A Framework for the Extension and Visualisation of Cyber Security Requirements in Modelling Languages

Curtis Leonard Maines

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy

August 2017

"The problem with the world is that the intelligent people are full of doubts, while the stupid ones are full of confidence." – Charles Bukowski

Acknowledgements

The completion of this work would have not been possible without the support and guidance of my family, friends and colleagues.

I would like to firstly thank my Director of Studies, Dr Bo Zhou. Although you started this project as my third supervisor, I could not imagine completing it without you as my first. Your support has been unwavering throughout the course of my studies. During times of uncertainty, you have always understood my doubts and were there to put them at ease. I hope I did not give you too many headaches, and am very grateful for your guidance.

Next, I would like to thank my second supervisor, Dr Stephen Tang. When I started University, I had no intention of postgraduate study, never mind a PhD. However, your support and encouragement gave me the confidence to apply. Both as an undergraduate and as a postgraduate student, you have always gone above and beyond to help me. You have always believed in my abilities and the quality of my work, and for that I am very grateful. I can say with a strong certainty, I would have never attempted a PhD without your encouragement, and as such, owe you any benefits it brings.

I would also like to thank my third supervisor, Professor Qi Shi. Your experience and guidance has ensured that my work reached its maximum potential. Regardless of our limited contact, I am very grateful for all your help.

Although no longer a member of my supervisory team, I would also like to thank Dr David Llewellyn-Jones. The few months you spent as my Director of Studies, improved my writing abilities more than five years of education throughout high school. So, for that, I am very thankful.

Next, I would like to thank my colleagues and friends, both in and outside of University. There are points during a PhD, in which only your colleagues can truly understand and help you with the levels of doubt and stress you reach. Irrespective of which of us finishes first or last, know I will be there to offer the same level of support in return.

Finally, there are no words to describe the appreciation I have for my family. Although it is difficult to describe the stress and uncertainties of a PhD, they have always been patient and supportive of my endeavour. I thank you not only for the support throughout my PhD, but throughout my entire life; I am eternally grateful for you all.

List of Acronyms

AR – Augmented Reality

BPMN – Business Process Model and Notation

BPEL – Business Process Execution Language

OMG – Object Management Group

PoN – Physics of Notations

RMIAS – Reference Model of Information Assurance and Security

SDLC – Systems Development Life Cycle

UML – Unified Modelling Language

VR – Virtual Reality

XML – eXtensible Markup Language

WS-BPEL – Web Services Business Process Execution Language

Table of Contents

List of Figures

List of Tables

Abstract

Almost half of UK firms claim to have been subject to some sort of cyber-attack or breach in the last 12 months, with an average cost per incident being around £20,000. Yet, even in the face of these ever-mounting threats, cyber security is still treated as an afterthought throughout the systems development lifecycle (SDLC).

Though literature is aiming to rectify this mindset through the proposal of multiple software security solutions, there is still a noticeable absence of any usable, expressive tool for designing cyber security into a system at the requirements stages of the SDLC. By not practicing secure by design, there is a risk of: poor defences, confused developers with no security guidelines to work from, a potential redesign of core functionality and very expensive patch management.

There have been several attempts at producing a solution, with modelling languages presenting themselves as the perfect platform to specify such designs. One can observe multiple publications throughout literature which propose the extension of these languages to include security expression. However, the ability of these propositions to provide comprehensive expression of the cyber security domain and remain usable alongside their parent modelling language, remains an elusive endeavour.

The aim of this thesis is to produce a solution which ensures the practicability of expressive and usable secure by design tool implementation. That is, by conducting an evaluation of existing attempts at security extension and extracting heuristics based on their current failings, combine them with proven scientific principles to produce a framework which will act as its own form of methodology to guide the development of a security extension to modelling languages.

1 Introduction

During software development, it is not unusual to model system requirements early in the systems development life cycle (SDLC). This not only helps to analyse and improve system functionality, but acts as a key resource during implementation. Yet, even though the same practice for cyber security requirements has proven to provide more robust systems, streamlined development and lower costs, there is still no standardised way of specifying cyber security at the requirements phase of the SDLC.

Business Process Model and Notation (BPMN) fulfils the requirements of visually representing business processes and is now the industry standard for their modelling (Chinosi & Trombetta, January 2012). Nevertheless, even though security directly affects business process functionality, and BPMN acts as the perfect platform early in the SDLC for specifying security requirements, it still offers no support for their specification (Saleem, et al., January 2012; Rodríguez, et al., April 2007).

There have been several attempts at implementing extensions, but they are being constructed unsystematically, without any empirical evidence to support their choice of concepts or notational design (Leitner, et al., 2nd-6th September, 2013). The frequent production of these extensions, further belies the need for a richer and more usable representation of security requirements in BPMN processes.

This thesis, presents work considering an analysis of existing extensions and identifies the notational issues present within each of them. It discusses how there is yet no single extension which represents a comprehensive, complexity managed range of cyber security concepts. Consequently, there is no solution even capable of becoming a standardised method of security specification within the industry.

To address this, a new framework is proposed that can be used to guide the extension, visualisation and verification of cyber security requirements within any modelling language. Using BPMN as a case study, this framework is justified through the implementation of a new security extension. This process provides several further contributions, including: the generation of an ontology of potentially modellable cyber security requirements; a new, complexity managed method of visualising different domain data within modelling languages with a potential of cross-discipline transference in the future; plus, a graphical framework for the creation of perceptually discriminable, semantically transparent, visually expressive symbols.

This solution is then evaluated against a set of proven scientific principles for assessing modelling languages and proves itself far superior to any other existing approach. Along with data from experiments involving novice and expert end users, the proposed solution shows real potential of becoming the first standardised means of specifying security requirements early in the SDLC.

1.1 Thesis Structure

Following this chapter, the thesis starts the literature review off by first providing some background on the themes that emerge within the work. That is, a general overview of cyber security, modelling languages, visualisations and finally the use of 3D in modelling languages.

After this overview, the “Physics” of Notations (PoN) (Moody, 2009) are discussed, detailing why the constituent principles act as the ideal set of heuristics for the scientific evaluation of existing security extensions to the BPMN language. Using the PoN, an evaluation is then conducted against six extensions, with publication dates from 2007 to 2015.

The aims and objectives of the work are then outlined, specifically regarding how this paucity of adequate secure by design tools within industry can be rectified. That is, through the creation of a framework for ensuring the expressive and usable extension and visualisation of cyber security requirements in modelling languages.

After this, a new set of heuristics are defined based on the failings of current extensions identified in Chapter 2. Unlike the PoN, which target modelling languages in general, these heuristics are focused specifically on security extension and the necessary requirements for avoiding their own issues.

This leads onto the core novelty of the work in which the proposed framework is outlined, detailing the various roles required, along with the individual components of each role for the successful implementation of a comprehensive, complexity managed cyber security extension for modelling languages.

From this, a case study using the BPMN language is conducted in which a new security extension is created to justify the claims of the framework. This section goes into explicit detail on how this new solution satisfies each component within the framework ensuring the previous issues are avoided. However, there are potentially a multitude of variations which can be created when using the framework. The purpose of the case study within this work, is to act as a form of verification for the framework to prove how ensuring the satisfaction of each component, can produce a comprehensive, complexity managed security extension.

After the case study is completed, there is a critical assessment of the produced solution. Upon comparing its PoN results against those of existing extensions, it is proven the far superior approach. There is also a discussion on the experiments that were carried out using both novice and expert security end users. That is, undergraduates who are relatively naïve to the cyber security domain, and industry professionals, who in contrast have a lot of experience within the domain. The combination of these, proving the extension and thereby framework as satisfying its claim.

Chapter 8 then concludes the work, outlining the novelties and contributions made within the thesis; primarily, the framework defined in Chapter 4. Along with the new approach visualising different domain data in modelling languages, the cyber security requirements ontology and the graphical framework for the creation of language symbols, this work provides multiple contributions to the area.

2 Literature Review

2.1 Background

The following sections provide a general overview of the themes that appear within the thesis. This then leads onto a more focused analysis of existing work within the area.

2.1.1 Cyber Security

One of the main themes within this work, is cyber security. Various dictionary definitions of this concept are as follows:

- *"precautions taken to guard against crime that involves the Internet, especially unauthorized access to computer systems and data connected to the Internet."* – Dictionary.com[1]
- *"The state of being protected against the criminal or unauthorized use of electronic data, or the measures taken to achieve this."* – OxfordDictionaries.com[2]
- *"ways of protecting computer systems against threats such as viruses"* – Dictionary.Cambridge.org[3]

Given that these definitions are aimed at someone who would typically have little to no knowledge of the area, they are somewhat crude and ambiguous in their attempts. A more refined and accepted definition of cyber security would be the one defined by Pfleeger and Pfleeger (Pfleeger & Pfleeger, 4th Edition, 2006). That is, cyber security is a combination of three core elements: *confidentiality, integrity* and *availability*. Together, these concepts define the main objective of security in computing: ensuring only authorised parties can access computer-related assets; only authorised parties can modify them in authorised ways; ensuring such computer-related assets are always accessible to authorised parties when required. This definition is more representative of the cyber security domain and acts as the three main goals every system should aim to satisfy.

Much like physical security, cyber security aims to reduce the number of vulnerabilities in a system to mitigate the possibility of any harm coming to valued assets (Pfleeger & Pfleeger, 4th Edition, 2006). For instance, a lock on a door is a physical security measure to protect the valued assets on the other side. Authorised personnel are then given keys which can be

[1] Dictionary; Browse : "cybersecurity"; (http://www.dictionary.com/browse/cybersecurity) accessed 02/11/2017
[2] English Oxford Living Dictionaries; Definition: "cybersecurity"; (https://en.oxforddictionaries.com/definition/cybersecurity) accessed 02/11/2017
[3] Cambridge Dictionary; Dictionary: English: "cybersecurity"; (https://dictionary.cambridge.org/dictionary/english/cybersecurity) accessed 02/11/2017

used to unlock this door and access the respective assets. In cyber security, a similar measure would be requiring credentials[4] to access assets on a system. As with the physical lock, authorised personnel will then be given access to the system's assets using their respective credentials such as usernames/passwords (keys).

There are multiple counter measures that can be used to try and protect a system, at a high-level, these can be categorised into: physical controls *(locks, guards, etc.),* procedural controls *(regarding people; laws, policies, copyrights etc.)* and lastly, technical controls *(access control, encryption, intrusion detection systems etc.)* (Pfleeger & Pfleeger, 4th Edition, 2006). All of which, contain many different solutions for countering many different threats.

The overall security of a system however, will generally depend on the cost of the counter measures (as well as the corporation's available funds) against the value of the assets they protect (Pfleeger & Pfleeger, 4th Edition, 2006). For example, expensive security on low value assets is illogical. In contrast, high value assets are worthy of more expensive counter measures. Therefore, just like physical security, a system is never classed as entirely secure, there will always be some vulnerabilities, the goal is to try and ensure these vulnerabilities will not compromise valued assets.

Just as cyber security is the term used for describing the protection of a system and reduction of vulnerabilities; cyber-attack is the one used for describing a successful intrusion into a system through the exploitation of a vulnerability.

In a recent UK government survey (April 2017) (Klahr, et al., April, 2017), it was reported that almost half of UK firms have been subject to some sort of cyber-attack or breach in the previous 12 months. This statistic rises to seven in ten when considering only large companies, with an average cost per incident being around £20,000, with the maximum in the millions. In terms of the global impact, McAfee published a report in 2014 claiming that the annual cost of cybercrime falls somewhere between \$375 billion *(£287 billion[5])* and \$575 billion *(£440 billion)*, more than the national income of most countries (Center for Strategic & International Studies, June 2014).

[4] Credentials refer to a unique knowledge based identifier. For example, a personal identification number (PIN), is a typical example of a credential used with Debit/Credit cards to authorise payment.
[5] Google Finance Converter; (https://www.google.co.uk/finance) accessed 17/07/2017)

Ciaran Martin, CEO of the National Cyber Security Centre, responded to the UK survey by imploring businesses to start prioritising security, stressing that for the most part, basic counter measures would be enough to mitigate most attacks (UK Government, 2017).

Nevertheless, for smaller businesses, their reasons for not implementing such security measures are based on their misconceptions of being too small to be a target, cyber security not being a priority to them, and that currently they do not regard themselves as being at risk (Klahr, et al., April, 2017). During this survey, however, it was discovered that of the smaller businesses contacted, 24% of them had experienced some sort of attack or breach in the previous 12 months (Klahr, et al., April, 2017), contradicting their beliefs of being insignificant targets to attackers.

Overall, this survey proves, big or small company, cyber security remains a huge problem in the UK (and globally for that matter), and shows no immediate sign of changing. The latest noteworthy attack at the time of writing this thesis is the ‘WannaCry’ ransomware infection that spread to organisations in 150 countries worldwide (Martin, et al., 17th May, 2017). While there may be no silver bullet to stopping these and other kinds of attacks from occurring, every piece of research conducted or new technology discovered that can aid in alleviating known issues, contributes to the overall solution.

One noticeable absence from current cyber security practice, is a formal or standardised way of specifying cyber security requirements early in the systems development life cycle (SDLC). There are various reports, which provide advice and guidelines on best practices for ‘secure by design’ (Arce, et al., 2014; National Cyber Security Centre, 2016). However, there are currently no adequate tools which allow developers to specify actual security requirements alongside their business process designs. Possibly the most likely reason for this, is the current attitude of industry, in which they regard security as an afterthought in the SDLC (Mohammed, et al., February 2017).

The absence of security designs during implementation can cause several issues, such as poor utilisation of existing techniques (Labda, et al., pp. 1399-1405, 24th - 28th March, 2014), degradation of a system (Saleem, et al., January 2012), confused developers (Khan, 17th - 20th August, 2015) and an overall higher vulnerability count (Sang & Zhou, 26th-28th October, 2015). Business functionality itself is also affected, as security directly impacts on these processes and therefore is a determinant in their designing (Saleem, et al., January 2012; Rodríguez, et al., April 2007). To put this into quantifiable data, fixing a vulnerability through software patching is 100 times more expensive than if it was done during development. As if that was not bad enough, patching almost always introduces new vulnerabilities as well (Mohammed, et al., February 2017). To summarise, by not practicing secure by design, there is a risk of: poor defences, confused developers with no security guidelines to work from, a potential redesign of core functionality and very expensive patch management.

Developing software which targets cyber security throughout the SDLC rather than after, although neglected by industry, is not something new to academic research. Mohammed *et al*. (Mohammed, et al., February 2017), recently conducted a survey of 118 studies in software security, to identify the various stages of the SDLC on which current research is focusing. The objective of their paper being to act as a foundation for any future research within the area. They identified the code level as the most investigated area for software security approaches, with the requirements modelling phase as the least investigated area, prompting the community for more work (Mohammed, et al., February 2017). This publication supporting the earlier statement on the absence of any security requirement specification tool.

2.1.2 Modelling Languages: BPMN and UML

The requirements modelling phase is typically quite early in the SDLC and is usually conducted using modelling languages such as UML (Unified Modelling Language). Using a textual approach has proven difficult to implement, understand, review and maintain (Silingas & Rimantas, 18th-21st May, 2008). This has subsequently led to a wider adoption of modelling languages, with UML becoming the industry standard in software design (Silingas & Rimantas, 18th-21st May, 2008). Furthermore, UML does not represent just one language, but allows for the specification of 13 different diagrams (Object Management Group, Inc., n.d.). Of these diagrams, however, the ones which deal specifically with the requirements modelling phase (use case, activity and state machine), have been rejected by business users for BPMN (Business Process Model and Notation) instead (Silingas & Rimantas, 18th-21st May, 2008).

UML is typically the language of choice when designing the object-oriented code structure for a piece of software, as far as business process specification goes though (requirements modelling), BPMN is the industry standard (Chinosi & Trombetta, January 2012).

There have been several comparisons between BPMN and UML Activity Diagrams (UML AD) within literature, however there is little evidence to support either language as the superior approach.

Peixoto *et al.* (Peixoto, et al., January, 2008), investigated whether BPMN could support its claim of being more understandable to a novice user than UML AD, by conducting experiments with undergraduates on their understanding of diagrams from each language. They concluded that BPMN showed no evidence of supporting its claim, as both languages provided very similar results.

Wang *et al.* (Wang, et al., 21st-23rd June, 2006)*,* also conducted a review of business process modelling languages. Although their paper was released quite early in BPMN's lifecycle (just 2 years after the language's release), they came to a similar conclusion as Peixoto *et al*. Basing their evaluation on the comparison of the languages' meta-models, graphical notations, serial representations and tool support, they found that both UML and BPMN standout as the best candidates for business process modelling. They did however, conclude their paper by claiming BPMN as the more likely potential standard.

Geambasu (Geambasu, 2012), compared BPMN and UML AD based on three criteria within their paper: *capacity of being readily understandable, adequacy of the graphical elements to represent the real business process of an organisation* and *mapping to Business Process Execution Languages (BPEL)*. They concluded their findings by stating that in relation to the first two requirements, both languages perform equally in this regard with neither outperforming the other. However, in terms of mapping the language to a BPEL, UML offers no support, whereas BPMN documents a mapping to the Web Services Business Process Execution Language (WS-BPEL).

Irrespective of which language is the superior approach (neither according to current comparisons), one conclusion that can be made, is that modelling languages offer the ideal platform for specifying cyber security requirements early in the SDLC. They have already proven themselves as effective tools for system design, and consequently, would ensure both system/business process and security are designed in parallel.

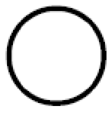
Based on that assumption however, a decision must be made regarding what language this work will focus on. In terms of extending a language with security requirements, it should not make a difference in any case, as the security extension should be as non-intrusive to the core language as possible and therefore not rely too much on the parents own approach. Given that BPMN is viewed as the industry standard by literature (Chinosi & Trombetta, January 2012), the Object Management Group (Object Management Group, Inc., n.d.) who represent an international, non-profit standards consortium and by consequence the industry itself; BPMN does present itself as the logical choice when neither language appears more superior. Therefore, given that both languages have been deemed somewhat equal by literature, the choice of language in this instance is BPMN.

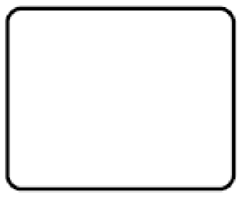
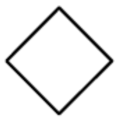
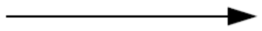
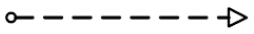
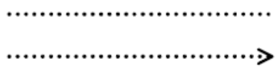
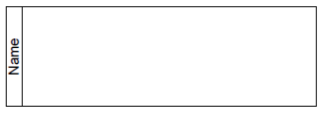
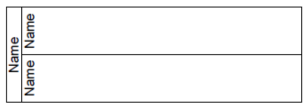
2.1.3 The Business Process Model and Notation (BPMN)

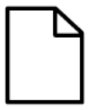
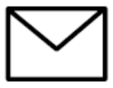
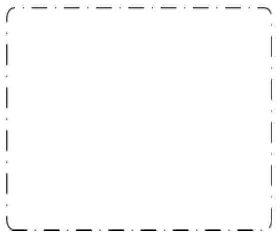
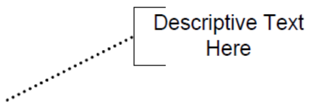
So, what is the Business Process Model and Notation (BPMN)? Firstly, a business process can be described as a set of linked tasks which must be executed in a specific order, collectively resulting in a business objective or policy goal being achieved. These tasks can be conducted across one or multiple organisations (Chinosi & Trombetta, January 2012; Meland & Gjaere, 20th-24th August, 2012). Business process modelling is the visual representation of business processes, offering an alternative form of understanding and allowing for the analysis and integration of the process into an enterprise (Aguilar-Saven, 2004). BPMN is a business process modelling language which is used by industry as a common standard between organisations (Object Management Group, Inc., n.d.; Bocciarelli & D'Ambrogio, 4th-7th April, 2011; Muller & Rogge-Solti, 21st-22nd February, 2011).

The BPMN 2.0 notation released in 2011 currently has 171 constructs that can be used for creating a business process diagram (Genon, et al., 2010). The Object Management Group (OMG) who currently maintain the BPMN language, categorise the notation into 12 basic elements. These, along with their descriptions and accompanying visual representation can be observed in Table 2.1. This table is an exact reproduction of the one located in the BPMN 2.0 handbook (Object Management Group, Inc., January, 2011) (pages 29-30), including all descriptions and notation. There is no claim to any information presented in this table and include it in this thesis only, as the official overview of BPMN's basic modelling elements as defined by OMG.

Table 2.1 BPMN 2.0 - Basic Modelling Elements

Element	Description	Notation
Event	An Event is something that "happens" during the course of a Process *(see page 238)* or a Choreography *(see page 339)*. These Events affect the flow of the model and usually have a cause (trigger) or an impact (result). Events are circles with open centres to allow internal markers to differentiate different triggers or results. There are three types of Events, based on when they affect the flow: Start, Intermediate, and End.	

Activity	An Activity is a generic term for work that company performs *(see page 151)* in a Process. An Activity can be atomic or non-atomic (compound). The types of Activities that are a part of a Process Model are: Sub-Process and Task, which are rounded rectangles. Activities are used in both standard Processes and in Choreographies.	
Gateway	A Gateway is used to control the divergence and convergence of Sequence Flows in a Process *(see page 145)* and in a Choreography *(see page 344)*. Thus, it will determine branching, forking, merging, and joining of paths. Internal markers will indicate the type of behaviour control.	
Sequence Flow	A Sequence Flow is used to show the order that Activities will be performed in a Process *(see page 97)* and in a Choreography *(see page 320)*.	
Message Flow	A Message Flow is used to show the flow of Messages between two Participants that are prepared to send and receive them *(see page 120)*. In BPMN, two separate Pools in a Collaboration Diagram will represent the two Participants (e.g., PartnerEntities and/or PartnerRoles).	
Association	An Association is used to link information and Artifacts with BPMN graphical elements *(see page 67)*. Text Annotations *(see page 71)* and other Artifacts *(see page 66)* can be Associated with the graphical elements. An arrowhead on the Association indicates a direction of flow (e.g., data), when appropriate.	
Pool	A Pool is the graphical representation of a Participant in a Collaboration *(see page 112)*. It also acts as a "swimlane" and a graphical container for partitioning a set of Activities from other Pools, usually in the context of B2B situations. A Pool MAY have internal details, in the form of the Process that will be executed. Or a Pool MAY have no internal details, i.e., it can be a "black box."	
Lane	A Lane is a sub-partition within a Process, sometimes within a Pool, and will extend the entire length of the Process, either vertically or horizontally *(see on page 305)*. Lanes are used to organize and categorize Activities.	

Data Object	Data Objects provide information about what Activities require to be performed and/or what they produce *(see page 205)*, Data Objects can represent a singular object or a collection of objects. Data Input and Data Output provide the same information for Processes.	
Message	A Message is used to depict the contents of a communication between two Participants (as defined by a business PartnerRole or a business PartnerEntity—*see on page 93*).	
Group (a box around a group of objects within the same category)	A Group is a grouping of graphical elements that are within the same Category *(see page 70)*. This type of grouping does not affect the Sequence Flows within the Group. The Category name appears on the diagram as the group label. Categories can be used for documentation or analysis purposes. Groups are one way in which Categories of objects can be visually displayed on the diagram.	
Text Annotation (attached with an Association)	Text Annotations are a mechanism for a modeler to provide additional text information for the reader of a BPMN Diagram *(see page 71)*.	

An example of a BPMN diagram, also taken from OMG can be seen in Figure 2.1.

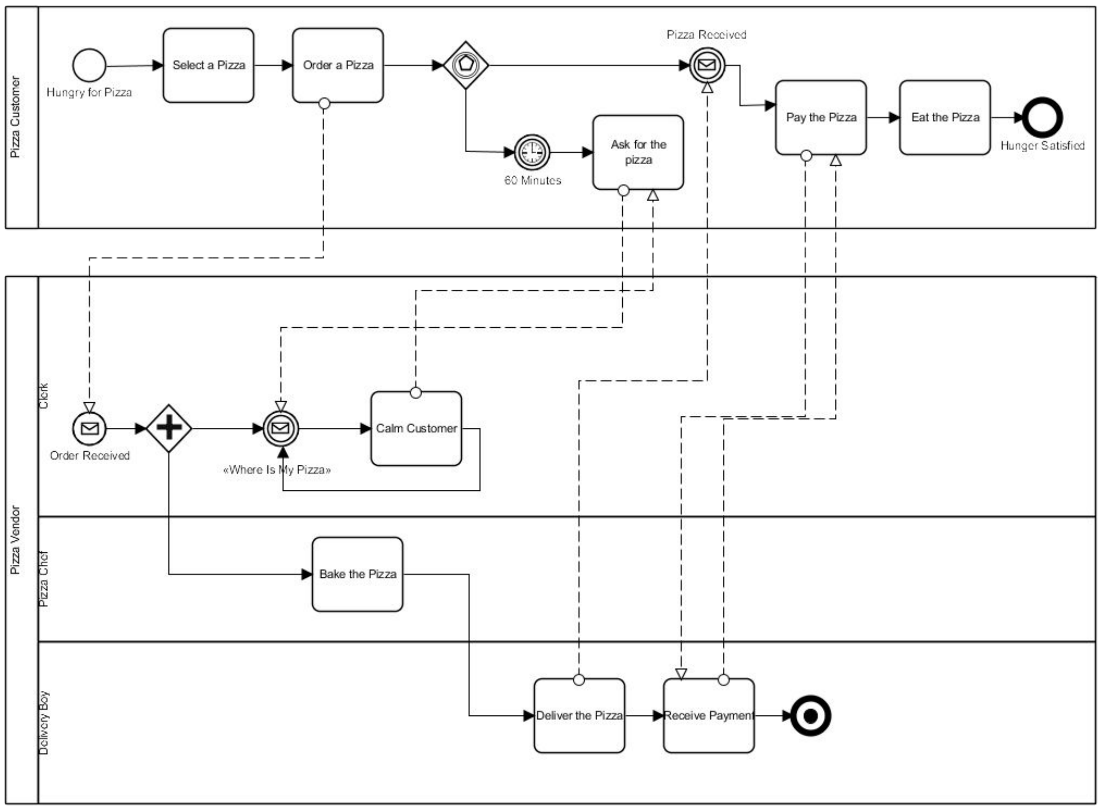


Figure 2.1 The Pizza Collaboration - Object Management Group, Inc.[6]

This diagram represents quite a popular business process amongst the BPMN community, visualising the process of ordering and receiving a pizza, showcasing the previously defined elements in a diagram which is a readily understandable process by both a novice and expert audience.

Nevertheless, although BPMN represents the standard for business process modelling, even though security directly affects the functionality of business processes, BPMN offers no support for specifying cyber security requirements (Saleem, et al., January 2012; Rodríguez, et al., April 2007). In the BPMN 2.0 handbook (Object Management Group, Inc., January, 2011), OMG provide a set of rules and guidelines for BPMN extensibility, providing explicit detail on what notation can be extended and in what way. In their own handbook on BPMN 2.0 (Shapiro, et al., 2012), Shapiro *et al*. summarise these into the following considerations:

- *Activity, event* and *gateway* elements must remain unaltered
- No new *flow* elements can be included

[6] Object Management Group; BPMN Examples: BPMN Quick Guide; (http://www.bpmn.org/) accessed 02/06/2017

- Any other existing element modified with new icons must maintain their core shape. For example, events should remain noticeable as a circle
- New artefacts which are connected to BPMN elements can do so through *Associations*, but must ensure that *Message Flow* and *Sequence* keep their integrity
- New artefacts should not conflict with any existing BPMN notation
- Colours and line styling can be freely used

It is worth noting that although BPMN is at the core of the work, the objective is not to produce a BPMN security extension but a framework for the extension of security in any language. Nevertheless, Shapiro *et al.* (Shapiro, et al., 2012) and OMG's guidelines (Object Management Group, Inc., January, 2011), for the most part, should be considered when extending any language not just BPMN, as their rules ensure the integrity of the core language.

However, this is not the case for all their guidelines. In one requirement, they detail how certain elements of existing notation can be modified to include new domain expression, this is a bad idea. It is likely that in certain circumstances, there may be existing notation which is already very close to a security requirement; i.e. *message event* and *encrypted message*. Therefore, one may choose to just replace the business notation with the security notation.

However, people in general, are quite poor at reading at what is in front of them, the two main flaws in human checking being omission and subjective judgement (Nakata, 25th - 26th November, 2014). These being a lack of urgency in the necessity of carefully checking and a poor ability in the practice itself, even at a cognitive level (Nakata, 25th - 26th November, 2014).

"Tkae for itanncse tihs snecente we are sltil albe to raed"

This ability is called typoglycemia, and proves that irrespective of what people are shown, their cognitive process tends to overlook abnormalities and extract only meaningful information (Nakata, 25th - 26th November, 2014). Regarding BPMN, a novice user to both the core language and a security extension will likely be able to cognitively distinguish between the two notations without issue. However, an expert user of the BPMN language, is likely to suffer from omission and/or subjective judgement.

This is because they have already mapped a large portion of the notation to memory. If a new extension is included which merely adds a few small icons to existing notation, it is very reasonable to theorise these could be overlooked for just the BPMN semantics. If this was to be translated into a potential scenario during implementation, say, a *message event* had an *encryption* icon on it, but the developer did not notice the icon due to the previous issues, this could manifest into sensitive details being transmitted as plain text. Therefore, the utilisation of existing notation for specifying new domain information is something to avoid not utilise. There are multiple methods for creating new symbols which should be sought out first.

Regarding existing BPMN security extensions, there have been several attempts within literature. However, very few have acknowledged the existing mechanisms in place for extensibility of the language (Braun, 13th - 16th July, 2015) (in fact, less the 20% of all domain extensions can satisfy BPMN's extension mechanism (Braun & Esswein, 12th-13th November, 2014)). The majority have also been constructed unsystematically, without any empirical evidence to support their choice of concepts (Leitner, et al., 2nd-6th September, 2013) or notational design. A more detailed evaluation of these extensions will be discussed later in the chapter.

2.1.4 Visualisation

Given that modelling languages and their extension are first and foremost a visual means of presenting information, it is prudent to provide a brief background on visualisations in general and their current application to cyber security.

Referring once again to dictionary definitions of "visualisation", these are as follows:

- *"to recall or form mental images or pictures…to make visual or visible"* – Dictionary.com[7]
- *"The representation of an object, situation, or set of information as a chart or other image."* – OxfordDictionaries.com[8]
- *to form a picture of someone or something in your mind, in order to imagine or remember him, her, or it"* – Dictionary.Cambridge.org[9]

[7] Dictionary; Browse : "visualize"; (http://www.dictionary.com/browse/visualization?s=ts) accessed 08/11/2017

[8] English Oxford Living Dictionaries; Definition: "visualization"; (https://en.oxforddictionaries.com/definition/visualization) accessed 08/11/2017

[9] Cambridge Dictionary; Dictionary: English: "visualize"; (https://dictionary.cambridge.org/dictionary/english/visualize?q=visualization) accessed 08/11/2017

In computing terms, visualisation has come to mean the representation of data or information in a “physical” visual form opposed to a textual one. For instance, one that relies on the perceptual system to identify information and trends, rather than the cognitive system. As an example, in the real world, someone buying an empty house would visualise their own furniture in each room, so they can create a mental picture of how the house would look if they lived there. In computing, rather than mentally creating this image, an actual graphic of some sort would be created so multiple people can view the same “mental image” or visualisation. This could be anything from a simple line-graph, to more elaborate 3D moving particle systems.

For instance, take the section of this literature review on cyber security, if one was required to provide an overview of what this section is about, a word cloud visualisation could be used. Word clouds have recently gained popularity not only due to their ability in providing an overview of text, but doing so in an aesthetically pleasing manner (Heimerl, et al., 6th-9th January, 2014). For example, see Figure 2.2 which demonstrates a word cloud based on the cyber security section of this work.

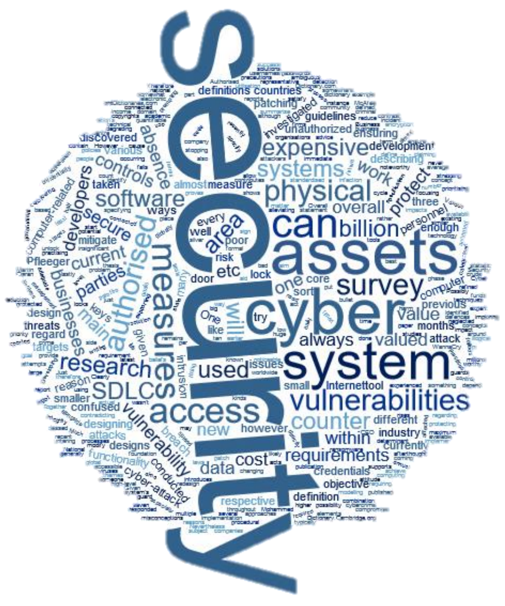


Figure 2.2 Cyber Security section – word cloud visualisation

A word cloud works by presenting words from a text at different sizes depending on their occurrences (larger size equalling more occurrences). From the visualisation in Figure 2.2, by extracting the enlarged words, one can see that the text discusses: *security, cyber, assets, system, vulnerabilities, access,* etc. Having read the section, the figure relates to earlier on, one can appreciate the effectiveness of word clouds and how they provide an accurate overview of the respective text by highlighting various keywords.

As mentioned, visualisations do not always have to be as elaborate as word clouds; graphs, pie charts and even road signs are all visualisations of some form of data/information.

2.1.4.1 Visualisation in Security

Returning to cyber security, there are a multitude of visualisation approaches being used within the domain, not only to aid in understanding data trends, but also to assist novices with tasks their lack of knowledge would otherwise make impossible.

One company, Kaspersky Labs (Lab, n.d.), created what they claim to be a real-time threat visualisation map of the world. Using their own detection network, every time an attack is identified, it is visualised on a 3D globe as a "laser" firing from one country to another. Where the practical use of this visualisation may only exist in specific scenarios, it is no doubt a very aesthetically pleasing and easier means of viewing such attacks. Compared with a textual approach, which may well be nothing more than a list of phrases such as "*country x attacked country y"*, viewing the visualisation in Figure 2.3, one can appreciate how the visualisation is not only a nicer way to receive the information but also a much faster way to comprehend it.

Figure 2.3 Kaspersky Cyberthreat Real-Time Map[10]

Another visualisation (Beltran, et al., 2012), proposed the use of game technology/interfaces to aid in the security of home networks. The concept of their work was to represent all devices connected to a home network as avatars (in this instance they chose to use pets). These would then be placed in an aerial perspective, graphical layout of

[10] Kaspersky Cyberthreat Real-Time Map; (https://cybermap.kaspersky.com/) accessed 15/11/2017

the home in their respective locations. Any threat that was then detected within the system, would be represented as a burglar avatar and animate differently depending on the type of attack (approach, grab or steal a device/pet avatar). The user would then be notified of this threat and given various actions to choose from to counter it: lock the burglar out (quarantine), closing doors (ports) or hiding the avatar/pet (switching the device off). This is a good example of how using an effective visualisation (gameplay/ game interfaces), can assist a user who would typically have very limited knowledge of the area, with performing somewhat advanced cyber security counter measures. A mock-up based on Beltran *et al.'* solution, can be seen in Figure 2.4.

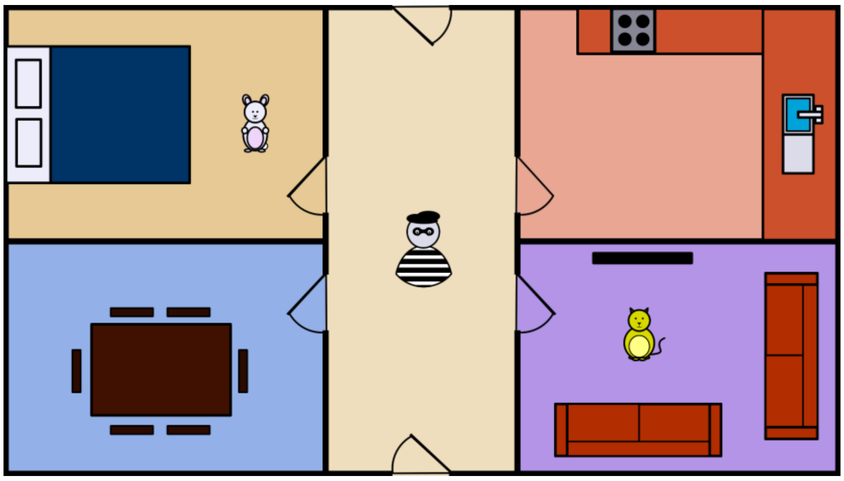


Figure 2.4 Mock-up of Beltran et al. proposed visualisation

An example of a visualisation which targets an expert in cyber security, but still aids the user in completing their task, is the solution proposed by Komlodi *et al.* (Komlodi, et al., 26th October, 2005). They present an intrusion detection toolkit that can be used for monitoring and analysing packet data. The visualisations they propose are very like that of a scatter plot (scatter graph), utilising different coloured circles (glyphs) to inform the user when packets meet certain criteria. One thing to note about their work, was the addition of 3D to aid in the analysis of the data. Where two axes, allow them to plot two sets of information about the data such as *time* and *source IP*, the inclusion of a third allows for more information to be represented such as the addition of *classification of alert*. A crude mock-up of the visualisation used by Komlodi *et al.* can be seen in Figure 2.5.

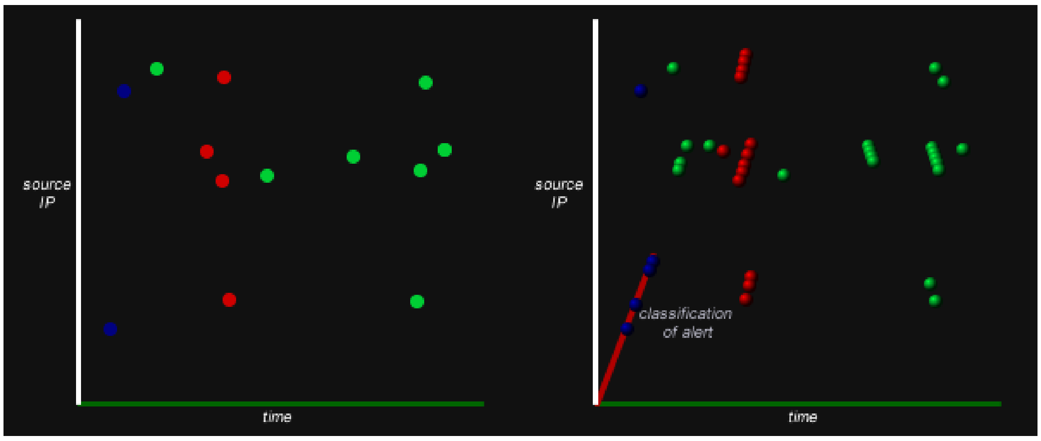


Figure 2.5 Mock-up of Komlodi et al. proposed visualisation (2D and 3D)

This is a good example of how utilising the full potential of visualisation (three dimensions), can greatly aid in the addition and representation of more information.

2.1.4.2 3D in Modelling Languages

As previously mentioned, modelling languages are their own form of visualisation and they too, have explored the possibility of 3D in their design. Nevertheless, the application of 3D to modelling languages, still feels somewhat in its infancy stage. There have been several publications in the area that propose the utilisation of 3D, but they do not take full advantage of what a third dimension can truly offer a language.

For instance, take Gil and Kent's proposal (Gil & Kent, 19th-2th April, 1998) of utilising the third dimension to visualise different kinds of 'edges' (relationships/arrows) between different diagrams. They suggest that by using the third *z* axis, one can represent more expressive relationships between diagrams than current 2D approaches allow for. A mock-up of their proposed solution can be seen in Figure 2.6.

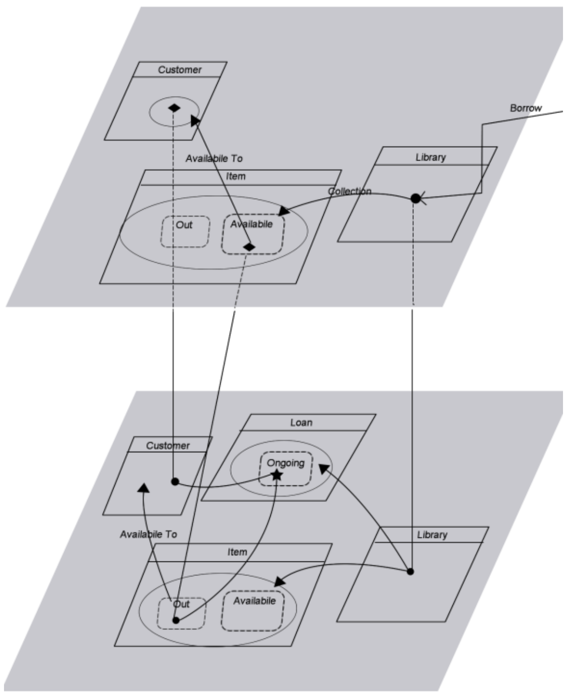


Figure 2.6 Gil and Kent 'Contract Box' diagram (mock-up)

As seen from the figure, they propose a ‘box-like’ approach, placing one diagram on top of a ‘box’, and another on the inside at the bottom. This then allows for various relationships to be drawn between each diagram. Although this approach allows for some sort of mental image in terms of how the diagrams could “overlay” each other, it still doesn’t utilise 3D to its full potential. For instance, this approach doesn’t make use of ‘the sides of the box’, or the *x/z, y/z* sets of axes.

Another example of 3D in modelling languages, is the work by Brown *et al.* (Brown, et al., 2011). They propose using the online virtual environment Second Life[11] as a means of collaborative business process modelling. They discuss how current 2D approaches to collaborative modelling do not allow for enough user information to be represented. For instance, it can be difficult to know what part of a diagram a user is currently working on in a 2D environment. However, in a 3D environment, each user can have their own avatar and be located ‘next to the element’ they are currently working on. For instance, viewing of Figure 2.7 demonstrates how the identification of each user’s location across a diagram is perhaps easier to see than different coloured cursors, which is a popular choice of user identification in current 2D collaborative software.

[11] Home: Second Life; (http://secondlife.com/) accessed 07/12/2017

Figure 2.7 Brown et al. Second Life 3D diagram creation environment

Nevertheless, 3D is once again not used to its full potential. The BPMN diagram seen in Figure 2.7 is still being represented across the same *x* and *y* axes, with very little use of the *z* axis (utilising it only as a simple menu system).

As seen earlier, effective use of 3D can provide more information than 2D alone. However, modelling languages are yet to utilise this ability to its full potential. Current approaches are opting for solutions which provide little to no advantage over their 2D counterparts. There is enough evidence from other areas of visualisation (Amini, et al., 2015; Marcus, et al., 11th-13th June, 2003; Teyseyre & Campo, 2009) to indicate that 3D can offer more to modelling languages than current attempts would have one believe. The area requires more investigation on how best to utilise the full potential of a third dimension.

2.2 Related Work

As discussed earlier in the chapter, there is a paucity of tools that support cyber security specification at the requirements phase of the SDLC. It was discovered that modelling languages offer the ideal platform for specifying such requirements, and that BPMN presents itself as the preferred language to base such a solution on (at least in the first instance). The remainder of this literature review now focuses on the evaluation of current BPMN security extensions, with an aim of identifying why no attempts thus far have presented themselves as a solution to the aforementioned problem.

2.3 'The "Physics" of Notations'

When it comes to the evaluation of these existing extensions (and modelling languages in general) there is one paper which stands out, "*The Physics of Notations*" (PoN), authored by Daniel L. Moody.

Moody defines nine principles that can be used for both creating and assessing visual notations (Moody, 2009). These principles are frequently used in the evaluation of modelling languages, including that of BPMN (Genon, et al., 2010; Popescu & Wegmann, pp. 166-173, 14th-17th July, 2014). However, this is not to say that they are without issue. Moody claims that the PoN can be used to *"evaluate, compare, and improve existing visual notations as well as to construct new ones"* (Moody, 2009). Nevertheless, Störrle and Fish rightfully state, that although Moody proposes nine principles (25 including their individual criteria), there is no clear definition of their relative weight (Störrle & Fish, Sep 29 - Oct 4, 2013). Moody discusses how there are trade-offs amongst principles, and satisfying one may negatively impact another. However, there is no guidance within the paper as to which principles should be prioritised.

Störrle and Fish attempted to quantify the first few principles and their criteria into a more usable equation within their paper. However, they also state how these equations are effectively placeholders and how most of the values used are assumption and guess-work, claiming that more empirical evidence is required before deciding on accurate, reliable values for each component (Störrle & Fish, Sep 29 - Oct 4, 2013). Given that their paper proposes the operationalisation of the PoN, as their equations are also based on assumption and opinion, they have turned a conceptual literature framework into a conceptual mathematical framework. Providing little in terms of contribution as they are still no closer to a quantifiable measurement based on evidence rather than conjecture.

Given that a solution to this issue is not an aim within this work, for the review of existing BPMN security extensions there will be an assumption that each principle has an equal value and simply stating whether an extension can satisfy each respective principle. Although this weighting may not be a true representation of each principle's value, the use of a consistent evaluation method across all extensions will still ensure a fair and scientific assessment of their notation, providing some form of grading system to use in their evaluation.

The nine principles within the PoN and their definition as they appear in the paper are as follows. All principles are named per their desired outcome rather than something to be avoided (Moody, 2009).

2.3.1 Semiotic Clarity

"There should be a 1:1 correspondence between semantic constructs and graphical symbols"

Moody expands on this, stating that modelling languages should aim to avoid the following issues: *symbol redundancy,* when one semantic construct can be represented by multiple symbols; *symbol overload,* when two or more semantic constructs can be represented by the same symbol; *symbol excess,* when symbols have no respective semantic construct, and *symbol deficit,* when semantic constructs have no respective symbol (Moody, 2009).

This principle (and its subcomponents), extend from ontological analysis. Therefore, to accurately assess or ensure this principle, one must first have an ontology of the respective domain. In this instance, the plan is to use the results from this review of current extensions to inform and later justify the creation of such an ontology for future solutions.

2.3.2 Perceptual Discriminability

"Different symbols should be clearly distinguishable from each other"

This principle is further divided into five more subcomponents: *visual distance,* describes how larger differences in visual variables (shape, colour etc.) can aid in better *perceptual discriminability*. *The primacy of shape,* states how shape is the strongest visual variable and therefore should be used as the primary method of ensuring this principle. *Redundant coding*, further emphasises the use of multiple visual variables, stating that although shape may be sufficient for discriminating between symbols, colouring them differently as well will help to further iterate this. However, *redundant coding* explicitly states that variables be used as a means of reinforcing an already existing notation; should these excess variables be removed, symbols should still be distinguished by a primary more robust notation, i.e. shape. *Perceptual popout,* refers to the idea that each symbol should have a unique value for at least one visual variable, for example, a circle in a row of squares will have high impact as it is very clearly different from the rest. *Textual differentiation,* describes how using text as a core distinguishing variable between symbols should be avoided, it is useful for symbol instances but as a core component of a symbol it is cognitively ineffective (Moody, 2009).

Visual distance is an example of what Störrle and Fish discuss in their paper (Störrle & Fish, Sep 29 - Oct 4, 2013). Although Moody acknowledges shape as the best distinguishing variable, the rest are open to conjecture, there is no clear framework as to what variables should always be included (maybe all of them?) or whether there is a limit and crossing this can ensue complexity issues. As for *perceptual popout*, the theory behind this principle appears logical. However, to some extent this is reliant on surrounding symbols being of a similar design. As mentioned earlier, a circle will stand out in a row of squares.

Nevertheless, if a yellow star, a red circle and a green triangle are placed alongside each other, will any of them stand out? They all have a unique colour and shape, but given that is the case for them all, negates the fact they are unique and that they stand out.

In this instance, the PoN are being used to evaluate an extension of an existing language, more specifically, an extension of a different domain. Therefore, this principle needs extending to state that as well as being discriminable from other symbols of the same domain, the extension should also aim to be distinguishable from BPMN itself. Moreover, there should be a way of relating symbols of the same domain whilst still ensuring they are distinguishable from each other.

2.3.3 Semantic Transparency

"Use visual representations whose appearance suggests their meaning"

Moody discusses three states any symbol can be categorised into: *semantically immediate,* a novice can infer the symbol meaning; *semantically opaque,* the symbol in no way infers its meaning, and *semantically perverse,* the symbol infers a different meaning to the one intended. *Semantic transparency* –like the previous principles– is also divided into more subcomponents: *icons,* refer to the perceptual goal all notation should aim for, where most notation in software engineering is made up of symbolic design (arbitrary distribution of shapes to concepts), an iconic design suggests that the shapes used are mnemonic to their underlying semantic. For example, a stickman to represent a person. *Semantically transparent relationships*, focuses more on relationships between symbols rather than the symbols themselves. For example, instead of using arrows to show inheritance, encompass all the symbols within another shape (Moody, 2009).

Semantic transparency is a strong example of one of the few principles that included a reasonable way of measuring success. Testing a notation with various audiences, one will quickly be able to identify whether it is semantically immediate, opaque or perverse. Nevertheless, although it offers a way of evaluating success, it does not offer anything in terms of symbol creation, specifying an iconic design approach is not enough to guide the creation of a notational construct.

2.3.4 Complexity Management

"Include mechanisms for dealing with complexity"

In the PoN, complexity refers explicitly to diagrammatic complexity, more specifically, dealing with large numbers of elements on a single diagram. Moody discusses two issues that can arise from poorly managed complexity: *perceptual limits,* the more diagram elements, the more difficult it is to distinguish between each of them; *cognitive limits,* only a certain number of elements can be comprehended at a time, if this is exceeded *cognitive overload* occurs. Two subcomponents to *complexity management* are proposed for dealing with these issues: *modularisation* and *hierarchy*. *Modularisation*, refers to the concept of breaking a diagram up into smaller, more manageable parts (subsystems). Whereas, *hierarchy* suggests representing diagrams at varying levels of detail. Unlike some of the other principles, these two complement each other well, making the incorporation of both into any notation a necessity as there are only positive outcomes (Moody, 2009).

Moody discusses how *complexity management* is one of the key issues most modelling languages fail to address. Although two proven concepts are proposed for dealing with complexity, this principle has the opposite issue of *semantic transparency*. That is, there is no quantifiable method for measuring whether the mechanisms in place are capable of *complexity management*. Terminology is proposed (*perceptual and cognitive limits*), but unlike *semantic transparency* it is not as simple as asking a novice user what a symbol means. Although Moody states that utilising *modularisation* and *hierarchy* will aid in *complexity management*, is this always the case?

2.3.5 Cognitive Integration

"Include explicit mechanisms to support integration of information from different diagrams"

The first statement made about this principle is that it is only relevant when multiple diagrams are required for one system, although this is almost always the case in software engineering and modelling languages. Nevertheless, it is worth noting that not satisfying this principle may be due to it being unnecessary. The subcomponents of this principle are *conceptual integration* and *perceptual integration*. *Conceptual integration,* is the utilisation of various mechanisms to support the user in constructing a mental representation of an entire system, this is typically done by providing the functionality to view all diagrams on screen at once. Although this renders elements near unreadable as they are scaled down, it provides a good representation of how each diagram fits into the entire system. *Perceptual integration,* is the incorporation of navigational assistance between each diagram, this can be done in various way such as labelling diagrams, signposting routes or including a map of the entire system (Moody, 2009).

As mentioned, this principle is only necessary when multiple interlinked diagrams are common within a language. However, this principle is somewhat ambitious in its objective, visualising multiple diagrams mentally or on a computer screen is a difficult task, more detail should be provided on how best to achieve *cognitive integration* within the PoN.

2.3.6 Visual Expressiveness

"Use the full range and capacities of visual variables"

This principle appears similar to *perceptual discriminability* and Moody acknowledges this by stating the difference: *perceptual discriminability* focuses explicitly on using multiple visual variables to distinguish between symbols. Whereas, *visual expressiveness* focuses on using all the available visual variables (horizontal position, vertical position, size, brightness, colour, texture, shape and orientation) to encode information and maximise computational offloading. Using the perceptual system to process symbols opposed to the cognitive system is a much faster means of symbol recognition. The first subcomponent of this principle is *use of colour*. Although some people with various visual impairments can struggle to distinguish between colours, it is a very effective notation. Differences in colour can be identified up to three times faster than shape and are also remembered more easily. Therefore, the utilisation of colour as a form of *redundant coding* is a very effective way of increasing the *semantic transparency* and *perceptual discriminability* of a notation. *Choice of visual variables: form follows content*, discusses how visual variables should be assigned per their properties. For instance, shape and colour have no logical order and can only represent nominal data. Whereas, position and size do and can be used to represent interval data. The last component is *textual versus graphical encoding*, again reinforcing the use of graphical notation opposed to text, stating how graphics are much more semantically transparent than text and should always be used when possible (Moody, 2009).

This principle is one of the strongest examples of a commonly ignored area within software engineering. The use of multiple visual variables is something that is very rarely seen, with most languages often providing very poor justification for their lack, such as an "unprofessional look" or "unusable by the visually impaired" (Moody, 2009). This is a dated outlook on notation design and evidence of how little progress has been made in the area. Modern operating systems across both desktop and smart devices are already making huge progress towards more visually expressive user interfaces (UI) with vastly positive feedback in terms of overall usability. Modelling languages need such a transition so they too can utilise the full effectiveness of visual variables.

2.3.7 Graphic Economy

"The number of different graphical symbols should be cognitively manageable"

Graphic economy discusses the other side of the *complexity management* principle. Where *complexity management* focuses on diagrammatic complexity, this principle focuses on the complexity of the notation itself. The graphic complexity of a notation is measured by the number of symbols it consists of, as the graphic complexity increases so too does the difficulty in successfully distinguishing and identifying notation symbols. *Reduce (Or Partition) Semantic Complexity*, is one of the methods Moody proposes for managing graphic complexity. As graphic complexity is driven by the number of concepts within the domain, reducing or categorising these will assist in lowering the graphic complexity. *Introducing symbol deficit,* is another means of decreasing graphic complexity. However, this does not mean in a negative way as discussed in *semiotic clarity*. In this instance, Moody advises the language designer to re-evaluate what concepts require a symbolic representation. If the graphic complexity of the language is unmanageable, it is likely that too many concepts have been assigned a symbol, it is better design to remove some of these and represent them all under one symbol, using text to define a more specific instance of each lower concept. *Increasing visual expressiveness*, can also assist in managing graphic complexity. As mentioned, increases in graphic complexity decrease the *perceptual discriminability* of symbols. Therefore, increasing the number of visual variables used in their design will increase their discriminability and counter the increased graphic complexity (Moody, 2009).

This principle is difficult to get right, the PoN discusses various ways of managing graphic complexity but there may be situations in which these principles cannot be applied. Although a language with too many constructs can overwhelm a user and lead to incorrect symbols being used, surely, it is still better practice to provide the ability to specify all concepts? Whether this be graphical or textual, when creating a diagram, a modeller requires explicit notation for specifying all concepts within the domain, high graphic complexity is preferable to a symbol (or textual) deficit language.

2.3.8 Dual Coding

"Use text to complement graphics"

This principle initially comes across as very contradictory. In the previous principles, Moody discusses at length about how text should always be avoided. However, this principle continues from *redundant coding*, stating how when used alongside graphical notation, text is useful for clarification of symbols and strengthening their underlying semantics. *Annotations* on diagrams are useful for description purposes and provide a more seamless reading of a diagram compared to the inclusion of a separate document. However, these should never be enclosed within or associated with a symbol, as this can cause the reader to misinterpret the annotation as notation. *Hybrid (graphics+text) symbols*, state the combination of both a symbol and text can reinforce the meaning of a construct by providing textual cues to users when they may otherwise be struggling to remember a symbol's semantics. This is especially useful for a novice as they do not need to access external resources when they inevitably end up in this situation (Moody, 2009).

This principle is already widely used in modelling languages but with slight variations to how Moody describes the most optimal way. *Annotations* for example, are frequently used in languages but they do normally include some form of enclosed shape around them. A combination of graphics and text however, is not something that has been utilised to its full potential. Taking BPMN as an example, the inclusion of the word "task" in one of the corners of a *task* element will have little to no effect on the symbol, but for a novice user be hugely beneficial when learning the notation.

2.3.9 Cognitive Fit

"Use different visual dialects for different tasks and audiences"

This principle defines why multiple visual representations should exist for a notation, stating that novice and expert users require different dialects for modelling. *Expert-Novice Differences* discusses how novice users require more discriminability between symbols and more *semantic transparency*, suggesting there should be two notations, one for "pro" users and one for "novice" users. *Representational medium*, suggests that notation should be designed so it can be easily represented in different forms. For example, simple enough that a software engineer (typically with limited drawing abilities) can draw it using pen and paper (Moody, 2009).

This principle is quite contradictory at times. Moody discusses how novices require more discriminability and more *semantic transparency*, yet makes no mention of what an expert requires, claiming that creating one notation targeted at just the novice, to meet their needs, is a poor practice. It appears "experts" have become accustomed to poorly designed notation and when presented with a well-designed (visually expressive) notation are dismissing it as “unprofessional”. It is difficult to comprehend that there is no single possible notation that is accessible by both the novice and expert user (traffic signs seem to work quite well as a prime example of such a notation). Having two separate notations for the novice and expert, creates a knowledge divide that can make the transition between the two an arduous task; it forces users to learn a new notation when they become “experts”.

As for the *representational medium*, this principle feels very dated now. Even though the PoN was published in 2009, it is difficult to understand how Moody can justify being able to draw notation using a pen and paper (or whiteboard) as a requirement. One paper (Luff, et al., 6th-10th November, 2004), was investigating ways to bridge the gap between traditional and digital environments five years before the PoN was published. There was an evidential acknowledgement that although paper still had its uses, the area was moving towards technological solutions. This is not surprising as the paper-medium is very restrictive in terms of what it can do, digital content is more expressive and offers a level of interaction unattainable by the traditional pen. Nowadays, there are technologies that include multi-touch interactive surfaces built into tables, which can allow for collaborative software development in a digital environment (Chaudron, et al., 23rd-27th October, 2016). Therefore, based on these technologies and the restrictions of a traditional pen and paper, it is difficult to prioritise this requirement in either the evaluation or creation of a modelling language.

2.4 Existing BPMN Security Extensions

Moody defines a visual notation as an explicit, predefined set of graphical symbols, compositional rules and definitions (or concepts) (Moody, 2009). Therefore, to meaningfully evaluate a BPMN security extension, it must first fulfil these three basic requirements.

From 2007 (Rodríguez, et al., April 2007) to 2015 (Sang & Zhou, 26th-28th October, 2015), there are six security extensions that can be identified which are capable of meeting these conditions, each solution provides a set of graphical constructs, at least one figure to demonstrate its composition (rules or otherwise), and naturally a set of domain specific concepts. Although a few other extensions exist in literature, it is impossible to evaluate them comparatively to these six, due to their inefficiency of figures showcasing their notation. Nevertheless, of the extensions that are reviewed, a broad range of attempts spanning across an eight-year period are explored.

Using the PoN as the form of scientific evaluation, the following section analyses these extensions and quantifies the total number of PoN principles they satisfy.

2.4.1 Rodríguez *et al.* (2007)

Although low in number of concepts, the Rodríguez *et al.* (Rodríguez, et al., April 2007) extension covers a broad range of the cyber security domain. They discuss how current approaches misinterpret security requirements for architecture specific restrictions, or altogether neglect specifying them and incorporate them in an *ad-hoc* way at implementation.

They propose an extension in which the business analyst can specify their own security requirements, providing security experts with a basis produced from a business expert perspective. The intention is that security experts can then refine these requirements ahead of implementation.

The constructs Rodríguez *et al.* include can be seen in Figure 2.8. Their underlying semantics are as follows (respectively from left-to-right): *non-repudiation, attack/harm detection, integrity, privacy, access control, security role* and *security permissions*; with the latter two being subcomponents of *privacy* and *access control* with no symbolic representation within the notation.

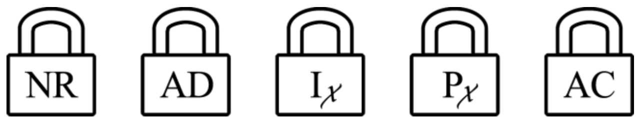


Figure 2.8 Rodríguez et al. security notation

The reason for the use of only high-level concepts is that the extension is targeted at the business analyst as opposed to the security expert, this does not however explain the absence of *availability*. Given the importance of *availability* for business processes and the fact it is a core component of cyber security (Pfleeger & Pfleeger, 4th Edition, 2006), it is a necessary requirement that should have been included.

By targeting the business analyst, the extension misses any potential of becoming widely accepted and utilised. Using only high-level concepts, it excludes the possibility of security experts being able to explicitly specify accurate security requirements; a truly comprehensive extension takes all users into account across varying levels of expertise.

Therefore, this extension has failed to meet the principle of *semiotic clarity* due to it being *symbol deficit*.

As for *perceptual discriminability*, this extension is quite poor. No visual variables are used to distinguish between symbols, relying solely on *textual differentiation*, thereby resulting in the notation having a *visual distance* of zero. The *primacy of shape* on the other hand, is used well to distinguish the notation from BPMN, a padlock is not currently used within BPMN, making it a robust shape for distinguishing from it. Nevertheless, this principle is not utilised for distinguishing between the security symbols themselves which is more important. Given that these symbols effectively have no *perceptual discriminability* from each other, this principle is unsatisfied.

The *semantic transparency* of the symbols is of a similar nature. Referring again to the padlock shape, although this represents a physical security measure as opposed to a cyber security measure, it is still something that is commonly associated with the domain. Therefore, the *perceptual resemblance* of the symbols is strong when trying to identify which are security. However, this principle focuses more on symbols resembling their own individual semantics rather than which domain they belong to. As these symbols have a *visual distance* of zero, it is virtually impossible for them to be *semantically transparent*, therefore, failing to meet this principle.

The *complexity management* of this extension is non-existent, having adopted the method of stamping symbols onto BPMN elements, see Figure 2.9. Where this may be an effective way of linking BPMN tasks and security concepts, diagrams quickly become overwhelmed when multiple concepts are placed on a single element. Thus, causing an end user to quickly reach both their cognitive and perceptual limits. Of the two mechanisms Moody suggests for managing complexity (*modularisation* and *hierarchy*), neither is utilised within this extension, nor is any other means of managing complexity.

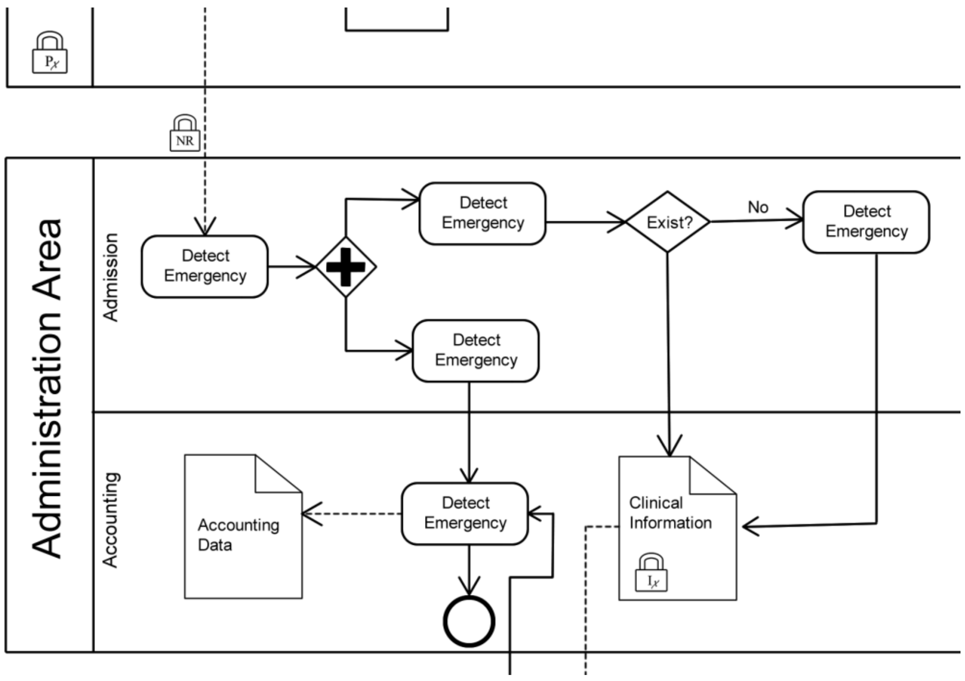


Figure 2.9 Rodríguez et al. BPMN diagram with notation (recreation from paper)

As mentioned, *cognitive integration* is not always required in a modelling language. In this instance, security is being used in a one-to-one relationship, each BPMN element can have its own security construct from the aforementioned list. Therefore, there is no scenario in which *cognitive integration* would be required for this extension, as each instance of security is encompassed within a single BPMN element. Primarily, this principle should be addressed by the parent language, as such, the extension has neither achieved or failed to attain it. It is stated instead, that although the principle's importance to modelling languages is acknowledged, it is difficult to evaluate security extensions against it. Therefore, it is omitted from this review, as it is inappropriate for the evaluation of language extensions.

The *visual expressiveness* of the notation is again non-existent, the only visual variable being used is *shape*, which as discussed, is identical for all symbols, effectively nullifying its use, resulting in a *visual expressiveness* of zero. As for *use of colour*, this extension opted for a simple black-and-white shell approach, with none of the symbols featuring a fill colour of any sorts. Missing not only an opportunity to target the perceptual system, but choosing to use the same colour scheme as the parent language, thereby, decreasing the core language/extension discriminability.

The *textual versus graphical encoding* of the extension is the exact opposite of the principle's goal. Rodríguez *et al.* have used text as the core distinguishing variable, with no visual elements to assist in the symbol recognition. Given that this extension utilises just one of the eight visual variables and fails to meet any of the principle's three subcomponents, it is a strong fail of *visual expressiveness*.

In this instance, *dual coding* is a difficult principle to judge. *Annotations* are used in the paper that introduces the extension, but they are semi-encompassed in a shape, something the PoN specifically states to avoid (Moody, 2009). As for *hybrid (graphics+text) symbols*, the notation does feature accompanying text on each symbol. However, this text is acronyms and not actual words. Given that the definition of *dual coding* states *"use text to complement graphics"* (Moody, 2009) and not "use acronyms as the core distinguishing variable", it is difficult to pass this principle. If the text on each symbol was full words, there would be no disputing the principle as satisfied. However, given that the pairing of acronyms and underlying semantics adds another step to symbol recognition, this is more of a hindrance to the symbol than benefit. The user is forced into identifying the acronym meaning rather than symbol meaning, more technically, the user is forced into using the cognitive system rather than perceptual system, so in this instance, the principle is not satisfied.

The *graphic economy* of this extension is satisfied, with only five symbols in the notation it is difficult to say the graphic complexity is unmanaged. However, as discussed earlier, the extension is very *construct deficit*. Economy is the management of available resources[12], this extension does not manage available resources though, it merely selects a few high-level concepts from the domain with several noticeable omissions (*availability*). To truly satisfy this principle, an extension needs to meet the requirements of *semiotic clarity* first. This economy is only managed due to its deficiency of security requirements. Nevertheless, given that the PoN states *graphic economy* as a notation whose symbols are cognitively manageable, the principle is fulfilled.

Cognitive *fit* is another principle which is difficult to accept as one of the nine PoN. These doubts being highlighted earlier, primarily, the disagreeing of needing two notations. Nevertheless, slightly reworded, this principle can be tested in the sense that a notation needs to be understandable and usable by both the novice and expert. In this instance, the notation is heavily dependent on having access to an external cheat sheet to reassert

[12] Oxford University Press; Dictionary: "economy"; (https://en.oxforddictionaries.com/definition/economy) accessed 16/01/2017

symbol (acronym) meanings. Therefore, the fact that a novice would almost definitely need to physically leave the language (software) to perform this action, is a fail of this principle.

Overall, this extension managed to satisfy just one (*graphic economy)* of the eight principles (disregarding *cognitive integration*), with the passed principle being solely down to the fact that the notation is symbol deficit. The extension feels negligent of modelling language design and as such fails to act as a suitable solution for specifying cyber security requirements in BPMN.

2.4.2 Saleem *et al.* (2012)

Another extension targeting the business domain expert is that of Saleem *et al.* (Saleem, et al., January 2012). They again emphasise the need for specifying security requirements at design-time as opposed to implementation, explaining how it can degrade the whole system if security is left entirely to developers. They propose an extension which allows business experts to specify their own security requirements which they can later refine prior to implementation.

They discuss several concepts to be included in this extension but only provide support for three concepts, these being, the core concepts of cyber security: *confidentiality, integrity* and *availability*; seen respectively from left-to-right in Figure 2.10. They also claim how *traceability* and *auditing* are important concepts but do not require modelling as they are inferred through the others.

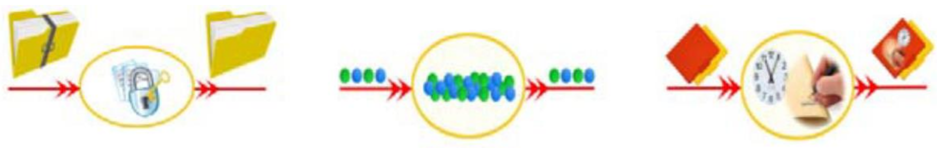


Figure 2.10 Saleem et al. security notation

Confidentiality, integrity and *availability* as mentioned are the core concepts of cyber security (Pfleeger & Pfleeger, 4th Edition, 2006), without question, they should be included within any security extension. However, to stop at this level of abstraction, irrespective of the user's expertise makes explicit specification of security requirements very difficult. Saleem *et al.* [5] state how *traceability* and *auditing* are inferred through these concepts. In truth, these three concepts together are broad enough to encompass any security requirement. If the aim of the extension is to allow the business expert to specify their own security requirements, they need specific enough concepts to do so, providing such high-level concepts restricts their ability to do this. Although in this case the aim is not to include an expert-level of concepts, there should be a sufficient and specific level of detail supported to allow meaningful expression (Recker, et al., 6th-9th August, 2009). Therefore, due to the extension suffering from *symbol deficit*, it fails to meet the requirements of *semiotic clarity*.

In this case, the *perceptual discriminability* of the symbols was done somewhat successfully. Although similar in theme, each notation has several distinguishing variables that do not include text, this makes the unique identification from any culture or language very easy. *Primacy of shape* is utilised well, each symbol has a similar outer shell, but they all feature a unique icon in and around them. The second visual variable in use is *colour*, it appears the authors used colour merely as decoration in their notation. Nevertheless, each symbol is distinguishable by a unique colour to some extent: yellow, green/blue and orange. As mentioned, it is unclear as to whether this was intentional, regardless *colour* has meaningful use in the notation. Therefore, this extension is capable of adequately satisfying the *perceptual discriminability* principle.

The *semantic transparency* of the symbols, shows that at least some thoughts have been put into their design, creating a notation whose symbols attempt to perceptually resemble their semantic meaning. Nevertheless, there are still some areas that allow for improvement. Take *integrity* for example, colour has been used as a primary notation to show an identical pattern both before and after a transmission, this is an appropriate visualisation. A better option however, would be to use shapes such as triangles and squares. Colour is very useful in notations, but like text should be a form of *redundant coding* and not a primary notation (Moody, 2009). Some people can struggle to distinguish between certain colours, with the likes of blue and green being a prime example for people who suffer from Tritanopia (Anon., n.d.), a variant of visual impairment. Although these symbols represent some of the more semantically transparent of existing notations, they are not yet at *semantic immediacy*. However, given that they are a moderate attempt, this notation satisfies the principle of *semantic transparency*.

The paper that introduces this extension includes a complete BPMN diagram, incorporating the security symbols in a real-world scenario, see Figure 2.11. As seen from this snippet of the diagram, the notation has poor scalability as the user relies too much on colour to differentiate between the symbols, this becomes clearer when viewing a grayscale version of them to the right of Figure 2.11. The symbols are also open and not enclosed in a uniform shape, when close together their boundaries become hard to see and they begin to corrupt each other. In this instance, a white rectangle has been placed behind each symbol (possibly due to poor software design), but the issue still stands, the *complexity management* of the notation is very poor. Although the symbols appear well designed against previous principles, when used in application there are several issues, failing *complexity management*.

Figure 2.11 Saleem et al. BPMN diagram with notation

The *visual expressiveness* of these symbols appears high when compared to other extensions. However, counting them out, the notation is only utilising two (*shape* and *colour*). From Figure 2.11 it can be seen that variables such as horizontal and vertical position are not considered, as each symbol is scattered across the diagram, the same can be said for size, brightness, texture and orientation. At no point do the authors make any suggestion through text or figures, that these variables are utilised within the notation. Although shape and colour are utilised well in this extension, to meet this principle, at least half of the visual variables should be met, as such, this extension fails *visual expressiveness*. Moody defines no criteria regarding what equates to a satisfaction of this principle (Moody, 2009). Therefore, for this work, it stated that at least half of the variables should be utilised as this represents the lowest amount in which a majority are used.

Dual coding is a straightforward principle to assess against this notation, none of the symbols include any text and as such fail to satisfy the *hybrid symbol* design approach, nor do any of the figures in the associated paper use annotations in reference to cyber security requirements. Therefore, it is easy to state the *dual coding* principle as unsatisfied in this extension.

As with the previous extension, this notation is also able to satisfy the principle of *graphic economy,* but only due to it being severely construct deficit. In truth, of all extensions reviewed, this one provides the least amount of support for security requirement specification, just three concepts are far too minimal for meaningful expression. This principle is only satisfied due to the failure of a more important one (*semiotic clarity*).

As for this work's version of *cognitive fit*, this extension is one of the strongest at satisfying the principle. Although not fully semantically transparent, the symbols in this notation have attempted to be a graphic mnemonic, therefore making them more accessible to a broader skill range of end users and not just the expert. As *dual coding* is not satisfied, it is very unlikely a novice could infer semantics from the symbols alone, thus the principle is not entirely satisfied. However, for competent to experienced users the symbols are distinguishable and recognisable enough to ensure the principle is met.

Of the eight principles, this extension satisfies four of them (*perceptual discriminability, semantic transparency, graphic economy* and *cognitive fit*). Although failing in comprehensiveness and *complexity management*, focusing on the design of the symbols themselves, this notation is the closest to meeting Moody's requirements (Moody, 2009). Though there is still much room for improvement, unlike other extensions it appears most of the author's time was invested in symbolic design as opposed to semantic constructs.

2.4.3 Salnitri *et al.* (2014)

Salnitri *et al.* (Salnitri, et al., 2014) propose a framework which aims to both model and verify security policies within a business process at design-time and after its deployment. The concepts they chose were derived from the Reference Model of Information Assurance and Security (RMIAS) (Cherdantseva & Hilton, pp. 546-555, 2nd-6th September, 2013): ***integrity, authenticity, accountability, non-repudiation, auditability, confidentiality, privacy, binding of duties, separation of duties, availability and non-delegation***, displayed left-to-right respectively in Figure 2.12.

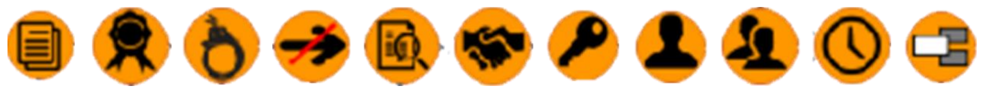


Figure 2.12 Salnitri et al. security notation

In comparison to others, this extension appears more extensive, but there are still issues and omissions within it. The level of abstractness amongst concepts needs to be consistent or display some sort of hierarchical structure to ensure the notation does not suffer from *symbol redundancy*. In this case, the concept of *authenticity* is being used to refer to the *authentication* of *identity*, determining whether a user is *authorised* to execute an activity, which is effectively three concepts in one. It seems more appropriate to change this concept to *access control* and have *authenticate, identify* and *authorise* as subcomponents. This will keep the level of abstraction the same, and through the subcomponents allow more specific concepts to be detailed within BPMN. Referring to the PoN, this issue is known as *symbol overload*, it is also enough to fail the principle of *semiotic clarity*.

The *perceptual discriminability* of these symbols is somewhat successful, but the *visual distance* between each symbol is dependent exclusively on *shape*, that being, the shape of the icons inside each symbol. Like other extensions, Salnitri *et al*. also opted for a consistent outer shell, which as discussed is an effective way of relating security elements. Nevertheless, unlike Rodríguez *et al.* who chose a unique shape not currently utilised within BPMN (padlock), Salnitri *et al.* have opted for one of the most highly utilised shapes (circle). This makes the *perceptual discriminability* more difficult than it should be, forcing users to not only distinguish amongst extension notations but that of the parent language. The authors have chosen to rely solely on an orange fill colour for distinguishing between the two domains, although a useful aid in distinction, colour is not robust enough for this, it should only be used as *redundant coding*. Nevertheless, with these issues considered, as the symbols within the notation are distinguishable from each other, this principle is satisfied.

As for *semantic transparency*, these symbols fall somewhere in the "*semantically translucent"* range. They are not capable of *semantic immediacy,* nor are they entirely unrelated-abstract symbols, there have been some thoughts put into their design but there are still uncertainties as to their exact meaning. Some symbols such as *authenticity, accountability* and *availability* can be inferred from their icon alone (assuming the user has prior knowledge of the concept definitions). Nevertheless, *integrity* and *non-delegation* are borderline *semantically perverse*, it is very difficult to determine their meaning from their icons alone. One can assume *integrity* is trying to infer an identical document both before and after a transmission (like Saleem *et al.* approach). However, this association is quite optimistic on the author's part, given how the documents are stacked and no form of transmission is represented. It is very difficult to identify. Overall, this extension has a weak satisfaction of the *semantic transparency* principle.

The *complexity management* of this extension is rather poor. Referring to Figure 2.13, which is a recreation of a diagram from Salnitri *et al.* paper (Salnitri, et al., 2014) (changing only the BPMN element text). It is very difficult to follow any logical route across the diagram. The use of colour ensures the security notation has a very high impact which immediately pulls the user's attention. Nevertheless, the relationships between security and business elements are defined using a dashed line. This form of notation is already heavily utilised across the diagram. As such, it is an arduous task identifying which security requirements link to which BPMN elements. Given that just eleven security elements are utilised on this section of the diagram, one is already nearing *perceptual* and *cognitive limits*. There has been no use of either of the mechanisms Moody proposes *(modularisation* and *hierarchy),* nor any other means of *complexity management*. As such, this extension fails to satisfy the principle of *complexity management.*

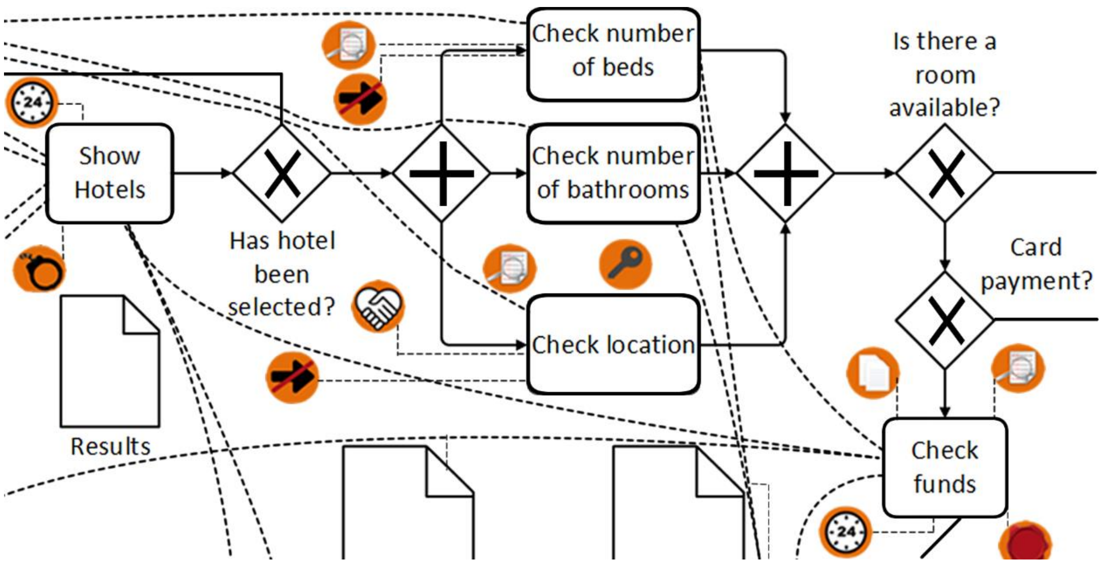


Figure 2.13 Salnitri et al. BPMN diagram with notation (recreation from paper)

As discussed, of the eight visual variables, Salnitri *et al*. only utilise one: *shape*. *Colour* is used to distinguish the notation from BPMN, but given that it is not used to encode domain specific information, it is excluded from the eight in this instance. Consequently, using just one of the eight variables makes the evaluation of this principle relatively straightforward; it is unsatisfied.

Dual coding as with previous extensions is a simple true/false evaluation. Salnitri *et al.* make no use of text within their notation, as either *annotations* or through *hybrid symbols*. Therefore, the principle is not satisfied within their extension.

Graphic economy also follows the same trend as other extensions. The principle is satisfied due only to its deficit. The number of symbols without question is cognitively manageable, using just eleven concepts from the domain, it is highly unlikely any user would feel overwhelmed. Nevertheless, referring to *complexity management* and Figure 2.13, it is easy to see how even a low number of concepts poorly managed can still cause several issues within a language.

The *cognitive fit* of this language is like that of Saleem *et al.,* as the symbols are *perceptually distinguishable,* have a fair *semantic transparency* and they are very usable by competent to expert users. The absence of *dual coding* again makes it difficult to conclude that novice users would infer symbol semantics. Nevertheless, the extension is capable of satisfying the principle of *cognitive fit*.

In summary, this extension is also able to satisfy four (*perceptual discriminability, semantic transparency, graphic economy* and *cognitive fit*) of the eight principles. However, in some ways this solution is more evolved than its equal scoring competitors. This is mainly due to the design of the extension, it appears to be built as a notation rather than as a set of unrelated symbols. Referring once again to Saleem *et al.* although similar in theme the symbols are arbitrary in terms of how their design relates to security or BPMN for that matter. Rodríguez *et al.* on the other hand, have a very relatable padlock to distinguish security from BPMN; failing on notation discriminability. Whereas, Salnitri *et al.'s* notation has a consistent outer shape and colour to distinguish from BPMN, and includes domain specific distinction through icons inside each shape. There is still much room for improvement, a circle for example is a poor choice given its high use in BPMN, the choice of concepts also leaves a lot to be desired. Nevertheless, in terms of notation design from Rodríguez *et al*. in 2007 to this notation from 2014, there is a noticeable evolution in their design. However, a domain-comprehensive solution which satisfies an adequate number of the PoN is yet to be created.

2.4.4 Labda *et al.* (2014)

Labda *et al.* (Labda, et al., pp. 1399-1405, 24th - 28th March, 2014), discuss how the increase of information online has greatly impacted privacy concerns, stating that most violations can be traced back to the lack of effective techniques for protecting privacy that could have been specified within business process modelling. With their extension focusing on privacy, the concepts included focus only on a specific section of cyber security, from left-to-right in Figure 2.14, the concepts they include are: *access control (allow), access control (prevent), access control (limited), separation of tasks, binding of tasks, user consent, necessity to know (high), necessity to know (medium)* and *necessity to know (low)*.

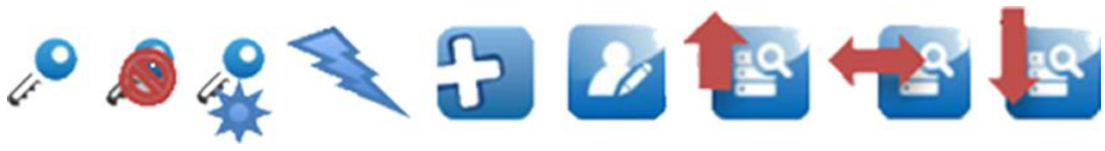


Figure 2.14 Labda et al. security notation

The subdivision of concepts is a refreshing change, finally allowing modellers to specify more detail. However, the choice of subdivided concepts is difficult to comprehend. *Allow, prevent* and *limited* are instances of *authorisation* (a subclass of *access control*), the ability to state whether access is granted or not within a design stage seems to offer the wrong level of abstraction. In practice, at the BPMN stage the analyst should only be specifying at what point it needs to be checked, it would be more appropriate for these concepts to be accompanied with text specifying 'who' access is allowed, prevented and limited to.

Nevertheless, in this instance, perhaps the author may have been referring to some form of classification instead. That being, each identity has a *trust level*, which then determines access to assets; like the UK government's security classifications (Cabinet Office, April 2014). If that is indeed in the case, the authors would benefit from the inclusion of more concepts. In this instance, sub-dividing *access control* into two lower level concepts: *identification* and *authorisation,* which each have their own concepts of *trust* and *asset classification level* respectively. These can then be assigned to the respective elements and provide a more meaningful representation within BPMN.

Along with former concepts, n*ecessity to know* subcomponents are also misleading, *high, medium* and *low* do not seem to portray any meaningful information. For an activity to be completed, information is either necessary or not necessary, *medium necessity to know* is too vague to be of any use. *Necessity to know* as a standalone concept is an appropriate security requirement, the subdivided concepts however are redundant.

The objective of this extension is to target privacy concerns, not the entire security domain. Nevertheless, given that other extensions are being reviewed as a "comprehensive security extension for BPMN" this one will be evaluated as such also. Naturally for the principle of *semiotic clarity* this will mean a dissatisfaction as there are multiple concept absences in comparison to the entire domain.

The *perceptual discriminability* of the extension is difficult to determine, each symbol is undoubtedly unique from the others, but the three *necessity to know* and *access control* constructs have little visual distance. The *visual distance* of the entire notation is just one, relying again on the *primacy of shape* for distinction. Nevertheless, given the robustness of shape as a notation and the fact each symbol can be distinguished, the extension satisfies the principle of *perceptual discriminability*.

There appears to have been an attempt at adhering to the principle of *semantic transparency.* However, the symbols are borderline between semantically opaque and perverse. For example, the concept of *separation of tasks* is represented by a lightning bolt; a strange choice of mnemonic. Likewise, *necessity to know* is represented by a magnifying glass and books/servers, something usually associated with searching and likely to cause some confusion. As such, although some thought has been put into *semantic transparency*, as the symbols are more *perverse* than transparent, the extension fails this principle.

Complexity management is again an issue with this extension, referring to Figure 2.15, one can see that Labda *et al*. have also opted for the symbol stamping approach like Rodríguez *et al.* Although this figure is a recreation from the original paper, one detail which was kept identical was the placement and size of the security symbols, this is to highlight the BPMN task element *"Request Medical data"*. As seen, this element includes two security symbols, both of which, are different sizes, and for the lack of a better term have been 'squashed' into the task element.

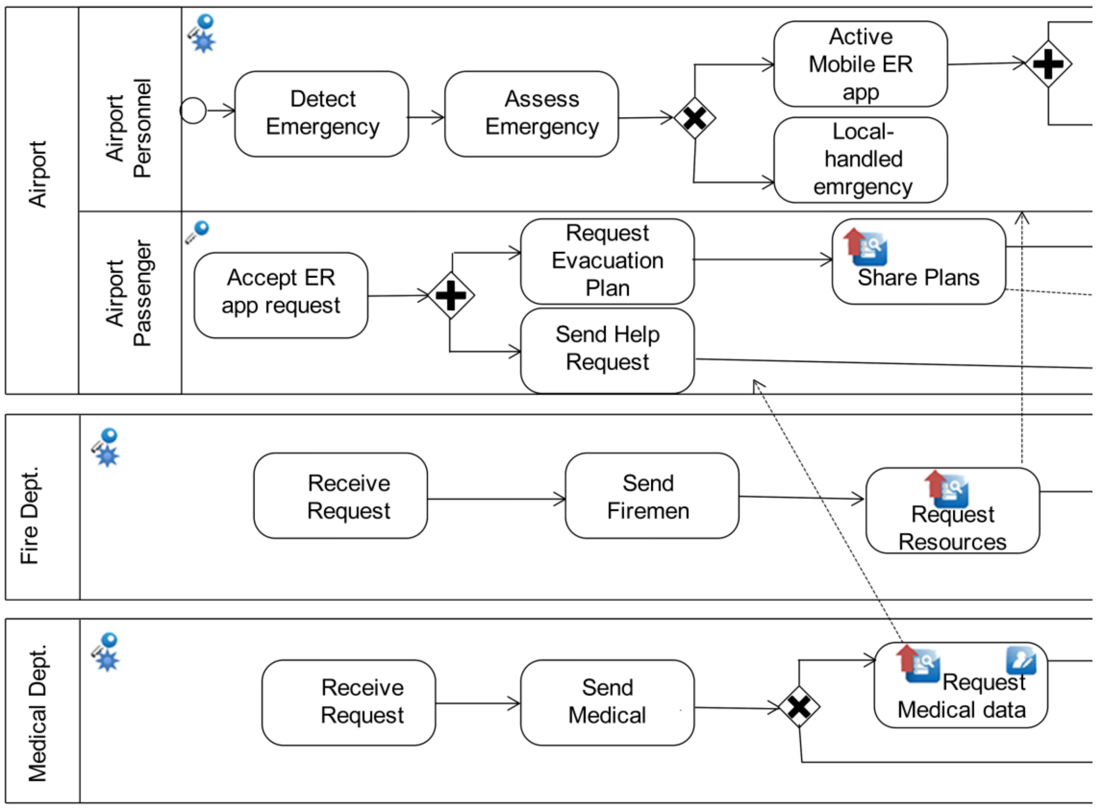


Figure 2.15 Labda et al. BPMN diagram with notation (recreation from paper)

Without even considering Moody's concepts of *perceptual* and *cognitive limits*, it is obvious how this approach to security/BPMN relationships can quickly cause several issues. In this instance, the authors chose to move the text slightly and scale down one of the symbols. The only other option would be to scale down the text or rephrase the element name to make it shorter, both of which are poor decisions from a usability perspective. As such, it is difficult to conclude that *complexity management* has even been considered in this extension, never mind satisfied.

The *visual expressiveness* of the symbols is like the previous extensions, the only utilised variable is shape, with colour once again acting as a weak secondary notation to separate the security notation from BPMN. Colour is also used to highlight some information within the symbols themselves. Looking specifically at the three *necessity to know* concepts, they are all identical except for an arrow changing orientation, to further iterate this, the authors highlight this difference using colour. This is a good and effective example of *redundant coding*. Without the red highlight, the arrows still convey the same information, the colour just helps to highlight this difference and allows for quicker symbol identification. However, even with this hint of design excellence, given that just two variables are used, the extension still fails the principle of *visual expressiveness*.

The *dual coding* principle was not satisfied in this extension. From inspection of Figure 2.15 it can be seen that no text is included within the symbols themselves *(hybrid symbols)* or as *annotations* across the diagram.

Without repeating the previous analyses, the *graphic economy* of this extension can be quickly summarised as being satisfied once again due to the languages deficit. Moody touches on how *complexity management* and *graphic economy* are very much influential on one another (Moody, 2009), that a language with good *complexity management* will be able to handle a larger *economy*. Given that no extension thus far can satisfy *complexity management,* supports this statement, as they all comprise of a very low number of modellable constructs, possibly to hide this known issue.

Cognitive fit in this extension is worse than the two previous solutions. As *semantic transparency* was not satisfied in this instance, the novice and even competent adoption of the language is relatively difficult. *Perceptual discriminability* although satisfied, was also quite poor in this extension with several similar looking symbols. Therefore, the extension is unable to satisfy the principle *cognitive fit*.

Overall, this extension managed to satisfy two (*perceptual discriminability* and *graphic economy*) of the eight PoN principles. Although created at a similar time to the Salnitri *et al.* extension, some of the approaches within this language are more dated. A quick review of existing solutions provides all the justification necessary to forewarn of the complexity issues associated with ‘symbol stamping’ relationships, yet Labda *et al.* still opted for this approach. It is also surprising that many of the extensions reviewed are all created after the publication of the PoN (2009), and yet no author appears to try and adhere to the principles.

2.4.5 Yulia Cherdantseva (2014)

Yulia Cherdantseva’s security extension (Cherdantseva, December, 2014), represents the most thorough attempt at extending BPMN with security requirements thus far. Given that their work was also completed as a PhD, this is not surprising. Salnitri *et al.’s* extension, as mentioned earlier, based their semantics on the RMIAS (Cherdantseva & Hilton, pp. 546-555, 2nd-6th September, 2013). This model was created during Cherdantseva’s studies and as such also represents the semantics for their notation. The motivation for Cherdantseva’s work is very much like that of this thesis, also noticing the clear absence of any standardised way of specifying security requirements early in the SDLC, aiming to solve this issue with a new extension.

The graphical symbols used within the notation can be seen in Figure 2.16.

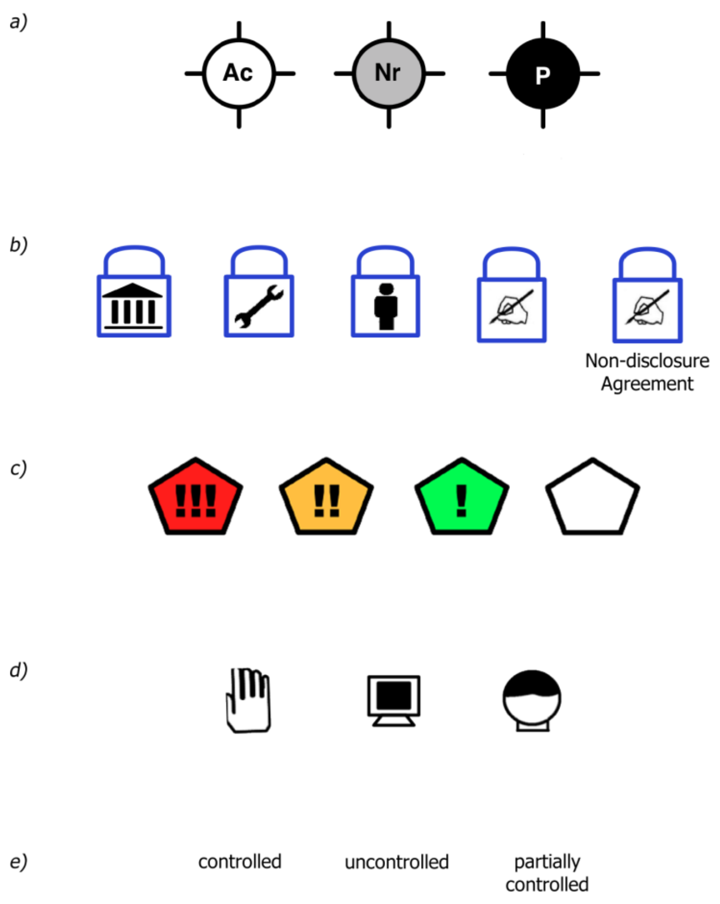


Figure 2.16 Cherdantseva security notation

This notation is categorised into five different sections with respect to the RMIAS. Observing Figure 2.16:

a. Represents a *security goal*. The darkness of the circle represents *criticality*, low darkness, low criticality and high darkness, high criticality. The letters inside each circle represent the concept the symbol is referring to: *confidentiality (C), integrity (I), availability (A), authenticity & trustworthiness (AT), non-repudiation (Nr), accountability (Ac), auditability (Au)* and *privacy (P)*.

b. Represents a *security countermeasure*. These are categorised into four types: *organisational, technical, human-oriented* and *legal,* respective to the first four symbols in Figure 2.16b. Specific instances of each type are instantiated using text below each symbol. For example, see *non-disclosure agreement* in the figure.

c. Represents *information sensitivity*. From left-to-right the symbols represent *confidential, restricted sharing, proprietary* and *public.*

d. Represents *information form*. From left-to-right the symbols represent *paper, electronic* and *verbal*.

e. Represents *information and secure swimlane location*. These “symbols” are self-explanatory, representing *controlled, uncontrolled* and *partially controlled* respectively.

Throughout the assessment of their extension, Cherdantseva did in fact evaluate the notation using the PoN. It was discussed earlier how Moody provides no information in regard to what constitutes a pass or fail of a specific principle and this is highlighted by observation of Cherdantseva’s own grading system and justifications, as they are very different to the interpretation within this work.

Firstly, the principle of *semiotic clarity* which requires a one-to-one mapping between semantic constructs and graphical symbols (Moody, 2009). Cherdantseva claims Secure*BPMN corresponds to the concepts within the RMIAS, stating that the reason for *location* and *security countermeasure* being text based is to ensure *complexity management*. However, referring to their *security goals,* it is clear to see they are also distinguished using only text; whether this be acronyms or words. Not only this, they openly admit to their notation suffering from *symbol deficit* and *symbol overload*. Therefore, it is difficult to conclude the notation as satisfying *semiotic clarity* based on their own analysis alone.

The *perceptual discriminability* of the language very much follows the same traits as the *semiotic clarity*. As mentioned earlier, when extending a language with a new domain, *perceptual discriminability* must be measured against the core language and between the extension notation symbols. Cherdantseva also acknowledges this, their work representing one of the very few who attempted to abide by the BPMN extensibility rules.

Nevertheless, although for the most part Secure*BPMN does separate itself from the BPMN language successfully, it is difficult to state the notation as satisfying the principle of *perceptual discriminability*, given just how much of the notation relies solely on textual differentiation amongst its own symbols. Specific instances of concepts which rely on text is typical modelling language practice. However, the *security goals* in-particular should have their concept encoded graphically into the notation and not rely on acronyms for differentiation.

Overall, not all the Secure*BPMN notation relies on textual differentiation, Figure 2.16c-d for example, represents somewhat successful *perceptual discriminability.* However, viewing this quantifiably, and giving each section of the notation an equal percentage in terms of overall notation, just Figure 2.16c-d satisfying the principle means 60% of the notation still does not. Therefore, Secure*BPMN also fails to satisfy the principle of *perceptual discriminability*.

The *semantic transparency* of Secure*BPMN is like that of Rodríguez *et al.* (Rodríguez, et al., April 2007). The language is almost capable of *semantic transparency* when trying to identify which concepts are security related, but this is not true for the entire notation. Although Cherdantseva states security goals as representing target icons, this is a bit of a challenge considering how much of the core symbol associated with a "target" has been removed (several smaller circles inside). Nevertheless, assuming Figure 2.16a-b, do in fact encode that they are security concepts, this still leaves a large portion of the notation without any justification.

Figure 2.16c for example, these symbols successfully satisfy the principle of *perceptual discriminability*. However, *semantic transparency* requires "visual representations whose appearance suggest their meaning" (Moody, 2009). This again is a bit ambitious, not many users would infer confidentiality ratings based on exclamation marks. As for the rest of the notation, this is either text-based or utilises existing BPMN notation. Therefore, the principle of *semantic transparency* is unsatisfied. Although this may be true when trying to identify security concepts in Figure 2.16a-b, for the remainder of the notation it is not.

Complexity management, the principle presenting itself as a consistent failure amongst every extension. Cherdantseva did not include any mechanisms within their work for managing the added complexity of a new domain, they do however, acknowledge the impact their work has on existing *complexity management* within BPMN. They also conducted a basic Q&A at a BPMN workshop and discovered that 40% of the audience did in fact feel their notation was too complicated (Cherdantseva & Hilton, pp. 546-555, 2nd-6th September, 2013). However, even in the face of all this feedback Cherdantseva responded:

> *"Therefore, Secure*BPMN does not include any specifically-developed mechanisms for managing complexity, but relies on the mechanisms inbuilt in BPMN because they are sufficient."* – Yulia Cherdantseva, Thesis (Cherdantseva & Hilton, pp. 546-555, 2nd-6th September, 2013)

This is a very surprising response for what is a huge issue within the software engineering domain (Moody, 2009), especially when there is enough evidence to support the need of more/newer *complexity management* mechanisms. To better grasp the level of complexity within the Secure*BPMN notation, refer to Figure 2.17, which showcases a BPMN diagram both before and after Secure*BPMN notation. (These were taken directly from Cherdantseva's thesis (Cherdantseva & Hilton, pp. 546-555, 2nd-6th September, 2013); these are in no way altered or manipulated, nor is there any claim to them within this work).

Figure 2.17 Cherdantseva security notation diagram (screenshot from thesis (Cherdantseva, December, 2014))

Although this diagram may have not quite reached *cognitive limits*, it is approaching it. Along with the evidence from the Q&A, this work claims Secure*BPMN as failing to consider the complexity of a new domain and by consequence the principle of *complexity management*.

Cherdantseva also acknowledged that *cognitive integration* is a parent language issue and as such is unfit to be evaluated against extensions.

The *visual expressiveness* of Secure*BPMN is one of the few principles Cherdantseva fairly assessed their notation against. As mentioned in their own thesis, Secure*BPMN does utilise six of the eight visual variables, these being: shape, brightness, colour, size, horizontal position and vertical position. Therefore, the extension is able to satisfy the principle of *visual expressiveness*.

As with previous extensions, *graphic complexity* is once again satisfied in Secure*BPMN. Cherdantseva discusses the various methods of adhering to *graphic complexity* and openly states their notation as purposely failing a lot of other principles to ensure *graphic complexity* is achieved. However, there is no justification on why such importance was placed on achieving this principle and not others. Moody states *complexity management* as one of biggest issues in software engineering (Moody, 2009). Not to mention, the other principles that were unsatisfied to ensure *graphic complexity,* using textual differentiation instead of symbolic. Given that BPMN itself contains 171 symbols (Genon, et al., 2010), it is perplexing as to why Cherdantseva sacrificed so many other principles for this one.

Within their work, Cherdantseva discusses the requirements of *dual coding* then discusses how text is used within Secure*BPMN. However, at no point is there a definitive answer on whether they believe they satisfy the principle of *dual coding*. This is likely because they are aware Secure*BPMN fails to satisfy *dual coding*. *Dual coding* requires “text be used to complement graphics” (Moody, 2009). However, throughout Secure*BPMN, text is only ever used as a primary means of distinction between symbols and never as *redundant coding*, nor is it used consistently throughout all notation, many symbols have no text. Therefore, failing the principle of *dual coding*.

Cherdantseva states their extension as being targeted at just the novice user, disregarding another notation for an expert. They also discuss how they developed their symbols to be easily utilised across multiple mediums, stating them as easy to draw and still meaningful when displayed monochromatically. As discussed earlier, there is no reason an expert will be unable to utilise a novice focused extension, therefore given that Cherdantseva’s other claims are also true, Secure*BPMN can indeed satisfy the principle of *cognitive fit*.

To summarise, Secure*BPMN can satisfy three of the eight PoN principles (*visual expressiveness, graphic complexity* and *cognitive fit*). Although Cherdantseva discusses at length the PoN and acknowledges the importance of various principles such as *complexity management*, there is almost no evidence to show there was an attempt at comprehensive satisfaction. Although Cherdantseva's thesis states its main contribution as a graphical extension for BPMN based on the RMIAS (Cherdantseva & Hilton, pp. 546-555, 2nd-6th September, 2013), it appears that like other authors, the real and only contribution in which effort was applied was in the choosing of semantics, or in this case the RMIAS.

It is stated multiple times how important semantics are in modelling languages. However, it is truly astonishing how many authors contribute so little in terms of the visual aspects of a modelling language, especially *complexity management*. Although within the PoN a total of eight principles can be observed which deal directly with symbol design and modelling language management, authors still insist on putting all their efforts in the one remaining principle: *semiotic clarity*.

2.4.6 Koh and Zhou (2015)

Koh and Zhou (Sang & Zhou, 26th-28th October, 2015) propose a security extension targeted directly at healthcare processes. They discuss how an increase in the digitising of healthcare data has increased the demand for more robust and secure systems, claiming that the lack of any formal means of representing security within BPMN causes more vulnerabilities within the final system. The underlying semantics for this notation are: *security task, authentication, access control, authorisation, harm protection, encrypted message, non-repudiation* and *secure communication* respectively from left-to-right in Figure 2.18. The final three symbols represent *confidentiality, integrity* and *availability*, with the stars visualising the required level for each one.

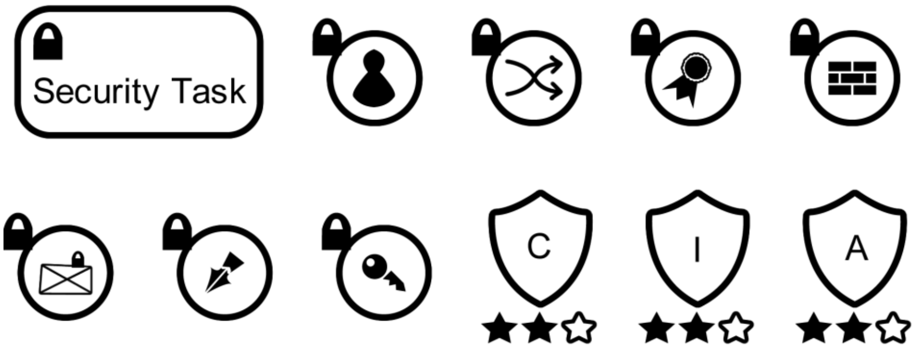


Figure 2.18 Koh and Zhou security notation

The concepts presented in this extension cover a broad range of the cyber security domain. However, there are still several issues. Firstly, Koh and Zhou once again incur *symbol redundancy* by not presenting constructs in a hierarchal structure. *Access control,* as previously stated, is capable of encompassing both *authentication* and *authorisation*, yet the authors place the symbols at the same level. It is difficult to envisage a scenario in which *access control* and *authentication* could be linked to the same task and still portray meaningful unique information. Likewise, *confidentiality, integrity* and *availability* all offer a user-driven star rating of desired security for each one. In this sense, the functionality is somewhat useless. No one would ever opt for lower security by choice, this is something which is determined by other factors such as budget etc. A dynamic rating, based on security used across a diagram is a potentially useful feature for a modeller, think grading system. A custom score however, is unlikely to ever change from the maximum three stars.

Given that this extension makes no use of any hierarchal structuring, it suffers from *symbol redundancy*. As such, the principle of *semiotic clarity* cannot be satisfied.

Continuing with the *perceptual discriminability* of the notation, this extension has both positives and negatives. The authors have used a padlock on each symbol as a way of identifying and separating them as security constructs, this is an effective way of separating the notation from BPMN (*primacy of shape*). However, this is almost nullified by the outer shape. Koh and Zhou have opted to use a circle for their outer shell which as mentioned earlier, is already heavily used within BPMN. Not only that, but their notation is not colourised in any way. Therefore, the symbols not only have similar shapes to BPMN, but given that they are also monochromatic too, makes distinction between the two domains quite difficult. As for the notation itself, the discriminability again relies on just shape as the core distinguishing variable. All symbols are near identical except for an icon inside each circle.

Although the PoN can be used as a scientific means of evaluating modelling languages, Moody is not explicit on what portion of each principle needs to be met before satisfaction can be granted. To some extent this may be obvious, in this instance however, it is difficult to say with confidence *perceptual discriminability* has been achieved. The authors rely solely on *primacy of shape* for distinction, three symbols also rely on *textual differentiation*, along with the fact that the notation is extremely like existing BPMN notation, this extension fails to satisfy the principle of *perceptual discriminability*.

The *semantic transparency* of the symbols is like Labda *et al.'s* notation, it is clear from inspection of each element that some thought has gone into achieving *semantic transparency*. However, some of the symbols are *semantically perverse*. Take *access control* for example (third symbol along on the top row). This icon is the universal symbol for "shuffle mode" on audio devices. To further justify this, see Figure 2.19, which is a screen capture of a Google Images search using the phrase "shuffle icon"[13].

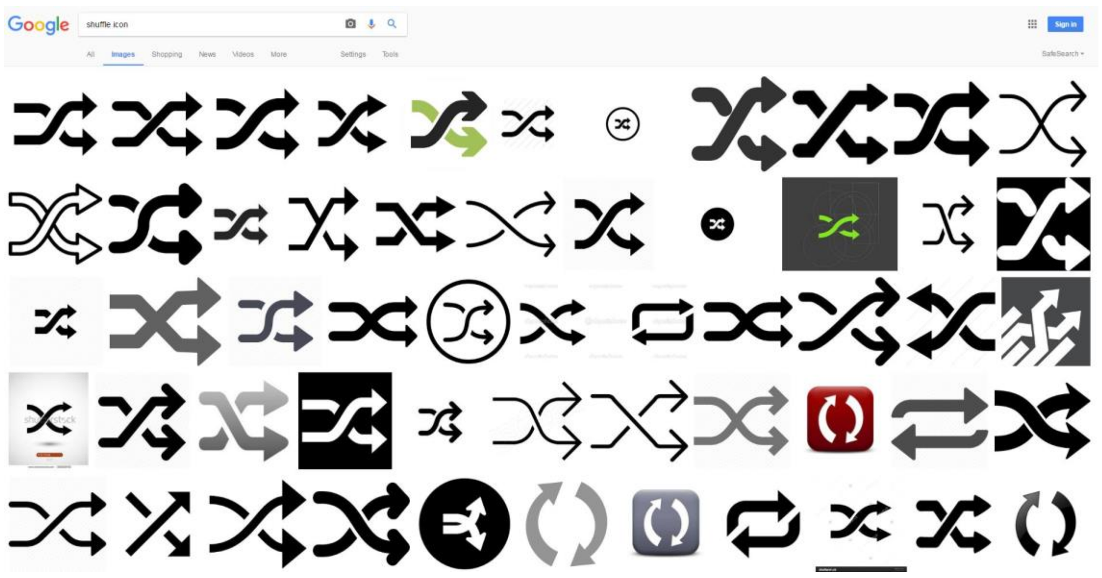


Figure 2.19 Google Images search, keyword "shuffle icon"

Google's search algorithm proves that the symbol Koh and Zhou use for *access control* is almost identical to over 80% of the images associated with "shuffle mode" (based on the first page). Although some users may infer a different meaning, from a usability perspective it is evident that this icon is already too familiar, a less generic one should have been used. Likewise, the symbol for *harm protection* appears to depict a *firewall*, undoubtedly a sub-concept, but a user may mistake this construct as just *firewall*. If this was the author's intention, naming the symbol *firewall* is better practice. However, if someone wishes to specify a different sub-concept to *firewall*, other readers may still associate the symbol to *firewall* given the focused nature of the icon. As *confidentiality*, *integrity* and *availability* also rely on textual differentiation, this extension again fails to satisfy the principle of *semantic transparency*.

The method of relating security and BPMN elements in this extension is a combination of symbol stamping and BPMN element replacement. That is, rather than use a BPMN

[13] Google; Google: Images "shuffle icon"; (https://goo.gl/USjwj9) accessed 15/03/2017

message event, one would use a security element *encrypted message*. However, as mentioned in an earlier section, this a poor way to model security.

Along with the reasons mentioned earlier, although business and security directly affect each other, readers do not always want or need to see both domains within the same sentence. Extensions should aim to be as non-intrusive as possible, allowing modellers the ability to remove (or hide) security requirements whilst still maintaining a complete business process. This is to minimise the cognitive strain on the reader. Remembering a single notation is difficult enough for the most part, if two separate notations are being used to construct one 'sentence', the difficulty in identifying the symbols is greatly increased. For example, imagine reading something which alternates between two languages every few words, it is not a user-friendly method of presenting information. There should be adequate relationships in place, but not to the extent they corrupt each other's domain and affect the integrity of the languages.

A portion of a BPMN diagram using Koh and Zhou's notation can be seen in Figure 2.20.

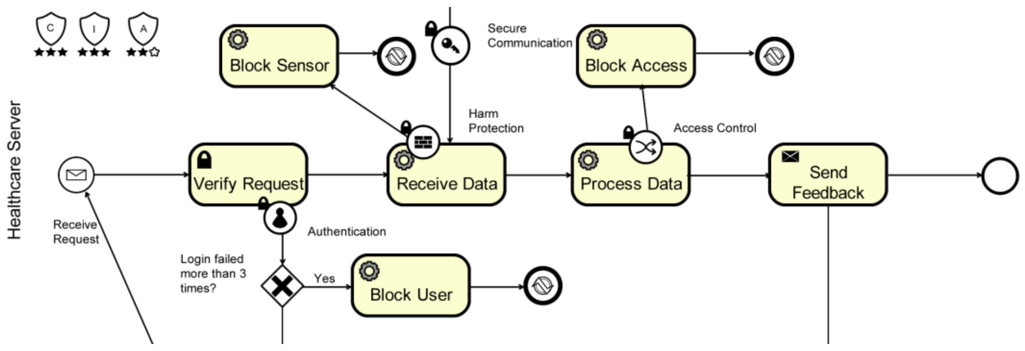


Figure 2.20 Koh and Zhou. BPMN diagram with notation (recreation from paper)

This figure emphasises the previous point on how the notation is far too similar to BPMN. It is not as clear in this figure as in Salnitri *et al.* what is security and what is business process. The use of colour would assist with this problem, though a more unique shape would be better still. "*Receive data*" particularly, demonstrates how Koh and Zhou's approach can at times cause symbols to almost overlap and corrupt each other. These issues emphasising there has been no attempt at managing the extra complexity created by the extension, a certain fail of the *complexity management* principle.

The *visual expressiveness* of this extension is rather poor, utilising only one variable, *shape*. *Confidentiality, integrity* and *availability* could be considered to use horizontal and vertical position, as they seem to consistently appear in the top left of each diagram in the paper. However, it is not explicitly defined whether this is a rule of the notation. Giving Koh and

Zhou the benefit of the doubt, the extension still only utilises three of the eight visual variables. Therefore, as stated earlier, given that at least half of these should be met to satisfy the principle, *visual expressiveness* is not satisfied in this instance.

Dual coding is partly satisfied within this extension. From inspection of Figure 2.18, one can see that *hybrid symbols* were not used. However, viewing Figure 2.20 it can be seen that *annotations* have been utilised, and to Moody's recommendations. That is, they are not enclosed within a uniform shape to indicate some form of notation. Therefore, the extension is able to satisfy the principle of *dual coding*.

Graphic economy is again achieved within this extension due to it being *construct deficit*. As for *cognitive fit*, given that neither *perceptual discriminability* nor *semantic transparency* has been achieved, it is difficult to conclude that *cognitive fit* has been satisfied. Although separate principles, they largely influence the outcome of *cognitive fit*. In this instance, there are potentially several issues with novices and the *semantically perverse* nature of this notation.

Overall, Koh and Zhou can satisfy just two (*dual coding* and *graphic economy*) of the eight PoN principles. The *perceptual discriminability* between the symbols themselves and that of BPMN is rather poor, along with the *perverse* icon design and the relationships used to link security and BPMN elements, this notation is far from an adequate security extension to BPMN.

2.4.7 Summary of Existing Solutions

Referring to Figure 2.21, this represents a plotting of PoN principles satisfied against extension publication year. In regards to research in general, one would expect to see an improvement across the years, in this instance that being a higher satisfaction of the PoN.

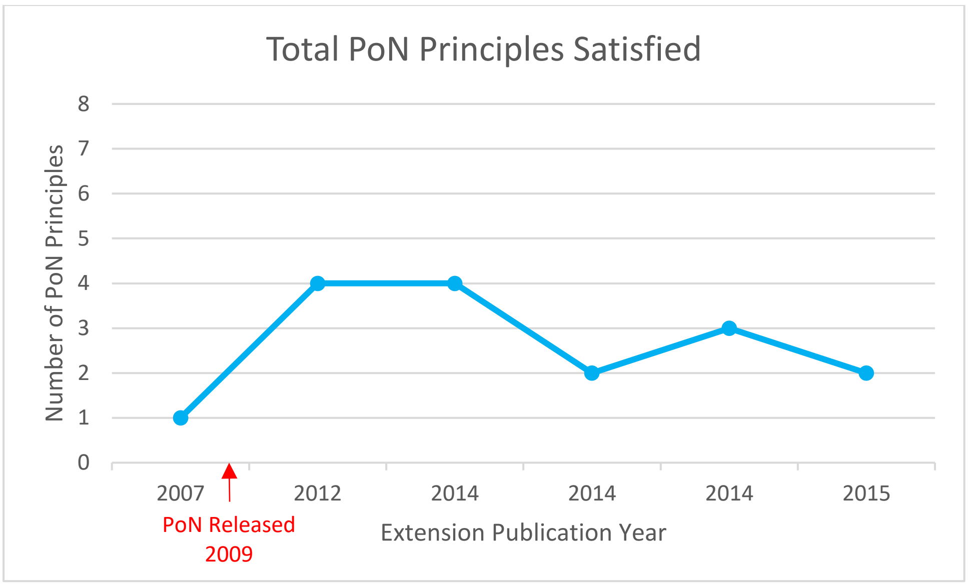
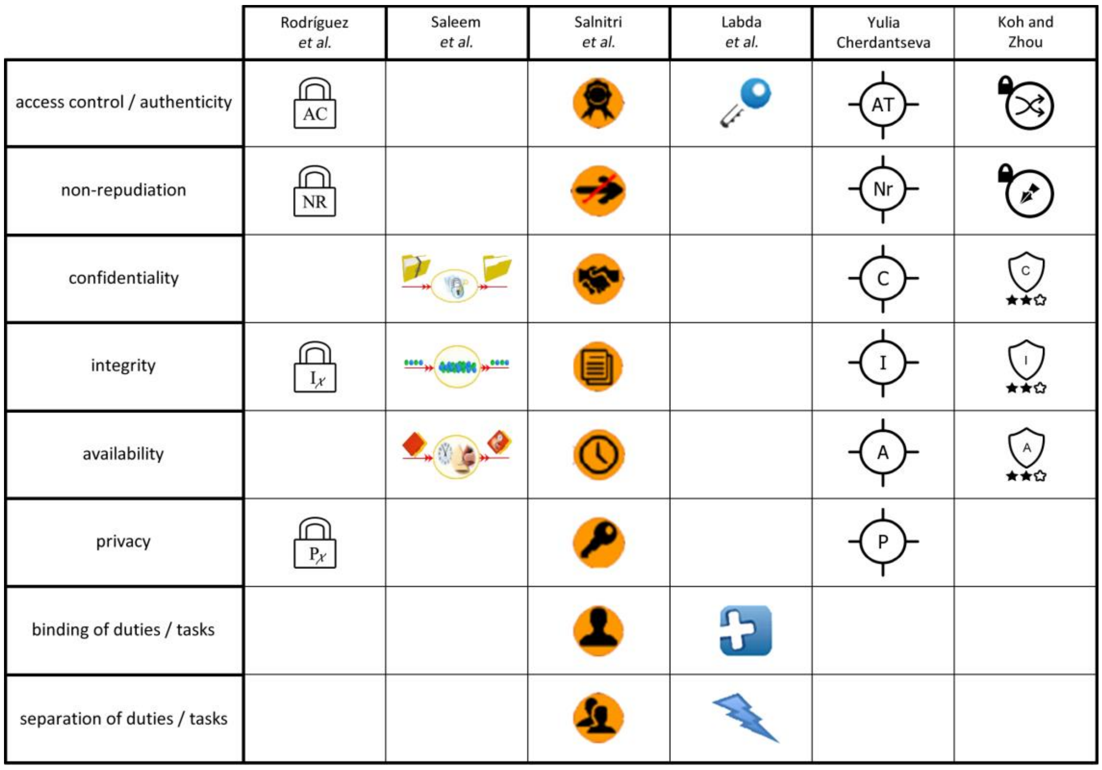


Figure 2.21 PoN principles satisfied against extension release year

As the PoN was not published until 2009, it is assumed that this is potentially the cause of Rodríguez *et al.* (Rodríguez, et al., April 2007) poor satisfaction. Likewise, as the next extension published by Saleem *et al*. (Saleem, et al., January 2012) was in 2012 (three years after the PoN publication), possibly explains the sudden increase in principle satisfaction. However, 2007 also marked one of the most substantial tech releases in history, that is the first iPhone (Apple Inc., 2007). In regards to the UI of this new device, this caused much controversy across the industry. Primarily, between whether a 3D skeuomorphic design or a cleaner 'flat' 2D approach was the more user friendly (Curtis, 2015; Stickel, et al., 22nd-27th June, 2014). Regardless of which was the better approach, UI and icon design was a hot topic. Therefore, it is difficult to conclude whether the PoN, or industry wide controversy on what contributes to good design, contributed to Saleem *et al.'s* satisfaction count. Regardless, the expected trend of more principles being satisfied as time goes on, occurred.

Nevertheless, observing the graph from 2012 onwards, the number of satisfied principles plateau and eventually decrease. The reasoning for this is not as easy to identify (or even guess at) as before. Although Apple did abandon their skeuomorphic approach shortly after 2012 (Curtis, 2015), the creation of apps (and thereby their icons), has only increased in recent years (Apple & AppleInsider, n.d.; Hou & Ho, 26th-30th August, 2013; Shu & Lin, 7th-9th August, 2014). Therefore, one would expect to see a further increase again in principles satisfied, given the abundance of well-designed icons that are now seen every day, evidently however, this is not the case.

Referring to Figure 2.22, one will be able to see a comparison of symbols from the previously assessed notations. This figure provides an interesting insight into existing icon association in the cyber security domain. Take for example, *availability*, (dismissing Cherdantseva's and Koh & Zhou's symbols which are based on *textual differentiation*), two authors include clocks in their icon. Therefore, one can assume when creating an *availability* icon, the use of a clock is likely to be a globally recognised symbol for the concept.



	Rodríguez *et al.*	Saleem *et al.*	Salnitri *et al.*	Labda *et al.*	Yulia Cherdantseva	Koh and Zhou
access control / authenticity	AC		●	●	AT	●
non-repudiation	NR		●		Nr	●
confidentiality		●	●		C	C ★★✩
integrity	I_X	●	●		I	I ★★✩
availability		●	●		A	A ★★✩
privacy	P_X		●		P	
binding of duties / tasks			●	●		
separation of duties / tasks			●	●		

Figure 2.22 Notation comparison

On the contrary to this, *access control, non-repudiation* and *confidentiality* have very different approaches in their icons, there is no obvious symbol which appears to be the globally recognised icon for each concept. Comparing Salnitri *et al*.'s *privacy* symbol and Labda *et al.'s access control* symbol, they are both the same. This is somewhat worrying from a modelling perspective, especially for the cyber security domain. Given that neither of the authors utilise *dual coding*, there is a very real chance that either of these symbols could be misinterpreted for the other. If this mistake is not rectified and developed into the system, the potential security issues could be very severe. The cyber security domain acts a strong case study on the potential impacts of poor icon and notation design, justifying the need for a more visually utilised security extension.

2.5 Chapter Summary

To summarise, throughout this chapter there has been a discussion on the various themes that emerge within this thesis. That is, an overview of cyber security in general and some of the issues with current practice. More specifically, the lack of support for specifying cyber security at the requirements phase of the SDLC. A comparison of BPMN and UML was then considered, detailing why modelling languages act as the ideal platform for specifying such security requirements. The background review was then finalised by exploring several visualisations within the cyber security domain and discussing the potential benefits of 3D to modelling languages.

This then led onto the more focused work of this chapter, which consisted of an evaluation of existing cyber security extensions to the BPMN language. The purpose of this evaluation being, to identify the specific areas of language design current extensions are failing in.

Throughout the next chapter, the aims and objectives of the work are extracted from the literature review, along with a set of heuristics for avoiding the identified issues in the previous evaluation.

3 Problem Analysis

3.1 Aims and Objectives

As seen from the literature review, cyber security remains a huge problem at an international scale; one reason for this being, the absence and practice of secure by design. Although there are multiple reasons for justifying secure by design, security remains an SDLC afterthought. Studies have shown that there is a lot of literature aiming to rectify this mindset by producing new tools to aid in the secure by design process. However, security specification at the requirements stage, is yet to receive the same attention. Although there has been some work in the area, there is yet no solution which has presented itself as an industry standard or even a potential contender.

Therefore, the aim of this work is as follows: "*to contribute to the global effort of secure systems by ensuring the practicability of expressive and usable secure by design tool implementation*."

To achieve this aim, the following objectives must be fulfilled:

1. Identify the areas of modelling language design current BPMN security extensions are failing.

2. Define a set of heuristics that can be used alongside the PoN for the avoidance of these problem areas in future solutions.
3. Produce a structured framework that can be used for the generation of a cyber security extension, which details the required components to satisfy the proposed heuristics and PoN.
4. Undertake a case study to produce a cyber security extension based on the proposed framework to evaluate its effectiveness.
5. Conduct end-user experiments with the proposed extension to evaluate its usability and user acceptance.

These objectives are very like the waterfall methodology (Balaji & Murugaiyan, 2012) of software development, with each point to some extent aligning with a specific step in the model. However, the aim of this work is not to produce a piece of software like current attempts, but to ensure the practicability of producing expressive and usable software (secure extensions). Therefore, in terms of the waterfall model, the focus of this work lies within the design component, as this is the area in which such criteria (heuristics/PoN) are achieved. This is something which as seen from the literature review, has thus far proven to be largely neglected by current attempts.

For that reason, this work aims to contribute to the area through the production of a framework. The aim of which, is to act as both a structured overview and set of guidelines/methodology for the design and implementation of a security extension. By conducting evaluations of existing attempts and extracting problem areas, a set of heuristics can be defined to further support and accompany the PoN in terms of good modelling language design practice. These can then be used as a foundation for the creation of the framework to ensure such principles are adhered to. The waterfall methodology can then be continued through a case study of the framework based on the BPMN language to evaluate its effectiveness. Should this evaluation come back positive and consequently prove the framework as a successful tool for ensuring the practicability of producing expressive and usable security extensions, the project aim can be considered accomplished.

The next section concludes the literature review and problem analysis by outlining the proposed heuristics that were extracted during the review of existing extensions.

3.2 Heuristics

As mentioned, the PoN can be used for the scientific evaluation of a modelling language as well as design requirements for the creation of one (Moody, 2009). Nevertheless, having identified new issues encountered with “extensions” to an existing language, a new set of requirements for avoiding these in the future can also be defined.

Throughout this section, a set of heuristics a security extension should aim to achieve are explicitly detailed. As with the PoN, these are also worded per their desired outcome rather than something to be avoided.

3.2.1 Comprehensiveness

Comprehensiveness in many ways is like Moody’s principle of *semiotic clarity*. However, *semiotic clarity* makes assumptions regarding what a language developer already has at their disposal, more specifically, a complete ontology of domain concepts ready for symbol design. In this instance, however, an ontology of potentially modellable cyber security requirements for modelling languages, which is also comprehensive to the domain, is something which to the best of this work’s knowledge does not yet exist.

Therefore, the principle of *semiotic clarity* is divided with respect to the underlying ontology. Stating that, before *semiotic clarity* can be truly assessed, one must first have an ontology which is *comprehensive* to the domain.

The reason for specifying this requirement, is due to the *construct deficit* current extensions suffer from. Although some extensions appear comprehensive to the domain, this is only relative to the *deficit* many of the others suffer from. In fairness to several extensions, they do state their focus on a specific area of security and as such their paucity may be excused. Nevertheless, needing multiple extensions to specify comprehensive requirements for the same domain is very poor usability, and will likely lead to the extensions being dismissed altogether. From the standpoint of an ‘ideal solution' the best approach is to include a comprehensive range of constructs which gives the modeller the ability to restrict the domain coverage as they see fit.

3.2.2 Coherence

In 2003, the phrase “far too much security terminology is vaguely defined”, was published in a short journal article (Donner, 2003). Although this statement is almost 15 years old, from the review of existing extensions, it is clear to see how relevant it still is today, there is still much discrepancy surrounding various concept meanings and uses. One concept which is an avid sufferer of this issue is *access control*. Many of the extensions reviewed, have very different approaches to this concept: one specifies access rights (Labda, et al., pp. 1399-1405, 24th - 28th March, 2014), another places parent and child concepts at similar abstractness levels (Sang & Zhou, 26th-28th October, 2015), and another disregards *access control* altogether and opts only for *authenticity* (Salnitri, et al., 2014).

The potential issues of misusing a construct due to symbol design were discussed earlier, the same rules apply when a modeller is unsure of semantics. Moody touches on this issue when discussing *construct overload*, stating that a user can become confused as to a symbol's true meaning when it can encompass multiple concepts (Moody, 2009). However, this does not state explicitly enough the requirement of what this thesis names *coherence*.

Expressing that a language developer should aim to avoid issues such as *construct deficit* is a reasonable requirement. Nevertheless, there should also be a principle which advises the explicit definition of a concept within a language. This ensures that any future modellers or readers of a diagram will always have a common archive of definitions should they require reminding. Current approaches are placing too much trust in the coherent definition of concepts from external sources such as the web, this requirement will ensure everyone is working in accordance with the same concept understanding.

3.2.3 Structure

The inclusion of a new domain in any existing language, will almost always cause an unparalleled increase in complexity (especially cyber security, given just how large the domain is). For novice users to the language, this can be overwhelming and potentially lead to only experienced individuals adopting the extension. As per the principle, *cognitive fit* suggests that modelling languages should prepare for this and include mechanisms to allow both the novice and expert equal opportunity to utilise the notation (Moody, 2009). Moody states that it is poor design to assume one notation can satisfy both the novice and expert. However, this is somewhat contradicted by the likes of traffic signs which all levels of expertise understand and utilise. Two notations can also cause an issue in which a reader may misinterpret one symbol from the expert notation with one from the novice notation.

Along with this issue, as discussed in the previous requirement, several authors placed parent and child concepts at similar levels of abstractness which can lead to *symbol redundancy*. A user needs to know without any doubt which symbol should be used to specify a specific construct.

Therefore, to overcome both these issues the principle of *structure* is proposed. This again links closely to Moody's principle of *hierarchy*. In the PoN however, Moody discusses using *hierarchy* as a form of complexity management to collapse diagrams into more manageable high level elements (Moody, 2009). In this instance, *structure* is targeting the notation symbols themselves. For example, rather than specify *access control* and *authentication* at the same level, one would specify *access control*, then if they so choose, they can specify the child symbol of *authentication* under it. This provides a level of cognitive difficulty for novices and experts. Novices only need to learn high level concepts to begin, then as they become more knowledgeable on the area, they can specify more specific sub concepts, ensuring the principle of *cognitive fit* and solving the issue of *symbol redundancy*.

3.2.4 Verification

Salnitri *et al.* (Salnitri, et al., 2014), were the only authors in the previous review to acknowledge the need for verification within a language. They also claim to have proposed their own framework for verifying (and modelling) security within BPMN. However, their paper is based heavily on their own extension of BPMN and provides little contribution in terms of any framework. Nevertheless, the verification aspect of their extension is something that should be considered a core requirement of a security extension.

Given the multitude of regulations surrounding security policies from both government and in many cases, in-house standards (Karjoth, 2nd-4th March, 2015), the ability to assess security requirements against such policies this early in the SDLC is a too advantageous opportunity to pass up (Arsac, et al., 2011). There is little benefit to blindly specifying security requirements at design if they later fail such regulations. Therefore, the requirement of a policy verification system is included within this set of heuristics.

3.2.5 Graphical Framework

From the review of existing extensions, only two can satisfy half of Moody's principles. Although the PoN are not entirely based on the graphical design of the notation, it is safe to generalise, the more principles satisfied the better the design of the symbols.

Although two extensions did satisfy half of the principles, neither of them appeared to have considered the longevity of their notation. Although both satisfy the principles of *perceptual discriminability* and *semantic transparency* as they are now*,* neither would be able to handle any further constructs being added to their notation (something they both require as neither satisfy *semiotic clarity*). The paucity of *visual variables* and *redundant* *coding* means that as more symbols are added the effectiveness at which a user can correctly distinguish between symbols will decrease quickly (Moody, 2009).

Therefore, a developer should first define a *graphical framework* for the symbols within their notation, one which uses the principles as defined in the PoN as a core foundation. This not only ensures the satisfaction of the principles, but also allows any future users to add to the notation as they see fit, without affecting any pre-existing design in place.

3.2.6 Complexity Management

Complexity management is potentially the single reason current security extensions have the issues they do. When viewing current extensions from a *complexity management* perspective, it is likely the authors of these extensions struggled to develop a visual solution in which they could be more comprehensive to the domain. Therefore, they opted for *construct deficit* as opposed to poor *complexity management*. Of course, without directly questioning each author it is difficult to confirm this was the case; not that anyone would admit to it. However, there is no disputing that all extensions either suffer from poor *complexity management* or are on the threshold of doing so. For this reason, this thesis views *complexity management* as the most important of the PoN.

Although principles such as *semantic transparency* can assist in the understanding of a language (especially as a novice), the satisfaction of *complexity management* can avert more severe impacts. BPMN and UML are good examples of languages which seem to have altogether ignored *semantic transparency* (Moody, 2009), yet they are both widely adopted. This is because their complexity and thereby ease of use is reasonably well managed. A notation could have *semantic immediacy* on all constructs but if its complexity is poorly managed, users will avoid or incorrectly make use of it (zur Muehlen & Recker, 2013). Therefore, when creating an extension (or modelling language) *complexity management* should be one of the key issues to address.

Complexity management already exists in the PoN, so why include it in this list? There is no claim to this principle within this thesis, it is included only to further iterate its importance in modelling language creation and to emphasise that it acts as one of the core motivations within this work.

The *complexity management* of a modelling language when using one domain across an empty canvas is a challenge itself. The *complexity management* of an extension to an already existing modelling language presents several new challenges. This principle represents the acknowledgement of these and the requirement to provide a suitable means of overcoming them.

3.2.7 Modelling Tool Functionality

When developing a security extension, most authors will typically expand on an already existing tool such as Microsoft Visio[14]. Most tools usually include functionality for the creation or inclusion of custom notation to assist with this process, Visio for example has functionality called Stencil. This allows users to create a custom toolbar of their favourite symbols for use later, it also allows for the saving of custom (imported) symbols, effectively a form of notation creator based on the Visio framework.

Current security extensions have opted to represent their notation in a similar manner to that of BPMN, stamped onto symbols or placed alongside elements with some form of graphical relationship. There has been little progress in terms of new modelling approaches, current extensions merely present a set of concepts and accompanying symbols, disregarding visual grammar[15]. The fact is that by adding more constructs, they have increased the languages complexity and potentially nullified any *complexity management* mechanisms the core language had in place.

The general approach to visualising security within BPMN follows the "*if it is not broke do not fix it*" rule with regards to existing modelling approaches. Where this may be the case for the extensibility functionality in existing tools, they typically only cover one domain (business process management for BPMN). If the authors simply wanted to extend a language with the same domain notation, utilising current software is the logical choice. However, extending a language with a new domain is more extensive than previous authors have given credit for, it is not simply a case of specifying a few concepts then creating symbols for them. There are a lot of other variables which must be considered (the objective of this section being to highlight these). Moreover, the tool which is used for creating and reading diagram instances requires just as much attention.

Although there may be an existing tool or framework which allows for the *complexity managed* specification of different domain elements across an existing language, this principle is defined to highlight the possibility that this may not be the case and a new tool featuring new functionality may be needed to satisfy the proposed heuristics and the PoN.

[14] Microsoft; Office Products: Visio; (https://products.office.com/en-gb/visio/flowchart-software) accessed 23/03/2017

[15] visual grammar – *a set of compositional rules for the notation symbols* (Moody, 2009)*; guidelines on how each symbol can be used to draw a diagram*

Complexity management in-particular, may require a new form of visualisation. Therefore, one should not be put off by the requirement of a new tool should this be the case.

3.3 Chapter Summary

In summary, this chapter defined the aims and objectives of the project that were identified during the literature review. There was a discussion on how achieving the specified objectives would consequently lead to the project aim of ensuring the practicability of expressive and usable secure by design tool implementation. The chapter then concluded the literature review, by defining a set of heuristics for avoiding the problem areas that were identified during the evaluation of existing extensions.

Continuing with the defined set of objectives, the specification of these heuristics now allows for the creation of the proposed framework in the next chapter.

4 Proposed Framework

The aim of this thesis is to provide the necessary tools for ensuring the implementation of expressive and usable security extensions. Therefore, the production of a security extension itself does little to satisfy this goal. However, a framework which can be used at the design and implementation stages of a security extension's SDLC, as a structured overview and set of guidelines to assist in its implementation, can satisfy the project aim. Therefore, the core novelty and contribution of this work lies within this framework.

Of course, the framework is not a "recipe" of how to create a very specific extension, but rather a set of blueprints or methodology which details the required components and order of development for ensuring a well-design solution. Take for instance the waterfall model mentioned earlier, this effectively acts as a high-level framework for developing a piece of software. The framework proposed within this thesis has a similar purpose, except for being focused on security extension opposed to general software development. Just as the waterfall model will claim to satisfy various aims through its use, the framework proposed within this work, claims to ensure the implementation of an expressive and usable security extension.

The following chapter discusses this framework, detailing the importance of each component to ensuring a complete solution is achieved. A published version of the framework can be found under the following reference (Maines, et al., 14th - 16th June, 2017). Likewise, a visual overview of the framework can be seen in Figure 4.1.

The framework consists of three core roles: *language architect, application developer* and *end user*. All of which, are necessary for the successful design, implementation and utilisation of a security extension. The roles are structured vertically in the framework and represent the strict chronological order of development from bottom-to-top (like that of a brick wall). For the most part, this rule is also true when reading the framework from left-to-right. Regardless of this however, each component is numbered to reiterate the order that should be followed when creating an extension.

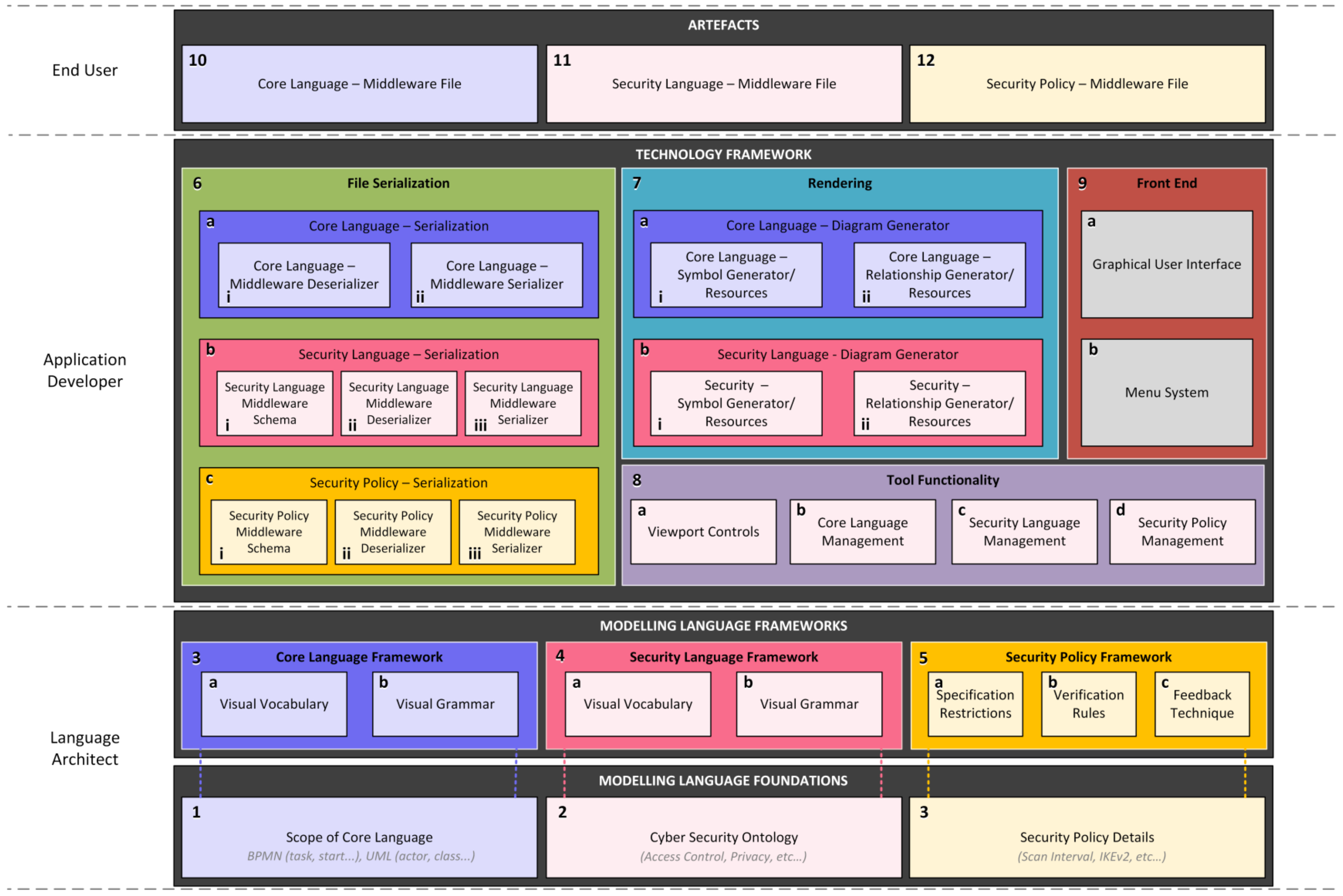

End User
Application Developer
Language Architect
ARTEFACTS
10 Core Language – Middleware File
11 Security Language – Middleware File
12 Security Policy – Middleware File
TECHNOLOGY FRAMEWORK
6 File Serialization
a Core Language – Serialization
i Core Language – Middleware Deserializer
ii Core Language – Middleware Serializer
b Security Language – Serialization
i Security Language Middleware Schema
ii Security Language Middleware Deserializer
iii Security Language Middleware Serializer
c Security Policy – Serialization
i Security Policy Middleware Schema
ii Security Policy Middleware Deserializer
iii Security Policy Middleware Serializer
7 Rendering
a Core Language – Diagram Generator
i Core Language – Symbol Generator/ Resources
ii Core Language – Relationship Generator/ Resources
b Security Language - Diagram Generator
i Security – Symbol Generator/ Resources
ii Security – Relationship Generator/ Resources
9 Front End
a Graphical User Interface
b Menu System
8 Tool Functionality
a Viewport Controls
b Core Language Management
c Security Language Management
d Security Policy Management
MODELLING LANGUAGE FRAMEWORKS
3 Core Language Framework
a Visual Vocabulary
b Visual Grammar
4 Security Language Framework
a Visual Vocabulary
b Visual Grammar
5 Security Policy Framework
a Specification Restrictions
b Verification Rules
c Feedback Technique
MODELLING LANGUAGE FOUNDATIONS
1 Scope of Core Language
BPMN (task, start...), UML (actor, class...)
2 Cyber Security Ontology
(Access Control, Privacy, etc...)
3 Security Policy Details
(Scan Interval, IKEv2, etc...)

Figure 4.1 A framework for the extension and visualisation of cyber security requirements in modelling languages

4.1 Language Architect

The *language architect,* represents the first and most important role in the development process, as implied, it is like that of a software architect. This role is responsible for the scope of each domain to include within the solution, along with the symbol design and visualisation approach. The importance of this role compared to the others, comes from the number of principles that can be satisfied at this stage of the framework.

4.1.1 Modelling Language Foundations

As the framework is based around the extension of an already existing language, one must take into consideration the restrictions of the parent language right from the beginning.

4.1.1.1 Scope of Core Language

At this stage of the framework, this is primarily to do with the scope of the core language. Naturally, incorporating the entire language is the most robust decision. However, when considering larger languages such as BPMN, which contains 171 constructs (Genon, et al., 2010), feasibility decisions must be made. Nevertheless, this component of the framework represents the acknowledgement of the core language scope, rather than the strict requirement of reducing it. Whether a language is reduced or not will be dependent on how the *language architect* envisages it will be used. For instance, if the framework is utilised for academic research, it is unlikely that the entire language will need to be included. Typically, academic research will not require comprehensive expression and may only require enough notation to test specific functionality (of course this may not always be the case). The intended use of the security extension will decide on the scope of the core language to include.

This component also requires the *language architect* to specify what elements of the core language will support security requirements. For example, in the case of BPMN, one may choose to only allow security specification on *task* elements and nothing else. Either way, a decision must be made to later inform the restrictions the *application developer* will implement into the solution.

4.1.1.2 Cyber Security Ontology

As stated by Moody (Moody, 2009), to satisfy the principle of *semiotic clarity* there must be a one-to-one mapping between modellable constructs and ontological concepts. Given that a new language is being created, to ensure adequate domain coverage, there is need of an ontology of cyber security requirements to assess against the final notation to determine whether this principle was satisfied.

Therefore, after defining the scope of the core language, the next step is to create an ontology of modellable cyber security concepts. The creation aspect naturally depends on the absence of an existing ontology. Following 'Step 2' of Noy and McGuinness' guide to ontology creation (Noy & McGuinness, March, 2001), there is no need to "reinvent the wheel" if such a solution already exists, it is easier and more efficient to adopt existing solutions and modify them where necessary.

Moreover, once an ontology of security concepts has materialised (one way or another), two of the core foundations of the final solution will have been completed. The *language architect* should now have two explicit lists of concepts each domain should include within their notation.

4.1.1.3 Security Policy Details

Although the framework is still usable without the inclusion of policy validation. Given the imposed regulations from governments and in-house standards, the ability to check a diagram's compliance with various policies at such an early stage in the SDLC must be utilised (Arsac, et al., 2011). Specifying security at design can have several benefits as discussed in previous sections. However, these can be negated if the specification later fails compliance checks due to inefficiencies. Although any form of security specification will always be advantageous compared to the current approach of post-development ad-hoc incorporation, if a tool can include functionality to test policy compliance before development, costs can be further reduced again.

Therefore, at this stage of the framework, a component to define the scope of the security policies is included. Unlike the previous section which explored modellable constructs, this requirement focuses more on specific instances of those constructs which may be stated within a security policy, such as *IPsec* as an *encryption* type.

4.1.2 Modelling Language Frameworks

As previously stated, utilising the functionality of existing software does not always provide the best results. They are very restricted in what they can do and typically only work for the same domain extension, hence the specified a requirement to consider the functionality of the extension end tool. Taking this into account, one must acknowledge the strong likelihood of current software being unable to effectively extend existing languages with security requirements.

If this is the case, only two possible options remain: build a new engine for the core language which also allows for the specification of security requirements or, create a separate tool solely for specifying security which extends from an existing core language framework. Either way, the *language architect* will require some form of core language renderer, whether this allows for creation of diagrams or just their drawing.

4.1.2.1 Core Language Framework

It is essential that the *language architect* defines the rules of the core language to later ensure the *application developer* implements them correctly into the tool. The PoN states that all visual notations are made up of three core components (Moody, 2009). Visual vocabulary, which can also be described as a set of graphical symbols. Visual grammar, which defines the compositional and relationship rules of the languages, and finally visual semantics or symbol definitions. As visual semantics are covered in the previous section, this component focuses on just visual vocabulary and grammar. That is, at this stage of the framework the *language architect* must specify in explicit detail what symbol represents each concept within the core language and what form of visualisation is required to ensure the composition and relationship rules are adhered to. For example, Table 2.1 details what symbols represent what concepts within BPMN, while Figure 2.1 showcases how these symbols are used to produce a diagram.

4.1.2.2 Security Language Framework

Contrastingly, the security language will require much more attention at this stage as no language yet exists, it is not simply a case of extracting information from an existing source. The end goal of the framework has little reliance on what methods of symbolic design and visual representation are chosen here, as all approaches will output solutions with identical functionality. However, the effectiveness of these solutions and their ability to satisfy the PoN, the previously defined requirements and be an appealing, usable extension are very heavily dependent on the decisions made here. Visual vocabulary and grammar represent two of the key areas almost every extension has thus far failed to successfully implement. They are also where a large majority of the proposed heuristics can be achieved.

Therefore, at this stage of the framework, the *language architect* must ensure that whichever vocabulary and grammar are used, both utilise a high number of visual variables and ensure the complexity of the extension is managed. Although no mechanisms are defined here, the objective of this framework is to provide a set of guidelines not a one solution tutorial. Therefore, although there is still a fair amount of work required by the *language architect*, this component most importantly, ensures the vocabulary and grammar receive their due attention, something current extensions have yet to do.

4.1.2.3 Security Policy Framework

The policy language framework is not as visual as the core and security frameworks, dealing largely with back-end checking as opposed to visual feedback. Therefore, the requirements are slightly different. For this component, the *language architect* must decide on specification restrictions or validation rules as well as the method of verification against a diagram. These will then be able to assist in determining what feedback technique can be used to inform the end user of any discrepancies between the diagram and policy.

Specification restrictions refer to minimising the human error element as much as possible. For example, certain requirements can sometimes negate others or even contradict them. Therefore, restricting the user's ability to define any of these disagreeable concepts will ensure any potential issues that could ensue in the future are nullified early on. Of course, this functionality could be included within the security language framework instead, stopping the issues before they even reach policy verification. However, given the ever changing and evolving methods of how security is used, placing these restrictions in the policy component provides a more robust extension. A user can always choose to ignore compliance warnings placed in the visual grammar rules, however, a modeller is restricted and unable to freely express their requirements.

Policy compliance represents a very important and complex section of cyber security. For this work, this is acknowledged at a high level as it could quickly take over the entire project. The framework is fairly brief with what is expected from language developers, requesting only that they specify a set of rules that can be used to verify a policy against a diagram, whether this be a simple Boolean check or something more elaborate. The option to expand and further refine this, of course is always there, and can be done at the language developer's own discretion (or in any future works).

The final component within this section is feedback technique. There are various approaches to providing feedback, many of which can be observed in other software. The most common scenario being that of coding languages, these utilise a set of predefined errors which are triggered when a certain condition has been met. Most of the time, they will also provide details as to which class and line of code the error occurred on.

There was a discussion earlier regarding the issues of relying solely on text, in this instance however, text is a necessity, as compliance feedback is too detailed for a visual-only solution. Therefore, it recommend to use some form of visual aid to alert the end user of the discrepancy location on the diagram, but to rely on text for informing the user of what this discrepancy is.

4.2 Application Developer

The previous section was based around the visual and conceptual construction of the languages, this portion of the framework focuses on the technical requirements (or technology framework) for implementing the application. As previously mentioned, current approaches to security extensions have all opted to extend existing tools with their new domain. However, so far, the results from these attempts have provided very poor solutions and the creation of a new tool or add-on appears the best approach to take.

Much like the entire framework follows a bottom-up implementation approach, the technology framework follows a left-to-right one. The middleware files act as the core foundation to the tool, with the rendering engine and tool functionality handling the visualisation of the diagrams and the necessary libraries for creating them. The front end then provides the user interface components, these allow for the interaction and manipulation of the tool (and thereby diagrams) based on user input.

4.2.1 File Serialization

One of the key technologies required for the development of a security extension (or software in general) is some form of middleware file for saving and loading data, and consequently, the functionality to serialize and de-serialize these files from within the tool.

4.2.1.1 Core Language – Serialization

As discussed earlier, whether a tool is created as an add-on or a standalone application, the core language will require a rendering engine. Therefore, it will also require some form of data file to hold the diagram information. Given that the core language has already been created, there will already be existing schemas for these files which can be sourced from other software. The tool simply requires the functionality to serialize and de-serialize these (potentially just de-serialize depending on whether the end tool allows for core language specification).

4.2.1.2 *Security Language – Serialization*

Once again, the security language is not as straightforward as the core language as no solution yet exists. Therefore, there is not only a requirement for serialization and de-serialization here, but also the creation of a schema for the data file. Depending on the quality of the cyber security ontology created by the *language architect* however, this can be a fairly straightforward process. The *application developer* may need only to translate the ontology into a middleware-framework, and specify the other necessary information such as data types and translation/rotational information alongside each node.

4.2.1.3 *Security Policy – Serialization*

The security policy data file is like that of the security language, also requiring a schema. Given that this policy will be assessed against the security requirements (language) though, these two schemas being similar, can help streamline the comparison process. Naturally, the policy file will not render any elements across a diagram and as such will not require positional data within its file. Nevertheless, the closer the two are, the easier the process of implementing verification rules will be.

4.2.2 Rendering

This component feeds directly from the visual vocabulary of the modelling language frameworks. The middleware files contain the necessary data to draw the languages, but without some form of rendering tool this data is useless. The *application developer* must implement (or utilise) some form of renderer capable of visualising the diagrammatic symbols and their component relationships as specified by the *language architect*. Nowadays, technology is readily capable of rendering 3D geometry (Shiode, 2000). However, the code level which deals directly with how something is rendered, requires "teaching". That is, some tools such as game engines may include functionality for applying materials and shaders to a 3D object. However, what object, material and shader are used will need to be defined in the first instance (Gregory, 2014).

4.2.2.1 *Core Language – Diagram Generator*

If the tool is a standalone application (or loads a middleware file from an external engine), it will require the ability to generate the symbols and relationships for the core language.

4.2.2.2 *Security Language – Diagram Generator*

Likewise, the security language will also require the same functionality. Although this is something an end user takes for granted, when creating a new tool, a rendering engine represents a significant component for doing so.

Both components will require the necessary resources to enable the rendering of their language. This may be through pre-created assets which are instantiated as needed, or more general assets which can be combined at run time. The latter being like existing approaches in tools like Visio, place a shape then add or edit a text box afterwards. Either way, these components represent the need for the *application developer* to implement into the solution, how each element of the languages should be visualised on screen.

4.2.3 Tool Functionality

The tool functionality component represents the core of the development process. This is effectively where the engine for the application is implemented. The serializer loads the necessary data into the tool. The tool functionality components then use the renderer to draw each element, and from the middleware file information, translate and rotate the respective elements to their correct positions. This component is also responsible for handling the functionality of policy verification.

4.2.3.1 Viewport Controls

Before implementing any of this functionality however, the framework first recommends to create the necessary tools for controlling the viewport. Depending on the size of the diagram, both as a developer debugging the application, or as an end user, without the ability to traverse a diagram, it is difficult to confirm elements have been placed correctly. As such, the *application developer* should first ensure the viewport has the necessary controls for panning throughout the environment.

4.2.3.2 Core Language Management

Similar to the renderer, the management components feed directly from the visual grammar elements of the previous section, representing the core libraries that are required for drawing a diagram as per the *language architect's* restrictions.

There are various approaches that can be taken when using the framework: a new core language engine for creating diagrams; a read-only engine which merely renders core language diagrams without the ability to edit them; or finally, an application which plugs into an existing solution, providing only the functionality for specifying security requirements. Irrespective of which route is taken, this component is a necessary one to complete.

As a bare minimum, the *application developer* must ensure the core language is loaded and rendered correctly within the extension tool. They must then implement the necessary functionality to interact with the core language elements. Otherwise, there will be no way for the end user to select which BPMN elements they wish to apply their security to. In summary, this component effectively acts as the motherboard, which connects the aforementioned components into a working solution for the visualising and interaction of the core language.

4.2.3.3 Security Language Management

Depending on the designs specified, the *language architect* will decide on the workload required for this component. Security language management represents the implementation of the security language and thereby the core functionality of the extension itself.

The rendering component as mentioned, is a vital constituent to this. However, this element deals directly with the implementation of the necessary functionality for specifying security, as well as the placement of these requirements onto a diagram. The key objective here being to ensure that an end user is strictly limited to the visual grammar approach specified by the *language architect* in the previous section. For example, if the *architect* specifies that core/security relationships are to use the symbol stamping approach, the language management component will be the stage at which the *application developer* implements the functionality to allow this and the necessary restrictions for disabling any other pairing method.

4.2.3.4 Security Policy Management

The implementation of policy verification and the feedback it generates, although different to the previous two components, still feeds from the security policy framework of the previous section. At this stage, the *application developer* is required to implement the necessary functionality for carrying out the tasks specified by the *language architect*. These being: policy specification, policy verification and policy feedback, each of which represents a significant portion of the extension. The creation of a security policy will typically not require any innovative functionality, a simple form input will suffice as this has little to do with the language complexity and will be done in a separate scene to the one in which the diagram exists. As for the verification itself, and the feedback it generates, these two will be largely dependent on the requirements of the *architect*. The objective of this component is to ensure that all these requirements are met and implemented into the application.

4.2.4 Front End

Finally, the last component of the technology framework is the front end of the application, otherwise referred to as the Graphical User Interface (GUI). In the framework, the GUI and menus are split into separate elements, even though menus are classed as GUI and can be categorised under them. The reason for this, is that the extension will require a lot of GUI strictly for creating diagrams though, the two are separated to effectively try and reduce the *construct overload* of framework components. Placing them at the same level of abstractness within the framework will have no negative implications as they are effectively only a checklist in this instance. However, a positive outcome is that it is easier to cognitively manage the requirement of the front end component.

The main requirement of these components, is to ensure that a usable interface is created which allows the end user to access and utilise the functionality from the previous section. This being: navigation of the viewport, the specification and/or selecting of core language elements, the specification of security on these elements and finally the creation and verification of policies.

Although a lot of the components within this section appear brief in comparison to what they require, it is a framework that is proposed and not a tutorial. Each developer will create their own extension, with the framework acting only as guidelines to ensure the end product acknowledges, and hopefully satisfies the aforementioned requirements.

4.3 End User

The end user represents the third and final role for the design, implementation and utilisation of a security extension. As expected, the end user has little responsibility in terms of the design and implementation of the extension. However, they do represent the end goal, and although they are not compulsory to produce an extension when creating diagrams and using the extension they will be producing the saved data-middleware files discussed earlier. These are compulsory for the application to even load a diagram. Therefore, the acknowledgement that without an end user there is no extension, is a necessary point to make within the framework.

4.4 Chapter Summary

Throughout this chapter, the main novelty of the work was defined. That is, a framework which ensures the implementation of a usable and expressive cyber security extension. The framework was initially proposed in a high-level visual form, followed by a detailed discussion on the importance of each component to ensuring a complete solution.

The following chapters represent the case study of the framework mentioned in the aims and objectives. Using BPMN as the core language, the framework is followed, detailing how each component is achieved. This not only acts as a form of justification on its own, but outputs a solution that can be evaluated to further strengthen the benefits of the framework.

5 BPMN Case Study – Language Architect

In the literature review, the necessity of being able to specify security requirements at design was explored, more specifically within a modelling language. As BPMN is the industry standard for business process modelling, efforts were focused here and found that several attempts had already been made regarding extending the language with security expression. However, of these attempts, it was discovered that none could provide adequate solutions. As such, a new framework was proposed for the extension of *any* modelling language with security requirements, with the ambition that any product of it would provide an adequate solution for security specification.

In the following chapters, the thesis returns to BPMN, which acts as the core language for testing the proposed framework. The chapters are structured the same way as the framework is defined, discussing the approach to achieving each component as well as detailing how each one was implemented into the solution. Providing not only justification for the framework itself, but a usable case study which can be referred to when creating an extension for another language.

5.1 Modelling Language Foundations

5.1.1 Scope of Core Language

BPMN contains 171 constructs (Genon, et al., 2010), if the objective was to commercialise the extension, without a doubt supporting all these elements would be the primary objective. However, in this instance there is merely a need for enough notation to test the framework. As such, the following BPMN elements are included: *pool, lane, start event, end event, message start event, message catch event, timer start event, error end event, terminate end event, parallel gateway, exclusive gateway, inclusive gateway, user task, business rule task, script task, receive task*, *service task* and of course the *sequence flows/arrows* which represent relationships between each element. Although missing several from the notation, these elements are sufficient for the creation of most general BPMN diagrams. The omission of concepts, is a common practice most users employ anyway when using BPMN. In most cases, engineers will only utilise a finite amount rather than the entire language (zur Muehlen & Recker, 2013).

With that said, another element is proposed at this stage, strictly for the use of security specification that this thesis chose to name *all elements*. The purpose of this element, is to hopefully reduce the *complexity management* of diagrams. In a certain scenario, should a modeller wish to add the same security requirement to every BPMN element (for example: *encryption*), rather than needlessly contribute to the construct count, they can simply specify it once on *all elements*. This of course only works on the understanding that later in the SDLC, whoever implements the system is aware any security on this element should be applied across the entire business process. Nevertheless, it is something not yet seen which can contribute to the *complexity management* of a language with very little effort required to implement.

As for which BPMN specific elements will support security specification, in this instance, all of the previously stated elements dismissing only the *sequence flows* will be capable of relating to security. Although this may appear a poor choice from a 'minimising human error' perspective as certain elements such as *start* and *end events* would likely never require security specification. It is a wasted effort to spend time on these negligible decisions in this project. The requirement is included in the framework, as commercial applications will greatly benefit from this added robustness. In this case, however, there is only a need to test the framework, with little gained from worrying about such trivial functionality. The purpose of this requirement is to minimise errors at a much later date in the SDLC, a stage this work has no intention of reaching.

5.1.2 Cyber Security Ontology

Unlike the previous component, the creation of a cyber security ontology is a much more extensive process. To assist in this task, Noy and McGuiness' ontology creation guidelines (Noy & McGuinness, March, 2001) will be followed. In Dahlem and Hahn's survey of ontology creation methodologies (Dahlem & Hahn, 6th-9th August, 2009), Noy and McGuiness satisfied more of their criteria than any other approach. Their guidelines have also been used multiple times in literature for a variety of different domains including business processes (Nitzsche, et al., 7th June, 2007), security (Karyda, et al., 20th-22nd April 2006) and games (Tang & Hanneghan, 6th-8th December, 2011). At the time of writing this thesis, the publication also has over 5,400 citations according to Google Scholar, a form of justification on its own.

In their paper, they define seven steps for the successful construction of an ontology. Throughout this section, each step is followed and discussed in detail with regards to the creation a new cyber security ontology:

1. Determine the domain and scope of the ontology
2. Consider reusing existing ontologies
3. Enumerate important terms in the ontology
4. Define the classes and the class hierarchy
5. Define the properties of the classes-slots
6. Define the facets of the slots
7. Create instances

5.1.2.1 Step 1: Determine the Domain and Scope of the Ontology

As already stated, the domain for this ontology is cyber security. Specifically, any requirements (or objectives) that are potentially modellable within BPMN. The distinction between this and the cyber security domain in general must be understood. The framework and thereby case study, make no claims to include security aspects such as threats. Although one must consider the attacks they hope to nullify, the specification of threats is outside the scope of this work. The aim instead, is security requirements or objectives which can potentially impact the development process and as such would benefit from early specification.

5.1.2.2 Step 2: Consider Reusing Existing Ontologies

From a review of existing literature, there was no ontology which was able to satisfy these needs. As stated by Elçi, current attempts have been extremely scenario focused (Elçi, 2014). Although there are existing publications which review multiple security ontologies (Blanco, et al., 4th - 7th March, 2008; Souag, et al., 25th-26th June, 2012; Singh & Pandey, 8th-10th October, 2014), there are none that combines them to form a solution which satisfies the framework's needs. Therefore, although it is not possible to reuse an existing ontology, of those that do exist, their constituent concepts can be combined with the concepts of existing BPMN security extensions. This will then produce a good foundation for Step 3 of Noy and McGuiness' guide.

5.1.2.3 Step 3: Enumerate Important Terms in the Ontology

The survey of existing solutions begins with security ontologies and taxonomies. In the same article mentioned earlier, Donner (Donner, 2003) expressed the need for an ontology in the cyber security domain. More importantly, several concepts were proposed that Donner believes should exist in such an ontology. Of the concepts specified, the following represent those that fit the previously specified criteria of being modellable within BPMN and impactful to implementation: *privacy, availability, integrity, detection, encryption, authorisation, authentication* and *roles*. (*Policy* was also specified, not something that would be classed as a modellable construct, but emphasising its inclusion within the framework nonetheless).

Denker *et al.* (Denker, et al., 2005) discuss the various trust and security issues between Web-based interactions in their paper. Although not directly linked to this work, they discuss several security requirements that can be extracted for the list of terms: *authentication, authorisation, access control, confidentiality, data integrity, privacy, encryption, digital signature, certification, public key infrastructures, anonymity* and *credentials*. They also include specific instances of *credential* requirements: *cookies, login username/passphrase, biometric;* as well as key types: *public key* and *symmetric key*. Although these concepts would not typically be modelled given their low level, they can be utilised for Step 7 of the guide, i.e. creating instances of slots (Noy & McGuinness, March, 2001).

Karyda *et al.* (Karyda, et al., 20th-22nd April 2006) share similar motivations with this work, also trying to address the paucity of security specification in application development. Their proposed solution however, opts for an ontological only approach, as opposed to modelling language extension like this thesis. Within their paper, they also follow Noy and McGuiness' guide, however at no point do they explicitly detail the entirety of their ontology, visual or otherwise. As such, it is difficult to ensure all their security requirements have been extracted for this review. Nevertheless, of the requirements that can be identified, these are as follows: *confidentiality, availability, integrity, accountability, non-repudiation, cryptography, encryption, access control, certificates, intrusion detection/malicious SW detection, firewall, anonymity, credentials* and *auditing*.

Kang *et al.* (Kang & Liang, 10th-12th October 2013) propose the 'NRL Security Ontology' within their paper, claiming it to be more comprehensive and organised than existing solutions. The purpose of their ontology, is to support the annotation of resources (primarily their functional aspects) with security requirements at varying levels of detail. According to the authors, the NRL ontology represents the combination of seven smaller ontologies in a more uniform and usable manner. The area of the ontology this work is interested in (as their concepts meet the previous requirements), is the *security objective* portion of the *main ontology*. Here, the concepts specified are: *confidentiality, availability, authentication, authorisation, integrity, key management, trust* and *anonymity*. Some of the other sub-ontologies also specify a few concepts that are modellable within BPMN, such as: *encryption* and *credentials*.

The publication that matches the previously stated requirements closest is that of Firesmith (Firesmith, 2004). Although the original list of requirements was defined in a different publication (Firesmith, 8th-12th September, 2003) *(same author),* the former (Firesmith, 2004) presents them in a more organised manner. Nonetheless, the main objective of the paper this work focuses on (Firesmith, 2004), is to specify reusable security requirements. The concepts proposed are: *access control (*with the child concepts of *identification, authentication* and *authorisation), attack/harm detection, integrity* (with the child concepts of *data integrity, hardware integrity* and *software integrity, immunity* being a child of *software integrity), non-repudiation, privacy* (with the child concepts of *anonymity* and *confidentiality*), *security auditing* and *physical protection*. In terms of what one would envisage to be the "end goal" ontology of this work, this list is by far the closest. The specified concepts are at the desired level of abstraction and their structure is the most logical seen thus far. However, this list still is not as comprehensive to the domain as it could be.

As mentioned, there are multiple scenario-specific security ontologies that exist in literature. Although the ontology created within this thesis represents a significant contribution to the work, the aim is not to create a 'survey-paper' thesis of existing solutions. Therefore, the focus of the thesis will now revert to BPMN security extensions and explore the concepts used within them.

Rodríguez *et al.'* extension (Rodríguez, et al., April 2007) include: *non-repudiation, attack/harm detection, integrity, privacy, access control, security role* and *security permissions*.

Saleem *et al.* (Saleem, et al., January 2012), the most construct deficit of existing extensions, specify: *confidentiality, integrity* and *availability*.

Salnitri *et al.* (Salnitri, et al., 2014) based their extension on the security goals specified in the RMIAS (Cherdantseva & Hilton, pp. 546-555, 2nd-6th September, 2013). These being: *integrity, authenticity, accountability, non-repudiation, auditability, confidentiality, privacy, binding of duties, separation of duties, availability* and *non-delegation*.

Labda *et al.* (Labda, et al., pp. 1399-1405, 24th - 28th March, 2014) with their rather unorthodox approach specified: *access control (allow), access control (prevent), access control (limited), separation of tasks, binding of tasks, user consent, necessity to know (high), necessity to know (medium)* and *necessity to know (low)*. For the ontology in this thesis, the *access control* and *necessity to know* concepts will be counted as one each, rather than the three instances specified by the authors.

Of Cherdantseva's RMIAS, it is the *security goals* that align with the proposed ontology requirements, these being: *confidentiality, integrity, availability, authenticity & trustworthiness, non-repudiation, accountability, auditability,* and *privacy.*

The last of the previously reviewed extensions is that of Koh and Zhou (Sang & Zhou, 26th-28th October, 2015). The concepts they specify being: *security task, authentication, access control, authorisation, harm protection, encrypted message, non-repudiation and secure communication, confidentiality, availability* and *integrity*.

Wolter *et al.* (Wolter, et al., 12th-14th March, 2008) presented a security policy and policy constraints model. Although not targeted directly at BPMN; they discuss how security should be specified at the modelling level so it can later be implemented into the system. Unfortunately, their paper did not include the explicit notation of the respective extension, making it impossible to evaluate against the PoN. Nevertheless, they do specify several concepts: *confidentiality, integrity, authentication, authorisation, traceability and auditing*, and *availability*.

Brucker *et al.* (Brucker, et al., 20th - 22nd June, 2012) present an extension called 'SecureBPMN', unrelated to Cherdantseva's version from what could be seen. They discuss how security requirements should be modelled at design-time instead of implementation, proposing a tool which can both model the security requirements for business process-driven systems at design time and enforce them at runtime. They express how functional behaviour and security are intrinsically interlinked and as such should be defined and executed together throughout the implementation stage. Again, only some notation is shown within their publication making it impossible to fairly evaluate it against the PoN.

The concepts covered in SecureBPMN are: *access control, separation of duty, binding of duty* and *need to know*. Brucker *et al.* state that their extension focuses on access control (like 'SecureUML' (Lodderstedt, et al., 2002)) but also provides support for the other mentioned concepts.

Mülle *et al.* (Mülle, et al., 2011) created a security extension to support the business process lifecycle from modelling to runtime. They discuss how enforcing security has high implementation and maintenance costs, stressing the need for such requirements to be specified as early as possible (during the modelling stages) and for the process to continue through to runtime. This extension also fails to include adequate notation for the evaluation against the PoN, hence its omission from the previous review. Nevertheless, the concepts included are: *authorisation, authentication, auditing, confidentiality* and *integrity*. Some of these are then subdivided to include the interaction methods of *role assignment, assignment mechanism, user assignment, separation of duty, binding of duty, delegation, adaptation, set interaction preferences, give consent, service selection, set data access policy, select data access policy, set trust policy, select trust policy* and *give trust feedback*. For this purpose, the previous concepts will be summarised into the following: *assignment mechanism, delegation, separation of duty, binding of duty, user consent* and *trust policy*. These concepts are more representative of the list as cyber security requirements and are therefore more useful to the thesis in this sense.

Throughout this review, concepts from five ontologies and nine security extensions have been extracted. Though there are other security extensions within literature (Peng, 6th-10th May, 2009; Paja, et al., 6th October, 2011; Ahmed & Matulevičius, 2014), as with existing ontologies, concepts quickly repeat themselves. Therefore, there is no need to overwhelm the thesis with excessive analysis of existing solutions.

Moreover, these reviews provide strong examples of the concepts experts want to include within a cyber security ontology. Many extensions have no intention of being fully comprehensive and thus omit many concepts they would otherwise have included. The areas they do focus on provide emphasis on their importance within the domain. To better visualise these concepts as a complete list, one can refer to Table 5.1, which specifies all the previously mentioned concepts and what authors included them within their extension/ontology.

		Ontology					Security Extension								
Total Ref.	**Security Requirement**	Donner	Denker *et al.*	Karyda *et al.*	Kang *et al.*	Firesmith	Rodríguez *et al.*	Saleem *et al.*	Salnitri *et al.*	Labda *et al.*	Cherdantseva	Koh and Zhou	Wolter *et al.*	Brucker *et al.*	Mülle *et al.*
6	privacy	✓	✓			✓	✓		✓		✓				
7	availability	✓			✓			✓	✓		✓	✓	✓		
11	integrity	✓		✓	✓	✓	✓	✓	✓		✓	✓	✓		✓
5	harm detection	✓		✓		✓	✓					✓			
5	encryption	✓	✓	✓	✓							✓			
7	authorisation	✓	✓		✓	✓						✓	✓		✓
9	authentication	✓	✓		✓	✓			✓		✓	✓	✓		✓
3	role/assignment	✓					✓								✓
7	access control		✓	✓		✓	✓			✓		✓		✓	
10	confidentiality		✓	✓	✓	✓		✓	✓		✓	✓	✓		✓
2	data integrity		✓			✓									
1	digital signature		✓												
2	certification		✓	✓											
2	public key infrastructure		✓		✓										
4	anonymity		✓	✓	✓	✓									
3	credentials		✓	✓	✓										
3	accountability			✓					✓		✓				
6	non-repudiation			✓		✓	✓		✓		✓	✓			
1	cryptography			✓											
1	firewall			✓											
6	auditing			✓		✓			✓		✓		✓		✓
3	trust				✓						✓				✓
1	identification					✓									
1	hardware integrity					✓									
1	software integrity					✓									
1	immunity					✓									
1	physical protection					✓									
1	permissions						✓								
4	binding of duties								✓	✓				✓	✓
4	separation of duties								✓	✓				✓	✓
2	non-delegation								✓						✓
2	user consent									✓					✓
2	necessity to know									✓				✓	

Table 5.1 Security Concept Reference Count

Although the number of concepts specified within the table is quite extensive, especially in comparison to any single review thus far, there are still more security requirements which are potentially modellable in BPMN which have not yet been specified. Therefore, along with these existing terms, a new list of concepts which aims be more comprehensive to the cyber security domain was created, these are as follows *(alphabetically)*:

- *access control*
- *accountability*
- *anomaly detection*
- *anonymity*
- *asset classification (data and service)*
- *asset maintenance*
- *asset management*
- *asset register*
- *attack/harm detection & prevention*
- *audit trail*
- *authentication (personnel and network)*
- *authorisation*
- *availability*
- *Bell-LaPadula model*
- *Biba model*
- *binding of duty*
- *biometrics*
- *confidentiality*
- *constraints*
- *credentials*
- *cryptographic protocol*
- *data backup (online and local)*
- *data retention*
- *data usage consent*
- *delegation*
- *digital signature*
- *encryption*
- *firewall (network and application layer)*
- *hardware backup*
- *hash function*
- *honeypot (low and high-interaction)*
- *identification*
- *immunity*
- *input validation*
- *integrity (data, hardware, personnel and software)*
- *intrusion detection & prevention system*
- *location*
- *need to know*
- *non-repudiation*

- *patch management software*
- *personnel*
- *personnel backup*
- *physical security*
- *privacy*
- *pseudonymity*
- *public key infrastructure*
- *role assignment*
- *sandbox*
- *separation of duty*
- *service backup*
- *signature-based detection*
- *smart card*
- *state transition system*
- *stateful protocol analysis detection*
- *trust level*
- *user consent*
- *virtual private network*
- *vulnerability assessment (service, personnel, environment and system)*

Of course, the cyber security domain is very extensive and it is difficult to claim any ontology or list of concepts as being a complete representation. However, when it comes to security requirements/objectives that should be specified at design stages, this list can be considered comprehensive to the domain.

5.1.2.4 Step 4: Define the Classes and Class Hierarchy

Reviewing the concepts specified in Table 5.1 along with the ones that were added, a total of six distinguishable classes can be deduced to split them into. These being: *access control, attack/harm detection and prevention, integrity, accountability, privacy* and *availability*.

These classes were identified by using the structure created by Firesmith (Firesmith, 2004) as a foundation. Firesmith lists seven key concepts at the highest level, these being: *access control, attack/harm detection, integrity, non-repudiation, privacy, security auditing* and *physical protection*. For the proposed ontology, *security auditing* and *physical protection* are not very applicable at such a high level. Although *physical protection* is a very wide domain itself, as this ontology is focusing on *cyber* security requirements, it was placed as a subclass of *hardware integrity* instead.

Non-repudiation is also an important concept within cyber security, but it too seemed more appropriate to place as a subclass, rather than as one of the main six. *Non-repudiation* ensures *accountability* effectively acting as a subcomponent to it, whereas it is difficult to find a scenario where *accountability* acts as a subcomponent to *non-repudiation*. Therefore, it seemed more appropriate to place *accountability* as the higher class and *non-repudiation* as the subclass.

Availability was established as a key concept simply due to its high use and lack of subcomponent classification within the domain. Seven of the fourteen reviewed papers included *availability*, plus it is also a core concept of cyber security (Pfleeger & Pfleeger, 4th Edition, 2006). As such, the placing of it within the hierarchy was clear.

Confidentiality is also a core component of cyber security (Pfleeger & Pfleeger, 4th Edition, 2006), however, there is justification for it not being one of the main six. Firstly, Firesmith (Firesmith, 2004) structured *confidentiality* and *anonymity* as subclasses of *privacy*. Researching around the terms more, it is more appropriate to place *privacy* as the higher class. *Confidentiality* tends to refer more to data and assets (Pfleeger & Pfleeger, 4th Edition, 2006), omitting the possibility of anything user related being a subclass. Whereas placing *privacy* as the higher class allows for user-related concepts to be included while still maintaining accurate relationships: *confidentiality* as a type of or an extension of *privacy*. *User consent* can ensure/act as a component to *privacy*, acting as a more meaningful hierarchy within the ontology.

Although possible, defining the entire ontology's hierarchies in this manner is not only an arduous task, but it is very difficult to visualise. Therefore, if one refers to Figure 5.1, the aforementioned list of concepts is presented in a structured diagram.

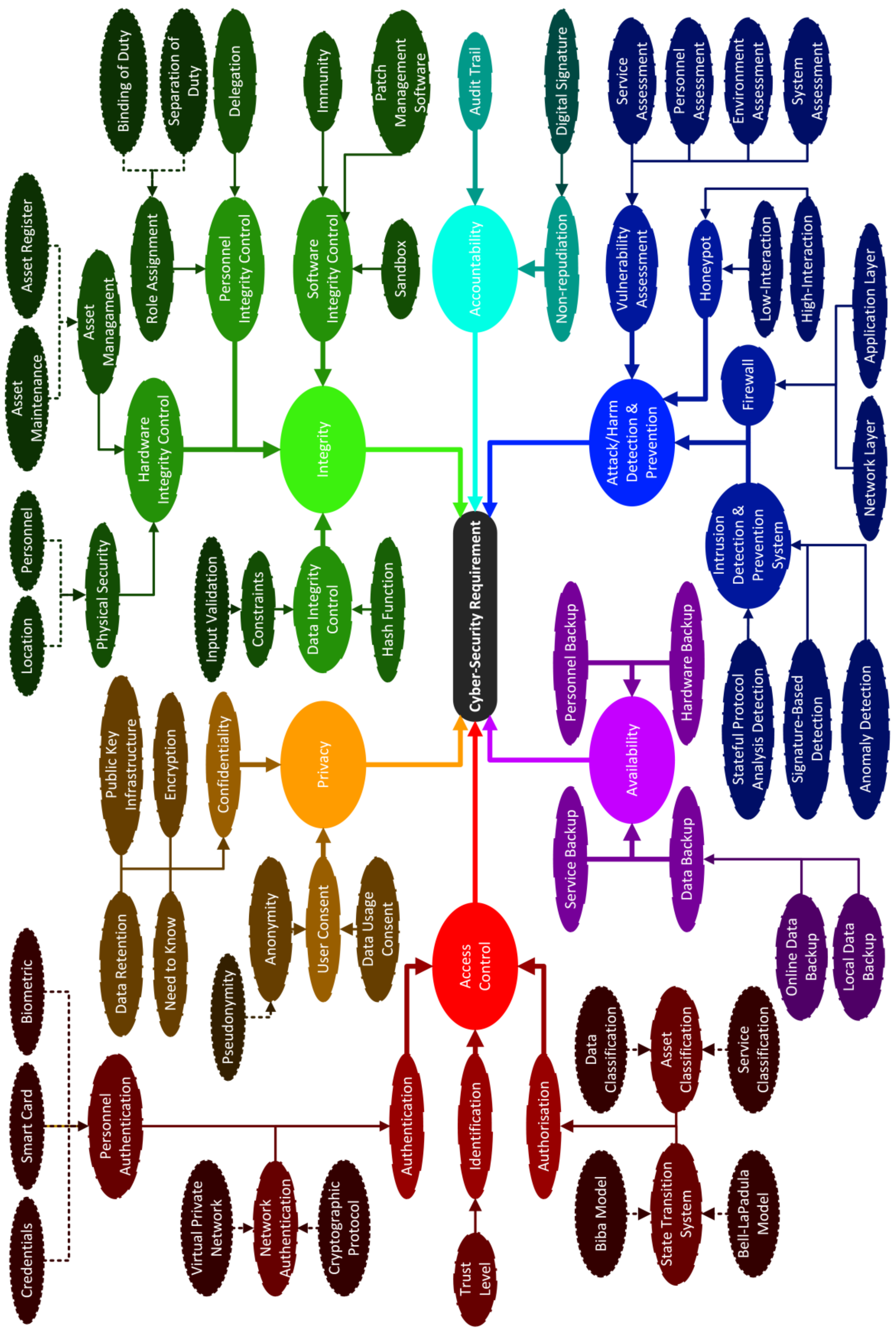


Figure 5.1 Cyber Security Requirements Ontology

A published version of the ontology can be found under the following reference (Maines, et al., 26th-28th October, 2015). *Separation of tasks, binding of tasks* and *digital forensics* have since been removed from ontology. This was due to confusion between tasks and duties and the fact that digital forensics is a post attack response rather than a security requirement or objective.

The concept structure used in this diagram is based on the understanding of what most accurately defines them as per this work. For example, *encryption* is placed as a subclass of *confidentiality* and thus *privacy*. *Encryption* is a widely-used security mechanism and could easily fit under several other classes such as *cryptographic protocol* or *data integrity control*. However, the purpose of this ontology is to act as a foundation for security extensions which can later be used for conducting ontological analysis. (As previously mentioned, ontological analysis is used to ensure a one-to-one mapping exists between the extension and ontology (Moody, 2009).) Whether *encryption* be placed as a subclass of *confidentiality* or *cryptographic protocol* makes no difference to its meaning or use within cyber security. The ontology is merely to act as a checklist that all concepts have been included. Placing the same concept at multiple points throughout the ontology prolongs this process as it is difficult to ensure each variation of the concept has been checked. One reference to each concept is all that is necessary, any more will over complicate the diagram with little benefit. If another user of the framework wishes to utilise or alter the ontology, or create their own, that is their own prerogative to do so, hence the high level of the framework.

Based on the previous analysis, this ontology is comprehensive enough to adequately satisfy any security extension irrespective of the target-user expertise or specific domain area. If another concept is needed, it can easily be included within the ontology without any major redesign needed.

As it is now, this portion of the ontology represents all the requirements to visually represent in BPMN. Although the remaining steps of the ontology creation guidelines have not yet been completed, this is the lowest level each concept can go before becoming a specific instance. As such, this represents the ideal place to stop in terms of modellable constructs.

Modelling language notation by design, provides high level concepts that can be used as a toolkit for designing specific instances. This is the stage the ontology is now at. If any lower level concepts were to be included, the final notation's graphic complexity will increase exponentially.

5.1.2.5 *Step 5, 6 & 7: Define the properties, facets and instances of the classes-slots*

For the final three steps, it is easier to collate them into one for this thesis. Although the steps are followed logically when creating the properties, it is a vast waste of time and space to repeat the same information with very little difference between them three times. Therefore, in one action, the properties, data types and given instances of the respective security requirements will be defined.

The definitions of each concept will also be specified here. As previously stated, there is a lot of discrepancy in regard to concept definitions, which has consequently caused the issue of incoherence amongst existing extensions. By no means does this work wish to challenge the security community at large by declaring the following definitions as the correct descriptions. However, there is need to emphasise that the explicit specification of definitions within a language will ensure all users of it are working under a common understanding.

Given that there are multiple concepts within the ontology, many of which have multiple properties, in this instance, only *access control* will be defined within the main body of the thesis. The remaining concepts and their constituent properties can be found in the Appendix.

Access Control	Concept Definition: Access Control refers to the act of authorising what parties are allowed to consume, enter or use a service.	
N/A	*N/A*	*N/A*

Again, there is likely a paucity of properties within the ontology which many would deem compulsory to any extension. However, the objective here is to test the proposed framework. The only real requirement of the ontology is to ensure there is enough notation to push the extension into a high enough complexity level to ensure the framework can still satisfy the aforementioned requirements. Nevertheless, that is not to say properties have intentionally been omitted, the objective was to include as many as possible within the given time constraints. However, there is always the possibility of potential absences.

In summary, a much larger number of concepts than any existing extension thus far has been proposed. Of modellable constructs, a total of 76 concepts have been defined, significantly more than the next highest at around 11 (Salnitri, et al., 2014; Mülle, et al., 2011). Therefore, at this stage of the framework, one can conclude that the extension is currently the most comprehensive solution to the cyber security domain, as well as the most graphically complex. This puts the final solution in an ideal position for assessing the *complexity management* later. Although a visual solution is yet to be specified, current attempts struggle to achieve *perceptual discriminability* with just 8/11 concepts (some even with just three (Saleem, et al., January 2012)). Therefore, if the extension created within this work is *perceptually discriminable* and capable of managing its complexity, even with an exceptionally larger number of concepts, the thesis is put in a strong position to declare the framework as a solution to the extension and visualisation of cyber security requirements in modelling languages.

5.1.3 Security Policy Details

Within this work, there is no aim to create an industry-ready policy compliance solution. The aim is to simply create a textual based version of the security extension itself to act as a proof of concept. For instance, a modeller (or security expert) will fill in a simple form stating which BPMN elements should have certain security requirements. The modeller will then be able to run this form across the diagram to ensure the respective elements do in fact specify those security requirements.

Although this represents a very basic Boolean checking and an actual policy would likely be more elaborate, this work is not yet at the stage to solve this problem. Currently, the primary focus is the creation of a framework, one which allows for the development of extensions that are capable of comprehensive, complexity managed security specification within modelling languages. A robust and dynamic policy compliance solution represents the immediate future of the work once the aforementioned issues are resolved.

Nevertheless, there is still need to a create policy verification within the case study and as such a need to specify the security policy details. Fortunately, due to the approach that is being taking in this instance, the concepts and properties used within the policy specification will be identical to those used in the cyber security ontology.

5.2 Modelling Language Frameworks

5.2.1 Core Language Framework

This component requests that the *language architect* should detail the visual vocabulary and grammar of the core language. Fortunately, as an already existing core language is being used in this instance (BPMN), the original language developers have already documented all this information for future use. Therefore, instead of retyping it into this thesis, refer instead to the BPMN 2.0 documentation (Object Management Group, Inc., January, 2011). (One can also view the earlier section in which the basic BPMN elements were outlined; Chapter 2).

The only element not in this documentation is the one created in the previous section named “all elements”. For this element, the thesis states it should always appear in the top left location of a BPMN diagram. As for the vocabulary of the element, it should look like the symbol in Figure 5.2.

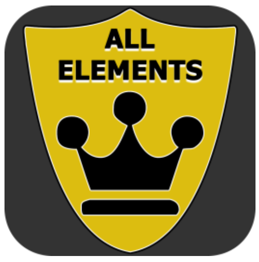


Figure 5.2 All elements symbol

5.2.2 Security Language Framework

Defining the security language framework requires an exceptionally larger effort than that of the core language. As previously stated, this component and subsequently its child components of visual vocabulary and grammar, represent the two most neglected areas of language design in current security extensions. It is also where a large majority of the PoN, and the previously proposed heuristics can be satisfied.

Therefore, this component(s) is highlighted again, as it represents another core contribution of the work. The main framework is intentionally abstract in terms of what visual approach to take when defining the visual requirements of an extension. As mentioned, the success of the framework is not dependent on just one perfect solution for this component, there are likely multiple avenues that can satisfy all the requirements. Nevertheless, the visual framework defined in this case study for BPMN, does represent an added contribution to the work. Whether this always be tied to the main framework, or simply used for extending another language with a different domain other than security, the approach taken here provides its own novel contribution to modelling languages as well the larger data visualisation domain.

5.2.2.1 Visual Vocabulary

Although the PoN deals with both the vocabulary and grammar of languages, it leans more towards the vocabulary side, detailing multiple principles that should be achieved in the design of any notation. As seen in the earlier reviews, these have customarily been ignored in current extensions. Therefore, rather than creating a notation based on conjecture, the approach of thesis is to use these principles from the very beginning as a foundation for creating the symbols.

Firstly, the *perceptual discriminability* of the notation will be discussed. When extending an existing language with a new domain, the *perceptual discriminability* of the extension language has two criteria to satisfy instead of just one. The new notation not only requires discriminability amongst its own constructs, but also against that of the core language - i.e. all business process notation is clearly distinguishable from security notation.

In most of the extensions reviewed, clear distinction against the core language (BPMN) was done rather successfully, one extension that stood out, was Rodríguez *et al.* (Rodríguez, et al., April 2007). Although their extension lacked colour, by using a padlock shape as an outer shell for each construct, it ensured the two languages were very distinct from each other. The main reason for this being that a padlock is not a shape currently used within the BPMN notation, nor does it resemble any other symbol within the language for that matter. Salnitri *et al.* (Salnitri, et al., 2014) and Koh and Zhou (Sang & Zhou, 26th-28th October, 2015) both opted for circles for their outer shape, which in contrast, are very heavily utilised within BPMN and therefore do not have nearly as high *perceptual discriminability*.

A quick search on Google Images using the keyword “security”[16] provides all the justification necessary for using a padlock as an outer shape for the notation, see Figure 5.3.

[16] Google; Google: Images “security”; (https://goo.gl/6bCjq0) accessed 11/04/2017

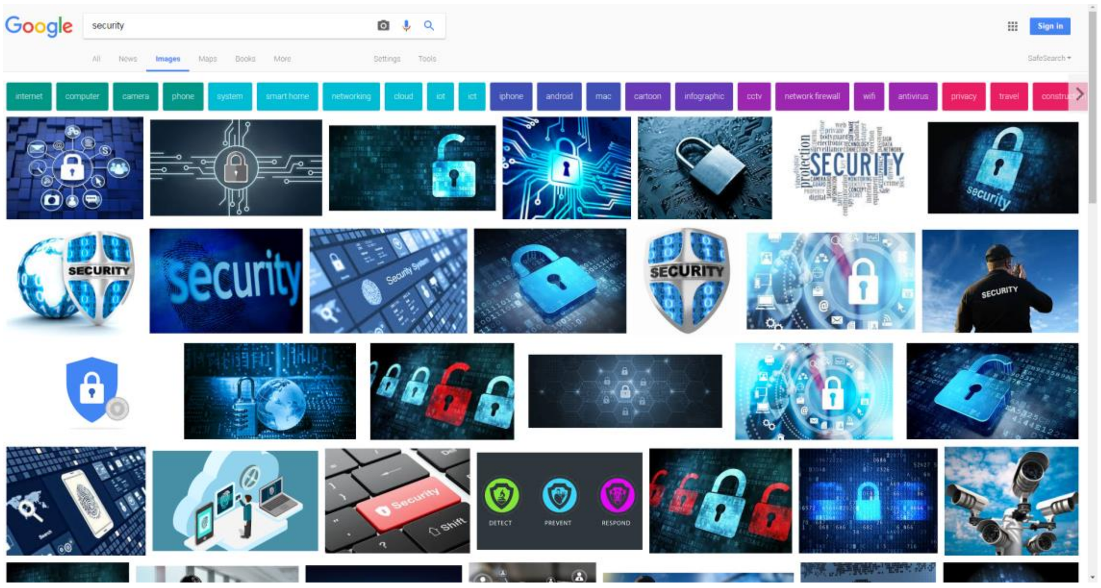


Figure 5.3 Google Images search, keyword "security"

Nevertheless, as mentioned, this is only one side of *perceptual discriminability*. The notational constructs must also be distinguishable amongst each other. If a consistent shell is to be used for all constructs, this means any distinguishing features must be encompassed inside that shape.

Upon further analysis of the padlock shell, the design space within the body of the shape (which is where much of the construct's graphic would go) is relatively small, especially by comparison with how much space the shape occupies on a diagram. If one refers to Figure 5.4 it shows that a padlock shape takes up roughly 40 squares on a diagram. However, the inner design space only allows for 20 compact squares, just 50% of the space the construct occupies. By comparison, when viewing a shield shape, (this happened to be the second most popular icon associated to security according to the study on Google), the shield takes up 39 squares and allows for a compact design space of 23 squares, i.e. 59% of the construct space. Of course, this comparison only took full squares into account, including incomplete squares as well, the shield could potentially offer up to four/five more. Whereas, most of the padlock’s inner design space is wasted with the locking mechanism. Therefore, based on this analysis of the two most *semantically immediate* shapes in regard to security, the shield has proven itself the better option for an outer shell.

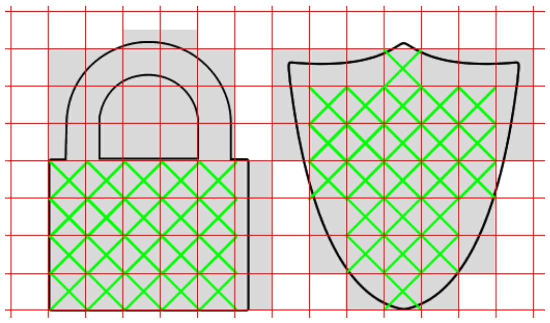


Figure 5.4 Padlock and shield design space comparison

The next principle to consider is *semantic transparency*. As mentioned, any discriminability between same notation symbols must be achieved inside the shield shape. Therefore, it is logical to ensure this inner shape is also capable of adequate *semantic transparency* in regards to the symbol's underlying semantics.

Leitner *et al.* (Leitner, et al., 6th-7th November, 2013) conducted a study on 11 security concepts (*user, role, risk, privacy, encryption, digital signature, data integrity, data confidentiality, availability, audit* and *access control*), in which they asked participants to draw a symbol they would most associate with each one, discussing their results. However, this is not a very effective approach to take. Although one could argue that a popular enough consensus will eventually produce a semantically immediate symbol, there are several drawbacks.

Firstly, the creative ability of the participants can directly impact results; abstract expression or poor drawing skills? Furthermore, for a large notation such as the one defined within this work, the duration of time required to conduct trial-and-error searching for popular icons is very inefficient. Moreover, the biggest issue of this approach, is that it effectively dismisses all of the research that has gone into icon design. Although at some point the "beauty" of the icon will be reliant upon the designer's artistic abilities, there are several principles (Moody, 2009) and methodologies (Chen, 10th-12th December, 2003; Lin & Lai, 2013) for aiding in the design of an icon that should be sought out first, Leitner *et al.'s* approach cannot guarantee these principles are adhered to.

The best approach for ensuring they are satisfied, is to use an icon design approach. This has been proven to increase usability, recognition and familiarity (Kascak, et al., 30th September - 1st October, 2013). Inevitably, every symbol will likely require some learning due to the complexity of most concepts. However, using this approach will improve the chances of attaining *semantic immediacy* given icons' natural goal of being a graphical mnemonic to their concept (Hartmann & Vossebeld, 2013).

Peirce defined three instances a sign can exist in: iconic, the sign resembles or is similar to its representative object; indexical, the sign indicates its representative object i.e. cause and effect; and symbolic, the sign is arbitrary and relies on users learning the connection with the respective object (Burks, 1949). For instance, a picture of a *fire*, would be an iconic sign as it resembles the object of *fire*. Likewise, smoke or burnt embers would be an index sign, as these indicate the presence of *fire* without showing it. Finally, the word fire itself, would be categorised as a symbolic sign, as it in no way resembles or hints at the presence of *fire*. One is only able to make the association, because previous experience and knowledge has taught that these letters, in this order, signify the object of *fire*. Therefore, when creating the symbols for the security concepts, the goal is iconic, then indexical, then symbolic.

Moreover, there are eight visual variables which can be used to construct a notational symbol, these being: horizontal position, vertical position, shape, brightness, size, orientation, colour and texture (Moody, 2009). The utilisation of *shape* has already been discussed, both as the outer shell of each construct and for the icons themselves. To attain *visual expressiveness* however, the aim should be to utilise at least half of these eight variables. (Half being the minimum requirement used when evaluating existing extensions.)

Given that horizontal and vertical position are elements of the visual grammar and not vocabulary, they will be dismissed at this stage and revisited in the next component. The next variable on the list is brightness. In this instance, brightness is utilised as a way of reiterating symbol depth within a hierarchy. Referring to Figure 5.1, some of the notation's vocabulary was used to subtly ease the comprehension of the ontology, defining the different areas and hierarchies. As seen, the lower in each hierarchy a concept/symbol is, the lower its brightness. Likewise, each of the six areas has a different colour, this translates directly into the notation itself, showcasing the use of colour to distinguish between each area of the domain. For selecting the colours to use within the notation, the range of available colours in graphics software was first observed, these can be seen in Figure 5.5.

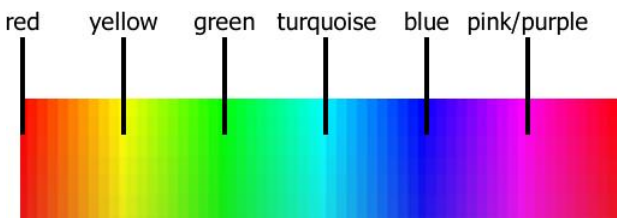


Figure 5.5 Range of available colours

Viewing the colour palette, six distinguishable colours can be identified right away, (the far right looping back to the colour red). Therefore, it would seem fitting to use the same colours within the notation, especially as it also has six distinguishable areas. For the most part, this was done. However, a bright colour yellow is an irritant to the eye and can cause fatigue when viewed for too long (Morton, n.d.), not to mention it clashes with a lot of other colours. Therefore, the yellow colour was changed to a less harsh light orange, it still provides a strong enough distinction from the others, with less strain on the end user's eyes.

There are multiple visual impairments that can impede an end user's ability to distinguish between certain colours (Ichihara, et al., 29th-31st January, 2008). Therefore, as per Moody's recommendations (Moody, 2009), colour should only be used for *redundant coding* and not as a primary source of distinction. The primary notation used for distinguishing between each of the six areas will be discussed in the section with the visual grammar.

At this stage, it is fitting to bring in a figure to better visualise the use of each variable. Referring to Figure 5.6, (a) represents the outer shell for each construct, with (d) providing an example an icon inside this shell. The final two elements, (e) and (f) showcase the use of colour to highlight the symbol depth and domain area.

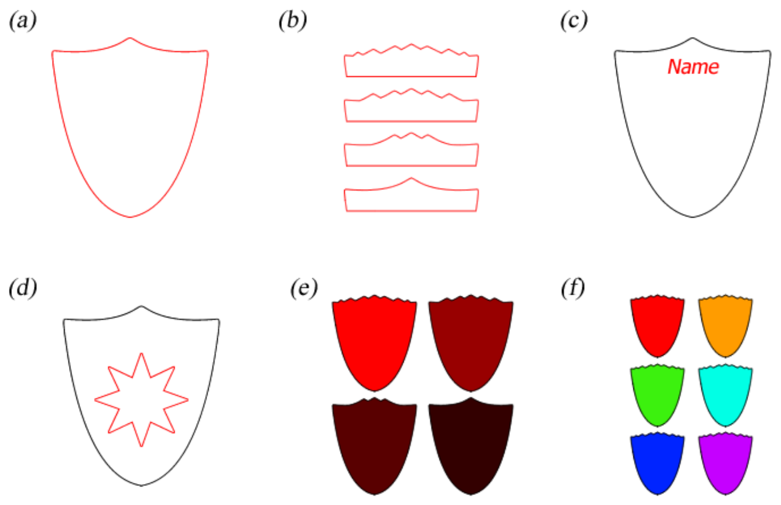


Figure 5.6 Usage of visual variables

Although brightness has already been used to define symbol depth, it is a weak form of notation, the end user's own vision as well as the brightness of the monitor they are using can both directly impact the effectiveness of this variable. Therefore, a more robust variable is needed; this will then push brightness to a form of *redundant coding* instead of a primary source of distinction. Referring to (b) in Figure 5.6, only the top section of each shield is visible, differing only on the number of peaks each one has. Referring then to (e), this peak count also corresponds directly to the symbol hierarchy depth. The lower the symbol, the lower the peaks and the lower the brightness. Size is also used to emphasise this distinction, although it is not showcased within this figure, each symbol is scaled down with each respective step down the hierarchy.

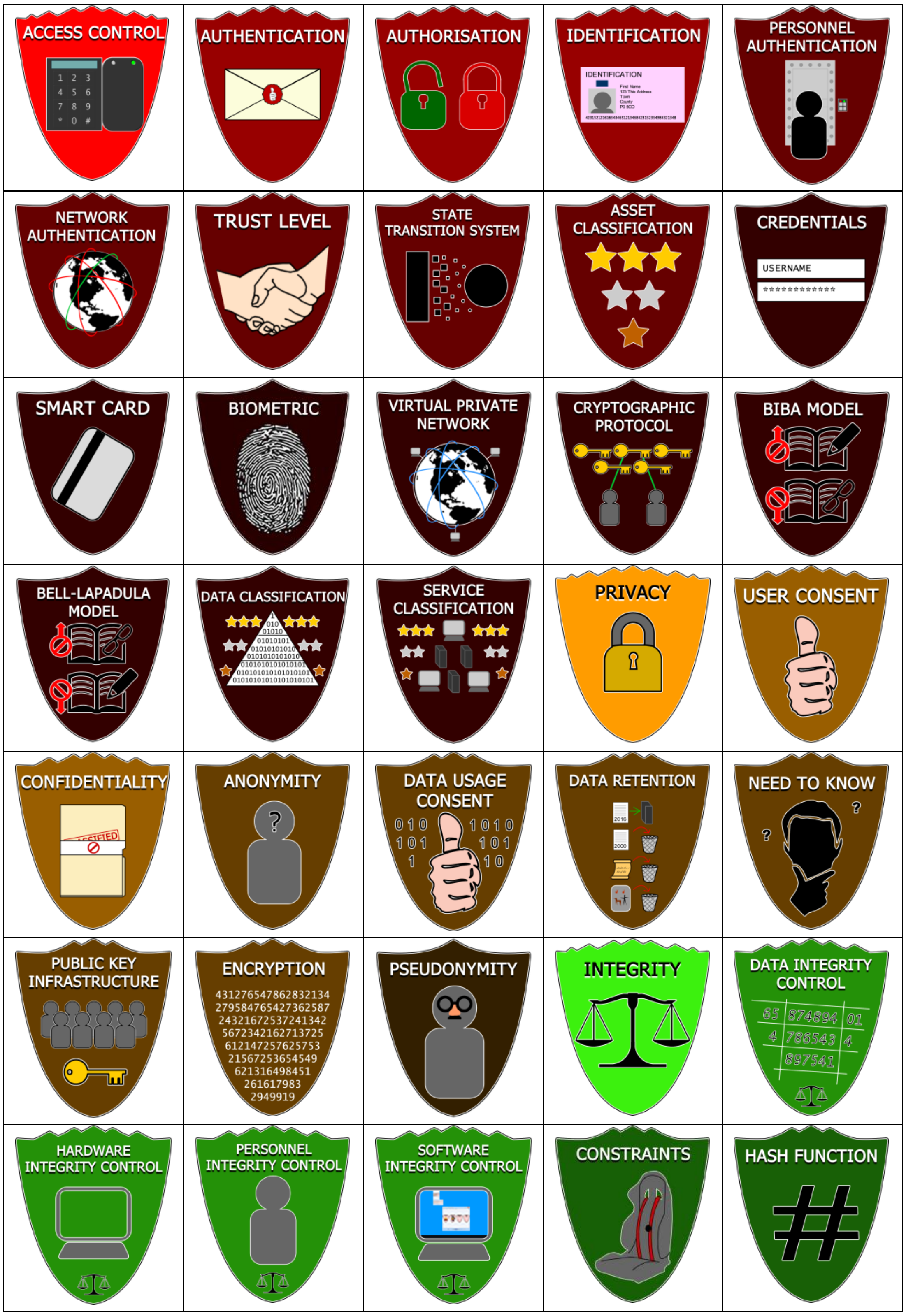
Of the eight visual variables, the vocabulary component alone utilises four of them, already satisfying the principle of *visual expressiveness* as per the previously mentioned standard. Nevertheless, as discussed later, there are still several variables left that can be effectively utilised within a notation.

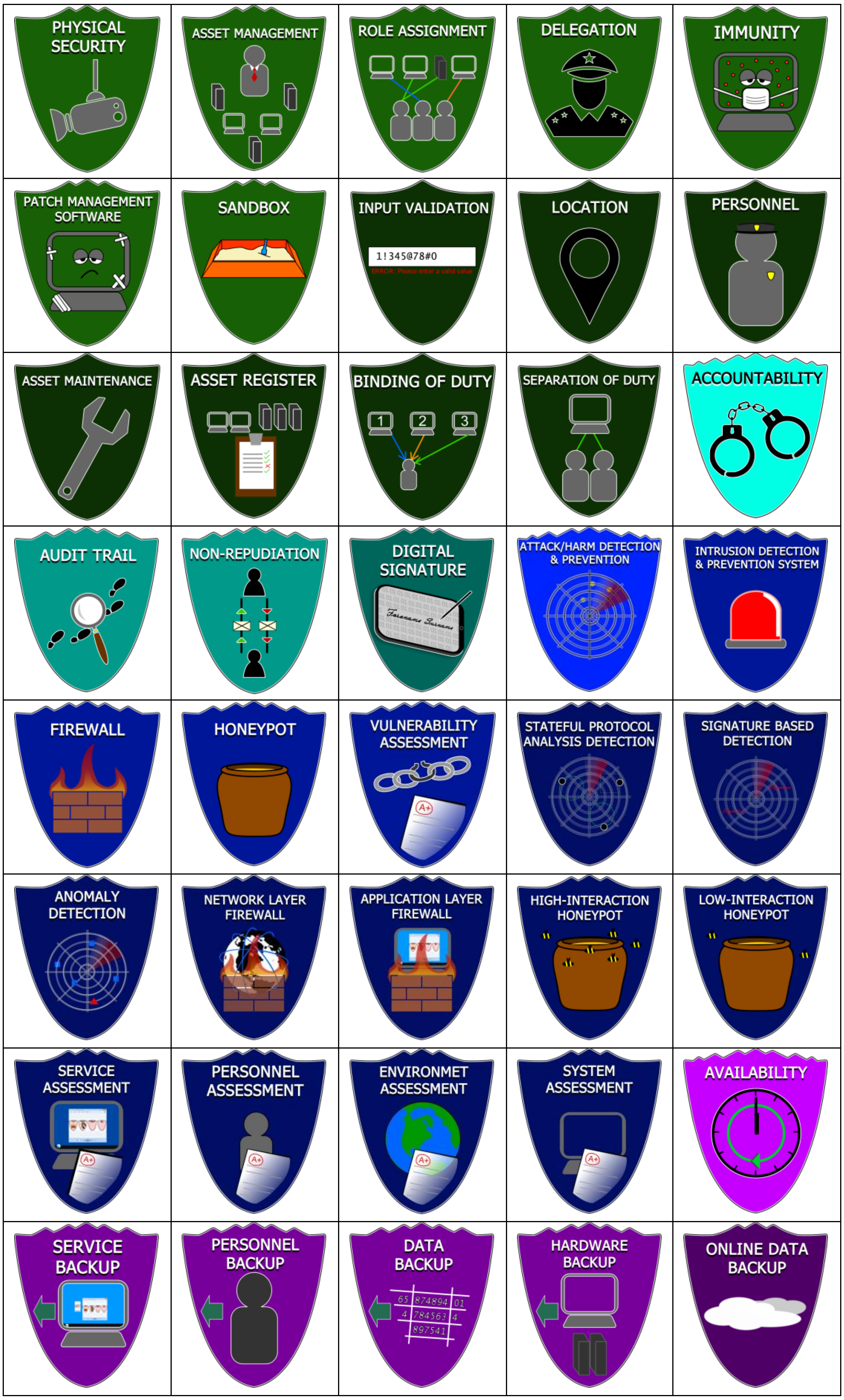
The final principle that is considered within the vocabulary component is *dual coding*. Referring once again to Figure 5.6 (c), the symbols utilise the sub-component of *hybrid (graphics+text) symbols*. Although *annotations* are also suggested within the *dual coding* principle, given that almost every core language will already be utilising some form of *annotations*, it is more user friendly to not contribute unnecessary complexity by also including security annotations.

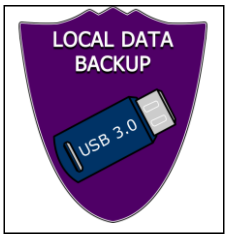
As promised when proposing the framework, this thesis has demonstrated how the visual vocabulary component alone can satisfy an extremely large number of requirements. Within this section: *perceptual discriminability, semantic transparency, visual expressiveness* and *dual coding,* have all been acknowledged. Of the proposed heuristics, a *graphical framework* has also been defined which can be used for the creation of any future notation.

Before considering future notation however, one must first specify the final design of each symbol within the current notation. Using the framework outlined in this section, the following 76 symbols represent the final notation of the case study extension:

Table 5.2 Entire security extension notation




PHYSICAL SECURITY
ASSET MANAGEMENT
ROLE ASSIGNMENT
DELEGATION
IMMUNITY
PATCH MANAGEMENT SOFTWARE
SANDBOX
INPUT VALIDATION
1!345@78#0
ERROR: Please enter a valid value
LOCATION
PERSONNEL
ASSET MAINTENANCE
ASSET REGISTER
BINDING OF DUTY
1
2
3
SEPARATION OF DUTY
ACCOUNTABILITY
AUDIT TRAIL
NON-REPUDIATION
DIGITAL SIGNATURE
Forename Surname
ATTACK/HARM DETECTION & PREVENTION
INTRUSION DETECTION & PREVENTION SYSTEM
FIREWALL
HONEYPOT
VULNERABILITY ASSESSMENT
A+
STATEFUL PROTOCOL ANALYSIS DETECTION
SIGNATURE BASED DETECTION
ANOMALY DETECTION
NETWORK LAYER FIREWALL
APPLICATION LAYER FIREWALL
HIGH-INTERACTION HONEYPOT
LOW-INTERACTION HONEYPOT
SERVICE ASSESSMENT
A+
PERSONNEL ASSESSMENT
A+
ENVIRONMET ASSESSMENT
A+
SYSTEM ASSESSMENT
A+
AVAILABILITY
SERVICE BACKUP
PERSONNEL BACKUP
DATA BACKUP
65 874894 01
4 784563 4
897541
HARDWARE BACKUP
ONLINE DATA BACKUP



Viewing these symbols, it is easy to appreciate how when used correctly, the PoN can drastically improve the overall design of a notation. It is also easy to see, how adding any more symbols to the notation will be a very straightforward process.

For example, if a concept called *star* was to be added to the notation. The first step is to identify in which area of the ontology it belongs, maybe *availability*. Then whether it is a child of an existing concept or its parent is *availability* directly. In this instance, it will be a child of *personnel backup*. From this information alone, before the user even begins to develop anything, the following is already known: the outer shape is a shield with three peaks, the colour of the symbol is purple, the brightness is at level three and the name of the concept on the symbol is *star*. All of which will produce a symbol like that in Figure 5.7.

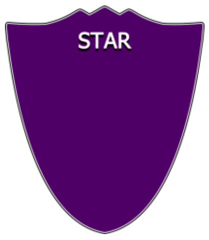


Figure 5.7 Star concept symbol

This means that whatever the final tool may be, the only input required is some form of icon. It is more than plausible to assume that functionality can exist which allows the user to input this basic information along with an icon, and the tool will be able to generate the symbol and any associated parent/child relationships. Not only providing functionality to extend a language, but extend it with already existing rules specifically designed to ensure various requirements are maintained.

The visual vocabulary as a framework can be seen in Figure 5.8.

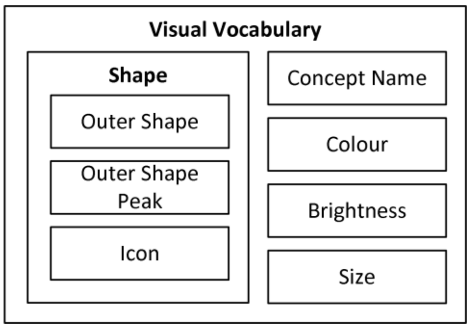


Figure 5.8 Visual vocabulary framework

5.2.2.2 Visual Grammar

5.2.2.2.1 Current Approaches

When discussing visual grammar, the main issue with current extensions is without a doubt their poor *complexity management*. This is the one principle that can be traced back as the root cause of all other issues in extensions, primarily *construct deficit*. Current extensions provide a very low number of concepts, yet, they still suffer from poor *complexity management*. Therefore, there is reasonable evidence for claiming that the reason current extensions have such a paucity of concepts within their notation is that they were unable to find a suitable solution for managing their complexity.

Before proposing a new method of visualising security requirements within a modelling language, it is logical to first establish the reason no extension has yet managed to achieve complexity management. There are currently three approaches being used to relate security requirements with BPMN elements. The first of these being symbol stamping, as seen in Figure 5.9.

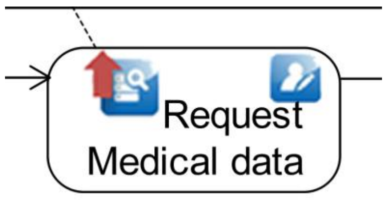


Figure 5.9 BPMN/security pairing - symbol stamping

The thesis has already discussed in a previous section the issues of this approach. At some point, there will be a situation where either the element text, or security notation will need to be scaled down to fit everything in. This is a workaround solution, until the point that either of these components becomes so small that they are illegible. In no way, is this an appropriate solution for relating to the two domains.

The second approach is to add further relationships (lines/arrows) to the diagram to pair the necessary concepts. For example, see Figure 5.10.

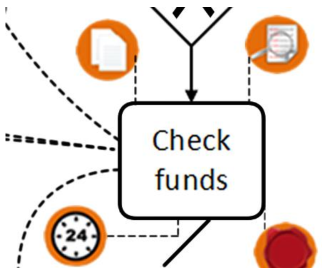


Figure 5.10 BPMN/security pairing - added relationships

This method overcomes the issues with the previous approach. However, a similar issue is now present in a different location. Instead of shrinking element text, the modeller will now have to determine whether to shrink entire BPMN elements. There will again be a point at which the modeller will be unable to fit their desired security requirements onto a diagram and must decide between usability and specification requirements.

The final approach to pairing BPMN and security requirements is the physical replacement of BPMN notation with security notation, see Figure 5.11.

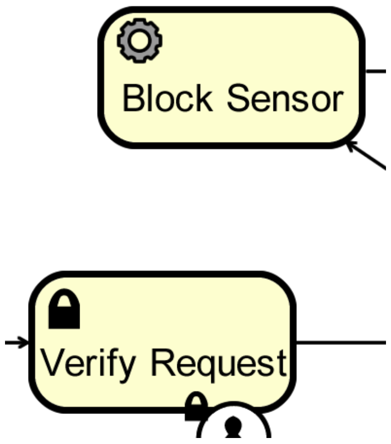


Figure 5.11 BPMN/security pairing - element replacement

In this figure, the focal element is the bottom task of the two, or a *security task* as named by the authors. The top task is included as an example of how Koh and Zhou (Sang & Zhou, 26th-28th October, 2015), opted to remove BPMN task type notation for their own padlock security symbol. An extension, unless necessary should aim to be as non-intrusive to the core language as possible. Primarily, this is to ensure the core language can still function as it was originally intended to do so. BPMN is a language for modelling business processes, should someone wish to see a visual representation of a business process, they have no interest in the security and as such no need to view security. Therefore, the extension should only compliment the core language and not alter it. This way, should the security be removed or changed the core process will remain the same and maintain its integrity.

The clear trend in existing approaches is the lack of design space. Whether this be inside BPMN elements or outside of them, there is no getting around the fact that any pre-existing BPMN diagram is simply unable to include sufficient security requirements alongside business process notation. A new approach is needed!

5.2.2.2.2 3D Extension

A possible solution to this problem can be found in a very different subject area. Looking instead to land and population growth, a very similar situation can be observed. Eventually, once a population has exhausted all available land, a point is reached where there is no longer any room for people to live. To overcome this issue, society began to build multiple floored buildings. By still utilising the same amount of land, building up instead of out allows for the possibility of incredibly dense cities across relatively small areas of land. For example, observe a comparison of New York City with and without manmade structures, see Figure 5.12.

Figure 5.12 New York City skyline comparison[17]

[17] Gizmodo (Kelsey Campbell-Dollaghan); Gizmodo: "What Every Block of New York City Looked Like 400 Years Ago"; (http://gizmodo.com/what-every-block-of-nyc-looked-like-400-years-ago-1733614748) accessed 18/04/2017

In this instance, numbers are not needed to emphasise the impact multi-storey buildings have had on society. The ability to build up (as high as current engineering will allow), has allowed for the accommodation of an exceptionally higher population than if everything was at ground level. The same benefits can be applied to modelling languages.

Surprisingly though, no one has yet considered representing security requirements across a different set of axes to BPMN. Disregarding this analogy with the built environment, research into 3D visualisations versus 2D visualisations has already provided empirical evidence supporting the use of 3D from both an efficient and user preference point of view (Amini, et al., 2015; Marcus, et al., 11th-13th June, 2003; Teyseyre & Campo, 2009).

As mentioned in Chapter 2, there have even been several publications within literature proposing the use of 3D to better aid in software engineering and modelling language complexity (Ware, et al., 24th-28th October, 1993; Gil & Kent, 19th-2th April, 1998). Although these approaches do not necessarily solve the issue which current extensions have, they are at least an indication that 3D may hold a potential solution. However, this potential never seems to gain any momentum.

This work aims to utilise as much of this potential as possible by proposing the representation of BPMN across one plane and security requirements across another plane perpendicular to the BPMN diagram. The advantage of this approach being, that BPMN can remain relatively unaltered, being represented without change across the *x* and *y* axes as seen in the left of Figure 5.13. Cyber security requirements can then be represented across the *x*, *y* and *z* axes, as seen in the right of Figure 5.13. This way, cyber security requirements are being displayed at a similar abstraction level as BPMN whilst still maintaining a comprehensible diagram.

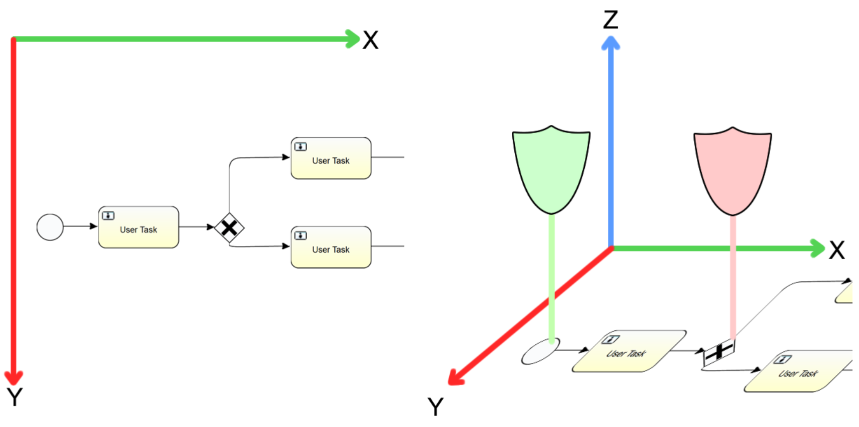


Figure 5.13 3D security proposition

5.2.2.2.3 2D versus 3D Experimentation

In theory, of course, this sounds a very plausible solution. However, to assess the feasibility of this hypothesis, experiments are needed to investigate whether a third dimension can provide any assistance to the complexity issues currently experienced by other 2D approaches in existing extensions.

The difficulty with testing this hypothesis however, is in the ability to focus the experiments explicitly on the *complexity management* and the *visual grammar* of the extension. BPMN itself requires some learning before understanding a diagram (or at least some prior experience with business process modelling (Recker & Dreiling, 5th-7th December, 2007)). As for security requirements, it was discussed earlier how there are multiple issues beyond *complexity management* which can impact a user's ability to read them from a diagram. The experiments must focus solely on 2D specification versus 3D regarding *complexity management*.

The first step in this process is determining which of the current 2D approaches is the best solution thus far, and therefore should be used to represent them all within the experiment. In this instance, this would be the extension created by Salnitri *et al.* (Salnitri, et al., 2014). Although they were not the only extension able to satisfy four of the PoN principles, their method of symbol design allows for a higher *diagrammatic complexity* than their equal scoring colleagues. Saleem *et al.* (Saleem, et al., January 2012), as discussed did not encompass their symbols inside of a shape, and as such they corrupt each other when placed close on a diagram.

In this case, one of the Salnitri *et al.'* diagrams[18] was chosen to act as the foundation for the 2D representation. This diagram can be seen in Figure 5.14.

Figure 5.14 Salnitri et al. SecBPMN original diagram

[18] Mattia Salnitri; SecBPMN: Home; (http://www.secbpmn.disi.unitn.it/) accessed 19/04/2017

As mentioned, just viewing this diagram alone, it is difficult to understand what business process is being shown. Likewise, without external resources or prior knowledge there is virtually no way of inferring the semantics of the security notation.

The PoN does not discuss any formal means of evaluating *complexity management*, only the concepts of *perceptual* and *cognitive limits* as previously discussed (Moody, 2009). However, a method of evaluation can be extracted from these concepts. If *perceptual limits* emphasise the theory of more elements means harder symbol distinction, this concept can be generalised into *accuracy*. That is, the accuracy at which a user can correctly identify security requirements and their respective BPMN counterpart. Likewise, if *cognitive limits* refer to the point in time in which a user can no longer comprehend the total number of elements, this concept can be generalised into *quantity of identifications in a given time*. Not quite as elegant a title, but it provides a quantifiable means of comparing 2D and 3D security specification. Utilising both approaches, it is now possible measure the accuracy at which a user can correctly identify symbols, as well as the total number of identifications in a given time frame. Given that these values are both based on existing and proven literature on *complexity management*, there is reasonable justification to state that the higher the value for either measurement the better the overall *complexity management* of the extension is.

Therefore, by conducting experiments with existing approaches to 2D security expression (acting as the control), and comparing them with a new approach which utilises 3D security expression, one will have a comparable and justifiable means of concluding which of the two approaches provides better *complexity management*.

Returning to Salnitri *et al.* diagram in Figure 5.14 and the difficulty in understanding each element, given the approach to measuring complexity, the actual information in each BPMN element or the security symbols used is relatively pointless. The only requirement is that each element (BPMN or security) is distinguishable enough to allow for unique identification. Therefore, for the experiments, the diagrams are generalised into their most simple form. That is, each BPMN element has a number, a very robust form of unique identification; with each security symbol being a different coloured circle, also a strong form of unique identification. Of course, for a final extension, colour as a primary notation is very poor design. However, for these experiments it is more than adequate. To better understand this, refer to Figure 5.15, where the diagram in Figure 5.14 has been translated into one purely designed for assessing *complexity management*.

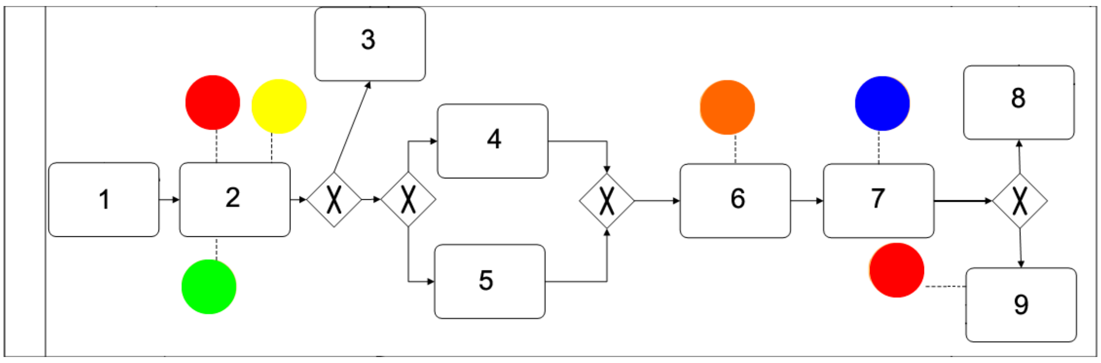


Figure 5.15 Salnitri et al. SecBPMN translated diagram (simple 2D)

The placement of the circles and their corresponding relationships to BPMN, are identical to those specified by Salnitri *et al*. in Figure 5.14. This is very important to the integrity of the experiments to ensure current 2D approaches are given a fair unbiased example. It would be very easy to manipulate the placement of these elements to satisfy a desired outcome. However as seen, an existing BPMN diagram with security notation already on it was used as foundation to merely overlay this temporary notation in the exact same locations.

For the 3D example, the exact same BPMN diagram was used alongside the current working visual grammar for the case study to assign security requirements. To ensure the integrity of the results again, each BPMN element in both the 2D and 3D diagrams had identical numbers of security requirements, the only difference being the colours of each circle. The 3D diagram can be seen in Figure 5.16.

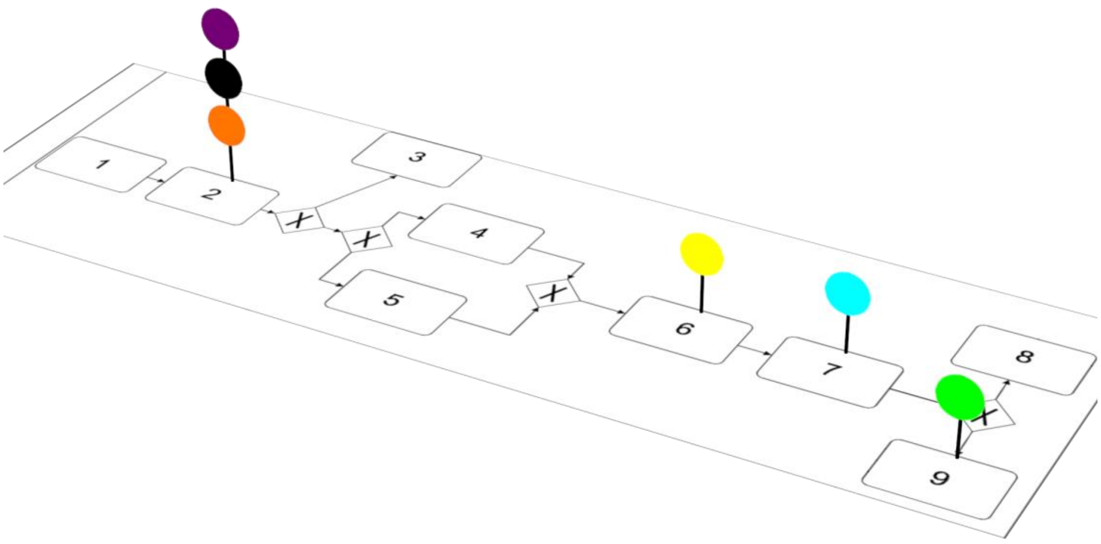


Figure 5.16 Simple 3D diagram

The experiment itself was relatively simple, participants were shown the BPMN diagrams via an overhead projector and given a set amount of time in which they had to identify what colours were linked to each task, noting their results through a paper based questionnaire.

They were given 30 seconds per diagram and upon completion of each one, were asked usability questions regarding the ease with which they could identify the colour and task connections (paper based). They were then given another questionnaire regarding the comparison between the 2D and 3D approach and which of them they preferred, providing both quantifiable data from the symbol pairing, and user opinion from their feedback.

The aim of these experiments being, to try and discover whether the proposed 3D approach provides any better *complexity management* than current 2D attempts. As mentioned, current extensions are struggling with their *complexity management*, given that 3D has proven itself in other areas of visualisation, it may be a potential solution to the complexity issues in language extension. These experiments will provide quantifiable data to either support or disprove this hypothesis, along with a multitude of user feedback on their preferred approach.

As mentioned, the 2D diagram within the experiment will act as the control. The chosen solution represents the best of current attempts and as such acts as the ideal baseline to compare the 3D results against. Likewise, the participants within the experiment also require a similar baseline. Therefore, given that modelling languages are a form a visualisation and within the area of computing, Multimedia Computing students were used to act as a control group for the experiments.

To recruit participants, students were emailed a participation information sheet detailing the experiment and inclusion criteria, asking for volunteers (there was no reward for taking part). The inclusion criteria consisted of participants having no prior knowledge about the project and no difficulties distinguishing between colours.

The experiment[19] was conducted initially with the Multimedia Computing students. This sample consisted of 30 second-year undergraduates, 22 of whom were male and 8 who were female, with an average age range between 18-24 years old. To ensure there was no collusion during the experiment, it was conducted under exam conditions. All experiment

[19] The experiment was pre-approved by an ethics board before it was conducted.

results are based on a 95% confidence level. Due to some anomalous results in this experiment (*incomplete questionnaires*), 5 of the participants' data had to be disregarded.

Results from the experiment for the 2D diagram, found that participants could complete an average of 90.67% ±6.85% of the diagram, with an accuracy of 94.32% ±5.06%. With complete, referring to the previous concept of *quantity of identifications in a given time*, or in lay terms: participants attempted to identify colours for 90.67% (or 8.1) of the 9 BPMN-task elements in the given time. Accuracy of course, being measured by the number of task elements, colours were correctly identified for. No half marks were awarded for incomplete or partially correct elements, only perfect identification of all colours connected to a task was awarded a mark.

For the 3D diagram, participants could complete 88.89% ±8.43% of the diagram with an accuracy of 94.73% ±3.42%. Of the participants, 12/25 managed to complete 100% of both diagrams, 7 managed to complete more of the 3D diagram and 6 managed to complete more of the 2D diagram. As for accuracy, 15/25 participants achieved an accuracy of 100% in both diagrams, 5 achieved a better accuracy on the 3D diagram and 5 achieved a better accuracy on the 2D diagram. A graph of these results can be seen in Figure 5.17.

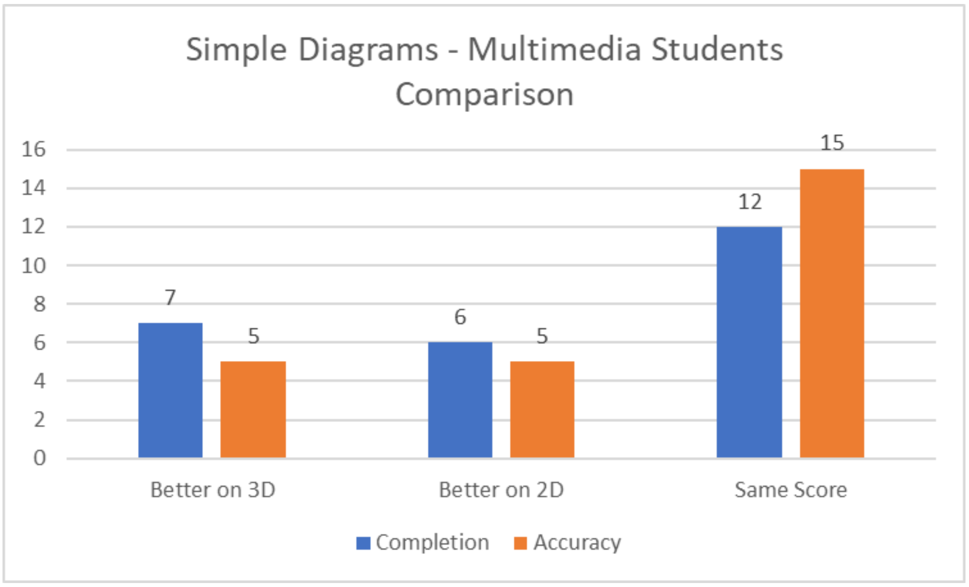


Figure 5.17 Simple diagrams – Multimedia Computing students' comparison

Given the statistical difference between the 2D and 3D results (a completion improvement of -1.78% ±9.04% and an accuracy improvement of 0.41% ±5.57%), it was concluded that in this instance, 3D provided no advantage over 2D with regards to *complexity management*.

However, this was somewhat expected. It was discussed in a previous section about how current authors have been very careful in the number of concepts they use within their extension, always ensuring their complexity is just manageable. Referring to Figure 5.14, this is evident by the low number of concepts used on the diagram. In anticipation of a result such as this, two more diagrams were created as though the security extension had a comprehensive number of concepts. More specifically, the previous diagrams (referred to as the simple diagrams), had six security requirements on them. The latter diagrams (or the complex diagrams), have 37 security requirements on them. The 2D complex diagram can be seen in Figure 5.18.

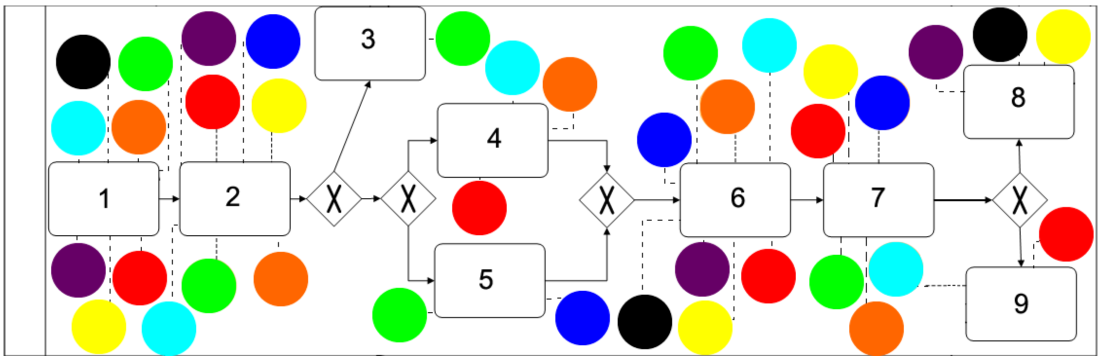


Figure 5.18 Complex 2D diagram

As no extension with this number of concepts is yet to exist in literature, it is impossible to overlay circles onto an existing diagram like with the previous examples. Nevertheless, the example endeavored to replicate Salnitri *et al.'s* approach as much as possible, following the previous visual grammar. The results of which can be seen in the previous figure. This of course looks very complicated, although some may suggest this was done intentionally, this is not the case. It merely proves the previous point that current extensions have been created with *construct deficit* to ensure their complexity is managed. Had they been created comprehensive to the domain, current 2D approaches would yield results like those in Figure 5.18. The 3D complex diagram, which features an identical number of circles per respective BPMN element, can be seen in Figure 5.19.

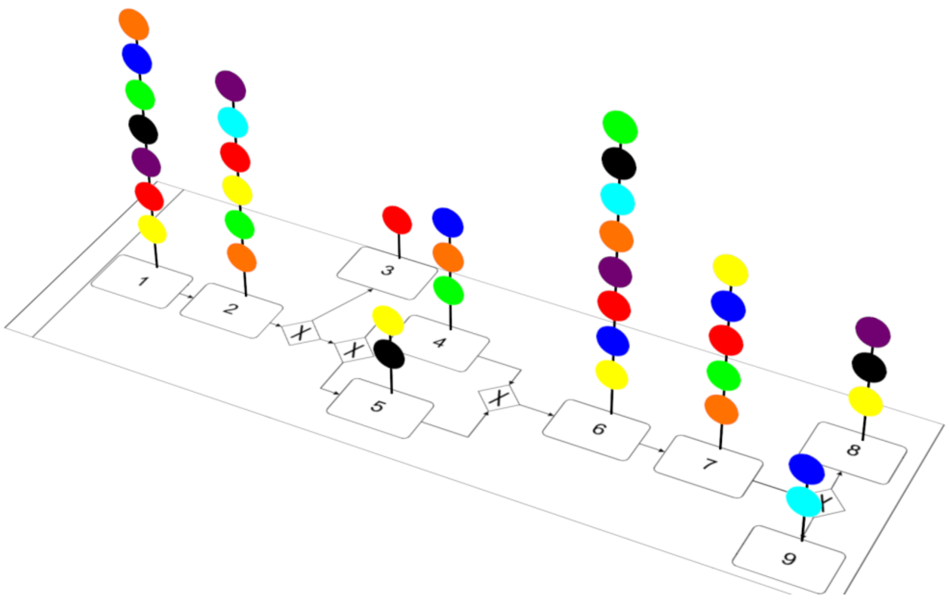


Figure 5.19 Complex 3D diagram

For the complex diagrams, as there are more elements to identify, the allocated time was increased from 30 seconds to 100 seconds; both times were chosen based on pilot studies. A few participants independent to those in any of the experiments, were asked to complete the experiment without a time limit, but their time to complete the task was noted. The main experiment times were then chosen based on these pilot studies, to ensure there was an unlikely chance of 100% completion. If all participants could comfortably achieve 100% completion, little could be learned from measuring this value. During the experiments themselves, it was made explicitly clear to participants not to rush through the questionnaire. It appears this request was acknowledged given that so far neither diagram has achieved an average of 100% completion.

The results for the 2D complex diagram gave a completion percentage of 83.33% ±5.88% with an accuracy of just 47.58% ±9.9%. Comparing these results to the 2D simple diagram, it is evident how much authors have relied on *construct deficit* to satisfy *complexity management*. Although the completion percentage is not that much lower; 83.33% compared with 90.67%, the accuracy percentage has halved, 47.58% compared to 94.32%. This result is very worrying from a security perspective. If any extension thus far had been comprehensive to the domain, these results indicate that end users would only be able to identify security requirements with an accuracy of just under 50%. Translating this into the possibility that an entire system would have only half the correct security requirements specified at design, this is a potential disaster waiting to happen. The number of vulnerabilities in a system would be exponential.

With the exceptions of one participant, who managed to improve their accuracy with higher complexity, and another who achieved 100% accuracy with both diagrams, every participant (23/25) had substantially worse accuracy when trying to identify requirements on a complex diagram compared with the simple diagram. Thereby proving the validity of the previous evaluation, current 2D approaches have thus far had very poor design and their *complexity management* is insufficient when a high number of concepts are utilised.

It is worth noting that at this stage, the previously mentioned five participants who had their results discarded for incomplete questionnaires, did manage to complete all the necessary documents for the complex diagrams. As such, their results were included for the second half of the experiment. Although lacking in comparison data for them between simple and complex diagrams, when comparing 2D and 3D at high complexity the data is only strengthened.

As for the 3D complex diagram results, the completion percentage was 97.41% ±2.26% with an accuracy of 89% ±5.91%. The first reaction to these results is naturally, how did the completion percentage increase when compared with simple 3D diagrams; 97.41% against 88.89%? The only possible cause that can be deduced is that, when first presented with the 3D visualisation in the simple diagrams, the participants were somewhat surprised, and wasted several seconds comprehending the change. The reason for this being that, the participants were unaware the next diagram (3D) would be so drastically different from the first (2D). Not only this, the 3D visual approach is not something they would be as familiar with compared to the 2D solution. Although participants were given simple instructions regarding what to expect, the only possible cause for the completion increase, is that participants were more prepared for what to expect and familiar with the visualisation second time. Irrespective of the true reason, an increase such as this in such a short space of time can only be a positive for the proposed solution. This positive being, the time needed to become familiar with the grammar is very short.

The best result from this experiment however, is how close the accuracy percentages are in comparison to the simple experiments; 89% compared with 94.73%. This allows for a reasonable claim that so far, the proposed working visual grammar's *complexity management* is relatively unaffected when a comprehensive number of concepts is introduced.

A total of 10 participants achieved 100% accuracy in both the simple and complex diagrams, compared with just the 1 participant from the 2D diagrams. Overall, the 3D approach has proved to be the much more robust solution when dealing with higher levels of complexity. For a comparison between the 2D and 3D complex diagrams please see Figure 5.20.

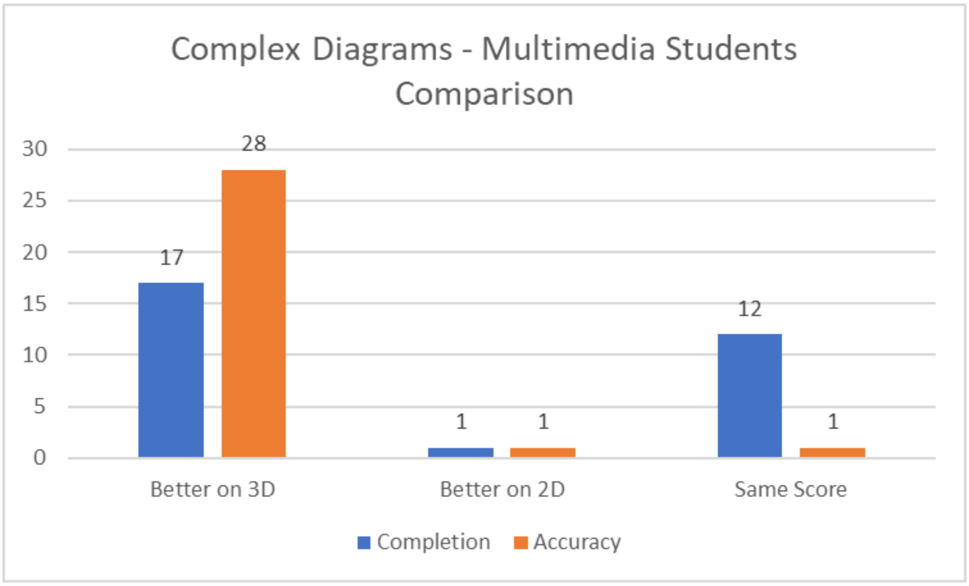


Figure 5.20 Complex diagrams – Multimedia Computing students comparison

This graph again illustrates the performance difference between the 2D and 3D approaches. As seen from the graph, 17/30 participants completed more of the 3D diagram, with just 1/30 completing more of the 2D one. As for accuracy, an impressive 28/30 participants performed better on 3D, with again just 1/30 performing better on 2D. Put into numbers, the average participant completed 14.07% ±5.32% more of the 3D diagram, with an average accuracy improvement of 41.42% ±8.82%.

Although the aim was to have a time limit in which 100% completion was impossible, for the 3D diagrams almost every participant achieved 100% completion (25/30). Nevertheless, given that most of the 2D diagram results did not achieve 100%, this data still provides very meaningful feedback, once again showcasing the superiority of the 3D solution compared to existing 2D approaches.

Overall, the quantifiable data from the control experiments has shown that on low complexity diagrams, 2D and 3D provide very similar results. However, on high complexity diagrams, current 2D approaches are inadequate, whereas 3D is still able to maintain *complexity management*.

As for user's preferences on the two approaches, participants were asked the following questions after completing each diagram:

1. *How easy did you find identifying what colours were connected to each task?*
   - ☐ *Very Easy*
   - ☐ *Easy*
   - ☐ *Difficult*
   - ☐ *Very Difficult*

2. *Did you feel overwhelmed by the number of symbols within the diagram?*
   - ☐ *Yes*
   - ☐ *No*

3. *What is your opinion on this method of matching colours to tasks?*
   - ☐ *Very Good*
   - ☐ *Good*
   - ☐ *Poor*
   - ☐ *Very Poor*

4. *Overall, how did you feel completing the task?*
   - ☐ *Happy*
   - ☐ *Neutral*
   - ☐ *Frustrated*

After completing each set of diagrams; both simple diagrams (2D and 3D) and both complex diagrams (2D and 3D). The following two question were also asked:

1. *Which extension did you find was easiest for identifying the colours connected to each task?*
   - ☐ *2D*
   - ☐ *3D*

2. *Which extension do you prefer?*
   - ☐ *2D*
   - ☐ *3D*

The results for the 2D simple diagram revealed that on average participants responded that they found the task *very easy/easy*, they *did not feel overwhelmed* by the number of symbols, the visualisation approach was *good* and their mood was *happy/neutral* during the experiment; these results being somewhat expected. As previously discussed, the simple diagrams were not necessarily created to test the participants, more to justify the substantial difference in *complexity management* when compared to complex diagrams. The results for each question can be seen in Figure 5.21.

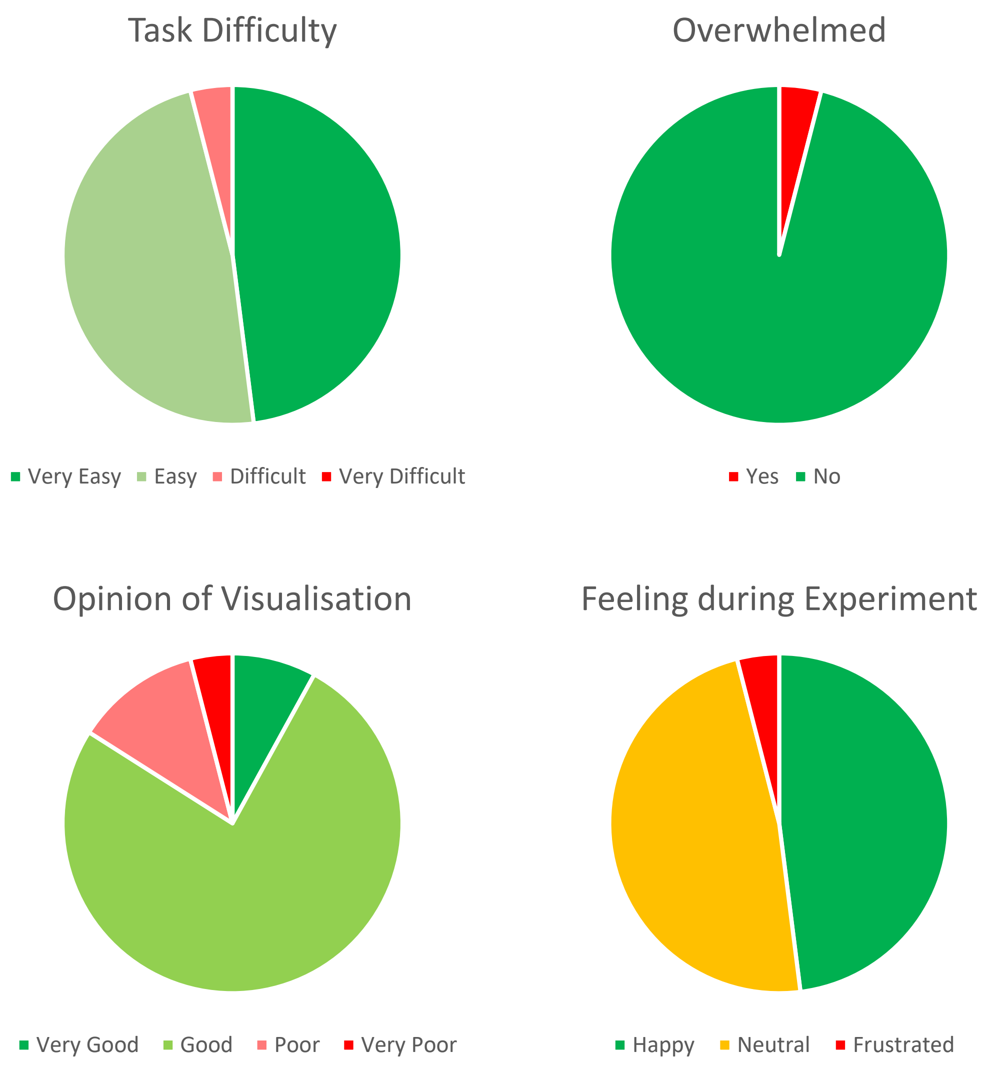


Figure 5.21 2D simple feedback results

As seen, most feedback for each question was positive. There were a few participants that expressed negative feelings, but for the most part, participants appeared to like and complete the simple 2D diagram with ease.

As for the 3D simple diagram, participants responded they found the task *easy,* they *did not feel overwhelmed,* the method of visualisation was *good* and their mood was *neutral*. The results for these questions can be seen in Figure 5.22.

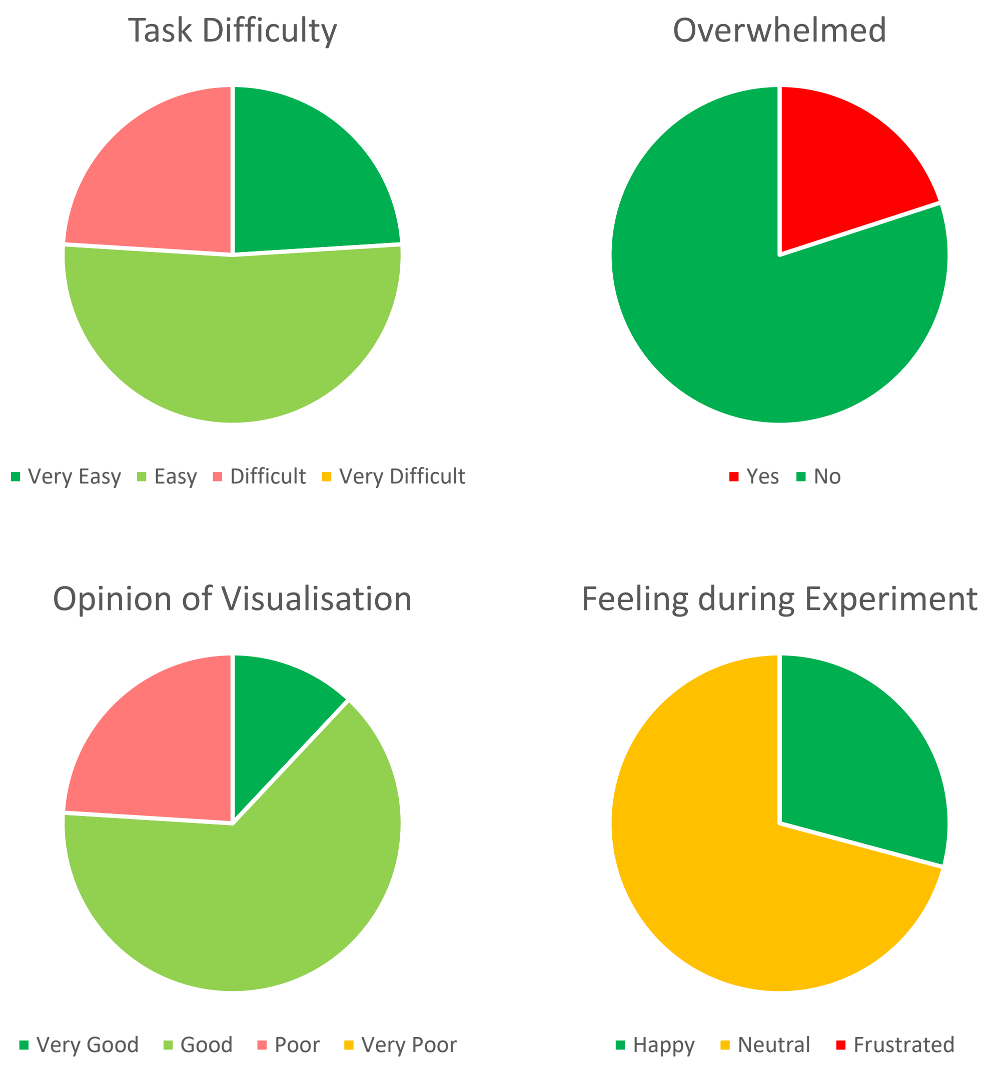


Figure 5.22 3D simple feedback results

As seen, although most feedback is once again positive, in comparison to the 2D results, the negative feedback is slightly higher in this instance. Unlike the 2D experiments, these results are slightly surprising. Referring to the quantitative results, there was very little difference between the 2D and 3D values for completion and accuracy, with 2D completion only 2% higher. Given that only 25 participants' data were used to determine this value, this percentage difference is relatively meaningless. Nevertheless, this feedback is necessary and still holds a lot of value. Irrespective of how the participants score, if they do not like the method of visualisation there is little chance it will be adopted.

When questioned with which method of visualisation the participants found easier, the results stated that 88% found 2D the easier extension. As for their preferred visualisation, 84% preferred 2D over 3D. This result is again somewhat surprising, given how similar the quantitative results are, one would assume there would be an even split between extension preference. However, this was not the case.

The 2D complex diagrams gave very different results to their simple counterparts. Participants responded that they found the task *difficult, overwhelming* and *frustrating*, stating the method of visualisation as *poor*. The results from these questions can be seen in Figure 5.23.

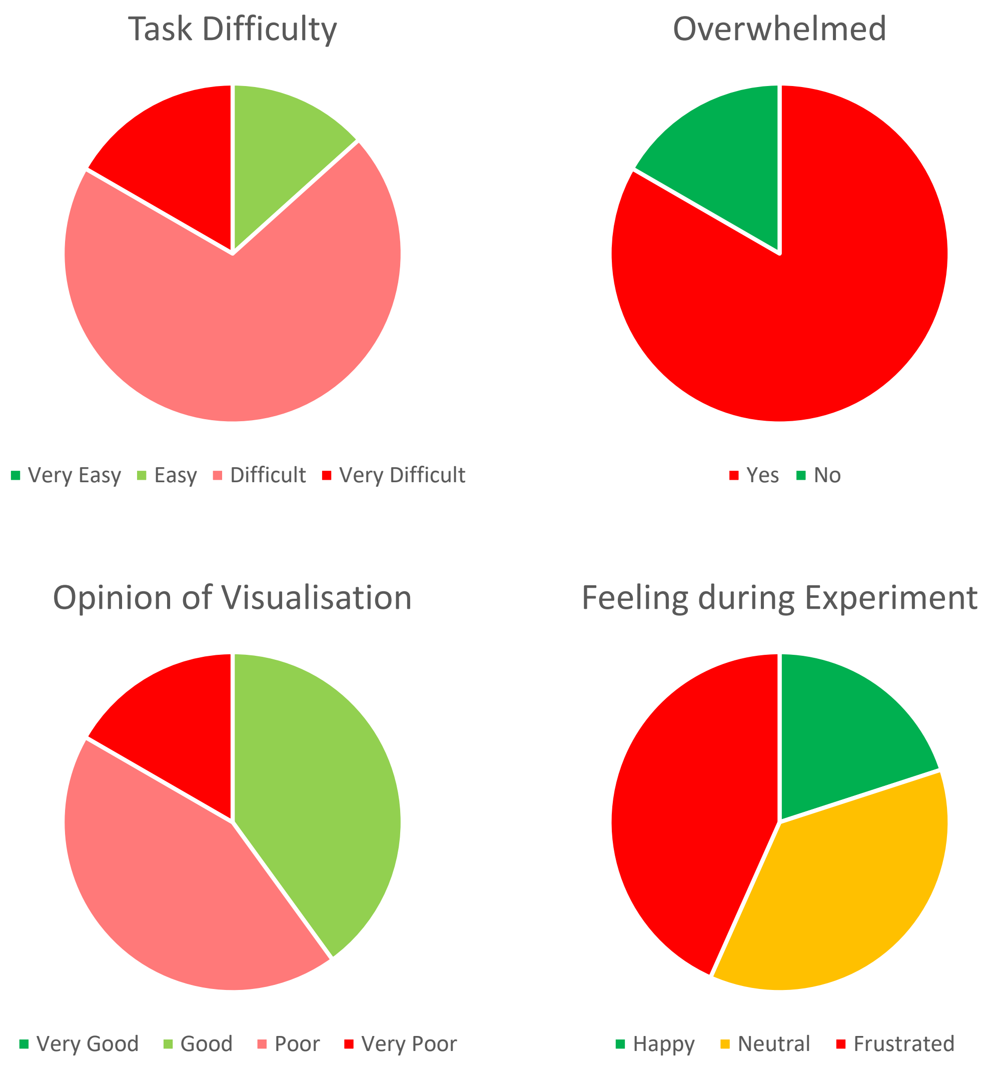


Figure 5.23 2D complex feedback results

As seen, the array of colours in these responses compared to the simple diagram is vastly different. The results from this feedback reflect the quantitative results much more accurately than the previous scenario. Although there is still some positive feedback about the visualisation, the majority of data shows that when the complexity of the diagram is increased, participants have very different opinions on current 2D approaches. Once again, proving the theory of current solutions only working when *construct deficit* is present in the language.

The 3D complex diagram on the other hand, provided more positive feedback than its simple counterpart. Even though the number of concepts had increased sixfold, participants found the extension just as easy to use; in comparison to the 2D approach, many had now changed their opinion on the 3D method. Responding they found the task *easy, did not feel overwhelmed*, the method of visualisation was *good* and their mood was now *happy* on average instead. The results from this feedback can be seen in Figure 5.24.

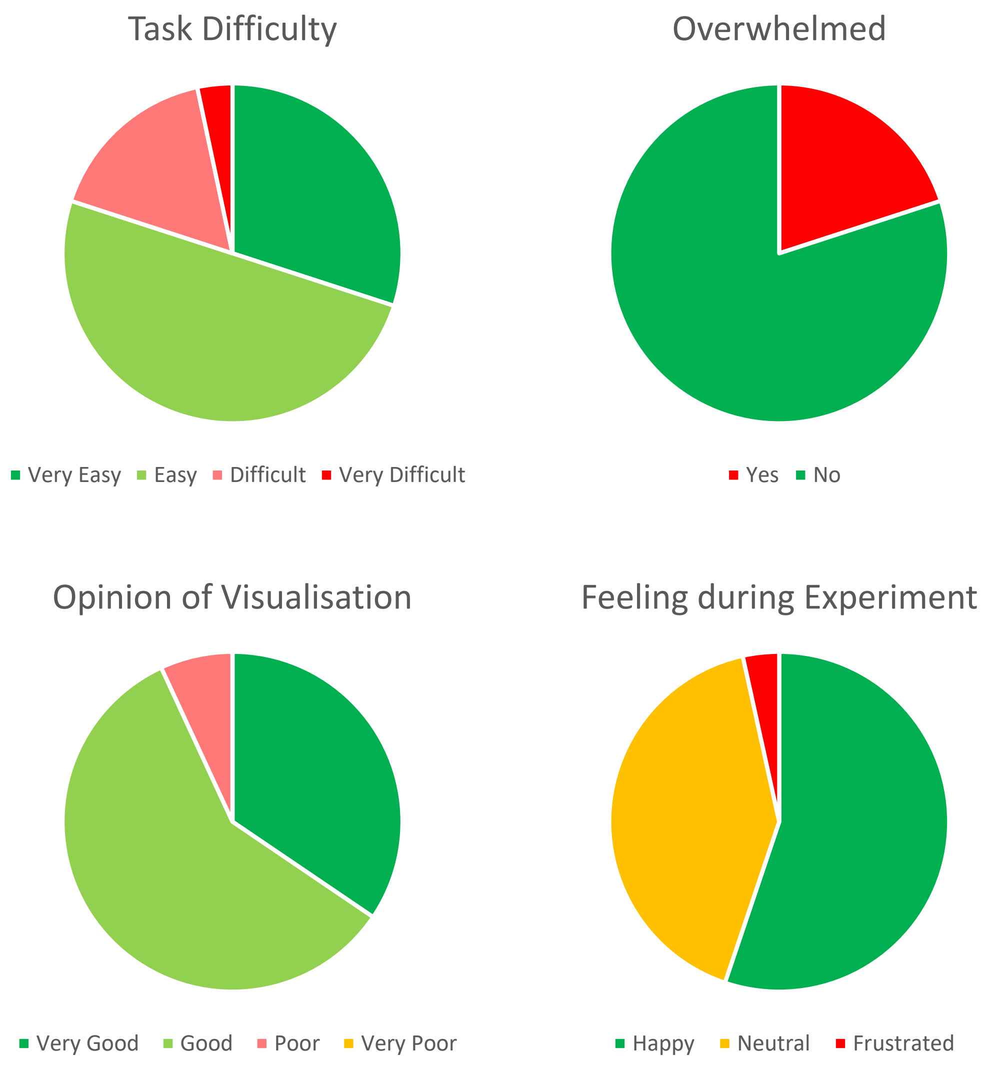


Figure 5.24 3D complex feedback results

The main differences that can be noticed when viewing these results compared to the simple diagrams is the opinion on the visualisation approach. Although the simple diagrams only had 25 participants' results and the complex diagrams had 30 results, one can still spot noticeable changes. For example, for the simple diagram, six participants rated the 3D approach negatively. However, even though the complex diagram had five more participants, only two rated the 3D approach negatively, showing that at least four completely changed their view of 3D when higher complexity was introduced.

This is further iterated when viewing the results from the extension preference. After completing both complex diagrams, the participants were once again asked which extension they found easier. This time around, 83% of participants stated the 3D extension as easier, with 80% now preferring 3D over 2D.

As mentioned, just as 2D acted as the baseline for comparing the 3D results against, the data from the Multimedia Computing students acts as the baseline for comparing further experiment results against. The intention of further experimentation being to ensure the reliability of the original results.

The experiment was conducted again with five other groups of participants, studying four different courses, across three different schools within the University. Of course, a full discussion around each of these experiments as extensive as the one carried out with the Multimedia Computing participants' data, would quickly overwhelm the thesis. Therefore, the analysis of these results will be somewhat brief in comparison, and included only as a means of supporting the results found in the control experiment.

As stated, there were five additional experiments conducted after the Multimedia Computing students, the background information for each these groups can be seen in Table 5.3. As with the control experiment, one participant across the other groups did not complete the simple diagram questionnaires correctly. Therefore, the total number of participant data for the simple diagrams comes to 94 (including Multimedia Computing), with the total participant data for complex diagrams coming to 100.

Table 5.3 Additional experiments' participant backgrounds

Number of Participants	Discipline	Education Level	Male/Female Split	Average Age Range
16	Cyber Security	Level 4	15/1	18-24
13	Business Studies	Level 6	8/5	18-24
18	Product Design Engineering	Level 4	9/9	18-24
12	Product Design Engineering	Level 5	12/0	18-24
11	Media Production	Level 5	6/5	18-24

Starting firstly with the results from the simple diagrams, the following graph represents a comparison of the percentage of participants who performed better with each approach in terms of diagram completion.

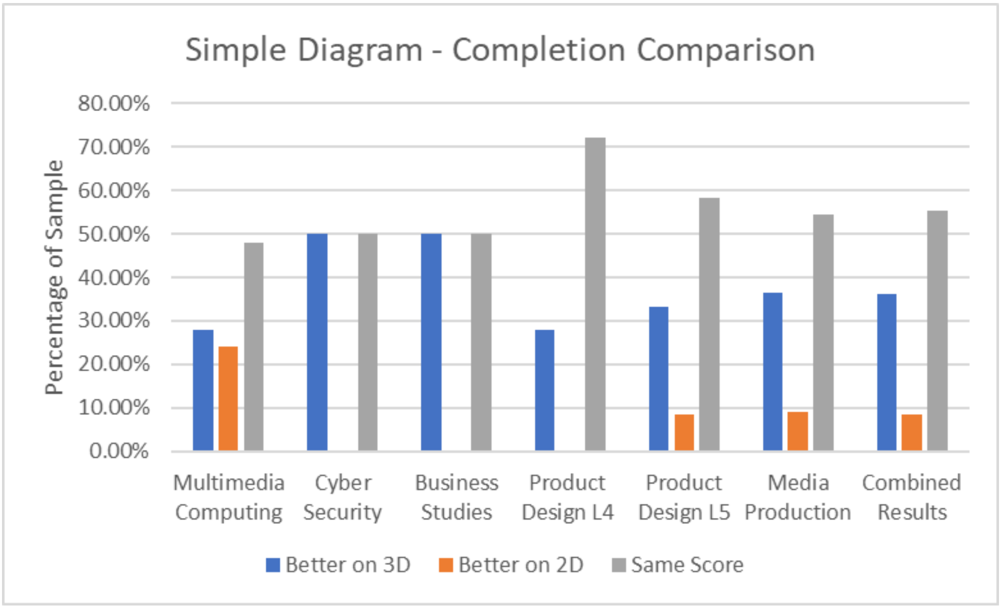


Figure 5.25 Simple diagram - completion comparison

When discussing the control experiment, it was concluded that in terms of completion and accuracy, 3D provided no benefit over 2D. However, viewing the results from the other experiments, this conclusion has been somewhat disproved when it comes to diagram completion. When combined with the results from the other experiments, the data reflects that 36.17% (35/94) of participants performed better on 3D, with just 8.51% (8/94) performing better on 2D. Viewing of the graph, one can see that for three of the experiments, no participants performed better on 2D. With emphasis on two of these groups being Cyber Security and Business Studies, both of whom represent potential end users.

Regarding the accuracy of the participants, Figure 5.26 details a comparison of who performed better with each approach.

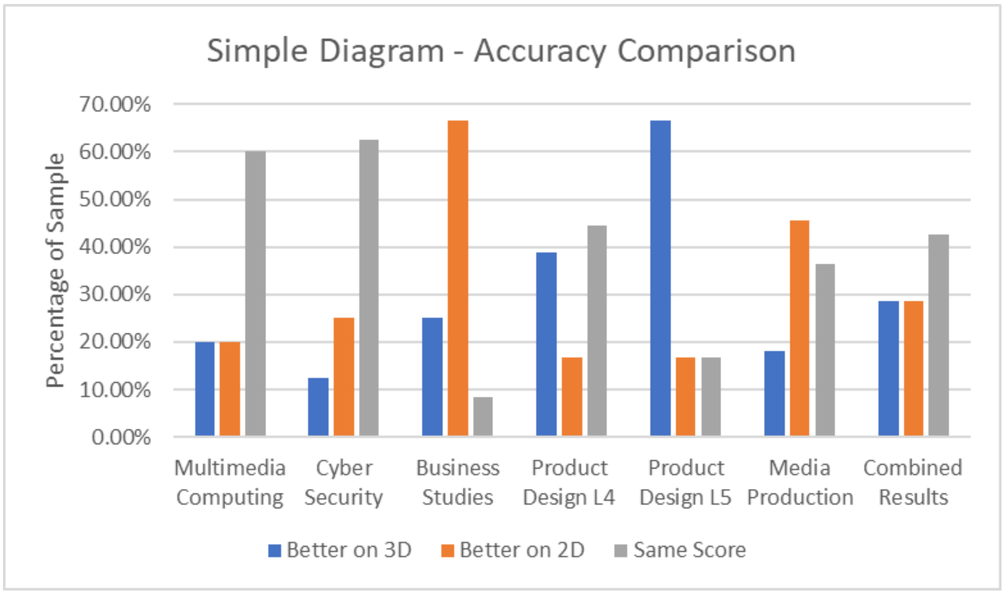


Figure 5.26 Simple diagram - accuracy comparison

Unlike the completion results, the accuracy results are very like the control. Although each group alone would produce different conclusions, when combined, the results come to the same conclusion as the control experiment. That is, in terms of accuracy, both 2D and 3D produce similar results with neither approach appearing the superior.

In terms of the average improvement when using the 3D approach compared to the 2D one, the following graph details each groups average improvement for both completion and accuracy for the simple diagrams.

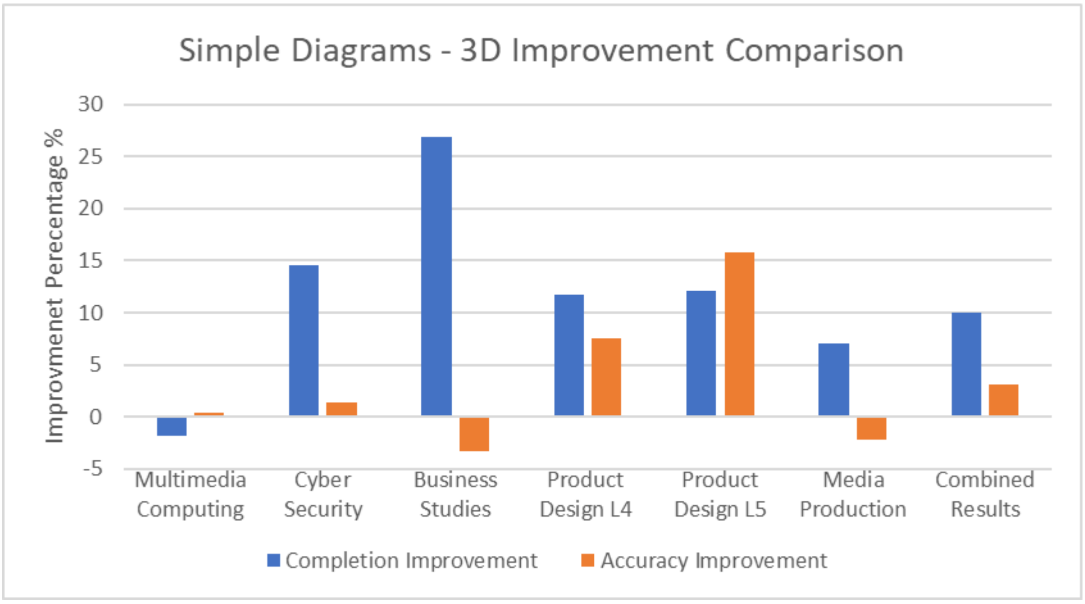


Figure 5.27 Simple diagrams - 3D improvement comparison

With the previous graphs showing the quantity of participants who performed better with each approach, this graph provides a more representative view of how well the participants performed. That is, should the results for any given value be positive, this would indicate that on average, participants improved by the respective amount when using the 3D approach compared to the 2D. Likewise, should a value be negative, this would indicate that on average, participants achieved the respective amount higher when using the 2D approach.

For the control experiments, the completion improvement for the simple diagrams came back as -1.78% ±9.04%, with an accuracy improvement of 0.41% ±5.57%. Hence the conclusion that both approaches performed equally, as each value is very close to 0%, which would indicate an identical performance.

Viewing the graph however, one can see that every other group performed significantly better on 3D regarding diagram completion. This result somewhat expected given the earlier data that 36.17% of all participants performed better on 3D. In terms of the average improvement across all participants, this comes to 10.05% ±5%. Therefore, although the control experiment deemed both approaches equal in terms of diagram completion, it can now be concluded with 95% confidence, that participants will be able to complete an average 10.05% ±5% more of the diagram when using the 3D approach compared to 2D.

As for the accuracy improvement, these results are again very like the control. With three groups performing better on 2D, and three groups performing better on 3D, the average accuracy improvement comes back as 3.14% ±3.8% across all experiments. Although this value is positive and would therefore indicate 3D will achieve a higher score on average, the result is very close to the 0% equal performance mark. Therefore, given that the experiment was graded out of 9, this percentage is too small to be of any significance, confirming the control experiment conclusion that 2D and 3D perform equally in terms of accuracy.

Nevertheless, in summary of the simple diagrams, further experimentation has shown that 3D performs slightly better in terms of diagram completion, with the original conclusion of equal accuracy performance, being strengthened.

As for the complex diagrams, the following graph represents a comparison of the percentage of participants who performed better with each approach in terms of diagram completion.

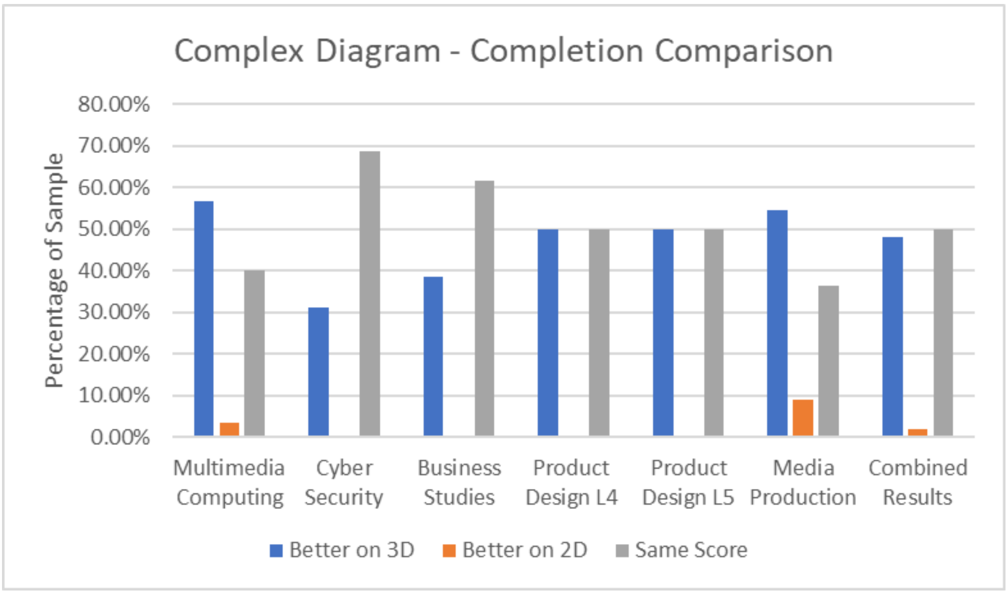


Figure 5.28 Complex diagram - completion comparison

As seen from the graph, the results for every experiment are almost identical to those in the control. Of the six groups, four of them had no participants who performed better on 2D, while they all showed a significant portion of participants performing better on 3D. Across all experiments, 48% of participants (48/100) performed better on 3D, with just 2% (2/100) performing better on 2D, and 50% (50/100) achieving an equal score. Comparing these results with the control experiment of 56.67% scoring better on 3D, 3.33% scoring better on 2D, and 40% achieving an equal score, the results are very similar. Although there has been a slight drop in the number of participants who scored better on 3D, the number who scored better on 2D remained relatively the same. Therefore, the results still reflect that 3D is a far superior approach when it comes to diagram completion.

As for the accuracy of the participants with each approach, the following graph represents a comparison of this data.

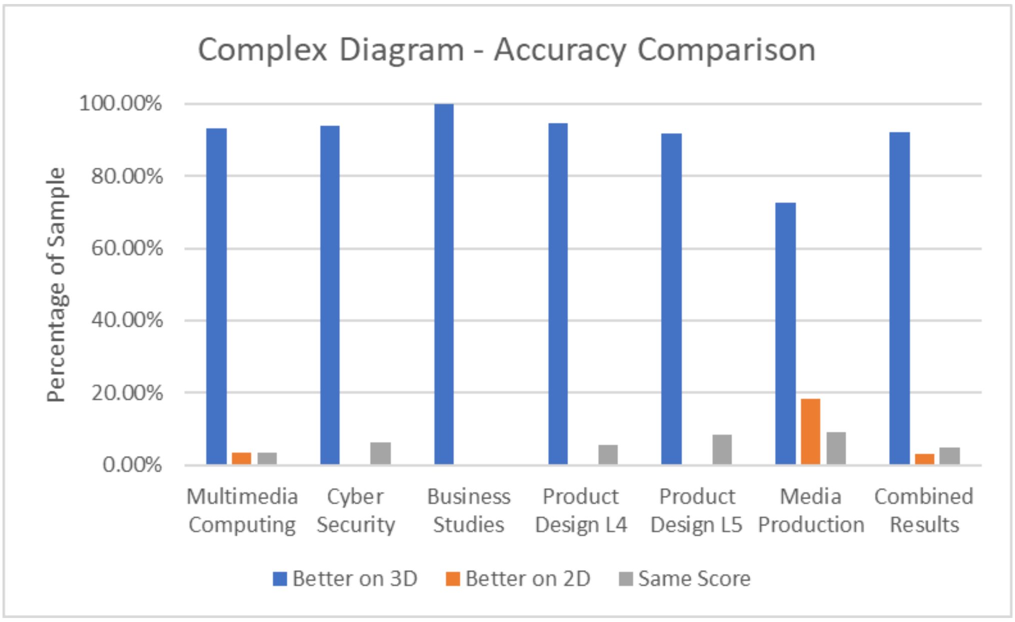


Figure 5.29 Complex diagram - accuracy comparison

This data is by far the strongest for supporting the proposed 3D approach over 2D. Although the completion comparison showed a large portion of participants who achieved better on 3D compared to 2D, there was still a large amount who scored equally on both approaches. However, in this instance, very few participants scored better on 2D, or equally on both. In fact, 92% of participants (92/100) across all experiments achieved a better accuracy with the 3D approach, with just 3% (3/100) performing better on 2D, and 5% (5/100) scoring the same. Given that the accuracy component of the experiment is measuring the user's ability to correctly identify a symbol and trace it to its respective element, which represents the *visual grammar* of the extension, these results are very positive for the proposed 3D solution.

Regarding the results from the original experiment, these showed that 93.33% of participants perform better on 3D, 3.33% perform better on 2D, and 3.33% perform equally. Comparing these to the results across all the experiments, it is safe to conclude that the original results came to the correct conclusion in this instance.

For the average improvement each participant made when using the 3D approach over 2D, refer to the graph in Figure 5.30.

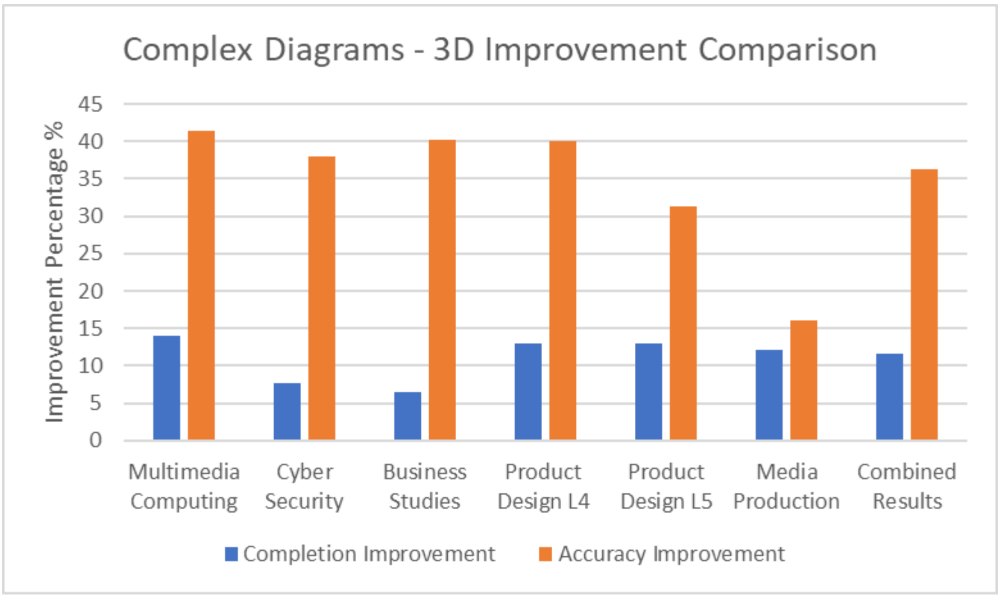


Figure 5.30 Complex diagrams - 3D improvement comparison

Viewing the graph, once again the results from the other groups support the initial experiment. Even without axes labels, one can instantly see that most groups have very similar values for both completion and accuracy improvement. The control experiment gave an average completion improvement of 14.07% ±5.32%, and an average accuracy improvement of 41.42% ±8.82%. The combination of all experiment results gave an average completion improvement of 11.67% ±2.86%, and an average accuracy improvement of 36.32% ±4.48%. Although the values have lowered slightly, the margin of error for each average has now halved, likely leading to the conclusion that the initial experiments were slightly out of range. Nevertheless, on the understanding that these experiments were graded out of 9, such a small percentage difference between the control and combined results is insignificant. The primary focus of this data is that it supports the initial findings from the control. When at high levels of complexity, 3D performs significantly better than 2D.

In summary, the 'grading' portion of the further experiments have generally supported the original conclusions. Although it can now be stated that 3D does provide some benefit over 2D in terms of diagram completion on simple diagrams, the remaining results are relatively unaltered. The accuracy of both approaches on simple diagrams are around the same. As for the complex diagrams, 3D can ensure a significant improvement in accuracy over 2D, as well as a fair amount more diagram completion.

Regarding the user feedback results from the further experiments, these will be summarised into which approach the participants found easier to identify symbols and their associated BPMN element for, and which approach of the two (2D and 3D) they preferred for both sets of diagrams. The remaining data for the other questions can be found in the Appendix. Starting firstly with the simple diagrams and the approach participants found easier to work with, the results of which can be seen in Figure 5.31.

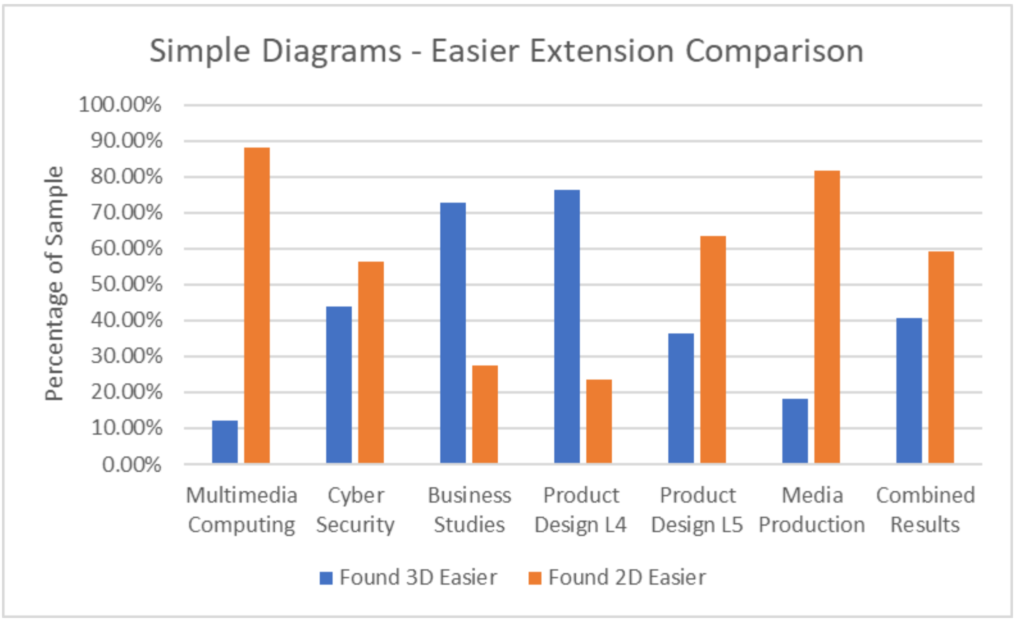


Figure 5.31 Simple diagram - easier extension comparison

The feedback for this question from other groups, is quite different than the results from the control. As seen, only Media Production reproduces such a drastic choice of 2D over 3D; two of the other groups have a similar relation of 3D over 2D. The combined results showed that 40.66% of participants (37/91) claimed 3D was easier, while 59.34% of participants (54/91) claimed 2D was easier. (Some participants failed to complete the respective portion of the questionnaire, hence 91 instead of 94 as seen earlier.) Nevertheless, even though the results still show that for simple diagrams participants find 2D the easier approach, the portion of participants who claim this, is not as large as the control experiment would have one believe.

Regarding the approach that participants preferred, the results can be seen in Figure 5.32.

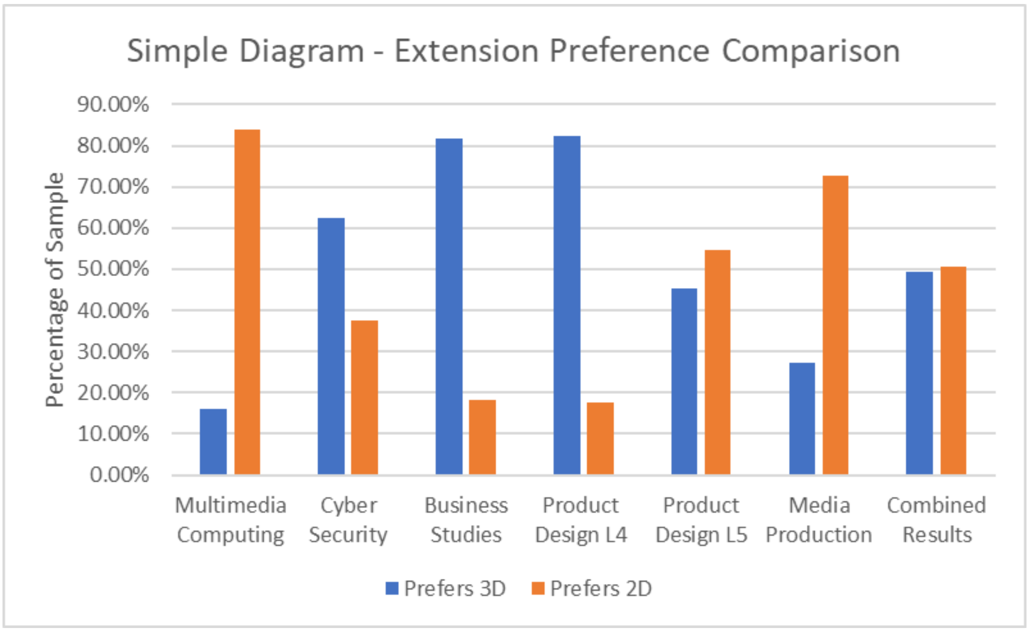


Figure 5.32 Simple diagram - extension preference comparison

The results from this question are very like their respective answers from the previous, an expected result as people typically prefer the easier option. The only group who gave somewhat different results are the Cyber Security students, although a majority found 2D easier, a majority also preferred 3D.

Moreover, the combined results from this question found that 49.45% of participants (45/91) prefer 3D, while 50.55% of participants (46/91) prefer 2D. Compared with the control data that showed 16% of participants (4/25) prefer 3D, and 84% of participants (21/25) prefer 2D, the combined results draw a very different conclusion. Further experimentation now leads one to believe that although a majority may have claimed 2D was easier, in terms of extension preference, each approach is equal.

The results for which approach participants found easier to work with for the complex diagrams can be seen in Figure 5.33.

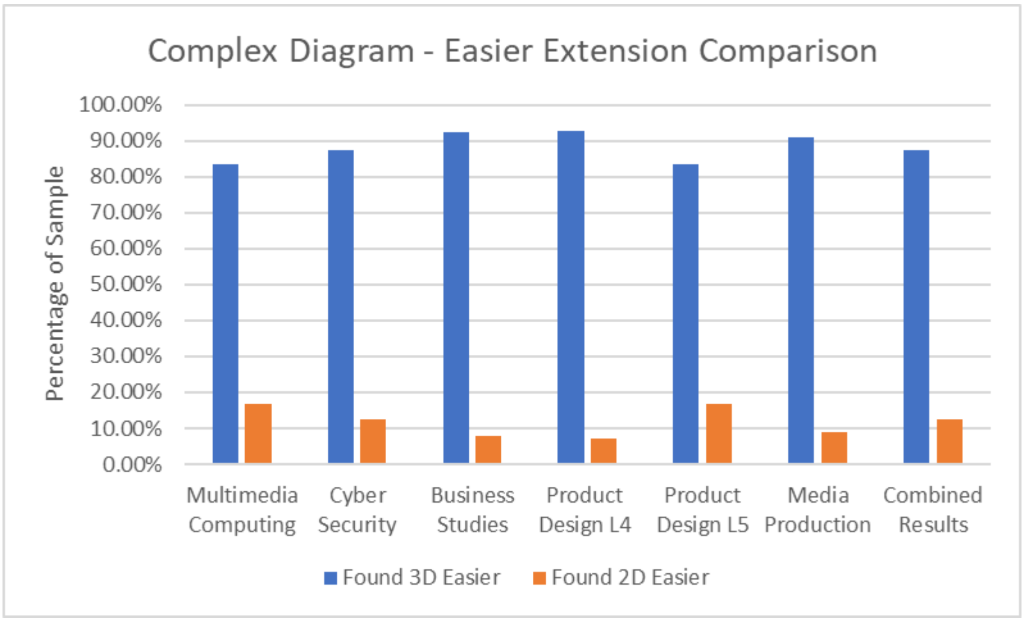


Figure 5.33 Complex diagram - easier extension comparison

Although the simple diagrams tend to give mixed results as each approach is relatively equal at that level, the complex diagrams have repeatedly shown consistency with the results from the control. As discussed multiple times, authors of current solutions appear to have developed their extensions with *construct deficit* to ensure low complexity across diagrams. However, when complexity is increased (or domain comprehensiveness/adequate expression is introduced) like in this experiment, the data speaks for itself regarding the inadequacies of existing approaches.

Viewing of the graph, one can see that the results from the control have again been only strengthened by the results of further experimentation. The control results found that 83.33% of participants (25/30) found 3D easier, while 16.67% of participants (5/30) claimed 2D was. The combination of results from all experiments show that 87.5% of participants (84/96) found 3D easier, with 12.5% of participants (12/96) claiming 2D is. A slight increase in the preference of 3D but otherwise an almost identical ratio of participant opinion on the ease of completing the task.

Finally, the results from which of the two approaches participants preferred for the complex diagrams, can be seen in Figure 5.34.

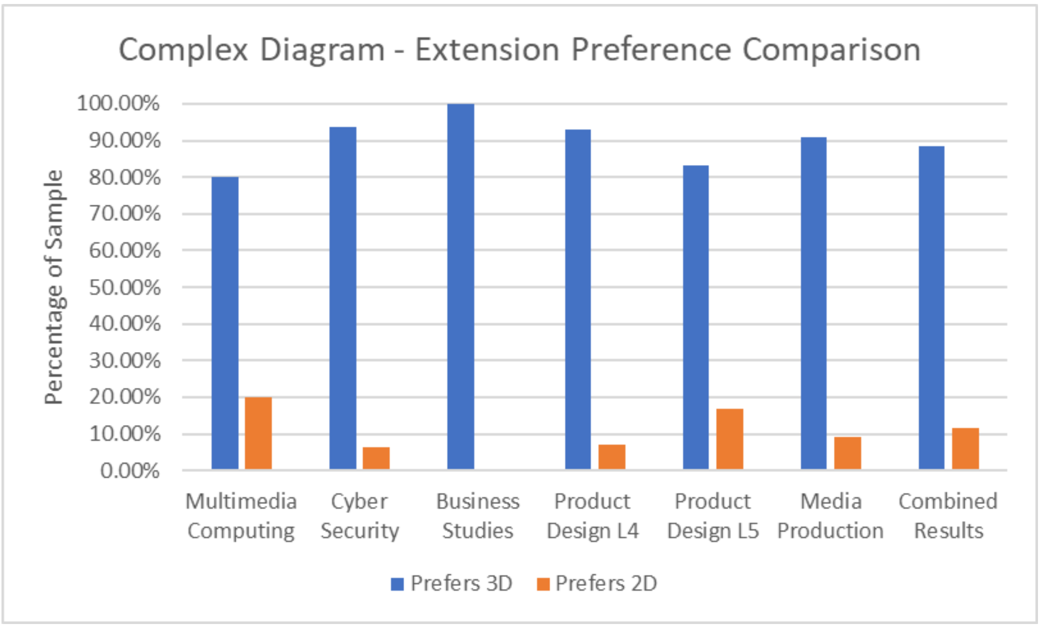


Figure 5.34 Complex diagram - extension preference comparison

Again, the results further support those from the control, with all groups giving a similar ratio of 3D preference over 2D. The most noteworthy group being Business Studies in which all participants claimed they prefer 3D. During the control, 80% of participants (24/30) stated they preferred 3D, with 20% of participants (6/30) stating 2D. For the combined results of all experiments, the data showed that 88.54% of participants (85/96) prefer 3D, with 11.46% of participants (11/96) preferring 2D. An almost 10% transfer of 2D preference to 3D preference compared to the control.

To summarise, further experimentation has shown that although most participants still find 2D easier on lower complexity diagrams, the margin of difference between each approach is much lower compared to the control results. Likewise, although the initial experiment found that nearly all participants preferred 2D on the simple diagrams, further experimentation showed this split to be almost 50:50, strengthening the support for 3D.

For the complex diagrams, further experimentation only strengthened the results from the control. An almost identical ratio of participants found 3D easier across all experiment groups compared with the control. Likewise, there was a 10% increase from already very strong data that most participants prefer 3D over 2D at high complexity.

Upon completion of these experiments several conclusions can be drawn. The previously proposed theory was that current authors have opted for *construct deficit* due to their inability to achieve a *complexity managed* extension. Viewing the results from the experiment, this is a very plausible argument to make, given that on low complexity diagrams current 2D extensions appear capable of *complexity management*. However, when complexity is increased, current approaches are inadequate. There is also the potential that because current authors have only used a low number of constructs in their notation, they have never realised the issue with their visual grammar. Regardless however, one thing the experiments prove, is that current 2D approaches are insufficient at achieving *complexity management* at high *diagrammatic complexities*.

On low complex diagrams, the data showed that the proposed 3D visualisation provided little advantage to current 2D approaches, with most participants stating they found the current 2D method easier. However, when complexity increased, the data showed that the 3D approach can achieve substantially better *complexity management* than 2D.

As stated, Moody provides no way of definitively confirming *complexity management* is achieved or not; using the concepts of *accuracy* and *completion* however, a quantifiable means of gauging its quality was proposed. With the 3D complex diagram results showing an average completion improvement of 11.76% ±2.86% and an accuracy improvement of 36.32% ±4.48% over 2D, this thesis is confident in stating that the proposed 3D visualisation is more than capable of adequate *complexity management,* even at high levels of concept count. This is further emphasised by the participants visualisation preference after the complex diagrams when compared to the simple diagrams.

A published version of these experiments can be found under the following reference (Maines, et al., 31st August - 2nd September, 2016). A table of results for the experiments can also be found in the Appendix.

5.2.2.2.4 Proposed Visual Grammar

Although the experiments state that “complex” diagrams were used, referring to the cyber security ontology and the list of modellable concepts specified, 37 concepts on a diagram does not truly reflect the level of complexity that a language which has 76 unique concepts will be attaining. Nevertheless, that is not to say the experiments have been a useless endeavour, they have still proven current 2D approaches insufficient and showcased the potential of 3D. Therefore, for the visual grammar of the BPMN security extension, the work will build on from this experiment. That is, each BPMN element will have its own unique holder[20] capable of specifying all security requirements should an end user so require.

Representing 76 concepts at once on a single BPMN element however, will almost always cause *cognitive overload* and incur several complexity issues on its own. Therefore, the current approach needs expanding to allow for the *complexity management* of concepts on a single element, not just the entire diagram. In this instance, the thesis once again refers to the PoN and Moody's concepts of *modularisation* and *hierarchy* structuring. Although in the PoN, Moody discusses using these concepts for diagrammatic *complexity management,* there is no reason the same concepts cannot be applied at this level.

For the visual grammar, the aim is to display six concepts at the highest level on each BPMN element (Figure 5.35a), respective to the six key concepts in the ontology; adhering to the seven, plus or minus two cognitive rule (Tufte, March, 1956). These symbols will then act as individual buttons to modularise their sub-concepts, once a symbol is selected the remaining five will collapse and the next level of concepts will display (Figure 5.35b). This functionality will then continue for the lower levels (Figure 5.35c-d). However, instead of collapsing the other symbols at lower levels, they will be hidden, this is to ensure complexity is still managed. Once collapsed the symbols become unidentifiable anyway, to hide them after the top level will ensure *cognitive overload* does not ensue.

[20] *A holder is the name assigned to the core relationship between BPMN elements and security requirements. Referring to any figure featuring the proposed visualisation, a holder is the grey line behind all the security requirements connected to the centre of the associated BPMN element.*

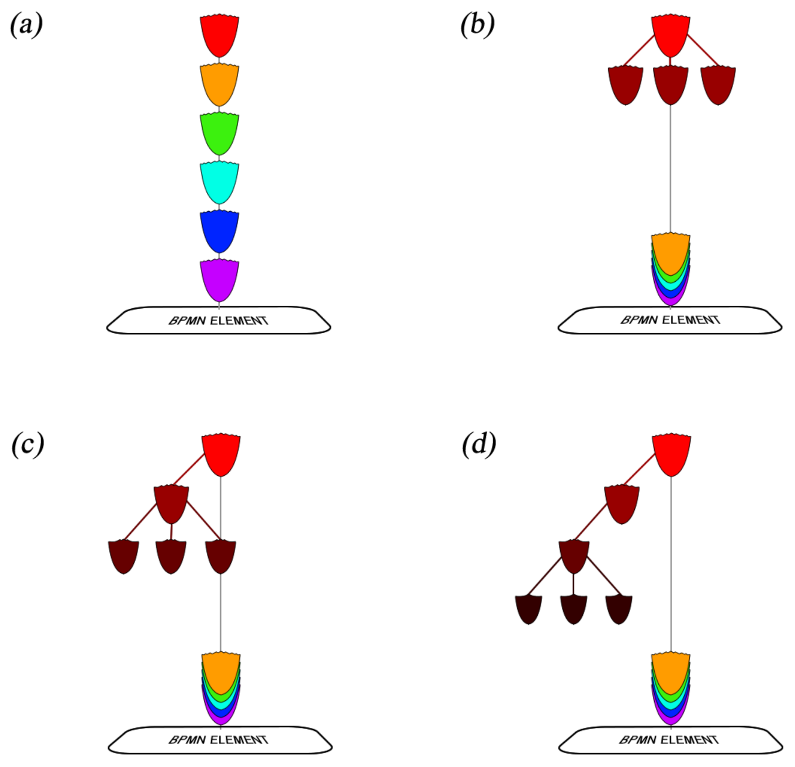


Figure 5.35 Proposed visual grammar

As seen, although Moody does not propose to use *modularisation* and *hierarchy* in this way, they are still very capable in aiding *complexity management*, allowing for the cognitively manageable specification of 76 concepts on a single element. (Although, it is highly unlikely there would ever be a situation in which one element requires all 76 concepts, especially as some negate others.)

Within the proposed visual grammar*,* to further iterate the concept hierarchy on top of brightness*,* shape and size another visual variable is included: vertical position. Once a symbol is selected, the sub-concepts display at a decreased size respectively to the lowest level (Figure 5.35d). They also appear below the parent symbol giving the impression of a tree structure and that a lower vertical position indicates a lower concept level.

When no symbol is selected, the six key concepts are displayed vertically as well. To ensure the user does not infer a similar hierarchy, new relationships (lines) are used to connect parent and child concepts (Figure 5.35). Along with the fact, the core six concepts also have different colours, outer shapes, size difference, and are typically horizontally aligned unlike child concepts, this is an unlikely issue to arise.

Along with vertical position*,* this work also makes use of horizontal position within the visual grammar. Unlike current extensions (and modelling languages), this extension always places constructs in the exact same position relative to their associated BPMN element, explicitly. For example, numbering each of the red symbols in Figure 5.35d, one to six respectively, symbol six (bottom right) will always appear in the same position irrespective of whether symbols four and five are specified. This effectively means that, even if every symbol looked identical, as horizontal and vertical positions have been used to encode a symbol's semantics, a user could still infer the construct from this alone.

This also means that, between the vocabulary and grammar of the extension, a total of seven of the eight visual variables are utilised, dismissing only texture. Thereby making this solution one of the first to explore and utilise the full toolset at a modelling language's disposal in terms of visual expressivity. The proposed method of visualisation also ensures the satisfaction of the *structure* requirement. With little explanation, observation of Figure 5.35 demonstrates how both a novice and expert can utilise the extension. Should an end user so choose, the novice can utilise six high level concepts, more than capable of providing basic security requirements across a diagram (similar to the approach of existing extensions). As the users' knowledge and domain expertise increases, they can then expand these concepts to specify lower levels of detail.

Although the grammar is stated as satisfying this requirement, in truth, the solution was developed based on this requirement. The analysis of existing extensions beforehand and the extracted requirement specification, provided a very strong foundation and justification when making such design choices.

Moreover, although the visual grammar for the language has been outlined, there is yet any details regarding the specification of specific instances of each concept. As stated earlier, there is no intention of representing these concepts visually, there is only so much a visual grammar can manage and the complexity will be unmanageable with too high a graphic complexity. Nevertheless, there must still be a way to specify this detail. Therefore, using current practice on such functionality, a simple form-GUI toolbar will be located somewhere on the screen to define such details. Although the ease and means with which this detail can be specified is important, there is always a point in every modelling language where the user must refer to text, to specify or read the level of detail required. In this instance, the plan is to have this details bar across the bottom of the user interface as seen in Figure 5.36.

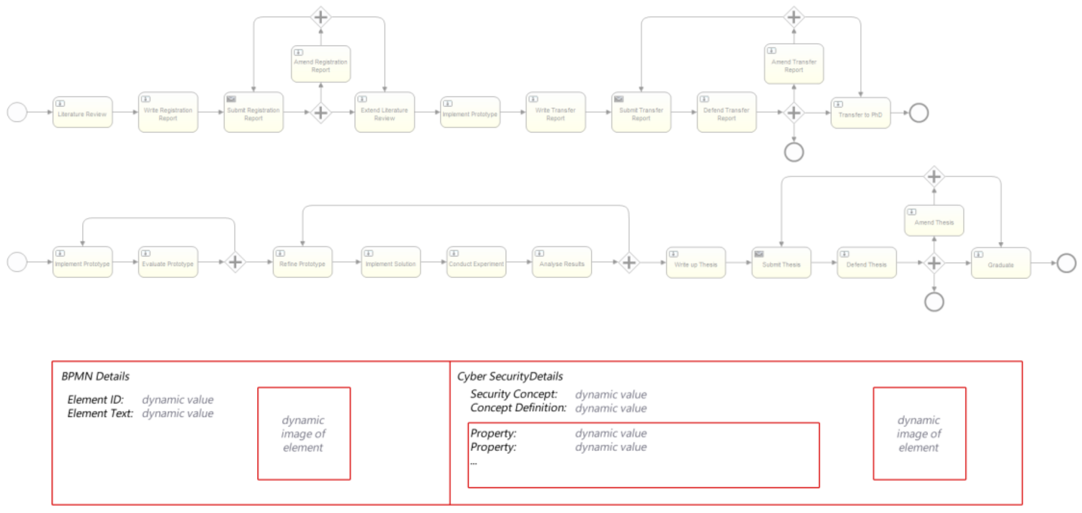


Figure 5.36 Visual vocabulary - details toolbar

Although this touches on the user interface component of the *application developer's* role, given that this functionality is directly linked to the specification of language concepts, it is declared here.

The toolbar is divided into two sections: the left being for BPMN and the right for cyber security details. In each section, the currently highlighted element is specified, which is the corresponding text value in the case of BPMN and the concept definition in the case of security requirements. Each section also features an enlarged image of the currently highlighted element for user reference. Inside the cyber security details side, is where the currently highlighted concept will have any properties specified. These will be listed below the concept definition allowing the end user to define each component as required.

Overall, the proposed grammar allows for the visual representation of 76 security requirements and the low-level specification of over 60 properties, a vast improvement over current approaches.

5.2.3 Security Policy Framework

5.2.3.1 Specification Restrictions

The specification restrictions as mentioned, focuses specifically on input validation to minimise potential human error. This is aimed more at the properties of each security requirement as opposed to the visual specification themselves. To highlight a few of these in the extension:

- Minimum levels of classification cannot be greater than maximum levels (vice-versa)
- When applicable, only numerical values can be entered in a text box
- When specifying further elements in *binding/separation of duty*, already specified elements are not allowed
- …and so on

As seen, this component represents a somewhat expected process of robust software development. As such, not every instance of validation will be specified, as it provides little contribution to the thesis, it is included again only to reiterate the benefits a basic system such as this can bring at run time.

5.2.3.2 Verification Rules

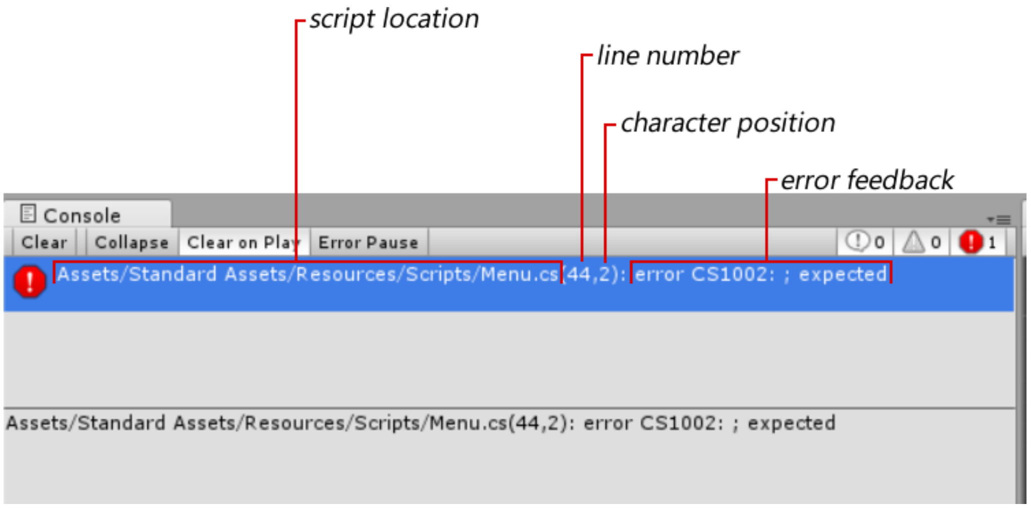
As for verification rules, these represent the method in which policy compliance will be assessed. Policy compliance is an important process in cyber security and represents a significant contribution itself. However, in this case, a very straightforward policy verifier is proposed. That is, the end user will simply specify what security is required on each BPMN element using only a textual form-input approach. This will then later be compared against the diagrammatic security requirements.

Although this is not typically the way a security policy is detailed, a policy compliance solution does not represent one of the core objectives of this work; the main focus here is on the matter in which an end user is made aware of a policy discrepancy and their ability to quickly fix or acknowledge these warnings.

Nevertheless, this approach still allows for existing policies to be translated into the application. Although the policy verifier is diagram based and this process would be required each time, the functionality is still there should any other format of policy require specification. As mentioned though, the focus is mainly on the feedback element as opposed to the method of checking. This is primarily because a robust, dynamic policy compliance verifier could potentially match the scale of this entire project if not more. As such, the expectations of this element are somewhat restricted in this work to ensure focus remains on the extension and visualisation of security requirements in modelling languages.

5.2.3.3 *Feedback Technique*

Moreover, the feedback technique component represents the core focus in regards to policy compliance. As previously mentioned, current approaches to user feedback in regards to warnings or errors use a programming error console, like that in Figure 5.37.



```
Assets/Standard Assets/Resources/Scripts/Menu.cs(44,2): error CS1002: ; expected

Assets/Standard Assets/Resources/Scripts/Menu.cs(44,2): error CS1002: ; expected
```

Figure 5.37 Programming error console (Unity game engine)

As seen, this console provides all the information necessary to locate an error, as well some indication as to what the cause of the error may be (the cause description being a case by case basis with regards to its usefulness). Usually, double clicking these errors will open the specific script and highlight the respective line in which the error is occurring. However, this method is aimed at errors in text. Given that this work deals with a more visual application, some visual feedback should also be included.

Therefore, for the feedback technique, the aim is to include a combination of both current error consoles and visual/symbolic representation. The choice of symbol to indicate a discrepancy in this case is a classic warning icon like that in Figure 5.38.

Figure 5.38 Warning icon

To notify the user, this symbol will be placed in front of the nearest respective symbol or BPMN element, and animated slightly in a pulsing motion. For example, referring to Figure 5.39a, the error icon can be seen over the red security requirement. If this concept is then expanded to show its children (Figure 5.39b), the warning icon has now moved to a lower level concept. If this is continued throughout the hierarchy, one will discover that the respective requirement to which the warning refers is that seen in Figure 5.39d (this of course will be more visible within the application, as in this instance the symbols are much smaller than their usable state and the warning icon is static).

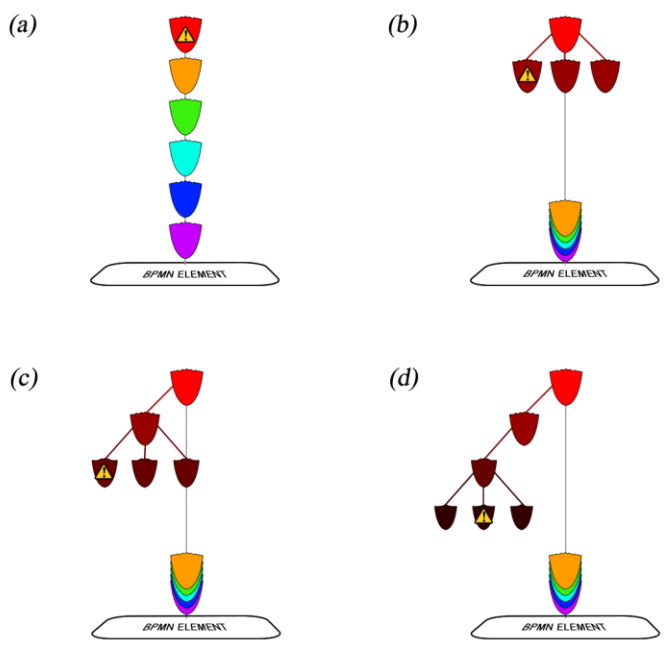


Figure 5.39 Warning icon example

Although an adequate means of locating the discrepancy, the user does not have much detail regarding what the discrepancy is. Therefore, this is where existing error console functionality can be utilised. Along with the warning icon, this work proposes the inclusion of an error console to further inform the user of what the discrepancy is between the diagram and policy. This should only be visible however when the user requests it, there is no need to over populate the GUI with unnecessary functionality. This console will be discussed more in a later section.

6 BPMN Case Study - Application Developer

As BPMN is a graphically complex modelling language, the implementation of a new engine is a rather extensive process. Given that the aim is not for an industry-ready software in this instance and just the evaluation of the thesis framework, a pre-existing tool for creating the business process side of diagram was used. In this instance, that tool was the Eclipse plugin Activiti[21]. Not only is this tool open source, but the output file is formatted in eXtensible Markup Language (XML), which has become a standard for semi-structured data representation (Tekli, 2016). The benefits of this being, the file and thereby integration into

[21] Alfresco Software, Inc.; Activiti:Home; (https://www.activiti.org/) accessed 26/04/2017

another tool, is relatively straightforward due to XML being a widely-supported middleware.

As for the cyber security visualisation side of the application, when it comes to interactive digitally generated real-time 3D scenes projected onto 2D displays, game technology provides a range of technologies that support and simplify the development of interactive visualisation. Technical advancements in game technology include rendering, realistic physics, lighting, audio, graphical user interfaces (GUI), head-up displays (HUD), inputs and scripting (Maines & Tang, 25th - 27th August, 2014). Therefore, for the development of the solution and the requirements of the *application developer*, the logical choice is to use a game engine as the foundation software for the tool. They already include libraries for several components of the framework, not to mention various file serializers to assist the importation of the BPMN Activiti file.

Most engines available offer relatively similar functionality, the main differences being licensing costs and final render quality. If making a realistic first person shooter (FPS) for example, one of the best engines would be Unreal 4 (Gregory, 2014). However, taking into consideration what is required from an engine, the choice of engine for this work was Unity[22]. This is mainly due to Unity's functionality of being able to port to most devices (Gregory, 2014). Although there is no immediate goal of developing on a platform beyond PC, this functionality is the most appealing in the long term when compared to what other engines offered. Moreover, of these devices, Unity is always one of the first engines to include support for virtual/augmented reality headsets. Although this is not a priority at this stage of the work, it again contributes to the decision, given the possibility of an AR solution in the future works. Unity is also free to download, making the decision a relatively simple one.

As mentioned, the plan is to use two applications to create a BPMN diagram and include security, one for each domain. In an industry environment, using one application for both is without a doubt better practice. However, given that the thesis is merely testing the language at this stage, to speed up development it is better to use two applications. An overview of how this workflow between the two applications will work can be seen in Figure 6.1.

[22] Unity; Unity:Home; (https://unity3d.com/) accessed 26/04/2017

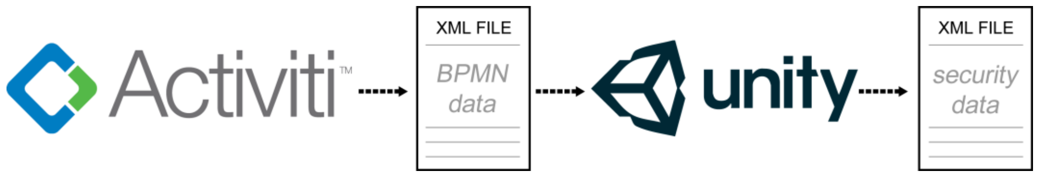


Figure 6.1 Application workflow

Although a crude overview, this figure provides a visual representation of:

1. Creating a BPMN diagram in Activiti.
2. Saving an XML file of the diagram
3. Reading the diagram into Unity.
4. Adding security requirements to the diagram in Unity.
5. Saving an XML file of the security requirements.

6.1 File Serialization

6.1.1 Core Language – Serialization

The Activiti XML file contains all the necessary data to redraw its respective BPMN diagram within a game engine environment, each BPMN element is represented by a node within the file, specifying the following details: *element type, unique ID, height, width, x* and *y*. An example of such a node can be seen in Figure 6.2.

```
<bpmndi:BPMNShape bpmnElement="usertask1" id="BPMNShape_usertask1">
      <omgdc:Bounds height="55.0" width="105.0" x="340.0" y="230.0"></omgdc:Bounds>
</bpmndi:BPMNShape>
```

Figure 6.2 Activiti XML - BPMN node data

As seen from this code snippet, no further work is required on this data. The Unity game engine already has built-in libraries for the serialization of XML files, and the data within the Activiti file itself is already enough to allow for the drawing of its respective diagram.

Many will argue that that the security requirements of a BPMN diagram should be encompassed within the same save file as the BPMN elements themselves. However, this has several drawbacks. In this instance, the use of multiple save files can speed up the development process. The inclusion of security in Activiti's save file would require a lot of time to be invested ensuring schema satisfaction. However, using two separate files allows for the mock-up a new schema targeting just security requirements in less time.

Furthermore, by separating the two files, end users have the ability to create multiple variations of security requirements across the same diagram. There may be a scenario in which a company is deciding between two different security standards, each requiring their own different requirements. By splitting the domains, the same BPMN save file can be used as a foundation for displaying the two different standards' requirements. Using an approach which edits the original save file to include security, consequently means the user would have to ensure they make a clean copy of the file at the beginning, or delete one set of requirements to show the other.

Regardless of this reasoning, the way security requirements are stored is insignificant to the framework and the evaluation of it. This is purely a case-by-case preference, in this instance it was decided to have two separate files.

6.1.2 Security Language – Serialization

Unlike the core language, the security language does not yet exist. As such, no XML file/schema exists which can simply be used within the application. Although OMG offer some documentation for extending existing BPMN save files (Object Management Group, Inc., January, 2011), as previously discussed, this work opted for separate save files. Therefore, new XML schemas must be created which can be used for the saving of diagram security data. As discussed, several times already, the concepts within the security ontology are structured. For the creation of an XML schema, this simplifies the process somewhat as there is already a meaningful way to store the data. Nevertheless, this data alone is not yet sufficient for the re-loading of diagram data.

Referring to Figure 6.2 again, there is a lot of data regarding element size and position, not to mention a unique identifier. However, the security ontology does not feature such data. Therefore, saving this data as it is, there will be no way of determining the location of a security element within a diagram, or the BPMN element it is targeted at.

Fortunately, due to the proposed visual grammar, the only additional data required for the schema is the respective BPMN element. As previously stated, the location of each security requirement is always the exact same in regard to its holder, just as a holder is always placed in the same location on a BPMN element. Therefore, there is no need to store any positional data within the security file, so long as the respective BPMN element ID is known, its position can be queried and the location of the holder and security element positions can be calculated from this.

An example of a node in an XML file derived from the proposed schema can be seen in Figure 6.3. This portion of the node only features *access control* requirements. As seen, the amount of data required is quite extensive and can quickly overwhelm the thesis should the entire ontology worth of concepts be included. Nevertheless, this portion fairly represents the schema and showcases the little data required in terms of the related BPMN element.

```
<bpmnElement elementID="pool1">
        <accesscontrol required="false">
                <authentication required="false">
                        <persauthentication required="false">
                                <credentials required="false" usernameRequired="false"
                                passwordRequired="false" pinRequired="false"
                                passwordChangePeriod="0"/>
                                <smartcard required="false" contactless="false"
                                pinRequired_SC="false"/>
                                <biometric required="false" biometricType="N/A"/>
                        </persauthentication>
                        <networkauthentication required="false">
                                <cryptprotocol required="false" protocol="N/A"/>
                                <vpn required="false"/>
                        </networkauthentication>
                </authentication>
                <identification required="false">
                        <trustlevel required="false" minimumLevel="0"/>
                </identification>
                <authorisation required="false">
                        <assetclassification required="false">
                                <serviceclassification required="false"
                                serviceLevel="0"/>
                                <dataclassification required="false" dataLevel="0"/>
                        </assetclassification>
                        <statetransition required="false">
                                <bibamodel required="false"/>
                                <belllapadula required="false"/>
                        </statetransition>
                </authorisation>
        </accesscontrol>
...
```

Figure 6.3 Cyber security XML - access control

6.1.3 Security Policy – Serialization

Fortunately, due to the proposed method of policy compliance, the exact same schema file for the security language can be used to store the policy data. As the policies are based on the BPMN diagram itself, the data required is identical to that of the security language. Usually one would have to remove unnecessary data such as location attributes. However, as mentioned, the visual vocabulary has streamlined what data is required and therefore made it so the security language and policy can both use identical schemas. This not only aids the development process, it also allows for a very straightforward comparison of XML files when assessing policy compliance.

6.2 Rendering

Although game engines already have the functionality to handle the rendering of assets, there are still a lot of resources required before this functionality can be utilised. As such, it is appropriate to display each element (BPMN and security) as individual frameworks consisting of all the necessary components for their construction.

6.2.1 Core Language – Diagram Generator

Firstly, the framework for the core language, or in this case BPMN, is shown in Figure 6.4. One will notice when first viewing the framework that some elements are coloured in grey, these represent abstract objects with no physical presence within the application. As for the rest of the framework, these represent physical components of each BPMN element. The only exception to this rule is the most outer shell titled *BPMN element*, this represents a *Class* within the tool engine, the remainder of the components are outlined below.

The first two properties are the *element type* and *unique ID*. The *element type* is used by the system to determine what visual representation the BPMN element will have in the diagram. The *unique ID* is a self-explanatory string variable.

Using the *element type* (obtained from the XML file)*,* the system can call a *prefab* of the respective construct. A *prefab* (Unity 3D, n.d.)*,* is an element of the Unity game engine that represents a prefabricated *GameObject* (Unity 3D, n.d.). They can also be understood as a saved *GameObject*, which can be instantiated multiple times with identical properties; in coding terms, these are like a *Class* object. Once instantiated they will be identical, however, the component properties can be altered to create varying instances. In this case, *prefabs* are very useful. BPMN only has so many elements, all of which have very similar properties. Therefore, *prefabs* can be used to create high level templates of each element, for example, *tasks, start events, parallel gateways* etc. These *prefabs* can then be instantiated based on the XML file, changing the properties as necessary.

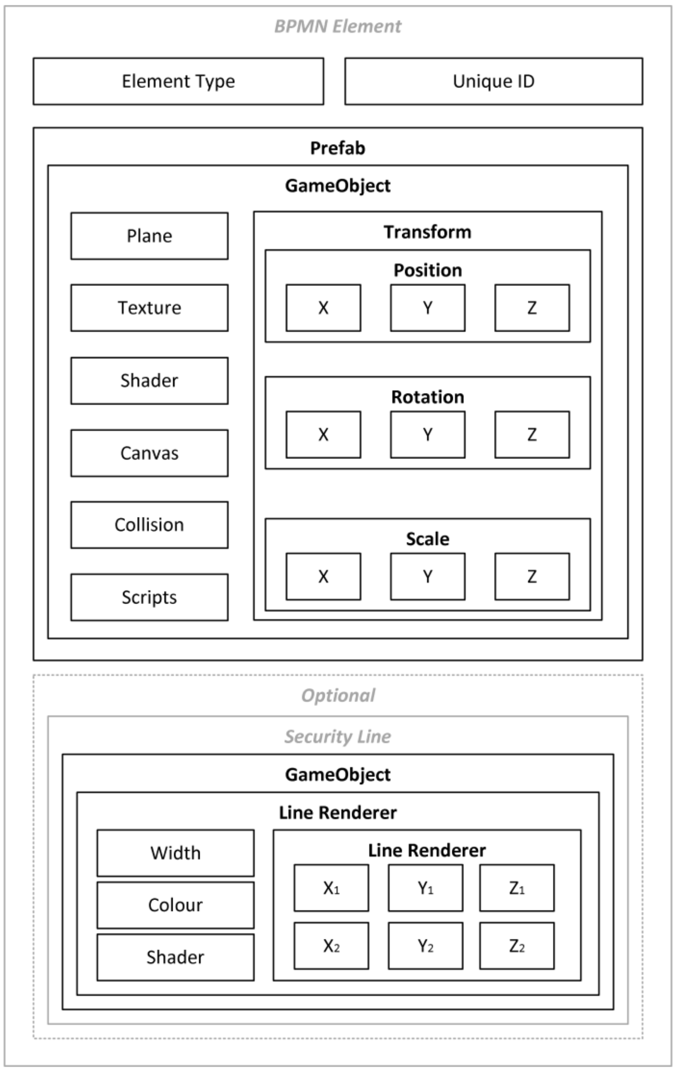


Figure 6.4 BPMN element framework

A *GameObject* is like a *prefab,* but not the same. In Unity, a *GameObject* acts a container for other components, these can be anything from main characters to sound effects. The key difference between a *GameObject* and *prefab,* is the method in which they are stored. *GameObjects* only exist at run time within the scene. Whereas, a *prefab* is stored in the application assets folder and can be viewed outside of the game engine in a file explorer. Nevertheless, the two are very closely related, a *prefab* can be created from a *GameObject* and vice versa. However, a *prefab* cannot be created without a *GameObject*, a *GameObject* can exist without a *prefab*. In this instance, pre-created *GameObjects* are being stored as *prefabs,* so they can later be instantiated upon request by the application.

Regardless of what BPMN element is requested by the application, there are several components that always need to be rendered, hence defining them in a framework. The first component of the *GameObject* is a *plane* object (one could also use Unity's own assets here, this would be a *quad*[23], however the two are identical with poly count being the only main difference). A *plane* is a flat 3D model, one which usually only renders textures on one side, these are best thought of as a 2D shape in 3D space. Although 3D is utilised within the visual grammar, it is only the 3D space that is being used, not 3D depth of objects. Therefore, to keep the poly count down and by consequence the processing power required by the application, a *plane* object is best the choice for each element. A *plane* can be created using Unity's primitive object assets or imported from an external 3D modelling software such as 3DS Max or Maya. Irrespective of which is used, all methods will provide an identical output. For the BPMN elements, a very basic plane consisting of two triangles is sufficient.

The next component is the *texture* for the *plane.* The *language architect* deals with most of the difficulties as far as this component is concerned, the *application developer* need only convert the *architect's* designs into usable textures should they not already be in this state.

After the *texture* has been added to the *plane*, the *application developer* will need to assign a *shader/material*[24] to it. A *shader* defines how the *plane* will display its *texture*. There are several preset *shaders* in Unity to assist with this process as well as the ability to create custom ones. As the application has used a *plane* for the 3D object to keep the poly count as low as possible, *shaders* will have to be utilised to remove any unnecessary background from the textures. As the BPMN notation (and the security notation), almost never have a perfectly quadrilateral shape, a *plane* object will be unable to do this without the assistance of a *shader*. In this case, existing *shaders* can be used for each *GameObject*, these being a combination of *diffuse and transparent cutout*.

After this, the focus now is on the *transform* component of the *prefab*. Although this is not a component the *application developer* will need to add, as it is a sibling to *plane* and added automatically with it, it is still of significant importance to the BPMN element, as this is where the diagram is effectively drawn. Taking the data from the XML file (*height, width, x* and *y*), these values can be translated into the *transform* component to determine where

[23] Unity; Unity:Manual:Primitive Objects; (https://docs.unity3d.com/Manual/PrimitiveObjects.html) accessed 27/04/2017

[24] Unity; Unity:Manual:Material"; (https://docs.unity3d.com/Manual/class-Material.html) accessed 27/04/2017

in the scene the BPMN element should be drawn. However, this is not a straight mapping of coordinates.

As the BPMN diagram is being converted into 3D space, the coordinate system used in Activiti will no longer be correct. A typical 2D graphic system uses the *x* and *y* axes, varying its origin from either top-left or bottom-left (usually top-left) from each application. In game engines and other 3D applications, it is very similar, the main difference is the introduction of a third *z* axis. This effectively comes out from the screen when comparing it to the current 2D system. The origin point is changed too, rather than placing it in the top-left, it is effectively moved to the bottom left and acts as a 'ground level'. Therefore, to adhere to the visual vocabulary, and save a lot of awkward rotation later, it is best to remap the current coordinate system to 3D space. This is better understood by viewing Figure 6.5.

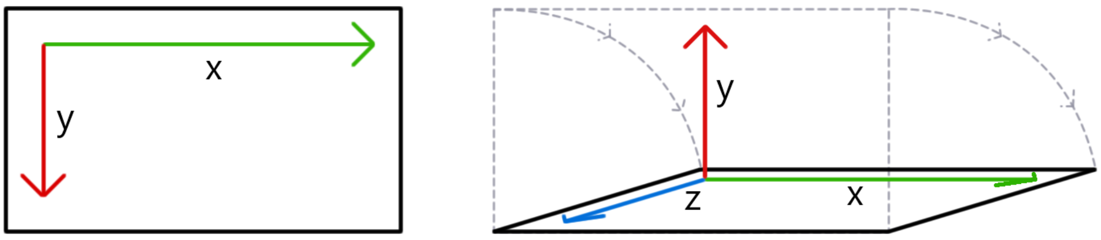
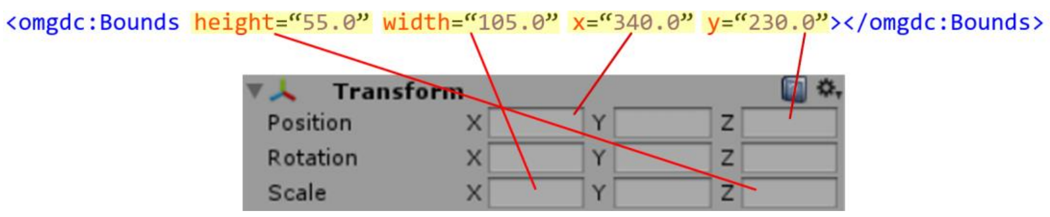


Figure 6.5 2D/3D space conversion

It is worth noting, this figure does not represent the typical conversion of 2D space to 3D space. This figure is being used to represent how this work maps the coordinates from the Activiti XML file into the 3D space in the application. As seen on the left, the current system uses a top-left origin across the *x* and *y* axes. By converting these to *x* and *z* instead, the coordinates will still render an identical diagram. It also makes the incorporation of the security notation easier, as the default viewport will be identical to the visual grammar. The mapping of the coordinates in the XML file to the respective *transform* components can be seen in Figure 6.6.



```
<omgdc:Bounds height="55.0" width="105.0" x="340.0" y="230.0"></omgdc:Bounds>
```

Figure 6.6 XML data mapping

The remaining *transform* values for *position and scale* will be defaulted to *1*. As for the *rotation* values, these will all be 0 except for *x* which will be *90*. This is to ensure the *plane* is at the correct orientation across the *x* and *z* axes (this can also be done in an external modelling software, meaning the imported *plane* could have an *x rotation* of *0* but still render the same in the application).

The next component to include in the *GameObject* is a *canvas*[25]. A *canvas* is typically used for UI content, but can also be used in 3D space as a way of displaying dynamic text content. Given that a lot of the BPMN elements will typically have dynamic text (*task* elements for example), this is a necessary component to include within the *prefab*. *Canvas* objects do include their own set of world space transforms. However, as it is a child of the BPMN element transforms, its position will always be relative to this and as such not require any editing past development.

The next components, although not necessary for the physical rendering of the elements, are required to ensure their interactivity at run time. Therefore, as the construction of the elements and their components are being covered within this section, their inclusion at this stage is justified. A *collision*[26] component is added to ensure the end user can interact with the element. Without *collision,* although the *GameObject* will still exist in the scene, there will be no listener to check for user interaction, Unity offers this functionality as standard built into the engine, requiring little implementation from the *application developer*.

The final component of the *prefab* is *scripts*. This is where all the functionality associated with element interaction is handled (these scripts are discussed more in a later section). A full workflow of all components that construct a BPMN element in the application can be seen in Figure 6.7. The *transform* data can be seen in the previous figure.

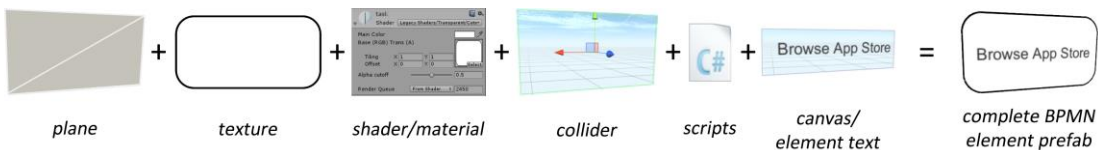


Figure 6.7 BPMN element prefab

The final section of the BPMN element is the *security line* or holder. This is classed as an optional component within the framework as not every element will require a holder. Nevertheless, the components required to add a holder to a BPMN element can be seen

[25] Unity; Unity:Manual:Canvas; (https://docs.unity3d.com/Manual/UICanvas.html) accessed 27/04/2017
[26] Unity; Unity:Manual:Colliders; (https://docs.unity3d.com/Manual/CollidersOverview.html) accessed 27/04/2017

here. Once again, Unity has existing functionality that can be utilised, this being a *line renderer*[27]. Much like the core components, a *line renderer* must be placed within a *GameObject*. However, this *GameObject* can and will be the same one used to hold each BPMN element. They are separated in the framework as the *line renderer* will not exist in the *prefab* and as such can be misleading. This component will be added via script when and as needed at run-time.

The components required to draw a *line renderer* are quite minimal. In this case, the *width* is set to *2*. The *colour* is set to *grey (#33333301 / RGBA (51, 51, 51, 1)* and for the shader, *Toon/Basic* was used, as no special effects are required. The only components requiring dynamic values are the element positions, these represent an array of two or more positions and determine the location of where the line will be drawn. As each element will only require one straight line for their holder, only two positions need to be specified. Although these values are dynamic, a basic equation can be used based on the BPMN element position to determine the holder location. For the position element, closest to the BPMN element, the *x* value can be calculated using the following equation:

$$x_h = x + \left(\frac{w}{2}\right) \qquad (6.1)$$

Where…

- x_h *= holder x coordinate*
- x *= BPMN element x coordinate*
- w *= width of BPMN element*

This equation will centre the holder of the BPMN element, horizontally. The *z* coordinate can be calculated using:

$$z_h = z + \sigma \qquad (6.2)$$

Where…

- z_h *= holder z coordinate*
- z *= BPMN element z coordinate*
- σ *= offset value*

[27] Unity; Unity:Manual:Line Renderer (https://docs.unity3d.com/Manual/class-LineRenderer.html) accessed 28/04/2017

The purpose of this equation is to offset the holder slightly from the edge of the BPMN element. The *y* value for this element of the holder will be set to *0*. This is to ensure it is at the same height as the BPMN element. A basic diagram of these coordinates relative to the BPMN element can be seen in Figure 6.8.

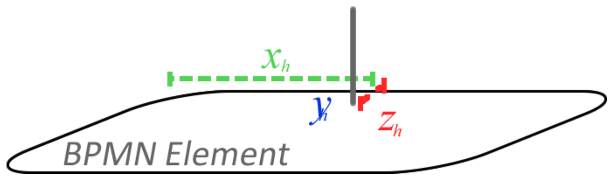


Figure 6.8 Holder coordinates

The coordinates for the second element of the holder uses the exact same equations. The only difference in this instance is that the *y* coordinate is now set to a higher value such as ~*500*. This ensures the holder is large enough to encompass all potential security requirements that need specifying.

Throughout this section, all the necessary components required for the successful rendering of a BPMN element within a game engine environment have been discussed.

6.2.2 Security Language – Diagram Generator

The framework for the security language elements is very like that of the BPMN elements, with only a few minor changes. An example of the framework can be seen in Figure 6.9.

As seen, a lot of the components are identical to the BPMN elements including that of the *prefab*. However, there are some extra elements that need specifying for security requirements. Firstly, the symbol *definition* and *BPMN element*. As mentioned in an earlier section, to ensure *coherence* amongst end users, one can specify a definition for each symbol within the tool itself, this was later outlined in the visual grammar where these definitions will be in the UI. Moreover, the *rendering* section of the framework deals specifically with the gathering and specification of necessary resources for displaying a security element. Therefore, it is at this stage that the symbols’ definitions are specified.

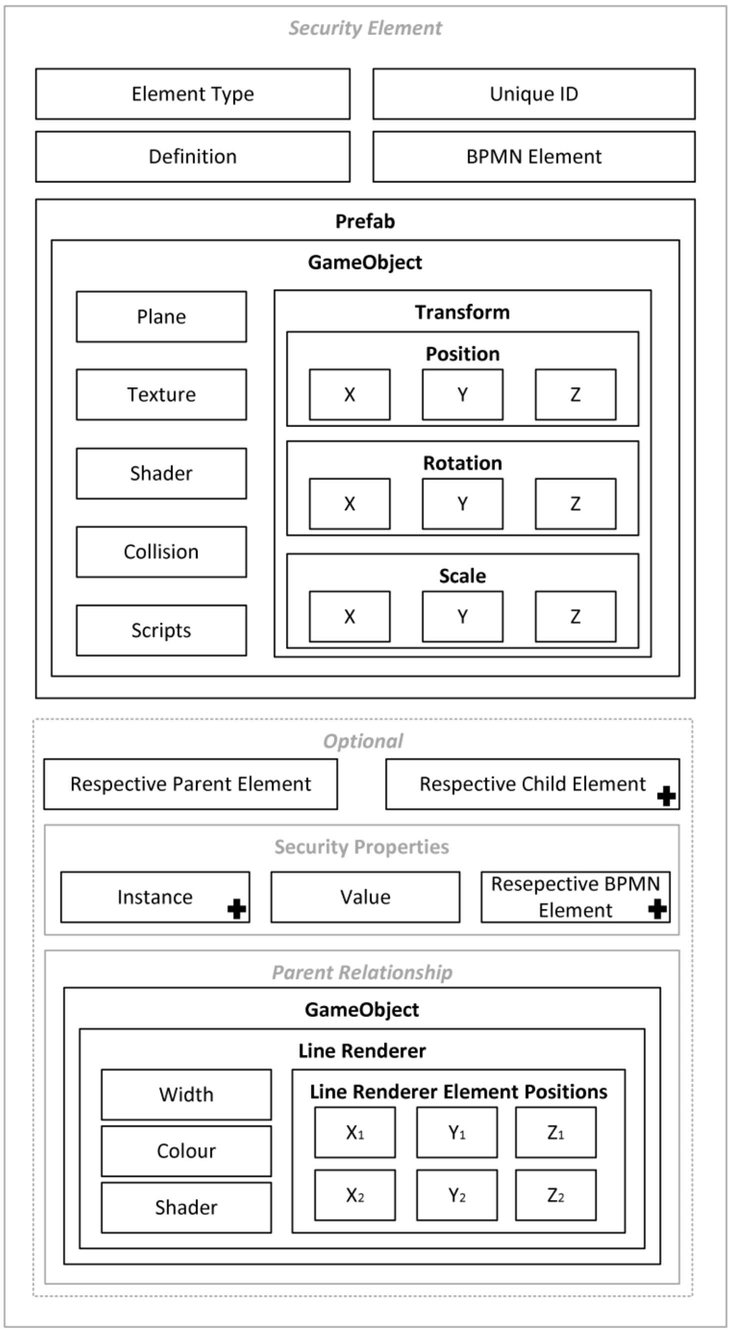


Figure 6.9 Security element framework

As for the *BPMN element* component, this is where the *unique ID* for the parent BPMN element is specified. Given that the whole purpose of the security extension is to specify security requirements on BPMN elements, the inclusion of this variable is compulsory for identifying what security is linked to what BPMN.

This identifier is also necessary for determining the *transform* values for each security element. Although the *prefabs* are identical for both BPMN and security, the key difference is that BPMN element locations are determined by an external XML file. Whereas, security element locations are calculated based on their parent BPMN element. As mentioned in the visual grammar, all security elements are always drawn in the exact same location relative to their respective holder/BPMN element. As such, like with the holder coordinates, there is no need to specify these values for every symbol. There is only a need to specify a list of predefined equations for each security concept. These can then take the coordinates of the respective holder/BPMN element and calculate their own location coordinates. For example, to calculate the *x* coordinate of a security requirement one can use the following equation:

$$x_s = x + \left(\frac{w}{2}\right) + \sigma_s \qquad (6.3)$$

Where…

- x_s *= security requirement x coordinate*
- x *= BPMN element x coordinate*
- w *= width of BPMN element*
- σ_s*= offset value respective to security requirement*

This equation is based on the BPMN element position. This can of course simplify this using the holder position instead:

$$x_s = x_h + \sigma_s \qquad (6.4)$$

Where…

- x_s *= security requirement x coordinate*
- x_h *= holder x coordinate*
- σ_s *= offset value respective to security requirement*

This equation, though relatively simple, will have multiple variances within the system. Most of the equation focuses on aligning the security requirement with the holder.

For the security element *prefabs*, the origin point was placed in the top-left. Therefore, when specifying the above equations in the system, this must be considered for each security requirement, the key value to note is σ.

Referring to the proposed visual grammar in Figure 5.35, one can see that no security requirements are aligned with the holder. Therefore, a σ is required for each one. With that said however, there are several security requirements which align with each other horizontally and can therefore share an equation for calculating their *x* coordinate. Take for example the key six concepts at the highest level within the ontology. To calculate the ơ for these concepts one can use the following equation:

$$ơ_s = -\frac{w_s}{2} \qquad (6.5)$$

Where…

- $ơ_s$ = *offset value respective to security requirement*
- w_s = *width of security requirement*

A similar equation for determining the ơ value of the *x* coordinate for each security requirement will be needed. However, as each unique security requirement is in the same location relative to its holder, these equations will be fully dynamic and reusable irrespective of the parent BPMN element's location.

The *z coordinate* is a lot simpler to determine:

$$z_s = z_h + 3 \qquad (6.6)$$

Where…

- z_s = *security requirement z coordinate*
- z_h = *holder z coordinate*

Unlike the *x-coordinate* equation which had a different ơ value for every security requirement, this equation has a consistent ơ value for them all: ~3. This places each security requirement slightly in front of the holder, just enough to stop the two objects corrupting each other.

The *y* coordinate is the only the value which has no relation to the holder/BPMN element. Irrespective of the BPMN element location on a diagram, the *y* value will always be the same respective to each security requirement. It is easier to just store a different *y* value for each element. The *x* and *z* coordinates on the other hand are very dependent on the respective BPMN element and as such the previous equations are the best option for determining their values.

As for the remaining components in the *prefab,* these are very much the same as the BPMN element only with security textures instead. The only other component slightly different is the *scale* values in the *transform*, these are also added via XML for BPMN. In the application, these are given pre-set values for security requirements. Although, BPMN element sizes can be adjusted in Activiti, for the most part they are usually one size.

Therefore, for the application that assesses the framework, a similar approach was taken, giving each security requirement a predetermined size within the tool.

The *optional* section of the security element framework feature several more components than that of the BPMN one. Firstly, the related security elements. Depending, on the hierarchy level of the security element and whether it has any child concepts, these will need to be specified within the security element *Class*. As mentioned, the visual grammar is interactive, with different elements acting as buttons to modularise their children. Therefore, it is important that each security requirement is aware of its immediate relationships so it can respond appropriately to user interaction. The plus icon inside the *respective child element* component represents a one or more instance possibility. For example, a security requirement can only have one parent, but it can have several children.

The same is true for the next component: *security properties*. As specified in the ontology, there will be several instances of security requirements that can be specified textually alongside the visual language. Given that these will be associated with a symbolic counterpart, they too will need specifying at this stage. Like with child elements, each security requirement can have multiple *instances* of *security properties*. Each of the instances can only have one *value*, but the respective BPMN element can be one or more. This is to allow concepts such as *binding of duty*, to encompass the necessary elements to adhere to its requirements.

The final component in the security element framework is the *parent relationship*, this is essentially the holder for child security elements. The main six security requirements are linked to the respective BPMN element via the main holder as specified in the previous section. As for the child elements, these are linked to the BPMN element through their parents, and therefore require their own *line renderer* to signify this relationship. The *width* and *shader* properties of the *line renderer* are identical to those in the main holder. The *colour* however is determined by the colour of child symbol; refer to the proposed visual vocabulary.

The coordinates for the *line renderer* once again rely on equations to determine their values. The *x* coordinates are determined using the following equations:

$$x_1 = x_c + \frac{w_c}{2} \qquad (6.7)$$

$$x_2 = x_p + \frac{w_p}{2} \qquad (6.8)$$

Where…

- x_1 *= relationship, position 1 x coordinate*
- x_c *= child security requirement x coordinate*
- w_c = width of child security requirement
- x_2 *= relationship, position 2 x coordinate*
- x_p *= parent security requirement x coordinate*
- w_p = width of parent security requirement

These two equations are used to calculate the *x* coordinates for both positions of the *line renderer*. As seen, both are identical except for the target security requirement. This equation ensures that the *line renderer* is horizontally positioned on both security requirements. The *y* coordinates are calculated using similar equations, this time to centre the *line renderer* vertically on the security requirements:

$$y_1 = y_c + \frac{h_c}{2} \qquad (6.9)$$

$$y_2 = y_p + \frac{h_p}{2} \qquad (6.10)$$

Where…

- y_1 *= relationship, position 1 y coordinate*
- y_c *= child security requirement y coordinate*
- h_c = height of child security requirement
- y_2 *= relationship, position 2 y coordinate*
- y_p *= parent security requirement y coordinate*
- h_p = height of parent security requirement

The *z* coordinates require no calculations and can simply be set to the same *z* coordinate as the main holder.

This concludes the frameworks for both the BPMN and security elements. The previous sections discuss all the necessary components that are required to construct and render the respective elements within the application.

6.3 Tool Functionality

6.3.1 Viewport Controls

As mentioned in the framework, the implementation of viewport controls is integral for allowing users to traverse a diagram. Given that the visual grammar is in 3D space, this is a potentially problematic area given the difficulty of 3D navigation (Hinckley, et al., pp. 1-10, 14th - 17th October, 1997; Câmara, et al., 21st-26th July, 2013).

For the application, a similar navigation method as that used in most 3D games and game engines is proposed. That is, a typical FPS camera which utilises the WASD keys and mouse input configuration (McClymont, et al., 19th-21st September, 2011). However, this may not be to all users' preferences and some will inevitably struggle to use this setup. Therefore, in this work, there will also be functionality to allow the user to focus and align the camera to specific BPMN and security elements using the UI. Although this is a slower form of navigation, it makes the application more accessible to users unfamiliar with FPS camera systems.

Of course, using the mouse to control the camera can cause an issue when trying to add security to the diagram, the user will unintentionally move the camera when moving the mouse across the UI. Therefore, the camera is something that needs to be enabled only when required, in this work, this is done by using the SPACEBAR and MIDDLE MOUSE BUTTON. At any given time, the camera controls will either be enabled or disabled; the previous buttons determining which of these states with a press.

Given that this method of navigation already exists in multiple applications, this will not be discussed any further here as it brings little contribution to the thesis.

6.3.2 Core Language Management

The management components of the framework do not necessarily represent a milestone of implementation as they do acknowledgement of their required parallel development alongside the previous components. In the last section, it was stated that *scripts* would be discussed at a later point in the thesis. The management components represent the development of these *scripts*.

Although *file serialization* and *rendering* are defined as two separate components, which they are, they are also largely encompassed under the *language management* components. *File serialization* and *rendering* both represent significant portions of functionality that need implementing into the system. However, it is in the *language management* or *tool engine* components where the two are linked together.

For the *core language management*, once a user opens the application, this component will use the *file serializer* to load the associated BPMN XML file. It will then read this file and based on the data, instantiate *prefabs* from the *renderer* into the 3D scene to draw the respective BPMN diagram. An example of a rendered BPMN diagram within the application (and 3D space) can be seen in Figure 6.10.

(dashed lines for axes reference only – not in final application)

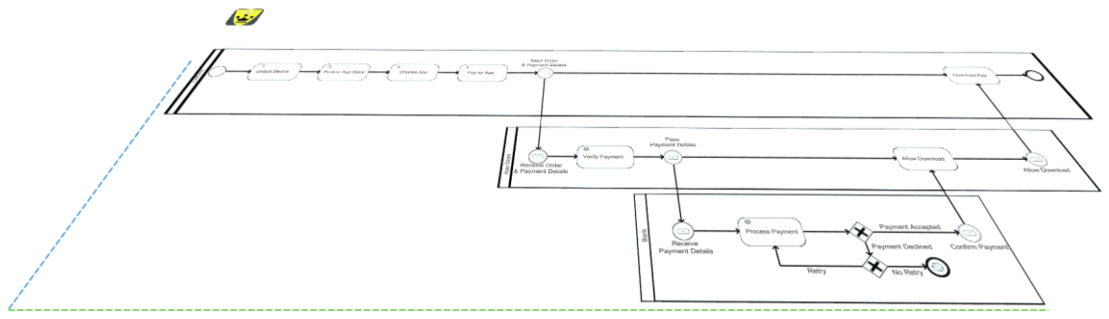


Figure 6.10 BPMN diagram rendered in the application (3D space)

6.3.3 Security Language Management

The *security language management*, follows a similar workflow. Once the system has rendered the BPMN diagram, the associated security XML file can be read to see if there is any existing data to render, utilising the *prefabs* from the *renderer* accordingly. An example of security requirements rendered within the application can be seen in Figure 6.11.

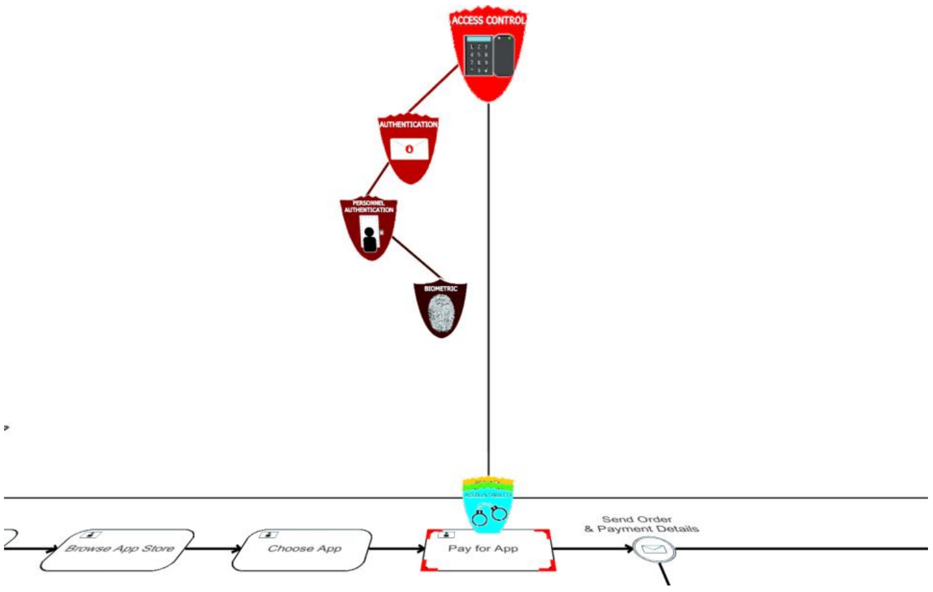
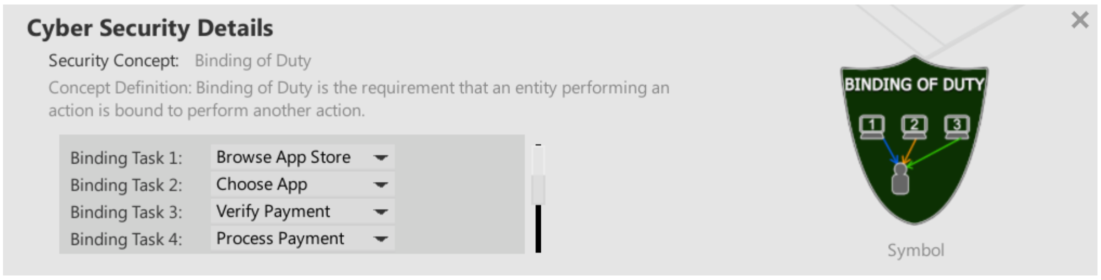


Figure 6.11 Security requirements rendered in the application (3D space)

This component also deals with functionality from the *front-end* component. Though this component is yet be discussed, this is where the application UI is handled. Given that users can add security elements at run-time using the UI, this component represents the libraries required to ensure the underlying functionality for each UI element work as intended when the user interacts with them.

This is also a good point to discuss the visual grammar for the security requirements that are linked to multiple BPMN elements: *binding* and *separation of duty*. These concepts require linking to multiple BPMN elements, but are yet to receive any attention on how this will be accomplished within this thesis. Although, it was stated that these links will be defined in the details toolbar, an example of which can be seen in Figure 6.12.



Cyber Security Details

Security Concept: Binding of Duty

Concept Definition: Binding of Duty is the requirement that an entity performing an action is bound to perform another action.

Binding Task 1:	Browse App Store
Binding Task 2:	Choose App
Binding Task 3:	Verify Payment
Binding Task 4:	Process Payment

Figure 6.12 Binding of duty - details toolbar

As for the visual representation of these relationships, these will be shown using *line renderers* which go from the bottom of the respective element to each linked BPMN element. These relationships however, will only be visible when the respective security element is selected. This is to ensure cognitive overload does not ensue having multiple lines across a diagram. An example of *binding of duty* with multiple BPMN element links can be seen in Figure 6.13.

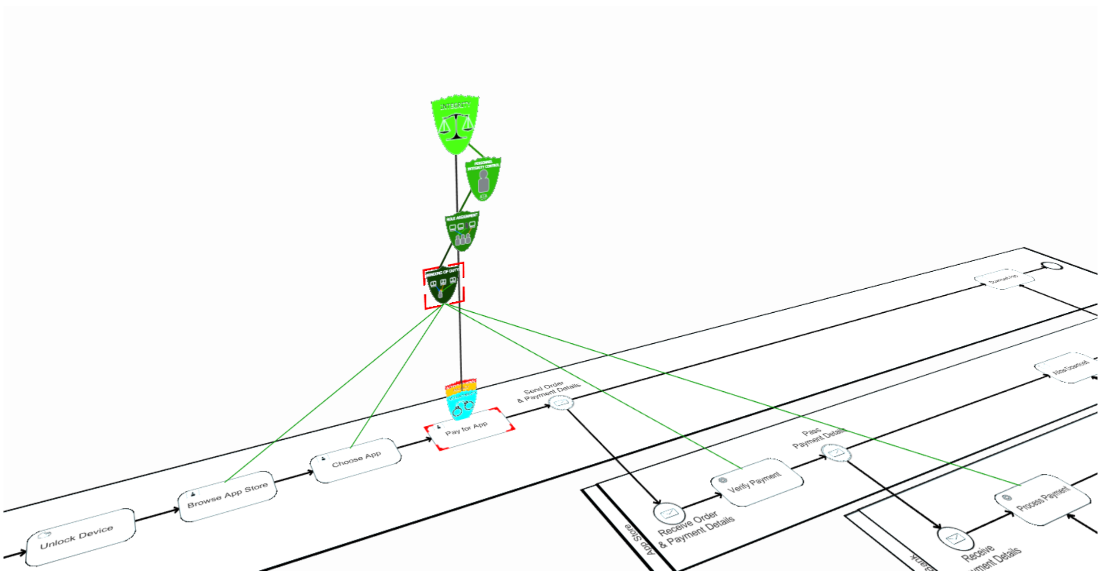
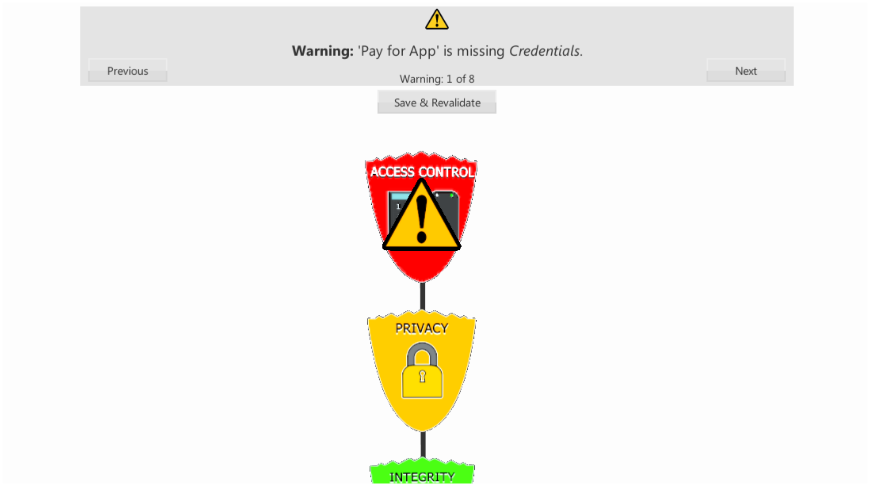


Figure 6.13 Binding of duty - diagram

6.3.4 Security Policy Management

The *security policy management* component is where the necessary functionality for both creating a policy and assessing its compliance is implemented. The visual design of the policy creation is covered in the *GUI* component. As for the compliance checking, this is handled in the application by1 comparing the XML files of the security language and security policy against each other. An example of the *feedback technique* within the application can be seen in Figure 6.14, showcasing both the error console and the warning symbol mentioned earlier.


Warning: 'Pay for App' is missing *Credentials*.

Warning: 1 of 8

Figure 6.14 Policy feedback in the application

6.4 Front End

6.4.1 Graphical User Interface

The GUI component is different to the other components in the *Technology Framework*, primarily because it is the first one which does not rely on explicitly defined designs by the *language architect*. Although GUI is an important component in the usability of an application and can therefore affect the end users' opinion of the security language, it is not something which is integral to ensuring that the visual vocabulary and grammar rules are adhered to. The BPMN language is used in multiple different software packages all with varying user interfaces, Visio and Activiti are two examples, without any issues regarding the language itself. Therefore, it is not something that should be defined by the *language architect*.

The GUI component can be further subdivided into lower concepts: *adding security* and *creating a policy*. The *adding security* component in regard to adding security requirements to the core language, BPMN. As mentioned in the visual grammar, the need of a details toolbar for specifying instances of security concepts as well as somewhere to hold concept definitions was discussed. This is not something unfamiliar in modelling language tools with many software packages including similar functionality for core language details.

The UI for the *adding security* component can be categorised into three areas: diagram creation toolbar, details toolbar and settings toolbar. The diagram creation toolbar as implied, deals exclusively with the required UI elements for adding security requirements to a diagram. The details toolbar as previously discussed holds security definitions and properties. The settings toolbar, something not yet mentioned, acts as a holder for system functionality. For example: camera controls for focus security/BPMN elements, hide security elements, saving of diagrams and help sections.

Although UI represents the unique selling point of most modelling language tools, it is not something which aims to be overemphasised within this work. Nevertheless, a usable UI is still of vital importance to ensure the end user can fully utilise the language. As such, to ensure the UI meets this criterion, inspiration will be taken from existing modelling language solutions. Firstly, that of Visio which can be seen in Figure 6.15.

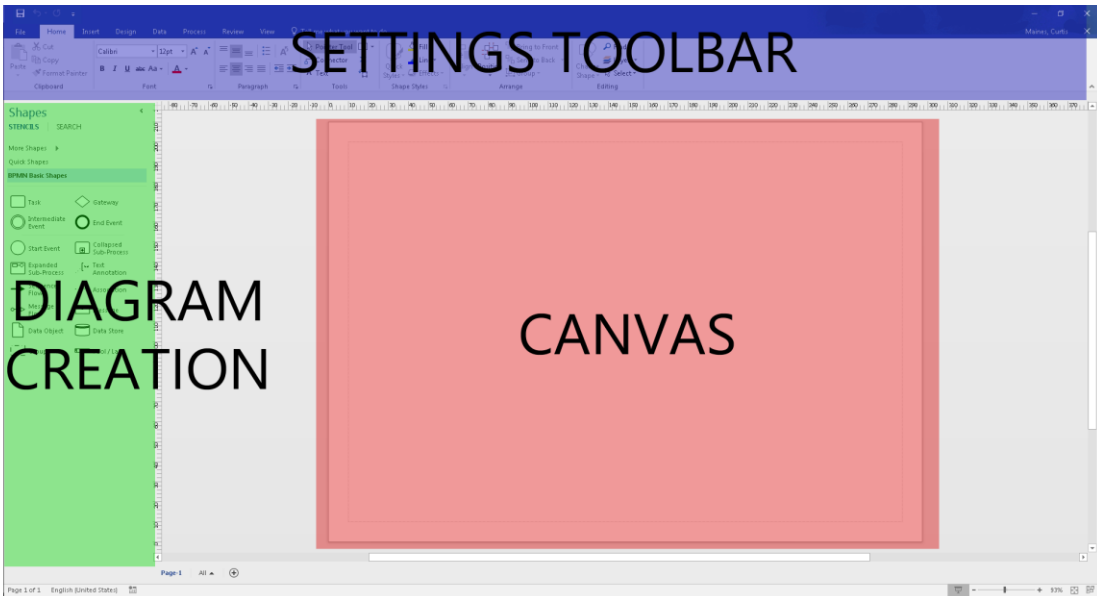


Figure 6.15 Visio UI layout

As seen in this figure, Visio places the canvas in which their diagram is created in the centre of the screen. The diagram creation toolbar which holds all the UI elements required for adding elements to the canvas, has a default location on the left of the screen (although, this can be moved to any location of the users choosing). The settings toolbar is like that of other Microsoft products and fills the entire top section of the screen, this is categorised into different subsections such as aesthetic based functionality which includes text/symbol colour, font type, font size etc. However, it is also where the settings for the software are located along with other functionality of high importance such as the saving of diagrams.

The UI layout for Activiti can be seen in Figure 6.16. Like Visio, Activiti is split into sections like those identified earlier. However, Activiti also includes a project file explorer section within their UI, not something which is necessary for this work.

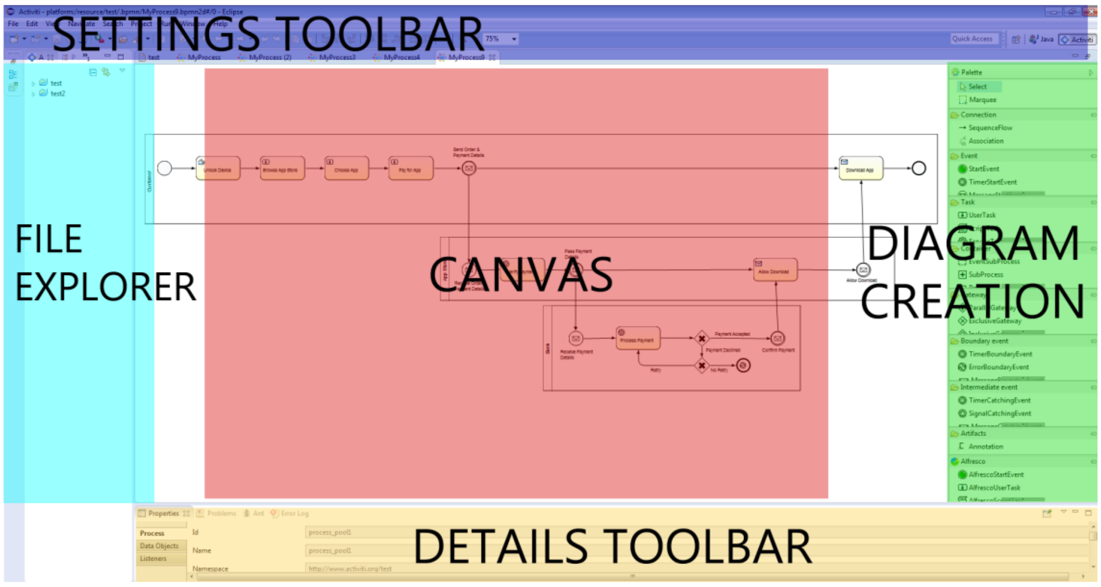


Figure 6.16 Activiti UI layout

As in Visio, Activiti also places their canvas in the centre of the screen. The diagram creation toolbar however, is located on the right of the screen as opposed to the left, with the left featuring the file explorer. As with most software (and Visio) the settings toolbar is placed along the top of the screen. Unlike Visio however, Activiti include a details toolbar within their software which can be seen across the bottom of the screen.

From these two applications, rough similarities in the UI layout can be deduced for modelling language tools. Although the location of each element somewhat differs between each piece of software, the pre-identified subsections of diagram creation toolbar, details toolbar and settings toolbar, are all justified from the viewing of existing solutions and their own categorisations. The main UI design choice which can be taken away from this light analysis of existing tools, is that each category of the UI has its own edge of the screen. For example, the settings toolbar typically fills the entire top of the screen. The diagram creation toolbar will then fill the entirety of either the left or right side, with the details toolbar filling the bottom of the screen. This gives a total of four potential areas to categorise the UI into. Although only three have been mentioned thus far, there is also the error console to consider for the policy compliance feedback, bringing the total categories to four.

Before deciding on where to place each category of the UI, the requirements of each section must be understood. For the diagram creation and settings toolbar, the only elements required will be basic buttons for adding/deleting requirements or calling some sort of system functionality. The details and policy feedback toolbars however, will require more design space as they have fair sizes of text they need to display. Therefore, given that the top and bottom of the screen offer the most in terms of design space, placing these two objects in these locations is the logical choice. A high-level layout of the UI can be seen in Figure 6.17.

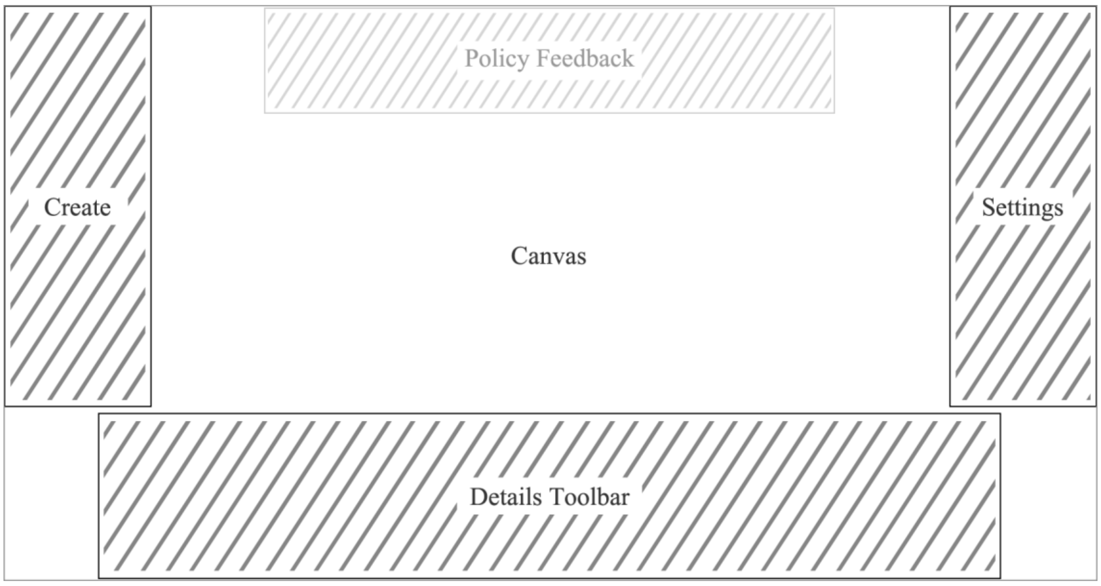
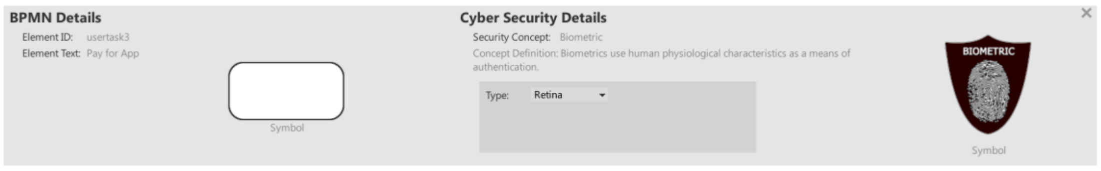


Figure 6.17 Application UI layout

The policy feedback section is faded in the figure as this will only be present when the user explicitly requests to check the diagram’s policy compliance; displaying this toolbar at all times uses up canvas space unnecessarily.

In the visual grammar, a similar high-level view of the proposed details toolbar was given. In Figure 6.18, one will be able to observe the final details toolbar as it appears within the application.



BPMN Details

Element ID: usertask3

Element Text: Pay for App

Symbol

Cyber Security Details

Security Concept: Biometric

Concept Definition: Biometrics use human physiological characteristics as a means of authentication.

Type: Retina

BIOMETRIC

Symbol

Figure 6.18 Details toolbar rendered in or application

As seen, the toolbar adheres to the specification of the grammar.

The settings toolbar within the application can be seen in Figure 6.19. The buttons that are included within this toolbar are: *save, reset camera, focus symbol, focus task, hide security, hide details, ontology, help* and *exit.*

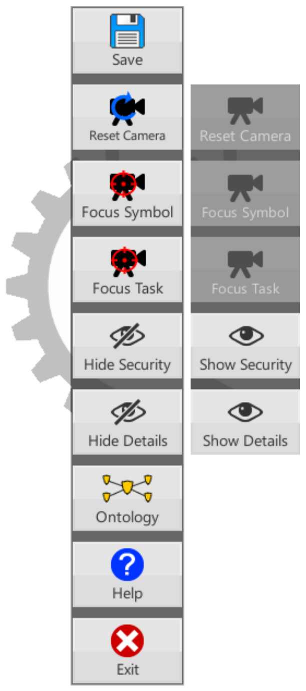


Figure 6.19 Settings toolbar rendered in the application

The functionality of these buttons are as follows:

- **Save:** *saves the diagram*
- **Reset camera:** *resets the camera to the default position (roughly 45° overlooking the diagram)*
- **Focus Symbol:** *focuses the camera on the currently highlighted security element*
- **Focus Task:** *focuses the camera on the currently highlighted BPMN element and its respective security requirements*
- **Hide Security:** *hides all security elements*
- **Hide Details:** *hides the details toolbar to allow for a larger canvas*
- **Ontology:** *opens a new window within the application showing the proposed ontology*
- **Help:** *opens a new window within the application with camera navigation tutorials*
- **Exit:** *exits the current scene to the main menu of the application*

One will notice in the figure, there are slight variances to the right of some of the buttons. These showcase alternative views the buttons have within the application depending on the user's current state. For example, if the camera controls are enabled, the *reset camera, focus symbol* and *focus task* buttons will be disabled. Likewise, if a security symbol or BPMN task is not currently highlighted the respective symbols will be disabled. The *hide security* and *hide details* buttons switch between their states depending on whether each is currently hidden. The functionality to hide security elements is something which was lightly touched on in an earlier section. Although there is a strong emphasise throughout the work on the importance of specifying security at design as it does impact the way in which a business process is constructed, that there is not always a need to view security requirements on a diagram. Depending on the audience (CEO or CTO/CSO), they will have different requirements and therefore providing the ability to hide security elements will provide a more user-friendly application.

The reason for including the ontology within the application is to assist with novice adoption of the language. As the visual grammar is based on *modularisation* and *hierarchy* structuring, knowledge of these hierarchies is integral to use the language. Although a competent user will be able to remember these, there is no need to discourage new users of the language and as such include the ontology in the application itself, dismissing the need for any external resources.

The diagram creation toolbar (or build toolbar) has similar functionality to the visual grammar. Given that 76 security requirements are used within the language, simply listing these in a dropdown gives a poor user experience. Therefore, the toolbar is structured like that of the language itself. At the highest level, highlighting a BPMN element, before any security is added to it, will present the user with a toolbar like that in Figure 6.20.

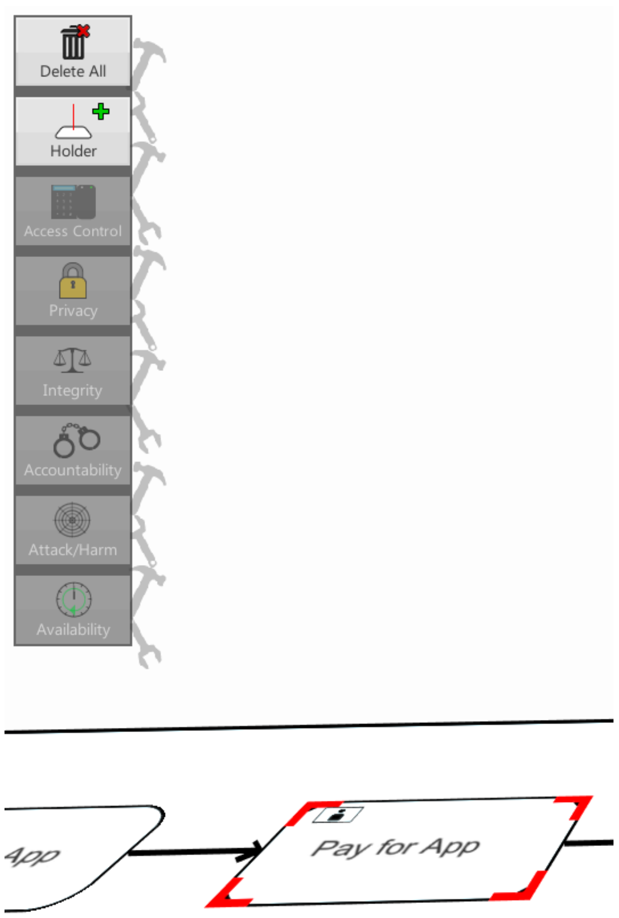


Figure 6.20 Build toolbar rendered in the application

As seen, the user has two possible options (the rest being disabled): *delete all* and *holder*. *Delete all*, gives the user the ability to delete all security requirements from the diagram. The *holder* button in this state, allows the user to add a security holder to the current BPMN element. By this state, this thesis refers to the green plus icon inside the button. This notifies the user that the button will add the respective concept. If the user adds a holder, the menu will change to the following:

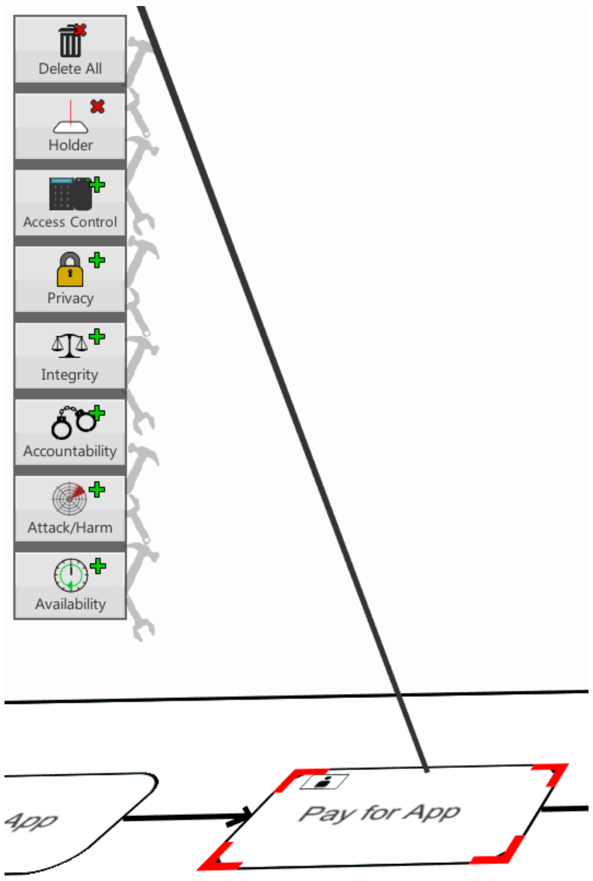


Figure 6.21 Build toolbar - holder on BPMN element

As seen, the holder button has now changed to a red cross icon instead. This is to notify the user that pressing the button will delete the respective concept. This process of adding and deleting components is consistent across the security requirements also. Viewing the figure, one can see that the core six security requirements of the ontology are now enabled and can be added to the BPMN element.

These buttons will change depending on the currently highlighted security element. As no element is currently selected, just a BPMN *task*, the key six will display. However, should the user select one of the key six after they have been added to the diagram, these buttons will change to the concept's respective children. This trend continuing to the lowest level of the ontology. For example, the following figure showcases how the build toolbar changes depending on the currently highlighted security concept:

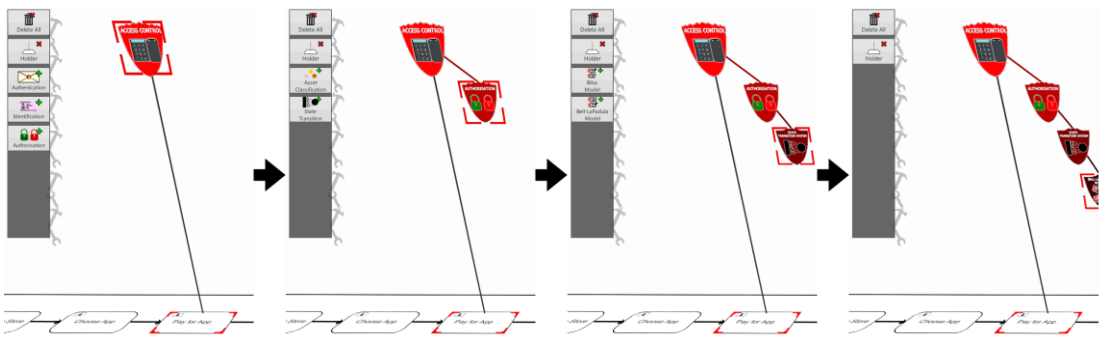


Figure 6.22 Build toolbar - visual grammar properties

A version of the above figure focusing on just the toolbars can be seen in Figure 6.23.

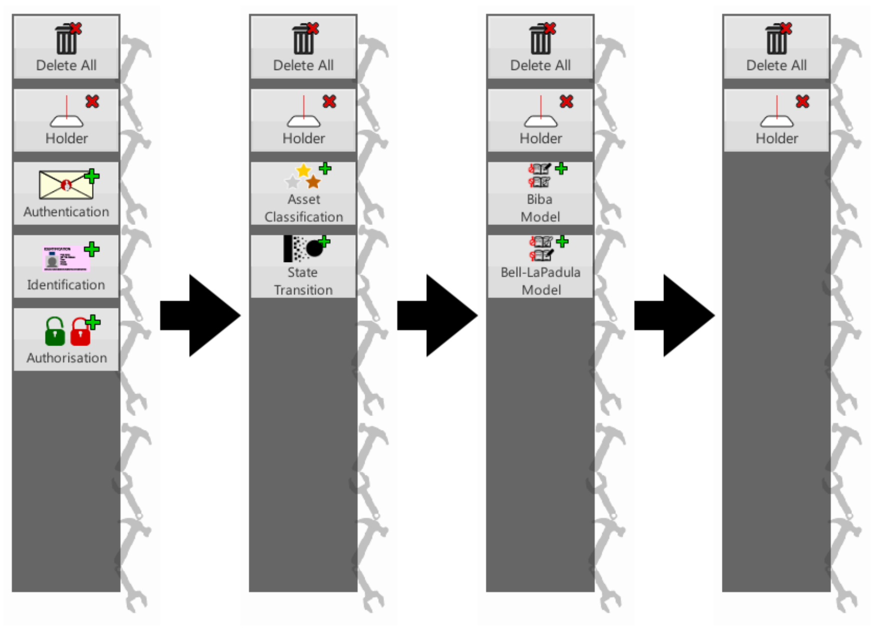


Figure 6.23 Build toolbar - visual grammar properties (toolbar focus)

As seen, by utilising the security language's visual grammar in the build toolbar, the end user is given a better experience than alternative options of long scrollable lists of elements like in other software. Given that the language requires the user to work in hierarchies and progress down the ontology to specify more detail, this solution is the logical approach to ensure cohesion with the language itself. It also assists the user in learning the various hierarchies that exist within the ontology, something they would be unlikely to pick up if using the usual list of elements instead. From a UI perspective, this method of specification allows for the creation of larger more meaningful buttons without compromising due to design space restrictions, again providing a more user-friendly experience.

One will notice in the previous figures that both BPMN and security elements had a red open rectangular shape around them. This is referred to as the *highlighter* throughout this work and the application. This acts as a form of visual feedback to notify the user of the currently selected element. For security requirements, this symbol also animates by pulsing its scale slightly. The highlighter is a necessary inclusion in the UI as there are times when it is difficult to know what the currently selected security element is in open trees. For example, refer to Figure 6.24.

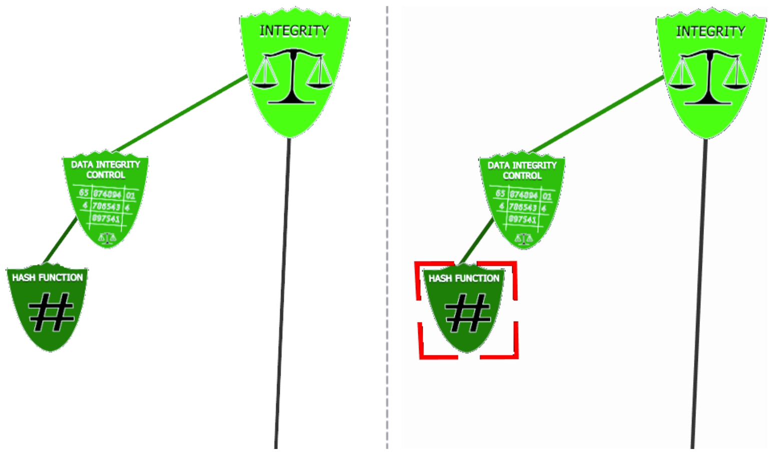
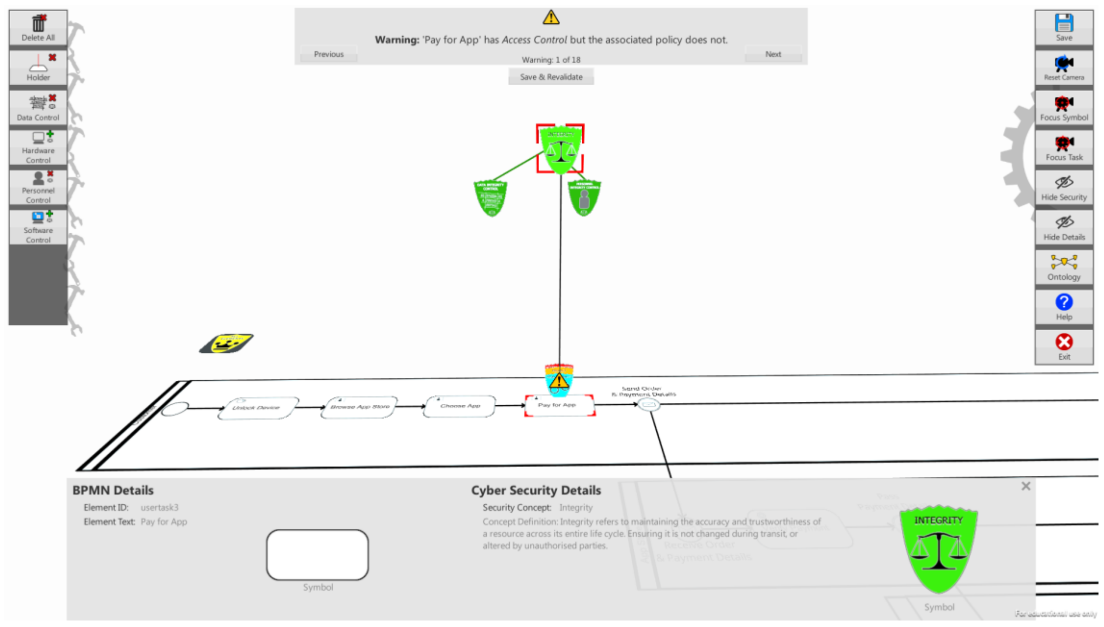


Figure 6.24 Highlighter

Viewing the left side of the figure, it is difficult to determine if the currently highlighted element is *data integrity control* and its child element of *hash function* is on display; or, if the currently highlighted element is *hash function* and it has no children. Viewing the same hierarchy using the highlighter in the right of the figure, one can see that the currently selected element is *hash function*, justifying its inclusion within the application.

An example of all the UI elements together, can be seen in a screenshot of the application in Figure 6.25.


Warning: 'Pay for App' has *Access Control* but the associated policy does not.

Warning: 1 of 18

BPMN Details

Element ID: usertask3

Element Text: Pay for App

Cyber Security Details

Security Concept: Integrity

Concept Definition: Integrity refers to maintaining the accuracy and trustworthiness of a resource across its entire life cycle. Ensuring it is not changed during transit, or altered by unauthorised parties.

Figure 6.25 Application UI - screenshot

This concludes the UI for the *adding security* component. The *create a policy* UI does not require as much effort as the diagram creation, as this is not something that can directly impact that evaluation of the framework. Nevertheless, for the sake of providing a thorough case study of the framework, its design will be defined here.

As mentioned earlier, the security polices in the application will be a textual version of the diagram security. That is, the user will have a list of BPMN elements in which they can add security requirements from the ontology, these will then be compared against the diagram to assess any discrepancies.

For the *create a policy* UI, there are four areas the elements can be categorised into, these being: *BPMN elements, security requirements, details* and *settings*. Unlike the diagram creation however, in this instance, there is no *canvas* element to consider and therefore have the entire screen for the UI. Taking this into account, the UI layout for the *create a policy* component can be seen in Figure 6.26.

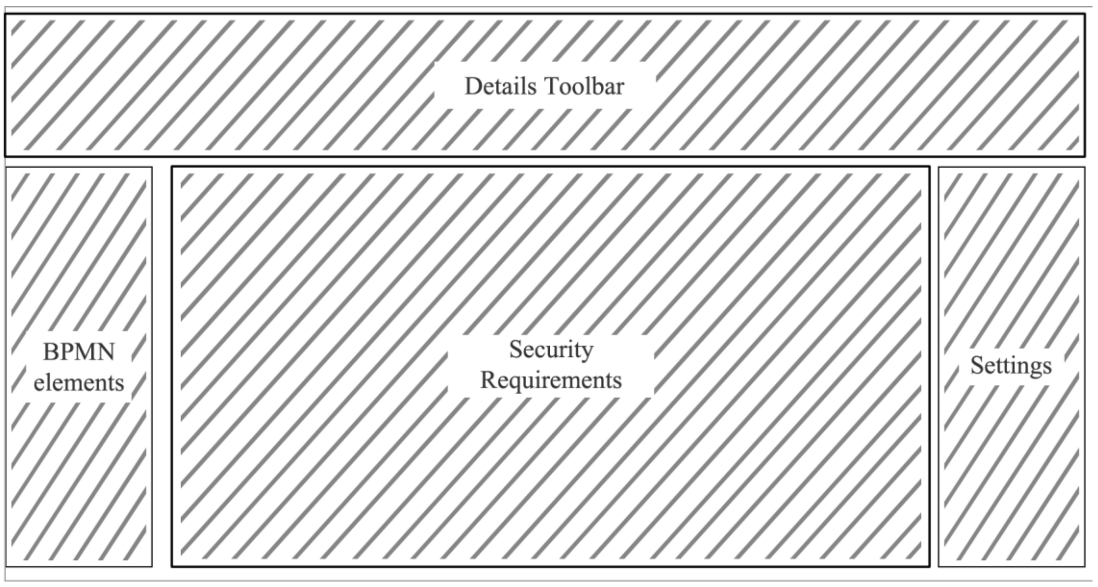
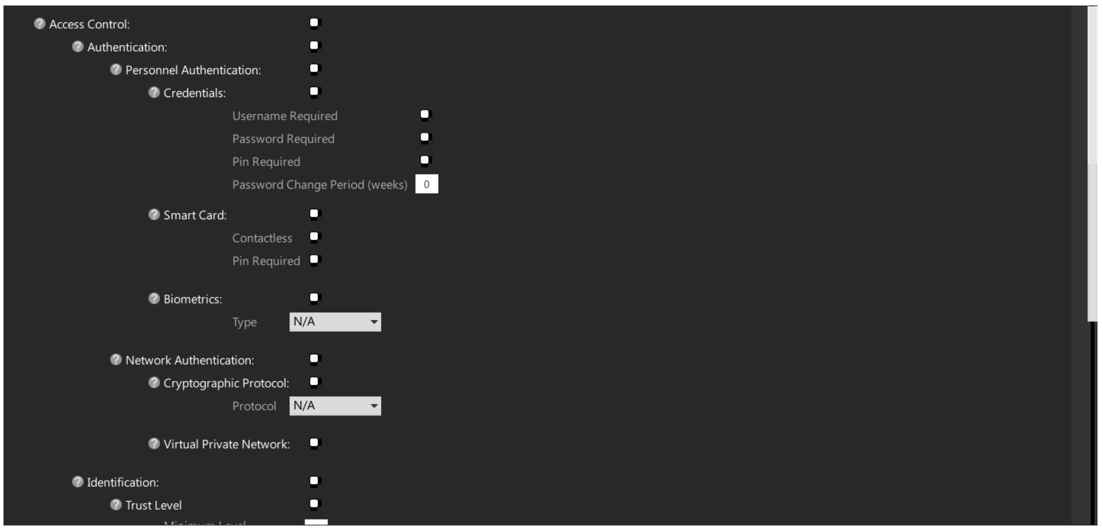


Figure 6.26 Policy creation UI layout

The *BPMN elements* section of the UI is simply a scrollable list of basic buttons. Their respective text being that of their BPMN counterpart. The security requirements and details toolbar somewhat interlink. Given that the security requirements are structured in a hierarchy, there is little benefit in simply listing them like the BPMN elements as they will lose their complexity management. Therefore, it is more fitting to once again utilise *modularisation* and split the security requirements into their six areas of the ontology. The user can then choose to show a list of concepts of the various areas for each respective BPMN element. Lower-level hierarchies can then be distinguished by indenting them on the page, like that in coding languages. An example of this layout for *access control* can be seen in Figure 6.27.



```
Access Control:
    Authentication:
        Personnel Authentication:
            Credentials:
                Username Required
                Password Required
                Pin Required
                Password Change Period (weeks)  0
            Smart Card:
                Contactless
                Pin Required
            Biometrics:
                Type  N/A
        Network Authentication:
            Cryptographic Protocol:
                Protocol  N/A
            Virtual Private Network:
    Identification:
        Trust Level
```

Figure 6.27 Policy security requirements

As seen, the use of indentation ensures that the hierarchy defined within the ontology is maintained. One will also notice a small question mark icon next to each concept. If the user selects this, a definition of the respective concept will appear below the form, ensuring all parties are working in cohesion with the same understanding of each concept.

The details toolbar in the policy creation can be seen in Figure 6.28.

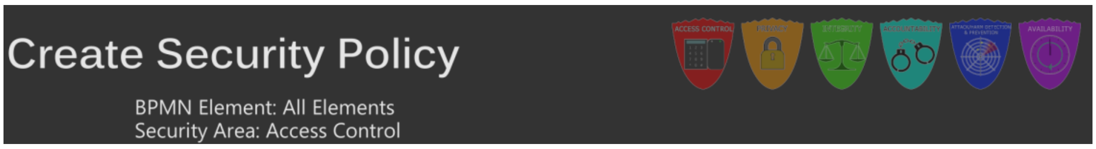

Create Security Policy
BPMN Element: All Elements
Security Area: Access Control
ACCESS CONTROL
ACCOUNTABILITY
ATTACK/HARM DETECTION & PREVENTION
AVAILABILITY

Figure 6.28 Policy details toolbar

The toolbar is split into two areas, one providing information to the user, the other, acting as a navigation bar. As seen on the left of the figure, the toolbar informs the user of the current BPMN element they are working on (in this case the *All Elements* element), as well as the current security area (*access control*). The right side then allows the user to choose one of the six security areas, which in turn will change the form seen in Figure 6.27 to the child elements of the respective concept.

The settings toolbar for the policy creation can be seen in Figure 6.29.

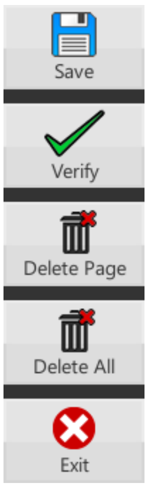


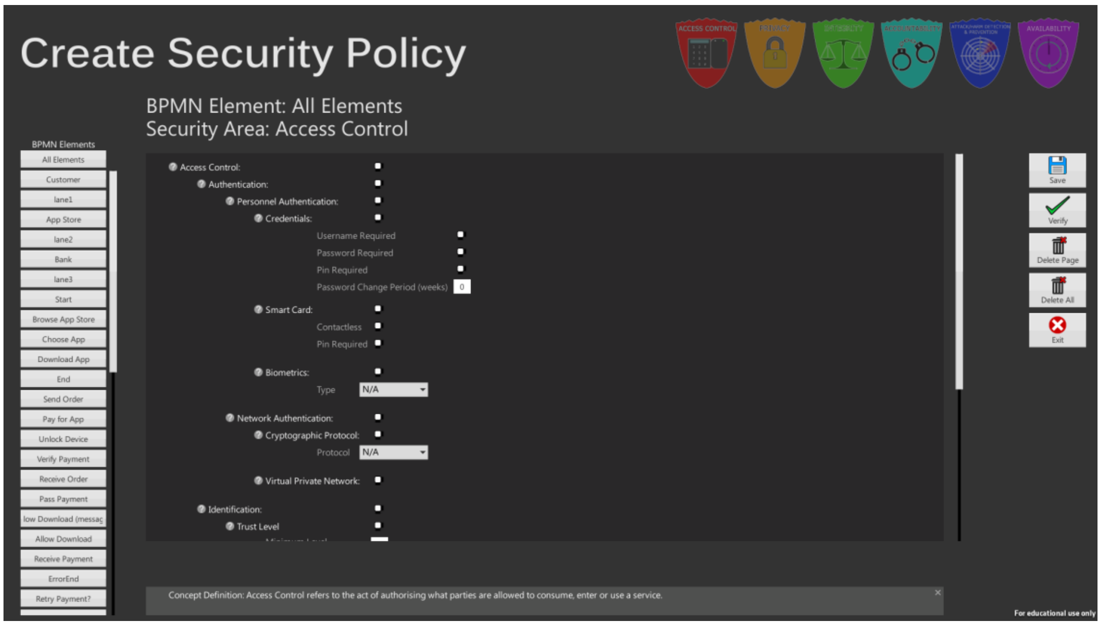
Figure 6.29 Policy settings toolbar

The functionality of these buttons is as follows:

- **Save:** *saves the diagram*
- **Verify:** *ensures respective hierarchies are adhered to (selecting a low-level concept will auto-select its parents)*

- **Delete Page:** *deletes all security requirements on the page the user is currently working on*
- **Delete All:** *deletes all security requirements on every BPMN element*
- **Exit:** *exits the current scene to the main menu of the application*

A full screenshot of the policy creation UI can be seen in Figure 6.30.


Create Security Policy

BPMN Element: All Elements
Security Area: Access Control

Access Control:
Authentication:
Personnel Authentication:
Credentials:
Username Required
Password Required
Pin Required
Password Change Period (weeks) 0
Smart Card:
Contactless
Pin Required
Biometrics:
Type N/A
Network Authentication:
Cryptographic Protocol:
Protocol N/A
Virtual Private Network:
Identification:
Trust Level

Concept Definition: Access Control refers to the act of authorising what parties are allowed to consume, enter or use a service.

For educational use only

Figure 6.30 Application policy creation - screenshot

6.4.2 Menu System

The *menu system* component of the *front end* refers to the UI elements that deal with navigating throughout the application. Most software applications will typically only consist of one scene and therefore do not require such functionality. However, in this case, there are several different scenes within the application: *diagram creation, policy creation* and *policy validation*. Therefore, this component requires that some form of 'home screen' or means of navigating between them is included. For this work, a home screen was utilised, with the previously mentioned 'Exit' buttons taking the user here when selected.

This component represents the final step in the *application developer*'s role. It also finalises the implementation process of the application. As mentioned, the *end user* represents the end goal of the framework in which the security language is used. For the BPMN case study, the role of the *end user* is effectively accomplished in the next section during experimentation.

A user manual for the application can be found in the Appendix. Apart from trying the application itself, this is the best way to experience how the tool is used.

6.5 Chapter Summary

As mentioned, this chapter details the case study based on the proposed framework. Using BPMN as the parent language, there is a full discussion on how each component of the framework is satisfied and by consequence several of the PoN, and the previously defined heuristics.

This chapter not only acts as justification for the proposed framework, but also encompasses several other novelties such as: a comprehensive ontology of cyber security requirements, a graphical framework for the designing of notation symbols, and a new 3D complexity managed visualisation for different domain extension within a modelling language.

Although this chapter encompasses a lot of justification for the framework itself, the following provides a more explicit focus on this analysis. Highlighting how each principle in the PoN and previously defined heuristics are satisfied, and consequently prove the proposed solution far superior to current attempts. Along with, user experimentation to further support the theoretical evaluation of the language's abilities.

7 Critical Assessment

Having used the framework to develop a security extension for a modelling language, the main contribution of the work has already been somewhat proven. The framework's claim is that it can be used for the extension and visualisation of cyber security requirements in modelling languages. Considering that the solution created within the case study is capable of this, the framework is to some extent validated. However, a security extension for BPMN is not something new in the area, within the literature review, several security extensions to BPMN are evaluated. Nevertheless, the extension of a modelling language with comprehensive security requirements which also adheres to existing design principles *does* represent something new to the area. This is where the framework surpasses that of existing solutions and provides its core novelty and contribution.

The framework detailed in Chapter 4, represents the highest level of the proposed solution. This system constitutes the basic foundations any security extension requires, with little detail on how best to achieve said requirements. That is not to say the framework does not provide its own contribution. Current attempts at security extension do not appear to have followed any logical structure, most efforts have been placed on semantics with the development of the language itself given little regard. The framework's objective is to end this practice and instead provide authors with a set of guidelines/methodology that encompasses not only semantics, but the necessary components for ensuring a well-designed language too.

Moreover, the BPMN case study in which a new security extension is created, further showcases how the framework fulfils this claim. Explicit detail is provided on how each component is achieved within the framework and by consequence how a much more comprehensive and visually expressive notation than any seen thus far is produced. This emphasises how the implementation of such a solution is not as difficult as current literature would make one believe. However, the ease of this process is only possible now due to the foundation work that was invested in this thesis' framework.

Nevertheless, that is not to say the other contributions of this work are diminished. Throughout the BPMN case study, several novel proposals to the area of language design are discussed; some of which, are potentially even more advantageous than that of the framework. These being: a comprehensive cyber security ontology, a visual vocabulary framework, a visual grammar framework and a development pipeline for creating the language tool itself.

Therefore, although the case study has proven the framework outlined in Chapter 4 will ensure the production of a cyber security extension to a modelling language, the sub-frameworks and the critical assessment of them, lies in the evaluation of the security extension solution which was produced in the case study. If successful and superior to current approaches, one can conclude that the sub-frameworks on which this solution was based, also provide their own novel contribution to the area. Further emphasising how the framework in Chapter 4, when fully utilised, can ensure the production of a comprehensive, complexity managed security extension.

7.1 Physics of Notations

One may argue that the only way to ensure a fair comparison between extensions, is to conduct experiments with end users with each language. However, there are several drawbacks to this approach which can consequently skew the results in the wrong direction.

Take as an example, the principle of *perceptual discriminability*, or the "ease of distinguishing between symbols within a notation". If an end user was asked to identify a set number of concepts in Salnitri *et al.'* extension and the one created within this thesis, it is very plausible the user would conclude Salnitri *et al*.' solution as the better of the two. The reason being, they have only 11 concepts to distinguish between, opposed to the 76 in this work's solution. Therefore, it would appear to an end user that Salnitri *et al.'* language offers the more *perceptually discriminable* notation. However, using the principles defined by Moody to assess the symbols based on scientific evidence opposed to user preference, one would conclude that Salnitri *et al.'* extension is actually *construct deficit* and utilises a very low number of visual variables to assist in symbol discriminability. Therefore, given that the paucity of constructs current extensions have, will undoubtedly lead to unjustified conclusions regarding which extension offers the best solution, end user experimentation will not be used as a means of comparing the proposed solution to existing attempts.

Throughout the review of existing security extensions, the PoN was used as a form of scientific evaluation, and as a way of giving each extension a quantifiable grade on their success as a modelling language. Therefore, evaluating the case study solution against these principles, will allow for a more realistic and fairer way of assessing the extension against existing solutions.

7.1.1 Semiotic Clarity

To ensure the satisfaction of the principle of *semiotic clarity* within this work, an ontology of potentially modellable cyber security requirements was created. This ontology was based on existing security extensions and other scenario specific ontologies as discussed earlier.

The principle of *semiotic clarity* requests that there should be a one-to-one mapping between domain constructs and graphical symbols (Moody, 2009). Throughout the case study, support is provided for the visual specification of all 76 concepts within the ontology as well as textual specification of multiple instances/properties of these concepts. Therefore, this thesis is confident in stating the produced extension as satisfying the principle of *semiotic clarity*.

7.1.2 Perceptual Discriminability

As previously mentioned, *perceptual discriminability* must be evaluated in two parts: against same notation constructs, and against that of the parent language. For this work's notation, the visual vocabulary was based on the successful traits of existing extensions; one element, being the use of an outer shape to distinguish the domain against that of BPMN. As with existing extensions, by using a consistent outer shape for all the symbols, the user can distinguish between each domain easily. Of course, domain distinction in the proposed language is most obvious by the fact that each one is represented on a separate set of axes.

As for distinction between the symbols themselves, multiple visual variables are utilised to ensure this is achieved. *Primacy of shape* is used the same way as Salnitri *et al*. (Salnitri, et al., 2014), using a unique icon inside each outer shape. The *visual distance* of the notation is difficult to measure as all the symbols have different values depending on the notation they are being compared against. Nevertheless, seven of the eight visual variables are used across the language both for *visual distance* and *redundant coding*. In comparison to existing extensions which also satisfy this principle, the notation created within this thesis can definitely satisfy *perceptual discriminability*.

7.1.3 Semantic Transparency

The *semantic transparency* of the symbols can again be assessed in two parts. As the outer shape is a shield, something largely associated with security, one can state the notation as achieving *semantic transparency* in regard to domain identification.

As for the individual symbols, the *semantic transparency* is encompassed within their inner shapes. These were designed with the intention of being icons and graphic mnemonics of their underlying semantics. Although the effectiveness of these icons to some extent is open to opinion, given that Saleem *et al.* (Saleem, et al., January 2012) and *Salntiri* et al. (Salnitri, et al., 2014) both satisfy *semantic transparency* in their notation, the icons created within this work at the very least are on the same level of transparency as these extensions, and as such also satisfy the principle of *semantic transparency*.

7.1.4 Complexity Management

As mentioned, *complexity management* is one of the key motivations for this work, stating on multiple occasions the inadequacy of current extensions in regards to this principle. To ensure the satisfaction of *complexity management,* an entirely new approach to modelling language extension was introduced. Placing security requirements perpendicular to BPMN elements in 3D space, along with *modularisation* and *hierarchy* structuring, the proposed language allows for the specification of 76 security requirements *per* BPMN element. This is an exceptionally higher number of concepts than any existing language and specified without manipulating any existing BPMN notation. In 1998, Gil and Kent (Gil & Kent, 19th-2th April, 1998) asked: "How can the introduction of 3D contribute to software and systems modelling?". This question can be answered now by simply replying, like this.

The principle of *complexity management* requests that a language includes mechanisms for managing a diagram's complexity (Moody, 2009). Viewing of the proposed visual grammar provides all the necessary evidence for confirming the satisfaction this principle. Unlike other existing approaches, the proposed language does not simply show a figure of how the elements are used within BPMN, explicit rules (mechanisms) are provided on the placing of security requirements in relation to their parent BPMN element. This allows for a substantially larger number of cognitively manageable concepts on a diagram compared to existing solutions. Therefore, this thesis is confident in claiming a strong satisfaction of the principle of *complexity management*.

7.1.5 Visual Expressiveness

Skipping over *cognitive integration* once again, the *visual expressiveness* of the proposed language greatly surpasses most existing extensions, utilising a total of seven of the eight possible variables. The *use of colour* within the notation makes the identification of the core security area very fast. Along with the *redundant coding* of multiple other variables, computational offloading is maximised, increasing symbol identification substantially.

The principle of *visual expressiveness* requests the use of the full range and capacities of visual variables (Moody, 2009). Given that, the only variable not utilised is *texture*, and the previous grading system suggested a minimum of half the available variables to fulfil the principle's requirements, the proposed extension is more than capable of satisfying *visual expressiveness*.

7.1.6 Dual Coding

Dual coding, as previously mentioned, consists of two sub components: *annotations* and *hybrid (graphics+text) symbols* (Moody, 2009). When defining the visual vocabulary of the language*,* there was a discussion about intentionally dismissing *annotations* to minimise confusion with the parent language. Nevertheless, the principle of *hybrid (graphics+text) symbols* is utilised within the notation, assisting with the learnability of each symbol by always having a text reference nearby for reassurance. Therefore, this thesis claims the proposed language as also satisfying the *dual coding* principle. Although *annotations* are not included*,* Moody's principles are based on a single language, reasonable decisions must be made in regard to whether satisfying certain principles for an extension truly does benefit the overall design of the language. In this case, the inclusion of *annotations* are more likely to damage the language than improve it.

7.1.7 Graphic Economy

One would assume that as the previously assessed extensions satisfy *graphic economy* due to their *construct deficit,* the extension proposed within this work would be unable to, given its comprehensiveness. However, referring to Moody's principle, it states: the number of different graphical symbols should be cognitively manageable (Moody, 2009). When using the proposed language, at no point is the user ever presented with more than six symbols at once. Due to the method of visual grammar used and the collapsing and hiding of symbols, the user is never required to deal with high numbers of graphical symbols. Of course, too many symbols in general can make remembering them difficult and this is a key purpose of this principle. Nevertheless, by categorising them and only displaying a few at a time, it is possible to achieve *graphic economy* at a high symbol count. This is justified by once again referring to The Highway Code and traffic signs (Driver and Vehicle Standards Agency, 2015), which include over 100 signs in a multitude of different variations and yet are still a usable form of notation by the novice and expert. This is because, although a lot of signs/symbols exist, if used in a cognitively manageable manner (small numbers at once, categorisation), the user can understand and infer meaning from them.

7.1.8 Cognitive Fit

Cognitive fit as discussed earlier, feels like a somewhat dated principle nowadays. The component of *representational medium* is shifting away from traditional methods (pen and paper) and moving towards more interactive, digital content. Likewise, the idea that a language should have two notations is an unreasonable requirement to make; from both a developer and user perspective. The requesting of intentionally abstract concepts for "expert users" is ludicrous. Based on the requirements of this principle the proposed notation fails to satisfy it. However, basing the notation on its ability to be understood by both the novice and expert, it is more than capable of achieving this. Not only through the *modularisation* and *hierarchy* of the visual grammar but from user feedback (more on this later).

Of Moody's principles (disregarding *cognitive integration*), the proposed extension can satisfy seven of the eight; however, this work is confident in claiming satisfaction of what *cognitive fit* stands for. In contrast, the current best was four principles, achieved by both Saleem *et al.* (Saleem, et al., January 2012) and Salnitri *et al.* (Salnitri, et al., 2014). A table of all extension results can be seen in Table 7.1.

Table 7.1 Physics of Notations Principle Satisfaction Comparison

PoN Principle	Rodrígu-ez *et al.*	Saleem *et al.*	Salnitri *et al.*	Labda *et al.*	Yulia Cherda-nseva	Koh and Zhou	Case Study
Semiotic Clarity							✓
Perceptual Discriminability		✓	✓	✓			✓
Semantic Transparency		✓	✓				✓
Complexity Management							✓
Cognitive Integration							
Visual Expressiveness					✓		✓
Graphic Economy	✓	✓	✓	✓	✓	✓	✓
Dual Coding						✓	✓
Cognitive Fit		✓	✓		✓		✓

Based on Moody's principles alone, the proposed language has already proven itself the most superior of existing extensions. Not to mention, of the principles Saleem *et al.* and Salnitri *et al.* did satisfy, none of them represents exemplary attempts. In fact, for the most part, they just about scraped a satisfaction of each principle. However, the proposed solution provides very strong satisfactions of each principle and far superior design approaches.

Nevertheless, although outscoring existing solutions provides a quantifiable means of proving that this work has pushed the area forwards and in the right direction, the main objective is to create a truly comprehensive, complexity managed security extension for BPMN. Satisfying seven of the eight principles (almost all), specifically *semiotic clarity* and *complexity management*, indicates that this objective has been accomplished.

7.2 Heuristic Evaluation

From the review of existing extensions, a new set of requirements that security extensions should aim to satisfy were extracted. Although some overlap with the principles in the PoN, the satisfaction of these requirements adds another level of reliability to the results and the evaluation of the solution. In some ways, even more so, as these principles are based on security extensions rather than entire modelling languages like the PoN.

7.2.1 Comprehensiveness

Comprehensiveness represents the proposed sub component of *semiotic clarity* that requires the underlying ontology the language is built against, should be comprehensive to the domain. Although one could try and claim the proposed ontology as being entirely comprehensive to the security domain, this is very difficult to justify. Nevertheless, comprehensive refers to the inclusion of a majority with the aim of including everything[28]. In this sense, the proposed ontology is comprehensive to the security domain. The ontology was based on 14 existing ontologies and extensions, plus the inclusion of several other concepts. Given that current "comprehensive" security extensions to BPMN include no more than 11 concepts (in regard to security requirements/objectives that is), as 76 are included within the proposed ontology, this thesis is confident in stating it as comprehensive to the domain.

Security ontologies and standards in general, can differ dramatically from one company to the next, with many priding themselves on their in-house standards, particularly consultants. The main objective of the proposed framework in this sense, is to ensure that the end user is not restricted in their ability to specify their desired requirements. As a released product, the best route to alleviate potential discrepancies between each company, is to allow custom ontologies within the application which still adhere to the proposed visual vocabulary rules. For example, ensuring each level respects the seven plus or minus two rule (Tufte, March, 1956). These ontologies can then be certified by their respective creators, for instance LJMU. In this way, so long as all parties are utilising the same ontology, the number of variations has little impact.

Nevertheless, the ontology still represents the first comprehensive solution in regards to modellable security requirements in modelling languages, as such, it is more than adequate

[28] Oxford University Press; Dictionary: "comprehensive"; (https://en.oxforddictionaries.com/definition/comprehensive) accessed 10/05/2017

at satisfying the requirement of *comprehensiveness*. Not to mention, it provides a novel contribution of its own to the area.

7.2.2 Coherence

The requirement of *coherence* was proposed as a means of overcoming the existing issue of inconsistency in concept semantics. Within the proposed tool, this requirement is satisfied by including the definition of each concept within the details toolbar, as seen in Figure 6.12 and Figure 6.18. Although this does not necessarily solve the issue of security experts having varying definitions for each concept, it does ensure that anyone using the application will use each of these concepts based on the definitions in the details toolbar.

Moreover, this is obviously another area of large controversy, there is no objective here to challenge the area at large, claiming the previously defined definitions as the absolute definitive answer. The key point of this requirement is ensuring everyone is working in *coherence* within the language with the same understanding of how each construct should be used.

7.2.3 Structure

As discussed earlier, this requirement was proposed as an alternative to *cognitive fit* and as a way of avoiding *symbol redundancy*. In the proposed language, the requirement of *structure* acted as the key foundation for its visual grammar. Moody discussed how *hierarchy* can aid in the *complexity management* of diagrams. Therefore, it was theorised that the same should be true for language symbols themselves. By basing the proposed grammar on a combination of *hierarchy* structuring and *modularisation*, not only did this method the principle of *structure*, but greatly assisted in the overall *complexity management* of the extension.

7.2.4 Verification

Verification as discussed, is something all security extensions should include, ensuring that any requirements included, at the very least satisfy the necessary security protocols for such a system.

The inclusion of such a policy verification system has a lot of presence within the framework, outlining the various components required to ensure an effective system is implemented. Within the case study, the verification was simplistic in comparison to what industry would require, for instance, the security policies were created within the tool itself as opposed to importing from an external source. Nevertheless, the system was still capable of assessing a diagram against security policies and providing meaningful feedback to assist the user in correcting any discrepancies.

Although the case study does satisfy the requirement of *verification*, this is still an area worthy of more investigation, and as such represents the immediate future works following this project.

7.2.5 Graphical Framework

The requirement of a g*raphical framework* maps to the visual vocabulary component within the framework. For the BPMN security extension, the *graphical framework* can be seen in Figure 5.8. As per the requirement, this framework ensures that the inclusion of any further symbols will adhere to the existing design rules, unlike existing extensions which rely on a developer's own interpretation of a language's symbolic design. Although a somewhat brief analysis, the framework was built with the intention of satisfying this requirement. Therefore, so long as the respective component was satisfied (like in the BPMN case study), the heuristic can also be confirmed as satisfied.

7.2.6 Complexity Management

As discussed, *complexity management* was not a principle that was proposed within this work. It was included within the set of heuristics purely to emphasise its importance and negligence in existing security extensions. For the BPMN case study, this principle was evaluated in the previous section.

7.2.7 Modelling Tool Functionality

Modelling tool functionality represents one of the key areas current extensions have failed to acknowledge. As stated by Teyseyre and Campo (Teyseyre & Campo, 2009), current research practice is to propose or prototype a new visualisation method (or notation in this instance), without considering usability factors and user experience. Although it is not always necessary to develop a new application for security visualisation, considering the size of the security domain and the complexity of most existing languages, it is highly unlikely any existing solution could allow for the specification of both areas without incurring any of the previously discussed issues. Hence the dedication of an entire role within the framework to this requirement: *application developer*. The design of the language is of utmost importance and as mentioned is where most principles/requirements can be satisfied. Nevertheless, without a tool capable of adhering to these rules, the designs are fruitless. In the BPMN case study, *modelling tool functionality* was ensured through the creation of a new security modelling tool, one this thesis entitles BPMN 3D.

7.3 End User Feedback

Although the PoN and the proposed heuristics are both based on empirical evidence and deduction from scientific evaluation, they do not provide end user feedback. They both indisputably prove that thus far, the overall design and approach of the proposed solution (and framework) are far superior to current extensions. Satisfying a substantially higher number of requirements whilst ensuring *complexity management*, it effectively acts as the first solution to comprehensive security specification in the requirements phase of the SDLC.

Nevertheless, along with ensuring the reliability of the results through triangulation (Jick, December 1979), without user feedback there is no way of knowing for sure an end user will like and use the solution. Therefore, experiments with end users must be conducted to discover the potential of the solution becoming an industry standard for security specification in the SDLC.

Throughout the visual grammar component, several experiments were discussed regarding 2D and 3D modelling approaches. These results showed that at low levels of diagrammatic complexity, 2D and 3D produce relatively identical outcomes. However, when diagrammatic complexity is increased, 3D proved itself far superior to existing 2D approaches, not only through quantitative results, but with many users changing their visualisation preference after viewing the complex diagrams. Although these results are not quite strong enough to conclude BPMN 3D (proposed solution) as an accepted security modelling solution, they do provide a very strong foundation and justification for the use of 3D as a means of visualising security requirements in modelling languages.

7.3.1 Experimentation

Therefore, although the benefits of the solution can be theoretically proven, end user feedback provides the necessary reassurance that the application could be adopted. Furthermore, conducting experiments in which participants “try out” the solution is the best way to attain these results.

To better control how users test the application, the experiment was constructed in such a way that participants would effectively use the software as a real end user would within industry. That is, they were given an example BPMN diagram and asked to assign certain security requirements to BPMN elements. The only real difference being, the pre-determination of what security would be required, this is likely a decision an industry user would make themselves. Nevertheless, it provides a way of grading users once again based on their accuracy of symbol placement.

The experiment required the participants to first complete an anonymised background questionnaire. They were then asked to open the BPMN 3D application and receive a brief overview regarding what the respective BPMN diagram represented. The BPMN diagram being that in Figure 7.1.

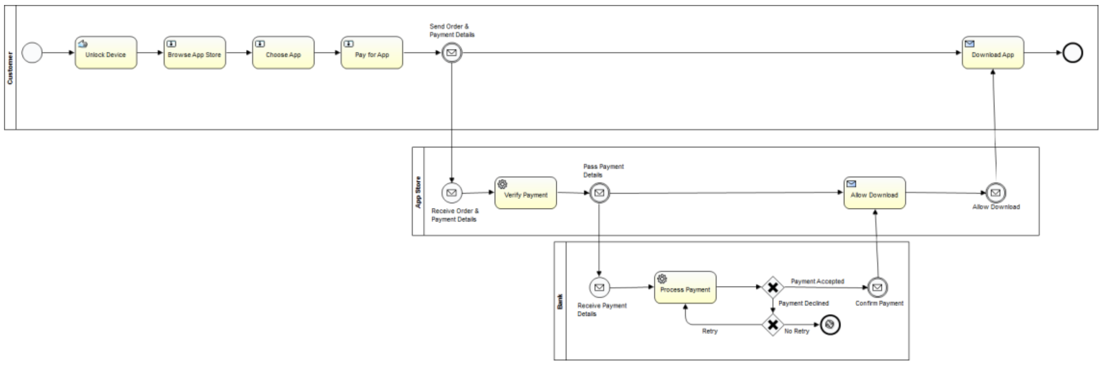


Figure 7.1 BPMN 3D experiment diagram

The diagram overview given to participants is as follows:

> *"This represents a common business process of a consumer purchasing an app from an app store. The consumer unlocks their phone using biometric authentication, they then navigate to the app store and begin to browse the available apps. Once they have chosen an app to purchase they input their payment details and send their encrypted order to the app store. The app store then pass these payment details to the customer's bank and request payment for the app. The bank decrypts these details and processes the payment, returning an error if the payment is unsuccessful. If successful however, the bank will then notify the app store of the payment transfer. The app store then sends a message to the consumer's device confirming the payment and authorising the app download."*

The accuracy of this diagram in terms of BPMN element usage, along with any other unnecessary or absent business process is of little importance to the experiment. This merely acted as a base template for assessing the language and tool.

As seen from the diagram definition, some indication as to what security will be specified was given. After participants had read over this and viewed the diagram, they were then asked to assign specific security requirements to certain BPMN elements.

There were seven questions in total, each one requiring some form of property specification as well as security requirement, with two of the questions requiring two security requirements to be specified. In total, the experiment required participants to specify 9 security requirements and 10 security properties. The hierarchy depth of these requirements varies from the highest level in the ontology to the lowest, ensuring the full functionality of the tool is explored during testing. Each correct question was awarded a total of one mark. This was earned by specifying the correct security requirement, on the correct BPMN element with the correct properties. In the instances where two security elements were required, this mark was split in half. If a participant managed to specify one correct security requirement they achieved a half mark; two correct requirements and they achieved the entire mark.

Before starting this task, participants were asked to note the time, noting it again upon completion. Once again, users were informed there was no award for completing the task at speed and were instructed to work at their usual pace.

As previously discussed, the application created within this work proposes several new techniques to both modelling languages and their respective UI. Therefore, it is important that feedback is obtained from end users regarding their opinion of this new functionality to better gauge whether it provides the intended user experience. Therefore, upon completing the main part of the experiment, participants were asked to fill out one final questionnaire with their feedback on the application itself.

Given that the objective of this final questionnaire is to gain feedback on the user's experience of the application (and its functionality) and not just a grading of how well they completed the task, the questions must be asked in such a way as to receive the required feedback. Therefore, it is logical to first define what functionality the users should provide their opinion on. Within the application, this functionality can be summarised into the following list:

- Adding/deleting a security requirement to a BPMN element
- Defining a security requirement property
- Locating elements within a diagram
- Navigating throughout the environment
- Focusing on a task/symbol using the UI
- Understanding and navigation the UI in general

This list summarises the core functionality of the application into high-level concepts. The next step, is to translate these concepts into questions. Taking the first concept "adding/deleting security requirements", this can be phrased in a questionnaire as:

1. *How did you find adding/deleting security requirements?*

Although some may argue this is an ambiguous question, that is only dependant on whether it is open-ended. In this case, a possible of four different answers are given to the participant, these being:

- *Very Easy*
- *Easy*
- *Difficult*
- *Very Difficult*

This list of answers now implies that the participant is being questioned on how difficult they found completing the respective task, removing any ambiguity. Likewise, the answers only allow for a positive or negative response. Although some may argue a middle ground or neutral answer should also be included, it can be difficult to draw any conclusions if all participants choose that answer. Therefore, in this instance, a neutral response was not included.

Moreover, some may ask why this question was phrased this way, opposed to say:

- *How easy did you find adding/deleting security requirements?*

This alleviates the previous issue of ambiguity in the question itself, effectively allowing for the question to remain open-ended and still provide similar results. However, the reason for not using this approach is that it leads participants to automatically rate the functionality based on how *easy* it was, removing any thoughts that it may have been difficult. Therefore, a less biased approach was used in the question, with any ambiguity being removed by the list of possible responses.

Regarding the remaining questions on the application's functionality, these are as follows:

2. *How did you find specifying security requirement properties?*
3. *How did you find locating security requirements and their respective tasks?*
4. *How did you find navigating around the 3D environment?*

5. *Did you find the 'Focus Task' and 'Focus Symbol' functionality useful?*
6. *How did you find understanding and navigating the user interface of the application?*

The possible responses for these questions being the same as those in the previous one. The exception being the *focus task/symbol* question (requiring a yes/no answer), as this is not necessarily a functionality all participants will have used. It was primarily included as a way of comparing participant data from those who struggled with 3D navigation. That is, one who struggles with navigation will likely use and benefit from the *focus task* functionality, those who did not struggle will not.

In summary, these questions will allow for a quantifiable way of grading the user's opinion on how difficult they found using each of the respective functionality. Although the main experiment offers a way of assessing how well or accurately a user can use the application, these questions offer an insight into the user's own experience.

With these questions focusing on the functionality of the application, to some extent a negative answer may not necessarily mean the user thought the language was poor, simply that the prototype is. Therefore, alongside these questions, it is useful to also ask the participant's opinion on the language itself in a more general manner. These questions being:

7. *What is your opinion on modularisation of security requirements (representing them in a hierarchical tree structure)?*
8. *What is your opinion on representing security requirements across the third dimension?*

The possible responses for these questions being like the previous set for similar reasons:

- *Very Good*
- *Good*
- *Poor*
- *Very Poor*

These questions will then provide more insight should a participant rate their experience poorly but the proposed language as good.

To conclude this questionnaire, it makes sense to simply ask the participants their opinion of the application as well. Although a poor user experience is not a desirable trait, sometimes the functionality an application offers can outweigh any other negativities it has. Therefore, the final questions the participants were asked are as follows:

9. *What is your opinion on this application?*
10. *Would you consider using an application such as this in the future to specify security requirements at design time?*
11. *Any further comments*

The first question uses the scale from the previous set of questions (*very good, good, poor* and *very poor*) for its responses; with the second question requiring a simple yes/no answer. The second question was asked to try and assess the possibility of the application being adopted in industry. Although one could assume that a user liking an application means they will use it, this is not always the case. Hence the addition of this added question to remove any doubts.

Once finished answering all the questionnaires, participants were asked to save their diagram and copy the saved file to a folder with their questionnaires. These folders were then given a randomised unique identifier and deposited in a shared folder for results collection.

To recruit participants, computing students were emailed a participation information sheet detailing the experiment and inclusion criteria, asking for volunteers (there was no reward for taking part). The inclusion criteria consisted of participants having basic knowledge of the cyber security domain, stating desired traits as experience with modelling languages.

7.3.1.1 Novice End Users

The experiment[29] was conducted initially with 19 first-year undergraduates studying cyber security-related degrees. The sample consisted of 18 males and 1 female with an average age range between 18-24 years old. To ensure there was no collusion during the experiment, it was conducted under exam conditions. All experiment results are based on a 95% confidence level.

Unfortunately, three of the participants failed to include the save file in their folder, leaving 16 participant data for the grading of the experiment. Two of the remaining 16 also failed to note their time, leaving 14 data for time averages.

[29] The experiment was pre-approved by an ethics board before it was conducted.

Nevertheless, of this 16, the average grade for each participant was 82% ±9%. The average time to complete this task for the participants who recorded it (14 in total), was 11 minutes 40 seconds ± 2 minutes 32 seconds. This equated to an average time of 79.56 seconds ± 16.7 seconds per security requirement and property specification.

Of course, no other security extension exists with the level of comprehensiveness the proposed solution offers. Therefore, the comparison of this data against another extension will provide relatively meaningless feedback. Nevertheless, several positives can be taken away from this data. For example, for this sample of participants, given that they are all in their first year of studies and aged between 18-24 years old, it is reasonable to state that they are novices in cyber security and modelling languages. This claim was confirmed from inspection of the background questionnaires. Therefore, they can offer valuable feedback and results from a novice perspective. (Remember Moody's principle of *cognitive fit.*)

The main objective of this experiment was to assess the likelihood of the solution being adopted by end users, maybe even becoming a standard. The experiment, acts as a way of controlling and ensuring that participants used the application as intended in a real-world scenario; this subsequently allowed for the collection of data on the accuracy and time it takes a user to complete these tasks. As stated earlier, these participants are novices in security and modelling languages and therefore have no prior knowledge to assist in this process. The only help they received was a basic user manual on how to use the application. Therefore, like with any discipline, it is highly unlikely one would ever see an average accuracy of 100% for a complete novice. However, comparing the grade of each participant against the average time they spent on each security element, there is a noticeable correlation between the data.

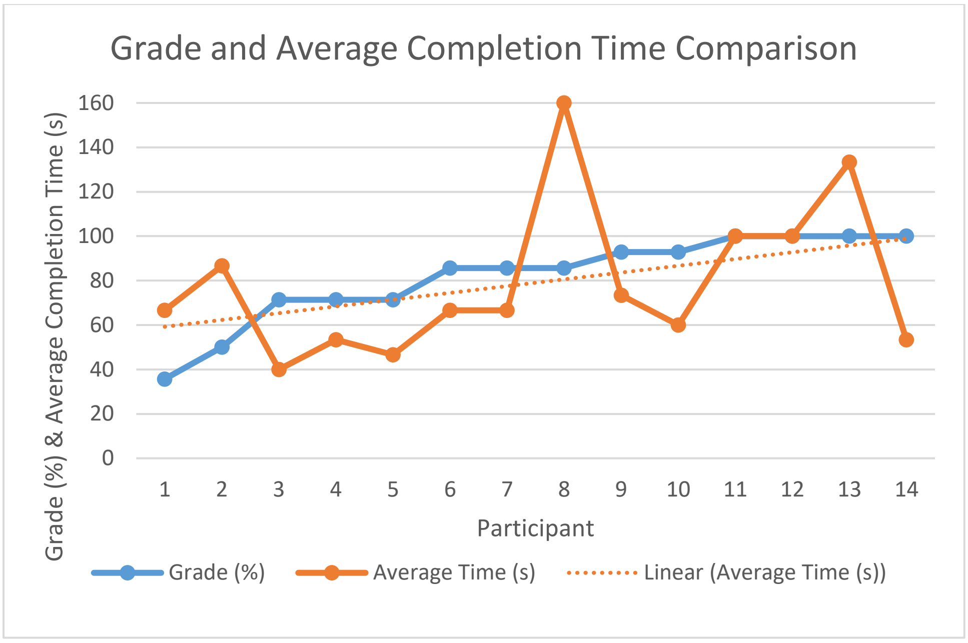


Figure 7.2 Participant grade and average completion time comparison

This graph plots both the participant's grade (in percentage) and average time per security element (in seconds) on the same set of axes. This allows for a better comparison of the data and assists with noticing things such as the previously mentioned trend. The results are ordered per their grade, from lowest to highest, represented by the blue line on the graph. The average time per element is represented by the orange line. The average time per element also has a trend line, as certain anomalous values such as participant 8 and 14 can make it difficult identify a correlation otherwise.

From inspection of the graph, it is reasonable to state, the longer a participant spends per each security element specification, the higher grade they will achieve. Although it was discussed earlier that novices to a discipline rarely score 100% on their first attempt, these results suggest that in this instance, the lack of a perfect grade is more likely due to participants rushing the experiment than not knowing how to perform the task correctly. Of the four participants who achieved a grade of 100%, three spent at least 100 seconds specifying each security requirement. The other participant scoring 100%, represents one of the anomalous results mentioned earlier (participant 14), who managed to achieve a perfect score with an average time per element of 53.33 seconds; the integrity of this value unknown. Likewise, the only other participant in the experiment who took longer than 100 seconds but did not achieve a 100% grade, is the other anomalous result (participant 8). Therefore, disregarding these results, focusing on the trend line, it can be said with some confidence, the longer a user spends specifying each requirement, the more chance there is of perfect specification. The ideal time to spend per element in this instance, being at least 100 seconds.

The results from the final questionnaire showed very positive feedback for the language and tool. When asked on the difficulty of adding/deleting security requirements, 5 participants responded *very easy* with the remaining 14 responding they found it *easy* (most questions providing results from all 19 participants). As for specifying properties of security requirements, 2 responded they found it *very easy*, 16 responded it was *easy*, with just 1 participant claiming it was *difficult*. This participant however, scored a grade of 92.86% in 9 minutes, getting just one security requirement wrong. Maybe not as difficult as they claim.

The results regarding the ease of locating BPMN tasks/security requirements came back as 4 found it *very easy*, 11 found it *easy* and 4 found it *difficult*. As for the 3D navigation, 3 claimed it was *very easy*, 9 stated it was *easy*, with 7 saying they found it *difficult*. Reading the comments from two of the participants who said they found navigation *difficult*, it appears they were unaware the application included a fly-cam mode. Both suggested to include the functionality in their feedback (this functionality was made aware to participants verbally, throughout the application itself and inside the manual they were given to support them during the experiment).

For the opposite side of this question, 9 participants stated they found the *focus symbol/task* functionality useful, 9 stated they did not and one participant did not respond. This functionality of course being a secondary form of navigation to assist users who struggle to use the fly-cam mode. The aim of this question was to discover if any participants found this functionality useful, not necessarily them all. Given that the results are an even split, the inclusion of this functionality within the solution can be justified. The final question in regards to the application itself, asked participants how they found understanding/navigating the user interface. 5 participants responded they found it *very easy*, 13 found it *easy* and just 1 claimed it was *difficult*.

Overall, these results are very positive in regards to the application, and subsequently the language itself. The only question in which negative feedback made a noticeable impact was on navigation of the 3D environment. This is something that was predicted at design stages and discussed several times. However, the feedback for this question was still generally positive with only some users having difficulties. Two results as mentioned, potentially may have given a different answer, had they paid more attention during the experiments and known the existence of the fly-cam.

Regardless, this issue could potentially be alleviated through more experience with the application. The controls are very like that of 3D video games, specifically first person shooters, likely explaining why so many participants in this age group had no issues. According to literature, the average age range for this genre is around 18-30 (Jansz & Tanis, 2007; Tekofsky, et al., 28th - 30th June, 2016). Given that 13 participants claimed to use 3D applications every day and 6 claimed at least once a week, leads to a reasonable assumption this was a contributing factor.

For the more language-focused portion of the questionnaire, the first question asked participants for their feedback on modularisation of security requirements. 8 participants responded that they thought it was *very good,* with the remaining 11 stating it was *good*. A complete acceptance of one of the core visual grammar mechanics by novice end users. As for their opinion on representing security in 3D, 11 participants responded they thought the concept was *very good*, 6 responded *good* with 2 responding it is *poor* (no comments were left by the participants who stated it as *poor*). Nevertheless, most participants gave the most positive feedback available (*ve*ry *good*) for 3D representation of security, this is a very good response to the language. As for participants' opinion on the application itself (bearing in mind it is a prototype), 7 responded it was *very good*, 10 that it was *good* and 1 that it was *poor*. Questioned on whether they would use an application such as this in the future for specifying security requirements at design stages, all 19 participants responded that they would.

Overall, the data from these experiments has been very positive, supporting the language and tool through both quantitative data (grading) and user feedback. Proving that, novice security users with no prior training in modelling languages can achieve reasonably high grades of accuracy when specifying security requirements, if the language is well designed. Given that this application is only a prototype, the feedback from these results is overwhelmingly positive, with the only negatives being a few users struggling to use the fly-cam. Most importantly though, all participants stated that they would be willing to use an application such as this for specifying their security requirements at design time, something which is impossible to deduce from heuristics and the PoN.

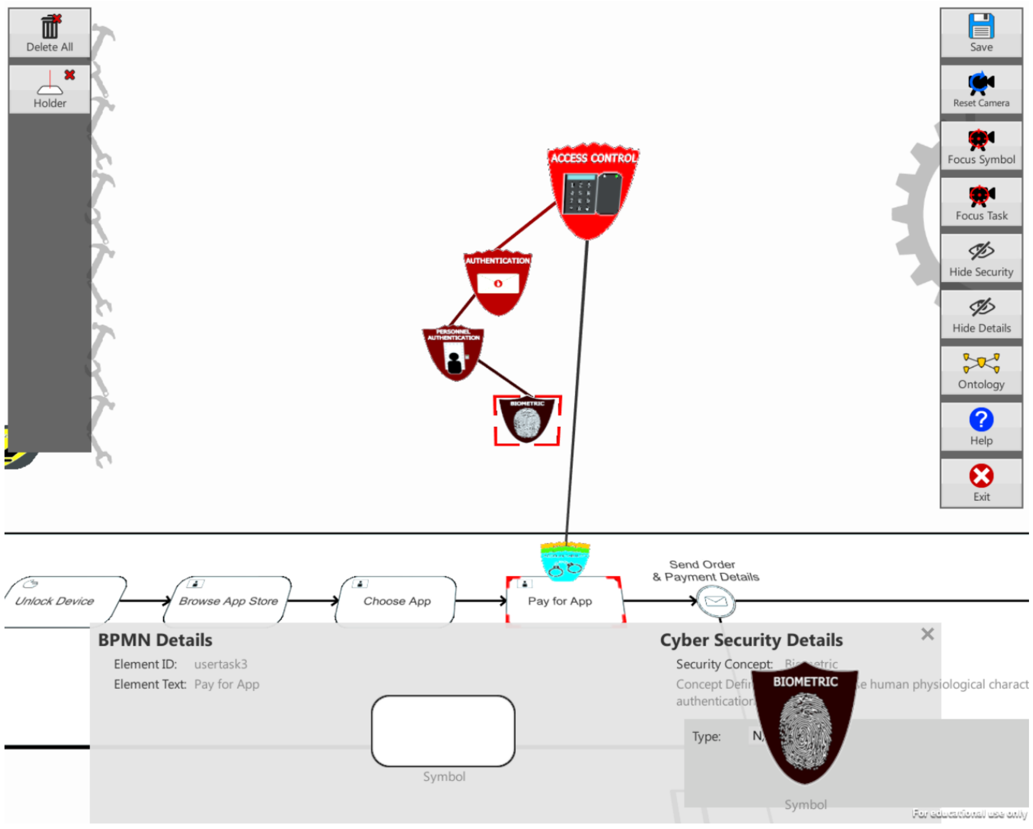
7.3.1.2 Expert End Users

Understandably, it is a lot harder to find expert end users with the time to spare for experimentation. However, an opportunity to conduct experiments with several participants at a cyber security company in China arose, and as such, provided the necessary sample (naturally, the application had to be localised to the Chinese language). In total, the experiment was conducted with 11 employees in the company; their years of expertise in cyber security averaged at 6.5 years. Two of the participants had previous experience with modelling languages, that being UML. The sample consisted of 4 females and 7 males, with an even distribution of age groups (1: 18-24 years, 5: 25-34 years and 5: 35-44 years). Once again, all experiment results are based on a 95% confidence level.

As with the novice end user experiments, two participants failed to submit their save file for the diagram, meaning their results cannot be included in the grades. Of these two, one also failed to submit the final questionnaire.

Nevertheless, the results for the remaining 9 participants' grades averaged at 68% ±15%, with an average completion time of 15 minutes 48 seconds ± 3 minutes 20 seconds. This equated to an average of 106.67 seconds ±23.46 seconds per security requirement.

Comparing these results with the novice users, there is a rather noticeable difference in both the average grade and completion time. Upon inspection of the questionnaire comments, and verbal feedback from the participants, it was discovered that several of the machines the experiment was conducted on, had a screen ratio of 4:3; the application was built and tested on a screen ratio of 16:9. Had this been an industry ready piece of software, this issue would have been alleviated through more robust testing at development. However, in this instance, the application was only tested on 16:9 ratio screens, unbeknown that this issue would occur. The impact the difference in resolution had was relatively minimal for the most part. Primarily, it affected the details toolbar making the specification of security properties very difficult, see Figure 7.3.



BPMN Details

Element ID: usertask3

Element Text: Pay for App

Cyber Security Details

Security Concept: Biometric

Concept Defi… e human physiological charact… authentication

Type: N

Figure 7.3 Application screenshot 4:3 screen ratio

Although some participants did try and work around this issue, it will have inevitably affected their overall time to complete the experiment and somewhat corrupted these results. Likewise, other participants did not attempt to specify properties, as understandably it is not a very user-friendly way to interact with an application. Therefore, the comparing of these results against that of the novices will not provide any meaningful data as the expert users were unable to specify properties as easily.

Nevertheless, a rather significant portion of this experiment can still be salvaged (given the difficulty of finding willing experts to conduct experiments, this was a necessity). Considering the visual grammar in terms of novel contribution, the method for specifying instances of security requirements does not represent any significance in this aspect. For property specification, typical functionality of form input and details toolbars was exploited, both of which are already proven and heavily utilised components within computing. The proposed grammar's novelty comes in the form of the users' ability to specify and interpret the modelling language itself. That is, the symbolic representation of security requirements and the associated tool's UI.

Although security properties are a part of the visual grammar, considering that their results bring very little in terms of proving any novelties and contributions, there is no need to dismiss an entire experiment with experts simply because they were unable to use this functionality. Instead, the element of time taken can be dismissed, as this data is too unreliable now given the impact of the resolution issue. The specification of security properties can then be removed from the grading system, awarding a mark instead for the correct specification of a symbol, as opposed to a symbol and its respective properties. If the novice and expert users are then re-graded based on this new grading criteria, there will be comparable data from both areas of expertise. The inclusion of property specification and time will of course have strengthened this data, but the core novelty of the work can still be assessed using this new approach.

Under the new grading system, the results from the novice end users provided an average score of 88.27% ±9%. As for the expert users, they provided an average score of 87.3% ±9%. The difference in these results is virtually non-existent, with both percentages equating to 6.18 and 6.11 marks out of 7 respectively.

Although these grades do not include the entire visual grammar as mentioned, they do focus on the novelty of the language extension. No claims are made to using forms for data input, they were merely adopted for specifying very low detail given their high use in industry and modelling languages themselves. However, specifying security requirements in 3D using a combination of the proposed *complexity management* approaches both in the language itself and through the UI approach, *is* something this work makes claims to and presents as a novelty. The above results represent the testing of this functionality. Although there is no reliable time data for comparison, given that users were told to work at their own pace, the grades they achieved in relation to security requirement specification are still very useful.

The measuring of security symbol specification in the language is of vital importance for many reasons. Unlike current extensions, which all adopted existing techniques of symbol specification, primarily drag and drop, the language in this thesis proposes several new approaches. For example, through the *modularisation* of UI elements and of the symbols themselves (at a symbol level not diagram level). Not to mention that the language is in 3D, something not yet seen by users. Therefore, although these experiments may provide very little when tested against existing security extensions, as the language offers brand new, untested approaches to the area, there is a need to assess the users' experience of this functionality as well as their ability to use it.

Going back to the results themselves, one thing that can be stated right away, is that no user group separated themselves as being more able to use the tool; the results were essentially identical for both. Not only in their average but in their margin of error. Given the emphasis Moody places on creating a language which is accessible by both the novice and expert (Moody, 2009), no such difference was identified between the groups in the results. Therefore, on the assumption that Moody's claim of there being two different identifiable groups of users with different requirements, is true, the proposed language must therefore be equally accessible to all areas of expertise and satisfy this requirement.

It also plausible that, given how close both averages are when dismissing properties, had the experts not experienced the issues with screen resolution that they did, they would have scored very similar results to the novice end users for property specification.

Inspecting the results from all users (both novice and expert), one other trend in regard to where marks were lost can be observed.

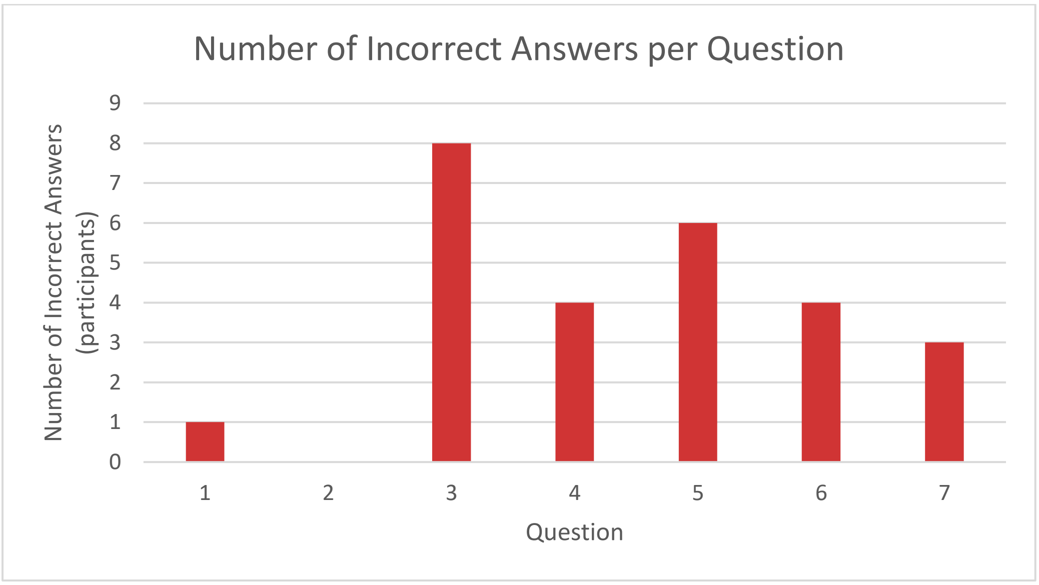


Figure 7.4 Number of incorrect answers per question

Although difficult to undoubtedly make any claims given the low numbers involved, there are two noticeable peaks on questions 3 and 5. Upon further inspection, these were identified as the questions that required participants to specify multiple security requirements.

When designing the questionnaire, it was quite difficult to specify the necessary information for each question in such a way the user would understand what they were required to do (if only there was a modelling language for visualising it?). It was eventually decided to take the approach seen in Table 7.2.

Table 7.2 Experiment Questionnaire 2 - Question 3

BPMN Element: **Send Order & Payment Details**	
Security Requirement	Properties
Encryption	Type of Encryption: **Symmetric** Key Size: **192**
Need to Know	n/a

The key difference between this example which requires two security requirements and one which requires one, is the absence of the extra row at the bottom. This approach of using a table, which details the respective BPMN element, required security elements and associated properties, presented itself as the best approach to take for instructing participants in regards to the required security specification. When trying it out with a few individuals in a light pilot study, they seemed to grasp the concept easily, consequently leading to its adoption in the main experiment (no data was recorded for these pilots, just observations).

However, as seen from the graph, it is apparent that there was obviously some confusion when multiple requirements were involved. For both questions (3 and 5), 14 participants across both experiments dropped marks. The fact that leads one to believe the multiple requirements was the issue, is that of the 14 participants, 13 of them specified one of the requirements correctly. Therefore, it is reasonable to theorise, participants either did not realise both requirements needed specifying or simply did not notice the extra one in the table.

Had these questions been better presented, or the participants been made more aware of how to read the table, it is plausible that they may have achieved full marks as opposed to just half, which would have equated to an average grade of 92% ±7%. However, without more data to confirm this, no reliable conclusions can be made on whether this was indeed the case.

Moving now onto the final questionnaire for the expert end users. Regarding the difficulty of adding/deleting security requirements, 4 participants responded with *very easy*, 3 with *easy* and 3 did not respond (as mentioned earlier, there are results from 10 participants for most of questionnaire 3). The question on the ease of specifying properties came back at 3 for *very easy*, 5 for *easy* and 2 for *difficult*. However, given the problems participants had with specifying properties, the reliability of this result is questionable, even if the majority is positive.

The ease of locating security and BPMN elements came back as 1 for *very easy*, 7 for *easy* and 2 for *difficult*. As for the ease of 3D navigation, 1 responded *very easy*, 5 with *easy,* 3 with *difficult* and 1 with *very difficult*. These numbers being around the same ratios as novice users. In this instance, however, there was 1 *very difficult* for 3D navigation. As mentioned earlier, the controls for this camera do translate from typical 3D video games, should an end user have had no experience with a system like this before, it can be quite difficult at first. Given that 5 participants claimed to have never used a 3D application before (highly unlikely), and 4 claimed to only use them once a month, this likely explains this difficulty with camera controls.

Nevertheless, the requirement for more and easier navigation techniques is becoming a trend in the results. With that said, when asked about how useful the *focus task/symbol* functionality was, this time 8 participants responded they found it was useful and 2 said they did not. This supports the earlier theory that the more difficult a user finds 3D navigation, the more they will utilise and appreciate the UI navigation.

As for the questions regarding the language itself, 3 participants responded that they found the *modularisation* of security requirements *very good*, with 7 responding it was *good*. Once again, a complete acceptance of *modularising* elements. As for visualising security elements in 3D, 3 responded with *very good*, 6 with *good* and 1 with *poor*. On their opinion of the application, 1 responded *very good*, 7 with *good* and 2 with *poor*. As for whether they would be willing to use an application such as this for specifying security requirements at design time, 8 responded they *would* use it, 1 responded they *would not* and 1 did not answer.

Overall, even with the issues surrounding property specification, the feedback for the application and language from experts has once again been very positive. There were a few complaints about the camera system as mentioned. For the most part though, participants showed very positive feedback. A novice stating they would use the application is good in terms of assessing its usability, but it does not hold much value considering their naivety to industry practice. However, to have most experts state they would use the application, given their wealth of experience, this represents a very strong acceptance of the language and tool. (A table of results for these experiments can be found in the Appendix.)

7.4 Chapter Summary

Throughout this chapter the proposed solution was compared against the PoN and proved itself to be far superior to existing approaches satisfying a total seven of the eight principles (disregarding *cognitive integration*). However, as discussed, regarding *cognitive fit* and the requirement that the notation should be readily understandable by both the novice and expert, the work is confident in claiming satisfaction.

End user experimentation was then conducted with novices (undergraduates) and experts (industry professionals) to the cyber security domain. The results from these experiments showed, that each group could utilise the solution equally irrespective of competency. Regarding user feedback, both groups of participants responded very positively to the application claiming it is a tool would they would use in the future for specifying security requirements at design.

In addition to the data from the 2D versus 3D experiments in Chapter 5, the application has proven itself as being more than capable of satisfying the requirement of an expressive and usable secure by design tool.

Throughout the next chapter, the conclusion of the thesis is detailed along with the novelties and contributions made within the work and future work.

8 Conclusions

8.1 Thesis Conclusion

At the beginning of the thesis, there was a discussion about how cyber security remains a large issue and one that cannot be solved with a single solution. The absence of no formal tool for specifying security at the requirements' phase of the SDLC was then identified. Along with a poor attitude towards the importance of secure by design, this has led to security being included ad-hoc after the implementation of core system functionality. This has been proven to produce poor defences, potential redesigns of system functionality and consequently large costs to the company.

Therefore, the solution to this problem, is to produce a tool which allows for the specification of security at the requirements phase. Typically, the requirements phase is conducted using modelling languages, with BPMN the industry standard for business process modelling. Therefore, this language in-particular offers the perfect platform for the specification of security requirements early in the SDLC.

Throughout the literature review, several attempts at extending BPMN with security specification were evaluated, all of which, proved inadequate solutions. This not only further emphasised the need for a solution, but highlighted the general poor methodology of existing approaches. Analysing each extension against the PoN, trends began to emerge in regard to how each author approached the problem. That is, a very intolerant attitude towards modelling language design. Almost all authors showcased limited or no attempt at adhering to modelling design principles, showing a complete disregard to the complexity increase associated with BPMN extension.

Therefore, a very different approach to existing attempts was taken within this work, considering all known issues before beginning any design of a new solution and opting for a careful consideration of required components. This came in the form of a framework which can be used for the extension and visualisation of security requirements in any modelling language. Although it is stated at several points throughout the work that BPMN represents the optimal language for targeting security in the requirements phase, the framework has no reliance on what core language is used. Therefore, there is no need to restrict its contribution by targeting just BPMN.

The benefits of the framework come in the form of providing the first set of proven guidelines which outline the necessary components needed for creating a security extension which does adhere to existing design principles. This is very much something that does not exist and can benefit the area at large. As mentioned, there have been several attempts at specifying security in BPMN, even more so in UML. However, current authors need either to conduct extensive literature reviews themselves to ensure a well-designed language, or simply ignore existing principles and implement their solution blind. The framework, negates this issue, providing authors with a detailed list of components which, if followed, will ensure the implementation of a well-design security extension.

This framework has been proven by developing a new security extension for BPMN, one which could satisfy most of the PoN. This of course possible because of the components within the framework, many of which, were largely influenced by the principles in the PoN and the proposed set of heuristics.

Moreover, although the BPMN 3D tool has the potential to be a large contribution on its own, considering the feedback to this solution, which was only a case study based on the framework, the framework itself has the potential to produce a similar tool for every other language and an overall much larger contribution.

In summary, the paucity of a secure by design tool at the requirements phase was identified. Through the evaluation of existing solutions, known issues were extracted. A framework was then implemented with the intention of overcoming these issues and creating a well-designed solution. The case study on this framework, managed to satisfy a large portion of the PoN, and received positive feedback through experimentation and industry contacts. Therefore, in regard to the identified problem at the beginning of the work, the methodology taken to alleviate this issue and the respective evaluations has proven the proposed solution. Overall, this thesis is confident in claiming that the work conducted throughout the project has provided sufficient novelties and contributions to the required standard.

8.2 Novelties and Contribution

Throughout the following section, the main novelties and contributions of the work will be highlighted and discussed.

8.2.1 A Framework for the Extension and Visualisation of Cyber Security Requirements in Modelling Languages

As implied by the thesis title and at several points throughout the work itself, the core novelty of this project is the framework which allows for the expressive and usable, extension and visualisation of cyber security requirements in modelling languages. As previously discussed, this framework represents the first methodological approach of implementing a security extension. Unlike existing solutions, by using the proposed framework and ensuring the satisfaction of each component, as proven by the BPMN case study, the outcome will generate a fully comprehensive, complexity managed security extension.

The beginning of the work discussed how secure by design should be considered a necessity for system development, but there are currently very limited solutions available for doing so. Current approaches to software development utilise modelling languages for the designing of their system. Therefore, they provide the perfect platform for specifying security. As mentioned, defining security at this stage, ensures any functionality is then built into the system *with* security rather than as a prerequisite to ad hoc security later.

Throughout the literature review several security extensions to BPMN were discussed which allow for the specification of security requirements. However, as discussed, these solutions are not only unable to adequately represent comprehensive security requirements, they contribute very little to the area in terms of specifying security within modelling languages.

The proposed framework, acts as a methodology (blueprint) for security extension which can be utilised alongside new or existing modelling languages. It provides developers with the necessary guidelines for implementing their own security extension, one which will satisfy existing design principles and ensure the end user has enough notation for meaningful expression. This is not just a one-language, one-software solution, but a framework that can be applied to any language to create infinite extensions. As mentioned, the potential and purpose of this framework to security extension is equivalent to the waterfall model in software development.

8.2.2 An Ontology of High Level Modellable Cyber Security Requirements

Although the framework can effectively encompass the other novelties, the others provide enough novel contribution on their own to deserve standalone recognition. The first of which, is the ontology of potentially modellable cyber security requirements. As previously discussed, there are a multitude of security ontologies in literature, far too many to cover within this work. However, to the best of this work's ability, none were identified which provide a comprehensive overview of high level security requirements which have relevant meaning when modelled within a language.

Therefore, after conducting a review of existing ontologies and security extensions within BPMN, the guidelines of Noy and McGuiness (Noy & McGuinness, March, 2001) were followed to create a new ontology. Although there will be a lot of controversy in regards to where concepts should be placed, this solution still represents the fist comprehensive security ontology which acts as a checklist for modelling language security extension. Whether this abides to the consensus of the area or not, this work represents the next (or even first) step towards a potential standard in the domain. By conducting a review of existing extensions, then contributing to this with several additional concepts, the proposed extension offers a significantly higher number of concepts than any other approach, making the ontology a significant contribution to the area of cyber security modelling on its own.

8.2.3 A Graphical Framework for the Creation of Perceptually Discriminable, Semantically Transparent, Visually Expressive Symbols

The next contribution, although maybe not as significant as the ontology, is the visual vocabulary framework used in the BPMN case study. Although this does not appear as large a contribution as the ontology, there is clearly a need of a framework for symbol design.

When viewing existing extensions, current approaches have followed more of an artistic expression approach rather than design principles. The PoN, although not entirely focused on symbol design, offer several proven principles to improve the robustness and potential lifetime of a notation. Using these, a new framework was created for designing the symbols within BPMN 3D, one which requires very minimal input from developers when adding to the notation, as most visual variables are predetermined. This framework can be applied to any other domain quite easily. For example, if there was a need to create a notation for cars categorised by their type, using the framework one could quickly create a notation like that in Figure 8.1.

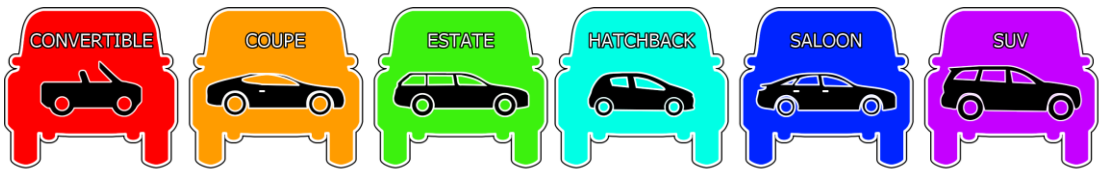


Figure 8.1 Visual vocabulary framework example

Of course, there is still some initial work required: deciding on an outer shell, choice of icon, categorisation method etc. However, this framework once again provides a foundation to build upon, one which has been successful at creating a notation consisting of 76 symbols. Existing approaches differ only on shape, by using this framework, there is an assurance a larger portion of visual variables will be fully utilised and by consequence an overall better notation will be created.

8.2.4 A Complexity Managed Visualisation for Comprehensive Domain Expression

The visual vocabulary, leads nicely onto the next contribution of the visual grammar used in the BPMN case study. Once again, although in BPMN 3D the visualisation focuses on security requirements, it is more than capable of showcasing alternative information.

The reason the visual grammar can act as another key contribution of the work, is because of its novel approach in managing large numbers of concepts whilst still maintaining *complexity management*. By utilising the third dimension and existing *complexity management* principles (*modularisation* and *hierarchy*) in a somewhat unconventional manner (at symbol level rather than diagram level), the proposed solution managed to successfully visualise a total of 76 security requirements per BPMN element without incurring complexity issues. Something existing solutions are struggling to accomplish with a measly ten concepts across an entire diagram.

As an example of how this visual grammar can be used for a different domain, National Grid information will be displayed across a map. Referring to Figure 8.2, one will be able to see an example of how using this visualisation method, information can be displayed on the three core utilities (*water, gas* and *electricity*). In this example, how much of each resource is remaining in the L3 postcode based in Liverpool was used, with the sub-concepts detailing resources at various locations. (This of course all hypothetical.)

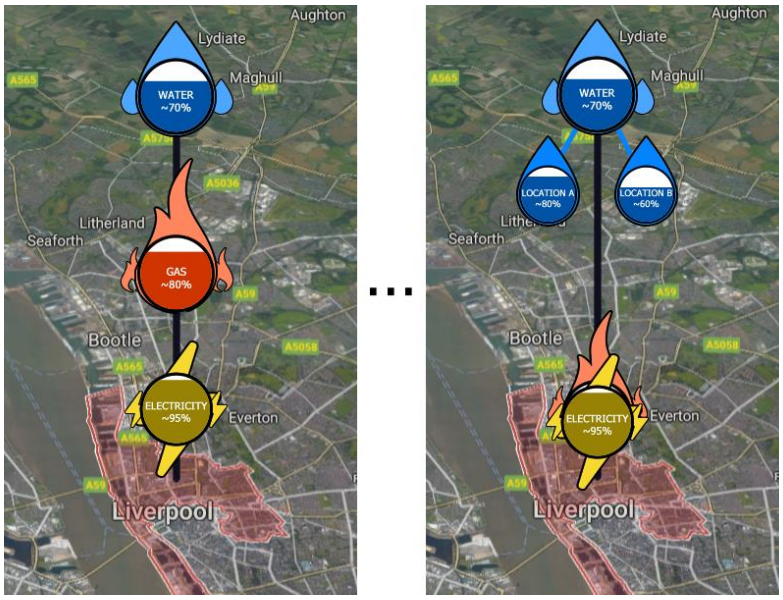


Figure 8.2 Visual grammar framework example

This is a good example of how the core two dimensions (*x* and *y*), have already maxed out their design space (with map data) and are unable to hold any more information. However, by utilising the proposed visualisation, a developer only need apply minimum effort and they will able to visualise detailed, complexity managed information on their domain.

As well as the BPMN case study, the two previous examples on the proposed visual vocabulary and grammar showcase the contribution they bring to the area. By using the respective frameworks to create each of these examples, mock-ups could be created in less than an hour. In a very short space of time, being able to design a visualisation for displaying a whole new domain of information across existing different domain data, which will also be complexity managed and comprehensive to its own domain, is something of great value to the area. The benefits of these sub-frameworks and their potential in visualisation reaches far beyond that of security and modelling languages.

As mentioned earlier, these sub-frameworks can be encompassed by the larger framework. However, in doing this, the contribution of everything is restricted. Although each of these methods has been proven in their respective area, to place them as “the one solution” for their respective components, greatly reduces the contribution of the frameworks at large. By no means are the proposed visual vocabulary and grammar the only possible solutions for representing comprehensive, complexity managed security requirements in modelling languages. However, the framework outlined in Chapter 4, does represent all the necessary requirements for constructing such a solution. Therefore, the main framework is proposed as the key novelty and contribution. With the sub-frameworks (vocabulary and grammar) as separate, usable and proven solutions for creating a notation and language extension (or data visualisation).

To better understand this, an analogy of the car industry can be used (somewhat of a theme at this point). Representing existing BPMN security extensions as cars that do not run, the framework would represent a blueprint of what is required to create a car that will run. The visual vocabulary and grammar then represent one possible outcome of a car that runs based on the framework.

To state the proposed visualisation approach as the only solution in the framework, would be the equivalent of stating there is only one solution to create a running car. The main framework and its components although complementary, provide their own set of contributions. Used in conjunction, they can produce a comprehensive, complexity managed security extension. However, used separately they bring their own novelties and contributions beyond that of the case study, ones that should not be restricted by encompassing them all under one title.

Should other researchers wish to explore alternative methods of security visualisation, the framework acts a solid foundation. Nevertheless, the proposed method of visualisation can also act as a foundation for further research into security visualisation or any other data for that matter. These frameworks can provide one large contribution together as well several smaller contributions on their own. Therefore, they are presented as one -*and* separate- novelties and contributions to the area.

8.2.5 Minor Novelties and Contributions

As well as the previous contributions, there are a few other minor contributions that can be discussed. Although their impact and potential in the area are not as significant as the former, they are still worth highlighting within this work.

The first being the set of heuristics defined in Chapter 3. As discussed, the PoN provides a set of principles for both the aiding in design and evaluation of modelling languages. However, this work focuses on the extension of an existing language with different domain requirements. As such, there are more principles that can be defined which will aid in this process, the proposed heuristics highlighting some of these. Primarily, the ones which are not as explicit within the PoN are: *coherence, structure, graphical framework, tool functionality* and *verification*. Each of these concepts represents a desired outcome for any security extension and as such provides a small level of contribution to the area.

The next contribution (BPMN 3D), though not providing much in terms of novelty, does provide the area with the first comprehensive, complexity managed security extension tool for the BPMN language. Although the BPMN case study was a means of evaluating the framework, given that it managed to successfully represent comprehensive security requirements within BPMN, it still acts as the first solution for true BPMN security specification at the requirements phase of the SDLC.

Overall, the primary contribution of this work is the framework outlined in Chapter 4; this, along with the ontology, symbol creation framework and visualisation approach. This thesis is confident in stating that the work conducted throughout this project has been significant and relevant to the area. Literature has proven there is a clear desire for a secure by design solution, and the work has produced a framework outlining how to achieve this, with a BPMN security extension tool proving it; advancing the area closer to a standard for modelling language security specification support.

8.3 Further Work

When discussing the future of this research, there are multiple avenues that can be taken, some which will provide significant contributions, and others which will refine the proposed solution with more usability features.

Firstly, from the discussions during ICURe (see Chapter 9), it was discovered that there is a large market for a security validation tool which can assess business processes against government security policies, amongst others. Although the proposed tool is not yet capable of achieving this, due to the work conducted within this thesis, the foundations for implementing one are now in place. Given that previous solutions are unable to achieve comprehensive security specification, a government policy verification tool is impossible to implement as many requirements will have no representation within their language. However, the proposed tool is comprehensive to the domain. In the unlikely event, there is a paucity of some concepts, the frameworks still provide the necessary guidelines for extending the solution without compromising its usability. Therefore, the implementation of an external policy verification tool is the logical next step in this work.

Another area worth investigating, which in many ways would benefit from parallel research alongside policy verification, is the grading of a diagram based on its security requirements. This could be determined either by comparison against the ontology or as a means of providing feedback on what portion of a policy a diagram satisfies. This system acting as the future works to determine the most accurate and optimal solution for grading the security of a diagram.

Another element of the work which still has potential, is the proposed visualisation. As discussed, there has been some exploration into 3D modelling. However, the approach in this thesis incorporates other practices to provide an overall more complexity managed solution. There is also the possibility of applying the visualisation to other domains. As seen earlier in Figure 8.2, the proposed solution is more than capable of visualising other domain data outside of modelling languages. Therefore, there is a chance it could be a potential solution to problems in other areas, this possibility alone is worth investigating.

Beyond this, there are several other elements of functionality this project aims to include within the BPMN 3D solution. Although not providing anything significant in terms of research progress, they will improve the overall usability of the application itself.

The first thing to include, being the ability to create and load custom ontologies. As discussed earlier, the controversy surrounding security ontologies feels like an eternal debate. Therefore, providing end users with the ability to create and publish ontologies to an external online database will effectively solve this issue for us. If company X creates an ontology, so long as their partners or anyone else using the tool in coherence are using the same one, there is little impact in terms of any potential issues. Issues mainly arise when company X are using one ontology and company Y are using a different one. Publishing and certifying an ontology ensures end users can collaborate and utilise whichever one they prefer.

Likewise, with the inclusion of new concepts comes the issue of new notation. Therefore, to once again overcome this problem, the aim is to implement a 'symbol generator'. That is, a tool in which the end user provides a concept name, definition, hierarchy location and then some form of unique icon. The tool will then handle the correct placing of the symbol in the respective ontology tree and the visual grammar itself, not to forget the creation of the symbol itself. As discussed earlier, the graphical framework used in BPMN 3D has a lot of predetermined visual variables based on the previous details. Therefore, the implementation of such functionality will be relatively straightforward and increase the lifetime of the tool itself, providing end users with some form of customisation of the security language.

As mentioned, the previous attempt at modelling BPMN in 3D was conducted by Brown *et al*. (Brown, et al., 2011). The main objective of their work was to use Second Life[30] as a means of collaborative modelling of the BPMN language. The fact that it was in 3D happened to be more of a consequence of Second Life being a 3D virtual world, rather than Brown *et al.* exploring any possibilities a third dimension can bring to modelling languages. In regards to collaborative modelling of the BPMN language, it would be much easier and usable to keep the language 2D and give each user a different coloured cursor in existing software. The introduction of 3D if anything, only hinders users in this instance.

Nevertheless, the potential and benefits of collaborative modelling within Brown *et al.'s* work (Brown, et al., 2011) is acknowledged. Given that BPMN 3D was developed using a game engine, it just so happens all the necessary tools and libraries have once again been

[30] Linden Research, Inc.; Second Life; (http://secondlife.com/) accessed 31/05/2017

included within the core files. Therefore, the inclusion of “multiplayer” functionality into the BPMN 3D solution will again be a relatively straightforward process.

Finally, it was discussed earlier how Moody’s requirement of different mediums is somewhat dated given today’s technology. To further emphasise this, the plan is to recreate the proposed solution in AR using Microsoft’s HoloLens. This will not only allow for better real-time collaboration of security specification, it also acts as a solution to the “must be easy-to-draw” requirement. This requirement largely comes from the idea that, groups of system designers will work together across a whiteboard to draw a diagram. However, using technology such as HoloLens, designers can utilise all the benefits of interactive software, while still maintaining that group atmosphere style of designing.

9 Publications & Achievements

Throughout the course of this research the following papers have been published:

- C. L. Maines, D. Llewellyn-Jones, S. Tang and B. Zhou, “A cyber security ontology for BPMN-security extensions,” in The 13th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2015), Liverpool, UK, 26th-28th October, 2015.
- C. L. Maines, B. Zhou, S. Tang and Q. Shi, “Adding a Third Dimension to BPMN as a means of Representing Cyber Security Requirements,” in Developments in eSystems Engineering (DeSE), Liverpool, UK, 31st August - 2nd September, 2016.
- C. L. Maines, B. Zhou, S. Tang and Q. Shi, “Towards a Framework for the Extension and Visualisation of Cyber Security Requirements in Modelling Languages,” in *Developments in eSystems Engineering (DeSE)*, Paris, France, 14th - 16th June, 2017.

During the project, the proposed solution was fortunate in securing funding from the SETsquared Partnership under the Innovation to Commercialisation of University Research (ICURe) programme. This funding totalled £35k for three months of ‘Market Validation’, in which researchers were expected to visit companies, discuss their work and gain otherwise unobtainable feedback through face-to-face contact with industry professionals. The main objective of this programme, being to assess whether the work has any potential market within industry, assisting with the transition of largely academic solutions into actual products.

Upon completion of the three months’ *market validation* stage, the work and its feedback was showcased to a panel who determine whether the results were positive enough for the extra funding to initiate an academic start-up. Given that the results were very positive from industry, the project was successful in obtaining an extra funding of £88k.

Therefore, throughout its entirety, the work has totalled £123k of funding during its three-year lifecycle, a form of very positive feedback on its own.

Another achievement, is the work’s involvement in The China UK Entrepreneurship Competition. The purpose of this programme being to encourage new business ventures between China and the UK by simulating the real-world process of entrepreneurs soliciting start-up funds from investors and venture capitalists. This programme is a competition between 32 teams, who must pitch their business ideas, plans, risks etc. to a panel of judges with the intention of securing funds to start-up a business. Through the efforts of the supervisory team, BPMN 3D managed to finish second in this competition with a lot of interest from those in attendance. Although this competition does not benefit much in research terms, it does represent another acknowledgement of the work's potential at plugging an obvious gap in cyber security industry practice.

10 References

Aguilar-Saven, R. S., 2004. Business process modelling: Review and framework. *International Journal of production economics,* 90(2), pp. 129-149.

Ahmed, N. & Matulevičius, R., 2014. Securing business processes using security risk-oriented patterns. *Computer Standards & Interfaces,* 36(4), pp. 723-733.

Amini, F. et al., 2015. The Impact of Interactivity on Comprehending 2D and 3D Visualizations of Movement. *IEEE Transactions on Visualization and Computer Graphics,* 21(1), pp. 122-135.

Anon., n.d. *Tritanopia – Blue-Yellow Color Blindness.* [Online]
Available at: http://www.color-blindness.com/tritanopia-blue-yellow-color-blindness/
[Accessed 30 August 2016].

Apple Inc., 2007. *Newsroom: "Apple Reinvents the Phone with iPhone".* [Online]
Available at: https://www.apple.com/newsroom/2007/01/09Apple-Reinvents-the-Phone-with-iPhone/
[Accessed 5 June 2017].

Apple & AppleInsider, n.d. *Number of available apps in the Apple Store from July 2008 to January 2017.* [Online]
Available at: https://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-app-store/
[Accessed 22 March 2017].

Arce, I. et al., 2014. *Avoiding the Top 10 Software Security Design Flaws,* s.l.: IEEE Center for Secure Design.

Arsac, W., Compagna, L., Pellegrino, G. & Ponta, S. E., 2011. Security Validation of Business Processes via Model Checking. *Lecture Notes in Computer Science,* 6542(216471), pp. 29-42.

Balaji, S. & Murugaiyan, M. S., 2012. Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management,* 2(1), pp. 26-30.

Beltran, L. P., Merabti, M. & Shi, Q., 2012. *The Use of a game-based interface for home network security.* s.l., 13h Annual Postgraduate Symposium on Convergence of Telecommunications, Networking and Broadcasting.

Blanco, C. et al., 4th - 7th March, 2008. *A systematic review and comparison of security ontologies.* Barcelona , Spain, s.n.

Bocciarelli, P. & D'Ambrogio, A., 4th-7th April, 2011. *A BPMN extension for modeling non functional properties of business processes.* Boston, s.n.

Braun, R., 13th - 16th July, 2015. *BPMN Extension Profiles - Adapting the Profile Mechanism for Integrated BPMN Extensibility.* Lisbon, Portugal, s.n.

Braun, R. & Esswein, W., 12th-13th November, 2014. *Classification of domain-specific bpmn extensions.* Manchester, UK, s.n.

Brown, R. A., Recker, J. C. & West, S., 2011. Using virtual worlds for collaborative business process modeling. *Journal of Business Process Management,* 17(3), pp. 546-564.

Brucker, A. D., Hang, I., Lückemeyer, G. & Ruparel, R., 20th - 22nd June, 2012. *SecureBPMN: Modeling and Enforcing Access Control Requirements in Business Processes.* Newark, pp. 123-126, s.n.

Burks, A. W., 1949. Icon, index, and symbol. *Philosophy and phenomenological research,* 9(4), pp. 673-689.

Cabinet Office, April 2014. *Government Security Classifications,* s.l.: Cabinet Office.

Câmara, M. et al., 21st-26th July, 2013. *Evaluating devices and navigation tools in 3D environments.* Las Vegas, Nevada, USA, s.n.

Center for Strategic & International Studies, June 2014. *Net Losses: Estimating the Global Cost of Cybercrime,* s.l.: McAfee.

Chaudron, M. et al., 23rd-27th October, 2016. *Interactive Surfaces for Collaborative Software Design.* Gothenburg, Sweden, s.n.

Chen, P. P., 10th-12th December, 2003. *Toward a methodology of graphical icon design.* Taichung, Taiwan, s.n.

Cherdantseva, Y., December, 2014. *PhD Thesis: Secure*BPMN - a graphical extension for BPMN 2.0 based on a Reference Model of Information Assurance & Security,* Cardiff, Wales: Cardiff University.

Cherdantseva, Y. & Hilton, J., pp. 546-555, 2nd-6th September, 2013. *A Reference Model of Information Assurance & Security.* Regensburg, s.n.

Chinosi, M. & Trombetta, A., January 2012. BPMN: An introduction to the standard. *Computer Standards & Interfaces,* 34(1), pp. 124-134.

Curtis, A., 2015. *Rhetoric of Flat Design and Skeuomorphism in Apple's iOS Graphical User Interface,* s.l.: University of Rhode Island.

Dahlem, N. & Hahn, A., 6th-9th August, 2009. *User-friendly ontology creation methodologies-a survey.* San Francisco, CA, USA, s.n.

Denker, G., Kagal, L. & Finin, T., 2005. Security in the Semantic Web using OWL. *Information Security Technical Report,* 10(1), pp. 51-58.

Donner, M., 2003. Toward a Security Ontology. *IEEE Security & Privacy,* 1(3), pp. 6-7.

Driver and Vehicle Standards Agency, 2015. *DVSA Official 2015 Highway Code.* s.l.:Driver and Vehicle Standards Agency.

Elçi, A., 2014. *Isn't the Time Ripe for a Standard Ontology on Security of Information and Networks?.* Glasgow, UK, s.n.

Firesmith, D., 2004. Specifying Reusable Security Requirements. *Journal of Object Technology,* 3(1), pp. 61-75.

Firesmith, D. G., 8th-12th September, 2003. *Analyzing and Specifying Reusable Security Requirements.* Monterey Bay, California, USA, s.n.

Geambasu, C. V., 2012. BPMN vs. UML Activity Diagram for Business Process Modeling. *Accounting and Management Information Systems,* 11(4), pp. 637-651.

Genon, N., Heymans, P. & Amyot, D., 2010. Analysing the Cognitive Effectiveness of the BPMN 2 . 0 Visual Notation. *Software Language Engineering, Springer LNCS,* pp. 377-396.

Gil, J. & Kent, S., 19th-2th April, 1998. *Three dimensional software modelling.* Kyoto, Japan, s.n.

Gregory, J., 2014. *Game Engine Architecture.* s.l.:CRC Press.

Hartmann, T. & Vossebeld, N., 2013. A semiotic framework to understand how signs in construction process simulations convey information. *Advanced engineering informatics,* 27(3), pp. 378-385.

Heimerl, F., Lohmann, S., Lange, S. & Ertl, T., 6th-9th January, 2014. *Word cloud explorer: Text analytics based on word clouds.* Waikoloa, HI, USA, Hawaii International Conference on System Sciences (HICSS).

Hinckley, K. et al., pp. 1-10, 14th - 17th October, 1997. *Usability Analysis of 3D Rotation Techniques.* Banff, s.n.

Hou, K.-C. & Ho, C.-H., 26th-30th August, 2013. *A preliminary study on aesthetic of apps icon design.* Tokyo, Japan, s.n.

Ichihara, Y. G. et al., 29th-31st January, 2008. *Color universal design: the selection of four easily distinguishable colors for all color vision types.* San Jose, CA, USA, s.n.

Jansz, J. & Tanis, M., 2007. Appeal of playing online first person shooter games. *CyberPsychology & Behavior,* 10(1), pp. 133-136.

Jick, T. D., December 1979. Mixing Qualitative and Quantitative Methods: Triangulation in Action. *Administrative Science Quarterly,* 24(4), pp. 602-611.

Kang, W. & Liang, Y., 10th-12th October 2013. *A security ontology with MDA for software development.* Beijing, China, s.n.

Karjoth, G., 2nd-4th March, 2015. *Aligning Security and Business Objectives for Process-Aware Information Systems.* San Antonio, TX, USA, s.n.

Karyda, M. et al., 20th-22nd April 2006. *An ontology for secure e-government applications.* Vienna, Austria, s.n.

Kascak, L., Rébola, C. B., Braunstein, R. & Sanford, J. a., 30th September - 1st October, 2013. *Icon Design for User Interface of Remote Patient Monitoring Mobile Devices.* Greenville, NC, USA, s.n.

Khan, M. U., 17th - 20th August, 2015. *Representing security specifications in UML state machine diagrams.* Belfort, France, s.n.

Klahr, R. et al., April, 2017. *Cyber security breaches survey,* s.l.: UK Government, Department for Culture Media & Sport.

Komlodi, A. et al., 26th October, 2005. *A User-centered Look at Glyph-based Security Visualization.* Minneapolis, MN, USA, IEEE Workshop on Visualization for Computer Security.

Labda, W., Mehandjiev, N. & Sampaio, P., pp. 1399-1405, 24th - 28th March, 2014. *Modeling of Privacy-Aware Business Processes in BPMN to Protect Personal Data.* Gyeongju, s.n.

Lab, K., n.d. *Home: Kaspersky Lab.* [Online]
Available at: https://www.kaspersky.co.uk
[Accessed 15 November 2017].

Leitner, M., Miller, M. & Rinderle-Ma, S., 2nd-6th September, 2013. *An Analysis and Evaluation of Security Aspects in the Business Process Model and Notation.* Regensburg, s.n., p. 262.

Leitner, M., Schefer-Wenzl, S., Rinderle-Ma, S. & Strembeck, M., 6th-7th November, 2013. *An Experimental Study on the Design and Modeling of Security Concepts in Business Processes.* Riga, Latvia, s.n.

Lin, T. S. & Lai, C. S., 2013. The recognition and comprehension of application icons on mobile devices. *The International Association of Societies of Design Research,* Volume 5, pp. 4069-4078.

Lodderstedt, T., Basin, D. & Doser, J., 2002. SecureUML: A UML-Based Modeling Language for Model-Driven Security. *The Unified Modeling Language, Springer, LNCS,* Volume 2460, pp. 426-441.

Luff, P. et al., 6th-10th November, 2004. *Only touching the surface: creating affinities between digital content and paper.* Chicago, IL, USA, s.n.

Maines, C. L., Llewellyn-Jones, D., Tang, S. & Zhou, B., 26th-28th October, 2015. *A cyber security ontology for BPMN-security extensions.* Liverpool, UK, The 13th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2015).

Maines, C. L., Zhou, B., Tang, S. & Shi, Q., 14th - 16th June, 2017. *Towards a Framework for the Extension and Visualisation of Cyber Security Requirements in Modelling Language.* Paris, France, s.n.

Maines, C. L., Zhou, B., Tang, S. & Shi, Q., 31st August - 2nd September, 2016. *Adding a Third Dimension to BPMN as a means of Representing Cyber Security Requirements.* Liverpool, UK, s.n.

Maines, C. & Tang, S., 25th - 27th August, 2014. *An Application of Game Technology to Virtual University Campus Touring and Interior Navigation.* Phapos, Cyprus, s.n.

Marcus, A., 2003. Icons, symbols, and signs: Visible languages to facilitate communication. *ACM Interactions,* 10(3), pp. 37-43.

Marcus, A., Feng, L. & Maletic, J. I., 11th-13th June, 2003. *3D representations for software visualization.* San Diego, CA, USA, s.n.

Martin, G., Kinross, J. & Hankin, C., 17th May, 2017. *Effective cybersecurity is fundamental to patient safety.* s.l.:British Medical Journal Publishing Group.

McClymont, J., Shuralyov, D. & Stuerzlinger, W., 19th-21st September, 2011. *Comparison of 3D navigation interfaces.* Ottawa, ON, Canada , s.n.

Meland, P. H. & Gjaere, E. A., 20th-24th August, 2012. *Representing threats in BPMN 2.0.* Prague, Czeck Republic, s.n.

Mohammed, N. M., Niazi, M., Alshayeb, M. & Mahmood, S., February 2017. Exploring software security approaches in software development lifecycle: A systematic mapping study. *Computer Standards & Interfaces,* Volume 50, pp. 107-115.

Moody, D., 2009. The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering,* 35(6), pp. 756-779.

Morton, J. L., n.d. *Which color is the most irritating?: Color & Vision Matters: Color.* [Online] Available at: http://www.colormatters.com/color-and-vision/color-and-vision-matters [Accessed 6 June 2017].

Mülle, J., von Stackelberg, S. & Böhm, K., 2011. *A Security Language for BPMN Process Models.* Karlsruhe, s.n.

Muller, R. & Rogge-Solti, A., 21st-22nd February, 2011. *BPMN for healthcare processes.* Karlsruhe, Germany, s.n.

Nakata, T., 25th - 26th November, 2014. *Improving Human Reliability On Checking.* Kochi, Japan, s.n.

National Cyber Security Centre, 2016. *Guidance: Security Design Principles for Digital Services.* [Online]
Available at: https://www.ncsc.gov.uk/guidance/security-design-principles-digital-services-main
[Accessed 1 June 2017].

Nitzsche, J., Wutke, D. & Van Lessen, T., 7th June, 2007. *An Ontology for Executable Business Processes.* Innsbruck, Austria, s.n.

Noy, N. F. & McGuinness, D. L., March, 2001. *Ontology Development 101: A Guide to Creating Your First Ontology.* s.l., s.n.

Object Management Group, Inc., January, 2011. *Business Process Model and Notation (BPMN) Version 2.0,* s.l.: http://www.omg.org/spec/BPMN/2.0/.

Object Management Group, Inc., n.d. *Business Process Model and Notation.* [Online]
Available at: http://www.bpmn.org/
[Accessed 25 August 2016].

Object Management Group, Inc., n.d. *What is UML: UML.* [Online]
Available at: http://www.uml.org/what-is-uml.htm
[Accessed 1 June 2017].

Paja, E., Giorgini, P., Paul, S. & Meland, P. H., 6th October, 2011. *Security Requirements Engineering for Secure Business Processes.* Riga, Latvia, s.n.

Peixoto, D. C. C., Batista, V. A., Atayde, A. P. & Borges, E. P., January, 2008. A Comparison of BPMN and UML 2.0 Activity Diagrams. *VII Simposio Brasileiro de Qualidade de Software,* Volume 56.

Peng, L., 6th-10th May, 2009. *A BPMN based Secure Workflow Model.* Milan, Italy, s.n.

Peralta, K. P., Orozco, A. M., Zorzo, A. F. & Oliveira, F. M., 28th September - 3rd October, 2008. *Specifying security aspects in UML models.* Toulouse, France, s.n.

Pfleeger, C. P. & Pfleeger, S. L., 4th Edition, 2006. *Security in Computing.* s.l.:Prentive Hall PTR.

Popescu, G. & Wegmann, A., pp. 166-173, 14th-17th July, 2014. *Using the Physics of Notations Theory to Evaluate the Visual Notation of SEAM.* Geneva, s.n., pp. 166-173.

Recker, J. C. & Dreiling, A., 5th-7th December, 2007. *Does it matter which process modelling language we teach or use? An experimental study on understanding process modelling languages without formal education.* Toowoomba, Australia, s.n.

Recker, J. C. et al., 6th-9th August, 2009. *Measuring method complexity: UML versus BPMN.* San Francisco, California, USA, s.n.

Rodríguez, A., Fernández-Medina, E. & Piattini, M., April 2007. A BPMN extension for the modeling of security requirements in business processes. *IEICE Transactions on Information and Systems,* E90-D(4), pp. 745-752.

Saleem, M. Q., Jaafar, J. B. & Hassan, M. F., January 2012. A Domain-Specific Language for Modelling Security Objectives in a Business Process Models of SOA Applications. *Advances in Information Sciences and Service Sciences,* 4(January), pp. 353-362.

Salnitri, M., Dalpiaz, F. & Giorgini, P., 2014. Modeling and Verifying Security Policies in Business Processes. *Enterprise, Business-Process and Information and Information Systems Modeling, Springer LCBIP,* Volume 175, pp. 200-214.

Sang, K. S. & Zhou, B., 26th-28th October, 2015. *BPMN security extensions for healthcare process.* Liverpool, UK, s.n.

Shapiro, R. et al., 2012. *BPMN 2.0 Handbook Second Edition: Methods, Concepts, Case Studies and Standards in Business Process Modeling Notation.* Lighthouse Point, FL: Future Strategies Inc., Book Division.

Shiode, N., 2000. 3D urban models: recent developments in the digital modelling of urban environments in three-dimensions. *GeoJournal,* 52(3), pp. 269-269.

Shu, W. & Lin, C.-S., 7th-9th August, 2014. *Icon Design and Game App Adoption.* Savannah, Georgia, USA, s.n.

Silingas, D. & Rimantas, B., 18th-21st May, 2008. *UML-intensive framework for modeling software requirements.* Gdansk, Poland, s.n.

Singh, V. & Pandey, S., 8th-10th October, 2014. *A comparative study of Cloud Security Ontologies.* Noida, India, s.n.

Souag, A., Salinesi, C. & Comyn-Wattiau, I., 25th-26th June, 2012. *Ontologies for security requirements: A literature survey and classification.* Gdańsk, Poland, s.n.

Stickel, C., Pohl, H.-M. & Milde, J.-T., 22nd-27th June, 2014. *Cutting edge design or a beginner's mistake?--a semiotic inspection of iOS7 icon design changes.* Crete, Greece, s.n.

Störrle, H. & Fish, A., Sep 29 - Oct 4, 2013. *Towards an operationalization of the "Physics of Notations" for the analysis of visual languages.* Miami, FL, USA, s.n., pp. 104-120.

Tang, S. & Hanneghan, M., 6th-8th December, 2011. *Game content model: an ontology for documenting serious game design.* Dubai, UAE, s.n.

Tekli, J., 2016. An overview on xml semantic disambiguation from unstructured text to semi-structured data: Background, applications, and ongoing challenges. *IEEE Transactions on Knowledge and Data Engineering,* 28(6), pp. 1383-1407.

Tekofsky, S., Miller, P., Spronck, P. & Slavin, K., 28th - 30th June, 2016. *The Effect of Gender, Native English Speaking, and Age on Game Genre Preference and Gaming Motivations.* Utrecht, The Netherlands, s.n.

Teyseyre, A. R. & Campo, M. R., 2009. An overview of 3D software visualization. *IEEE Transactions on Visualization and Computer Graphics,* 15(1), pp. 87-105.

Tufte, E., March, 1956. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *The Psychological Review,* 63(2), pp. 91-97.

UK Government, P. R., 2017. *News: Almost half of UK firms hit by cyber breach or attack in the past year.* [Online]
Available at: https://www.gov.uk/government/news/almost-half-of-uk-firms-hit-by-cyber-breach-or-attack-in-the-past-year
[Accessed 1 June 2017].

Unity 3D, n.d. *GameObject: Manual: Unity.* [Online]
Available at: https://docs.unity3d.com/Manual/class-GameObject.html)
[Accessed 27 April 2017].

Unity 3D, n.d. *Prefabs: Manual: Unity.* [Online]
Available at: https://docs.unity3d.com/Manual/Prefabs.html
[Accessed 27 April 2017].

Wang, W., Ding, H., Dong, J. & Ren, C., 21st-23rd June, 2006. *A Comparison of Business Process Modeling Methods.* Shanghai, China, IEEE International Conference on Service Operations and Logistics, and Informatics.

Ware, C., Hui, D. & Franck, G., 24th-28th October, 1993. *Visualizing object oriented software in three dimensions.* Toronto, Canada, s.n.

Wolter, C., Menzel, M. & Meinel, C., 12th-14th March, 2008. *Modelling Security Goals in Business Processes.* pp. 197-212, Berlin, s.n.

zur Muehlen, M. & Recker, J., 2013. We Still Don't Know How Much BPMN Is Enough, But We Are Getting Closer. *Seminal Contributions to Information Systems Engineering,* Berlin Heidelberg(Spinger-Verlag), pp. 445-451.

11 Appendix

Ontology Creation: Step 5, 6 & 7 (Properties, facets and instances)

Access Control	Concept Definition: Access Control refers to the act of authorising what parties are allowed to consume, enter or use a service.	
N/A	*N/A*	*N/A*

Authentication	Concept Definition: Authentication is the process of confirming an identity.	
N/A	*N/A*	*N/A*

Identification	Concept Definition: Identification is the process of establishing who or what someone (or something) is.	
N/A	*N/A*	*N/A*

Authorisation	Concept Definition: Authorisation is the function of specifying access rights to resources.	
N/A	*N/A*	*N/A*

Personnel Authentication	Concept Definition: Personnel Authentication is the process of confirming the identity of a person.	
N/A	*N/A*	*N/A*

Network Authentication	Concept Definition: Network Authentication is the process of confirming the identity of a network.	
N/A	*N/A*	*N/A*

Credentials	Concept Definition: Credentials refer to a unique knowledge based identifier.	
Username Required	Boolean	True/False
Password Required	Boolean	True/False
Pin Required	Boolean	True/False
Password Change Period (weeks)	Integer	0,1,2…*n*

Smart Card	Concept Definition: Smart Cards are small pocket-sized cards that have embedded integrated circuits, used for authentication and security clearance.	
Contactless	Boolean	True/False
Pin Required	Boolean	True/False

Biometric	Concept Definition: Biometrics use human physiological characteristics as a means of authentication.	
Biometric Type	String	Iris, Retina, Facial, Fingerprint, Voice

Cryptographic Protocol	Concept Definition: Cryptographic Protocols describe how cryptographic primitives (low-level algorithms) should be used.	
Protocol	String	IKE, IKEv2, IPsec, Kerberos, PPTP, TLS, SSL

Virtual Private Network	Concept Definition: Virtual Private Networks extend a private network across a public network, such as the Internet. Enabling users to send and receive data across shared or public networks as if working from a private network.	
N/A	*N/A*	*N/A*

Asset Classification	Concept Definition: Asset Classification refers to categorisation and restriction of resources based on their authorisation level.	
N/A	*N/A*	*N/A*

Data Classification	Concept Definition: Data Classification refers to categorisation and restriction of data based on their authorisation level. (1 = highly classified, 5 = public access)	
Level of Classification	Integer	1, 2, 3, 4, 5

Service Classification	Concept Definition: Service Classification refers to categorisation and restriction of services based on their authorisation level. (1 = highly classified, 5 = public access)	
Level of Classification	Integer	1, 2, 3, 4, 5

Trust Level	Concept Definition: Trust Level refers to a ranking system that determines what level of access the party has. (1 = highly classified, 5 = public access)

Minimum Level Required	Integer	1, 2, 3, 4, 5

State Transition System	Concept Definition: State Transition System is used to describe the potential behaviour of discrete systems, consisting of states and transitions between states.	
N/A	*N/A*	*N/A*

Biba Model	Concept Definition: The Biba Model describes a set of access control rules designed to ensure data integrity. Subjects may create content at or below their own integrity level. However, subjects can only view content at or above their own integrity level.	
N/A	*N/A*	*N/A*

Bell-LaPadula Model	Concept Definition: Bell-LaPadula Model The Bell-LaPadula Model describes a set of access control rules which use security labels on objects and clearances for subjects. Focusing on data confidentiality as opposed to data integrity.	
N/A	*N/A*	*N/A*

Privacy	Concept Definition: Privacy is the ability of an individual to determine for themselves when, how and to what extent information about themselves is communicated to others.	
N/A	*N/A*	*N/A*

User Consent	Concept Definition: User Consent refers to the required authorisation of a party before allowing access to their sensitive information.	
Once Only	Boolean	True/False
Every Time Accessed	Boolean	True/False

Anonymity	Concept Definition: Anonymity is the ability to hide one's true identity by not using real credentials.	
Compulsory	Boolean	True/False

Pseudonymity	Concept Definition: Pseudonymity is a state of disguised identity, where a user will use false credentials as opposed to just hiding them.	
Compulsory	Boolean	True/False

Data Usage Consent	Concept Definition: Data Usage Consent refers explicitly to the required authorisation of a party before allowing access to their data.	
Once Only	Boolean	True/False
Every Time Accessed	Boolean	True/False

Confidentiality	Concept Definition: Confidentiality is the property, that information is not made available or disclosed to unauthorised individuals, entities or parties.	
N/A	*N/A*	*N/A*

Need to Know	Concept Definition: Need to Know enforces restrictions such that even if one has all the necessary approvals, unless one has a specific need to access a resource it will remain confidential.	
N/A	*N/A*	*N/A*

Encryption	Concept Definition: Encryption is the process of encoding messages or information in such a way that only authorised parties can read it.	
Type of Encryption	String	Symmetric Key, Public Key
Key Size	Integer	128, 192, 256…

Data Retention	Concept Definition: Minimum Data Retention is an aspect of records management. It represents a minimum period of time a resource should be kept before it can be discarded or destroyed.	
Minimum Retention Period (years)	Integer	0, 1, 2, 3…n
Maximum Retention Period (years)	Integer	0, 1, 2, 3…n

Public Key Infrastructure	Concept Definition: A Public Key Infrastructure is a set of roles, policies and procedures needed to create, manage, distribute, use, store and revoke digital certificates.	
N/A	*N/A*	*N/A*

Integrity	Concept Definition: Integrity refers to maintaining the accuracy and trustworthiness of a resource across its entire life cycle. Ensuring it is not changed during transit, or altered by unauthorised parties.

N/A	N/A	N/A

Data Integrity	Concept Definition: Data Integrity Control refers to maintaining the accuracy and trustworthiness of data across its entire life cycle. Ensuring it is not changed during transit, or altered by unauthorised parties.	
N/A	N/A	N/A

Hardware Integrity	Concept Definition: Hardware Integrity Control refers to maintaining the accuracy and trustworthiness of hardware across its entire life cycle. Ensuring it is not changed or altered by unauthorised parties.	
N/A	N/A	N/A

Personnel Integrity	Concept Definition: Personnel Integrity Control refers to maintaining the trustworthiness of personnel across their entire employment cycle. Ensuring their honesty and loyalty to the company.	
N/A	N/A	N/A

Software Integrity	Concept Definition: Software Integrity Control refers to maintaining the accuracy and trustworthiness of software across its entire life cycle. Ensuring it is not changed or altered by unauthorised parties.	
N/A	N/A	N/A

Hash Function	Concept Definition: A hash function is any function that can be used to map data of arbitrary size to data of fixed size. In cryptography this allows one to easily verify that some input data maps to a given hash value, but if the input data is unknown, it is deliberately difficult to reconstruct it by knowing the stored hash value.	
N/A	N/A	N/A

Constraints	Concept Definition: Constraints impose a set of conditions that variables must satisfy.	
N/A	N/A	N/A

Input Validation	Concept Definition: Input Validation refers specifically to a set of conditions that data must satisfy before they are accepted. This reduces the possibility of human input error and aids in maintaining data integrity.

Date	Boolean	True/False
Numerical Values Only	Boolean	True/False
Textual Values Only	Boolean	True/False
No Numerical Values	Boolean	True/False
No Textual Values	Boolean	True/False
No Symbols	Boolean	True/False

Physical Security	Concept Definition: Physical Security describes security measures that are designed to deny unauthorised access to facilities, equipment and resources, and to protect personnel and property from damage or harm.	
N/A	*N/A*	*N/A*

Location	Concept Definition: Location describes locality security measures that are designed to deny unauthorised access to facilities, equipment and resources, and to protect personnel and property from damage or harm.	
Physical Barriers	Boolean	True/False
Video Surveillance	Boolean	True/False
Alarm System/Sensor	Boolean	True/False
Security Lighting	Boolean	True/False

Personnel	Concept Definition: Personnel play a central role in all layers of security. In this instance, Personnel refers to security guards who are paid to protect property and its assets from hazards.	
Personnel Required	Integer	1, 2, 3…n

Asset Management	Concept Definition: Asset Management refers to any system that monitors and maintains assets of value to an entity or group.	
N/A	*N/A*	*N/A*

Asset Register	Concept Definition: Asset Registers are records that keep track of all the assets belonging to a business.	
Description Required	Boolean	True/False
Serial Number Required	Boolean	True/False
Data of Purchase Required	Boolean	True/False
Purchase Price Required	Boolean	True/False
Data Asset Assigned Required	Boolean	True/False

Estimated Lifespan Required	Boolean	True/False

Role Assignment	Concept Definition: Role Assignment is a security policy that determines whether a user or group can access a specific item or perform an operation. A role assignment consists of a single user or group account name and one or more role definitions.	
N/A	*N/A*	*N/A*

Binding of Duty	Concept Definition: Binding of Duty is the requirement that an entity performing an action is bound to perform another action.	
Binding Task 1	String	*Task name*
Binding Task 2	String	*Task name*
…	…	…
Binding Task n	String	*Task name*

Separation of Duty	Concept Definition: Separation of Duty is the concept of having more than one entity required to complete a task. It can also mean when two or more tasks cannot be performed by the same entity.	
Minimum Entities Required	Integer	2, 3, 4…n
This Task Only	Boolean	True/False
Separation Task 1	String	*Task name*
Separation Task 2	String	*Task name*
…	…	…
Separation Task n	String	*Task name*

Delegation	Concept Definition: Delegation is the assignment of responsibility or authority to another entity (normally from a manager to a subordinate) to carry out specific activities.	
Forbidden	Boolean	True/False

Immunity	Concept Definition: Immunity (anti-virus) is computer software used to prevent, detect and remove malicious software.	
Scan Interval (days)	Integer	1, 2, 3…n

Patch Management Software	Concept Definition: Patch Management is the process of using a strategy and plan of what patches should be applied to which systems at a specified time.	

Patch Scan Interval (days)	Integer	1, 2, 3…n
Patch Install Time (time)	Time	00:00

Sandbox	Concept Definition: A Sandbox is a security mechanism for separating running programs. It is often used to execute untested code, or untrusted programs from unverified third parties.	
N/A	*N/A*	*N/A*

Accountability	Concept Definition: Accountability refers to one being answerable for the correct and thorough completion of a deliverable to task, and the one who delegates the work to those responsible.	
N/A	*N/A*	*N/A*

Audit Trail	Concept Definition: An Audit Trail is a security-relevant chronological record, set of records, and/or destination and source of records that provide documentary evidence of the sequence of activities that have affected at any time a specific operation, procedure, or event.	
User ID Required	Boolean	True/False
Time Stamp Required	Boolean	True/False
Affected Entity Required	Boolean	True/False
Read Required	Boolean	True/False
Write Required	Boolean	True/False
Modify Required	Boolean	True/False

Non-Repudiation	Concept Definition: Non-Repudiation is the assurance that someone cannot deny something. For example, a party is unable to deny the authenticity of their signature on a document.	
N/A	*N/A*	*N/A*

Digital Signature	Concept Definition: A Digital Signature is a mathematical scheme for demonstrating the authenticity of a digital message or documents. A valid digital signature gives a recipient reason to believe that the message was created by a known sender and that the sender cannot deny having sent the message.	
N/A	*N/A*	*N/A*

Attack/Harm Detection and Prevention	Concept Definition: Attack/Harm Detection and Prevention refers to devices or software that monitor network and systems for malicious activity.	
N/A	*N/A*	*N/A*

Vulnerability Assessment	Concept Definition: A Vulnerability Assessment is the process of identifying, quantifying, and prioritising the vulnerabilities in a system.	
N/A	*N/A*	*N/A*

Honeypot	Concept Definition: A Honeypot is a mechanism set to detect, deflect, or, in some manner, counteract attempts at unauthorised use of information systems. Honeypots consist of data that appears to be a legitimate part of the site but is actually isolated and monitored.	
N/A	*N/A*	*N/A*

Firewall	Concept Definition: A Firewall is a network security system that monitors and controls the incoming and outgoing network traffic based on predetermined security rules.	
N/A	*N/A*	*N/A*

Intrusion Detection & Prevention Systems	Concept Definition: Intrusion Detection & Prevention Systems are network security appliances that monitor network and/or system activities for malicious activity. The main function of intrusion prevention systems are to identify malicious activity, log information about the activity, attempt to block/stop it and report it.	
N/A	*N/A*	*N/A*

System Assessment	Concept Definition: A System Assessment is the process of identifying, quantifying, and prioritising the vulnerabilities in a system.	
Assessment Interval (days)	Integer	1, 2, 3…n

Environment Assessment	Concept Definition: An Environment Assessment is the process of identifying, quantifying, and prioritising the vulnerabilities in an environment.	
Assessment Interval (days)	Integer	1, 2, 3…n

Service Assessment	Concept Definition: A Service Assessment is the process of identifying, quantifying, and prioritising the vulnerabilities in a service.	
Assessment Interval (days)	Integer	1, 2, 3…n

Personnel Assessment	Concept Definition: A Personnel Assessment is the process of identifying, quantifying, and prioritising the vulnerabilities of personnel. This could be to ensure integrity or skills-based to ensure personnel are capable of performing their duties.	
Assessment Interval (days)	Integer	1, 2, 3…n

High-Interaction (Honeypot)	Concept Definition: High-Interaction honeypots imitate the activities of the production systems that host a variety of services and, therefore, an attacker may be allowed a lot of fake services to waste his time, monitor his activity and learn his strategy.	
N/A	*N/A*	*N/A*

Low-Interaction (Honeypot)	Concept Definition: Low-Interaction honeypots simulate only the service frequently requested by attackers.	
N/A	*N/A*	*N/A*

Network Layer (Firewall)	Concept Definition: Network Layer firewalls, also called packet filters, operate at a relatively low level of the TCP/IP protocol stack, not allowing packets to pass through the firewall unless they match the established rule set.	
N/A	*N/A*	*N/A*

Application Layer (Firewall)	Concept Definition: Application Layer firewalls work on the application level of the TCP/IP stack, and may intercept all packets travelling to or from an application.	
N/A	*N/A*	*N/A*

Stateful Protocol Analysis Detection	Concept Definition: Stateful Protocol Analysis Detection identifies deviations of protocol states by comparing observed events with predetermined profiles of generally accepted definitions of benign activity.	
N/A	*N/A*	*N/A*

Signature-Based Detection	Concept Definition: Signature-Based Detection monitors packets in the network and compares with pre-configured and pre-determined attack patterns known as signatures.	
N/A	*N/A*	*N/A*

Anomaly Detection	Concept Definition: Anomaly Detection determines the normal network activity and alerts the administrator or user when traffic is detected which is anomalous.	
N/A	*N/A*	*N/A*

Availability	Concept Definition: Availability refers to resources or systems being available as and when needed.	
Availability Percentage	Float	90, 91…99.99
Downtime per Month (hours)	Integer	24, 48…n

Service Backup	Concept Definition: Service Backup refers to a separate system which can be used to provide core functionality should the main system become unavailable.	
Minimum Backups	Integer	1, 2, 3…n

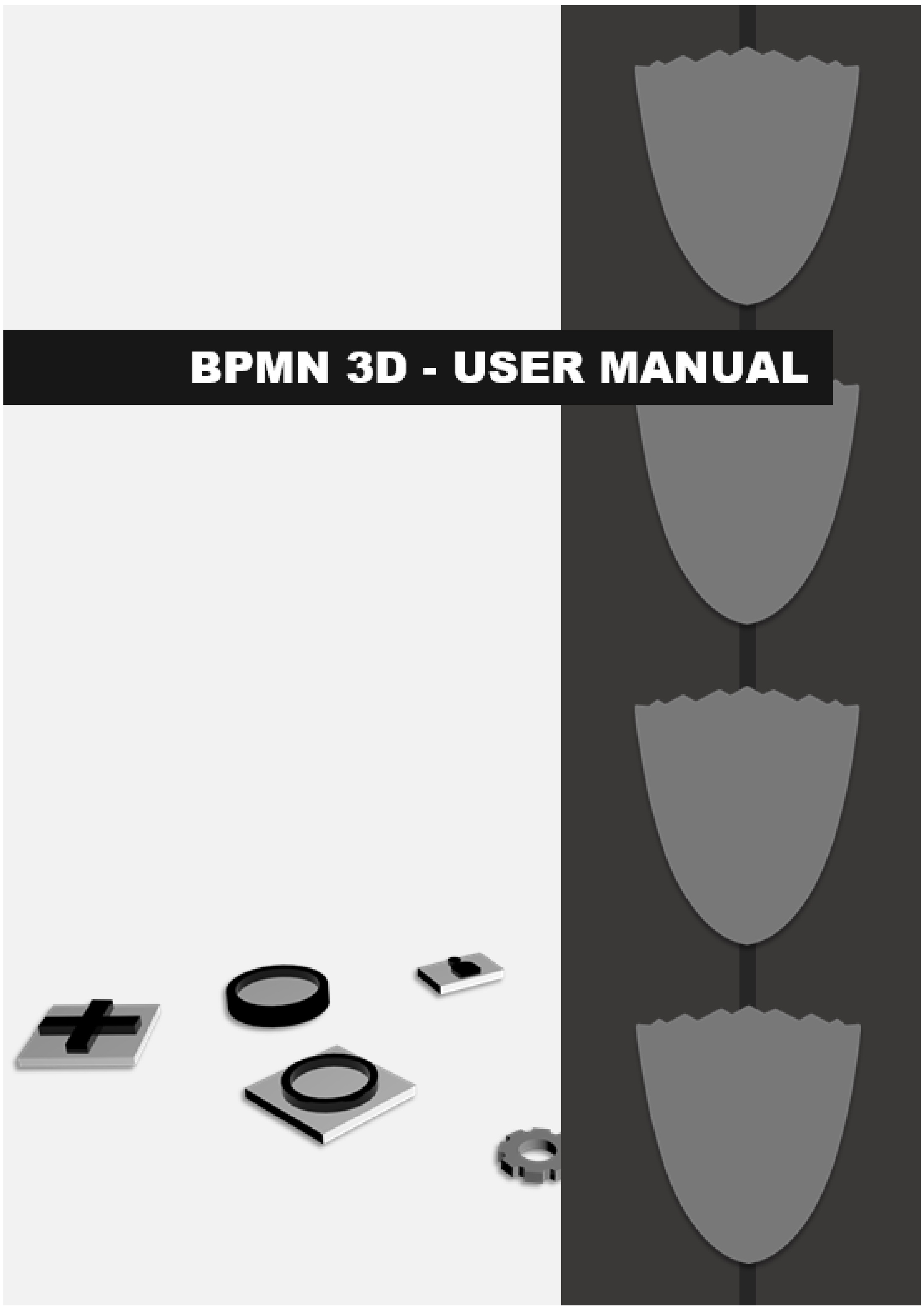
Data Backup	Concept Definition: Data Backup refers to the copying and archiving of data so it may be used to restore the original after a data loss event.	
Minimum Backups	Integer	1, 2, 3…n
Backup Frequency (days)	Float	0.5, 1, 2…n

Personnel Backup	Concept Definition: Personnel Backup refers to having surplus personnel capable of completing a task. In the case an employee is unavailable, there are backup staff capable of performing their duties.	
Minimum Backups	Integer	1, 2, 3…n

Hardware Backup	Concept Definition: Hardware Backup refers to having reserve hardware which can be used to provide core functionality should the main hardware become unavailable.	
Minimum Backups	Integer	1, 2, 3…n

Local Backup (data)	Concept Definition: Local Backup refers to the copying and archiving of data to local storage, so it may be used to restore the original after a data loss event.	
Minimum Backups	Integer	1, 2, 3…n

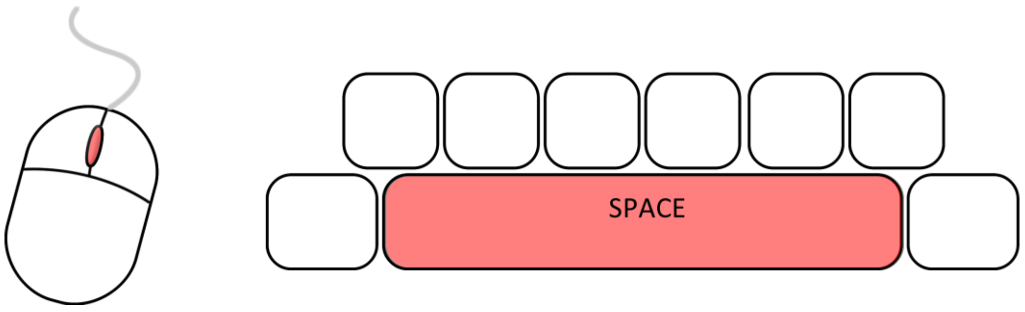
Online Backup (data)	Concept Definition: Online Backup refers to the copying and archiving of data to online storage (cloud), so it may be used to restore the original after a data loss event.	
Minimum Backups	Integer	1, 2, 3…n



BPMN 3D - USER MANUAL

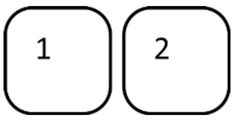
Controls

To navigate the diagram in 3D space, you must first enable **free camera mode** by clicking the RIGHT MOUSE BUTTON or by pressing the SPACEBAR.

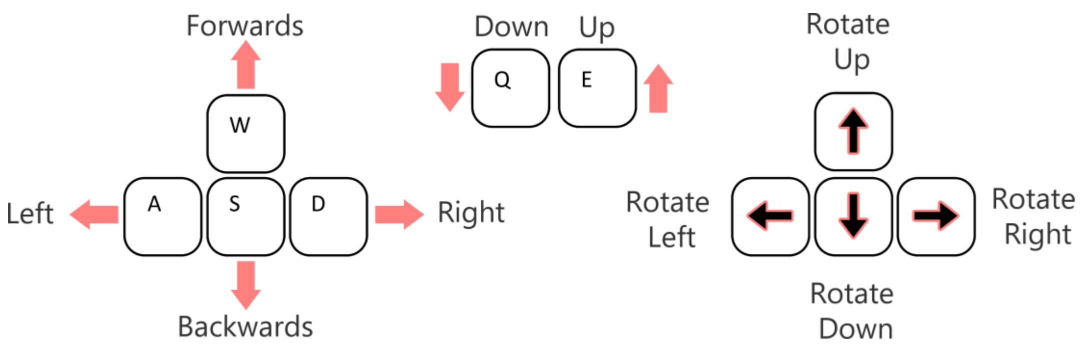


(The same buttons are also used to disable ***free camera mode****.)*

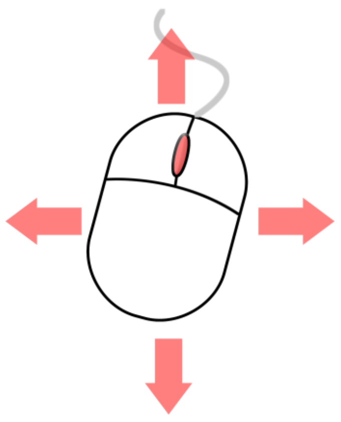
There are then **three** different methods of camera control you can choose between within the application. You can switch between these modes by pressing the 1 and 2 keys. 1 being the **default** setup. 2 being **classic**, first person shooter video game controls.



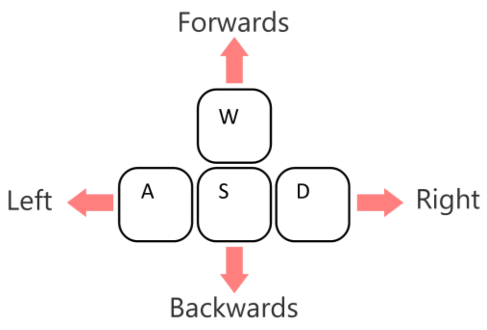
In default mode, navigation can be accomplished in two ways. Firstly, by using the WASD keys to strafe forwards, left, backwards and right respectively. With the E key being used to move upwards and the Q key to move downwards. The arrow keys can then be used for rotating the camera.

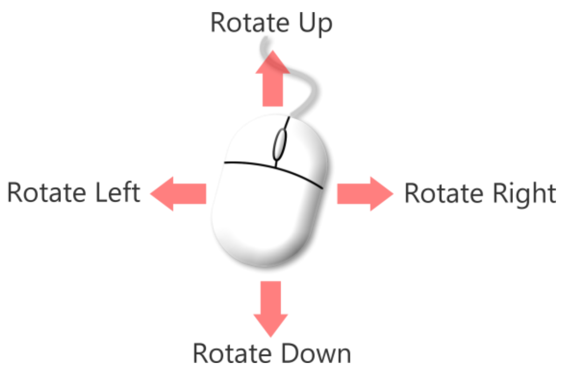


The second method of camera control while in default mode, uses only the mouse for navigation. By pressing and holding the LEFT MOUSE BUTTON, you can drag the camera across the scene. By pressing and holding the RIGHT MOUSE BUTTON, you can control the camera rotation. The SCROLL WHEEL can be used for zooming the camera in and out.

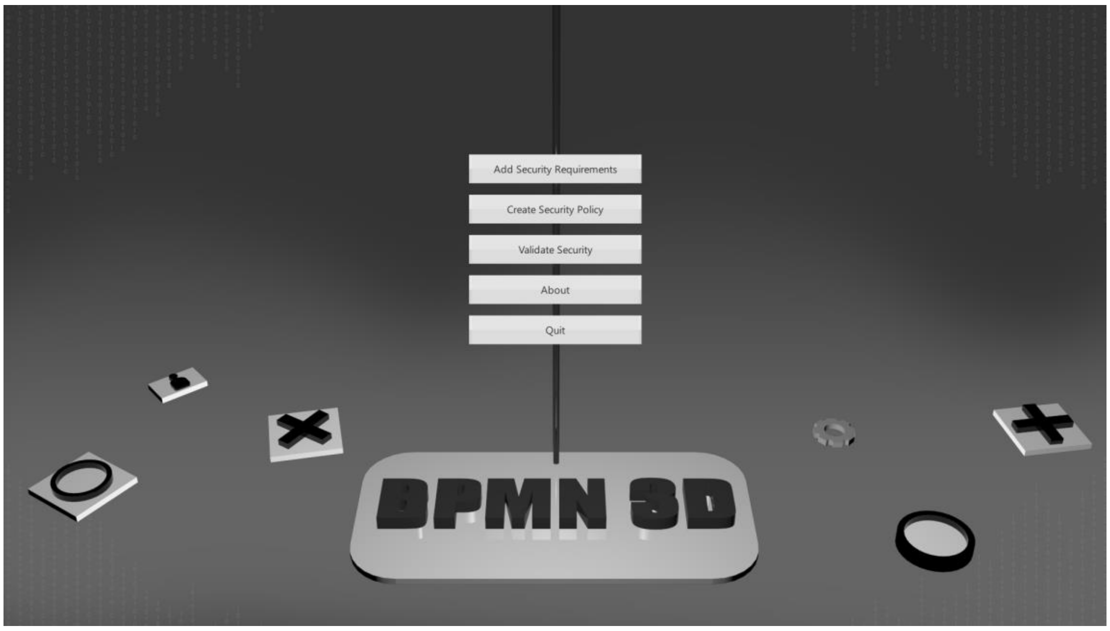


While in classic mode, navigation is achieved by using the WASD keys to strafe forwards, left, backwards and right respectively. This time, mouse movement controls the rotation of the camera.





Main Menu



Add Security Requirements

Opens the associated BPMN diagram in BPMN 3D ready for adding security requirements.

Create Security Policy

Lists the associated BPMN elements for creating explicit security policies.

Validate Security

Compares the BPMN 3D diagram against the specified security policies.

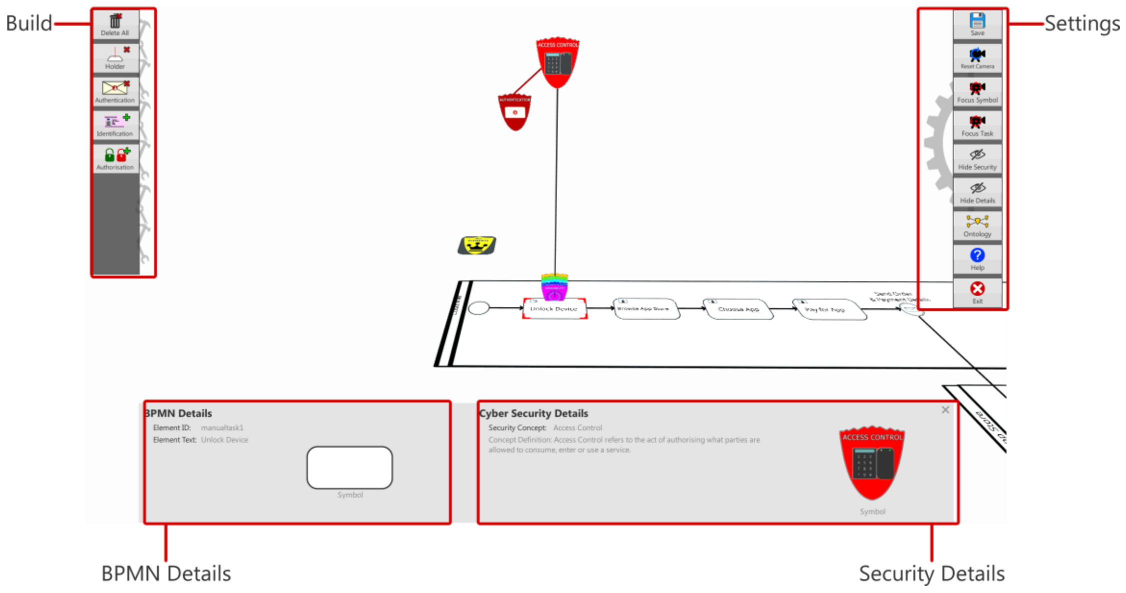
About

View the credits for the BPMN 3D developers.

Quit

Quits the application.

User Interface



BPMN Details
Element ID: manualtask1
Element Text: Unlock Device

Cyber Security Details
Security Concept: Access Control
Concept Definition: Access Control refers to the act of authorising what parties are allowed to consume, enter or use a service.

Build

This toolbar contains all the functionality required for adding and deleting security requirements.

Settings

This toolbar contains various settings and application assists: save, reset camera, focus symbol, focus task, hide security, hide details, ontology, help and exit.

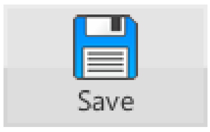
BPMN Details

This side of the details bar displays information about the currently highlighted BPMN element, consisting of the element ID, element text and a graphical representation of the element itself.

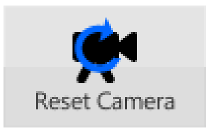
Security Details

This side of the details bar displays information about the currently highlighted security symbol, consisting of the symbol concept, concept definition and a graphical representation of the symbol itself. It also allows for more detailed specification of security requirement properties.

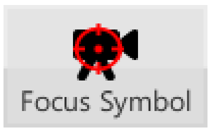
Settings



Save as implied saves the diagram and your security requirements.



Reset Camera will move the camera to the original starting position above your diagram.

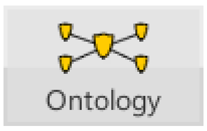


Focus Symbol will move the camera to a new position in front of the currently highlighted security symbol for better viewing.

Focus Task will move the camera to a new position in front of the currently highlighted BPMN element for better viewing of the security requirements.

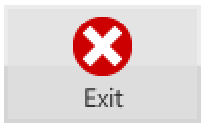
Hide Security will hide all security requirements on the diagram for better viewing of the BPMN diagram. *Note: This does not delete security requirements.*

Hide Details will hide the details toolbar at the bottom of the screen for better viewing of the diagram.



Ontology will open the security requirements ontology which shows the full hierarchy of BPMN 3D's security requirements.

Help will open a small help page which includes the controls for the diagram.

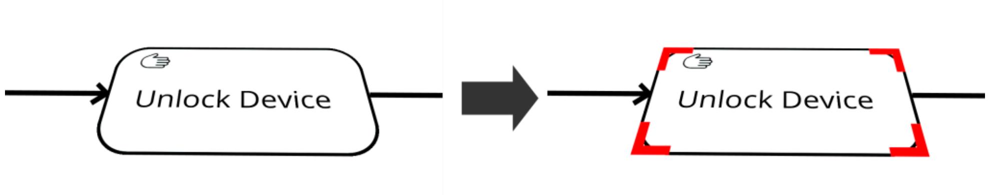


Exit will quit to the main menu of the application.

Adding Security Requirements

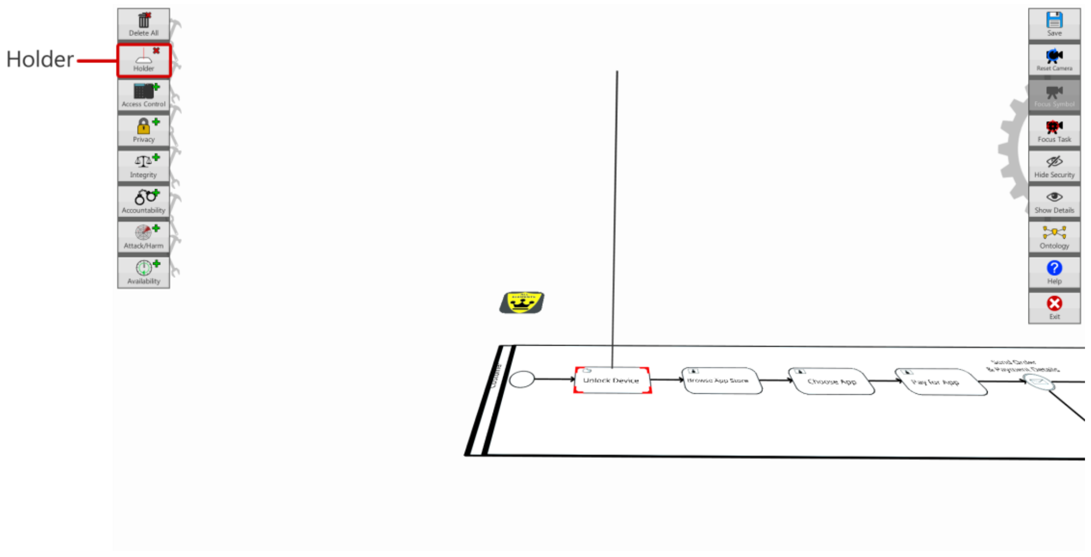
Adding security requirements to elements in BPMN 3D is a straightforward process.

1) Once you have navigated to **Add Security Requirements** from the main menu. The first step is to highlight the element you wish to add security too. This is done by moving the mouse over the element and clicking the left mouse button. This will then highlight the element as seen below.



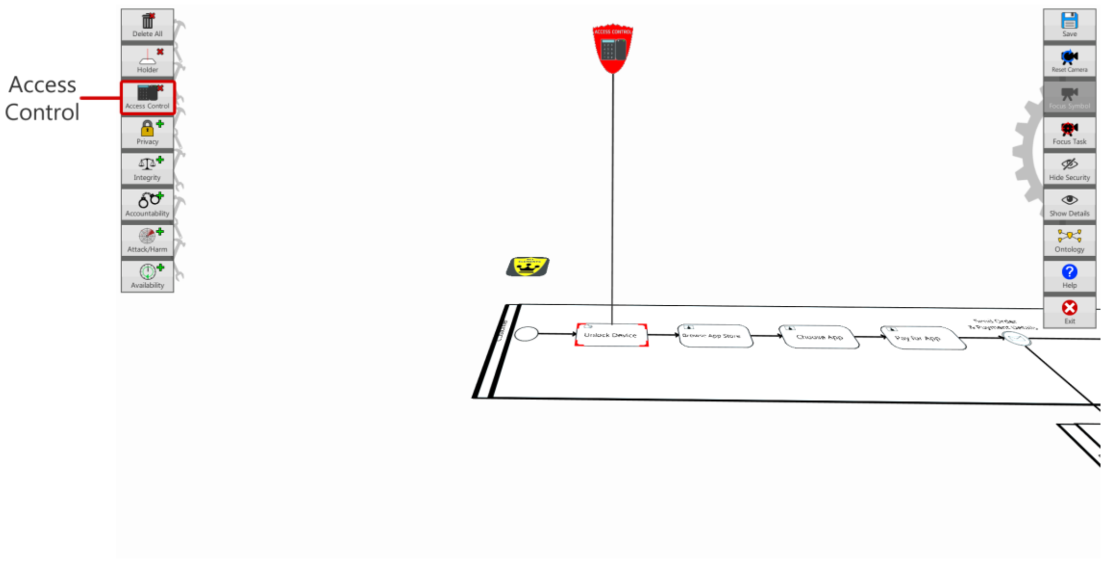
(Note: If the camera rotates when you move the mouse, you have free camera mode enabled. Press the right mouse button to disable it; see pg. 3 for more info.)

2) The next step is to add a holder to the element. (A holder is used to show the relationship between the element and its respective security requirements.) To add a holder -with the element highlighted- navigate to the **Build** toolbar and press the **Holder** button. This will add a holder to the respective element as seen below.



You will notice the **Holder** button icon has now changed from a green plus to a red cross. The functionality of the button has also changed from adding a holder to deleting a holder. This functionality is respective to each individual element and will change accordingly.

3) The final step is to choose a security concept and add the respective symbol. This is done by once again navigating to the **Build** toolbar and pressing any of the add security buttons. In this case we will choose **Access Control**. As seen below, the security symbol for *Access Control* has now been added to the highlighted element.

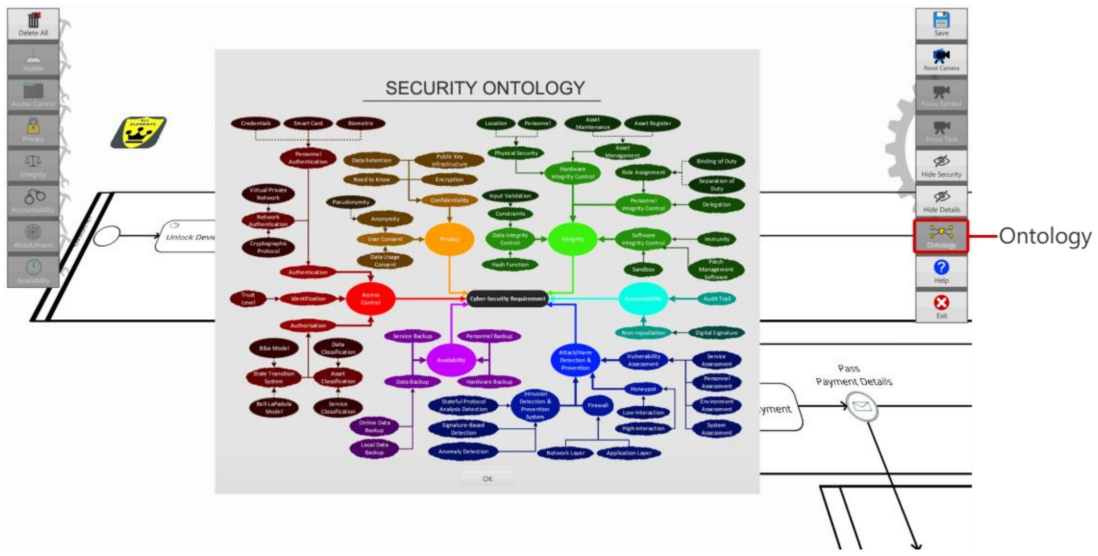


You will notice once again the icon for the **Access Control** button has changed. Like with the **Holder** button, the add security symbol buttons also change functionality depending on whether the element already has the security requirement.

Low Level Security Requirements

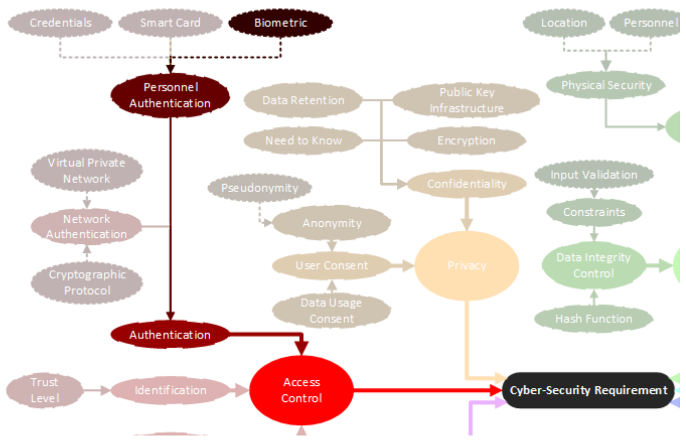
BPMN 3D has more than just these six security requirements. In total, the application is capable of specifying 76 unique requirements along with their properties. These requirements however are structured in a hierarchy and added to an element slightly differently to the core six.

1) The first step in adding a low level requirement is to find which of the core six it belongs to. You can find this information out by opening the security requirements ontology. This can be done by navigating to the **Settings** toolbar and pressing the **Ontology** button as seen below.



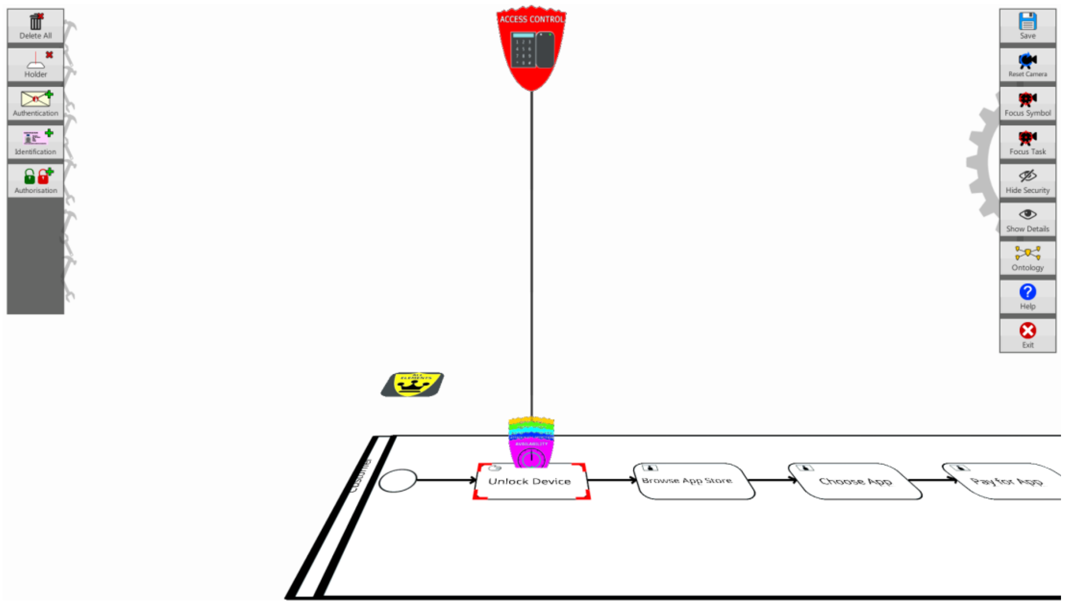
2) With the ontology open you can now find which branch your requirement belongs to. In this case we will choose *Biometric*. From the ontology we can see that *Biometric* is structured under the following requirements:

Biometric > Personnel Authentication > Authentication > Access Control



This tells us that to add *Biometric* as a security requirement we must first add *Access Control* then *Authentication* then *Personnel Authentication*.

3) Now that we know where *Biometric* is located we can add it to an element. As discovered in the previous step we must first add *Access Control*. Once *Access Control* has been added to the element, the symbol must be highlighted to allow for lower level requirements to be added. This is done by moving the mouse over the symbol and clicking the left mouse button. This will then highlight the requirement as seen below.

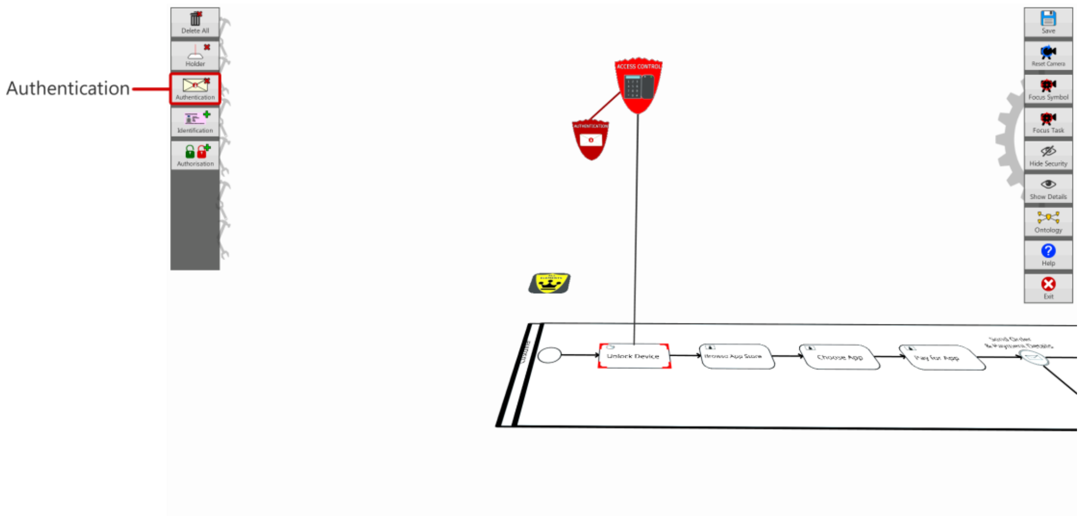


As seen, when a symbol is highlighted the other five requirements will collapse. This provides workspace to add lower level requirements to the highlighted symbol.

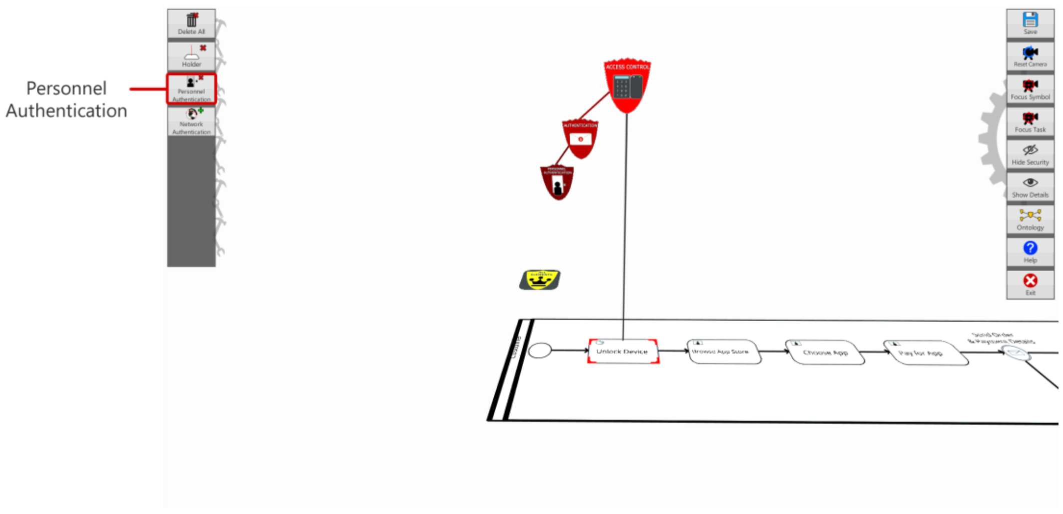
It also acts as modularisation for the application. If all 76 requirements were constantly visible on each element it would cause cognitive overload and make the diagram difficult to interpret. This method provides an elegant solution which allows the user to view lower level requirements by simply highlighting their respective parent.

4) Now that *Access Control* has been added to the diagram we need to add *Authentication*. You may have noticed that when you highlighted *Access Control* in the previous step, the **Build** toolbar changed. As well as collapsing or hiding other requirements, when a symbol is highlighted the **Build** toolbar will change to the symbols children security requirements. In the case of *Access Control* these are: *Authentication, Identification* and *Authorisation*.

   To add *Authentication* to the element you simply need to ensure you have *Access Control* highlighted then press the **Authentication** button in the **Build** toolbar. This will then add *Authentication* to the element as seen below.

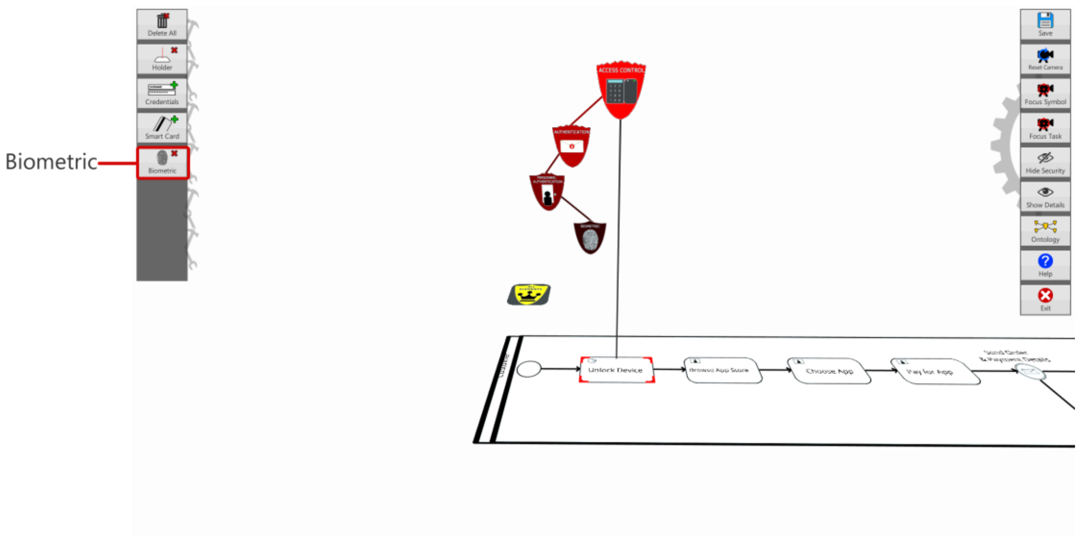


5) Next we need to add *Personnel Authentication*. This is done the same way as *Authentication*. First we must highlight the parent symbol by left clicking on it; in this case *Authentication*. This will then add the child requirements to the **Build** toolbar (*Personnel Authentication*). Now you simply press the **Personnel Authentication** button to add the requirement to the element as seen below.



(Note: If you have multiple child requirements on the same level, you will notice that when you highlight one of them the others disappear. This is to once again modularise the requirements and ensure the user is not overwhelmed when creating or reading a diagram.)

6) Now that we've added *Personnel Authentication*, all parent symbols are now present on the element. This means that we can now add *Biometric*. Like in the previous steps, highlight the parent symbol (*Personnel Authentication*) by left clicking on it. Now navigate to the **Build** toolbar and press the **Biometric** button. Like below, you will see the *Biometric* has now been added to the element.

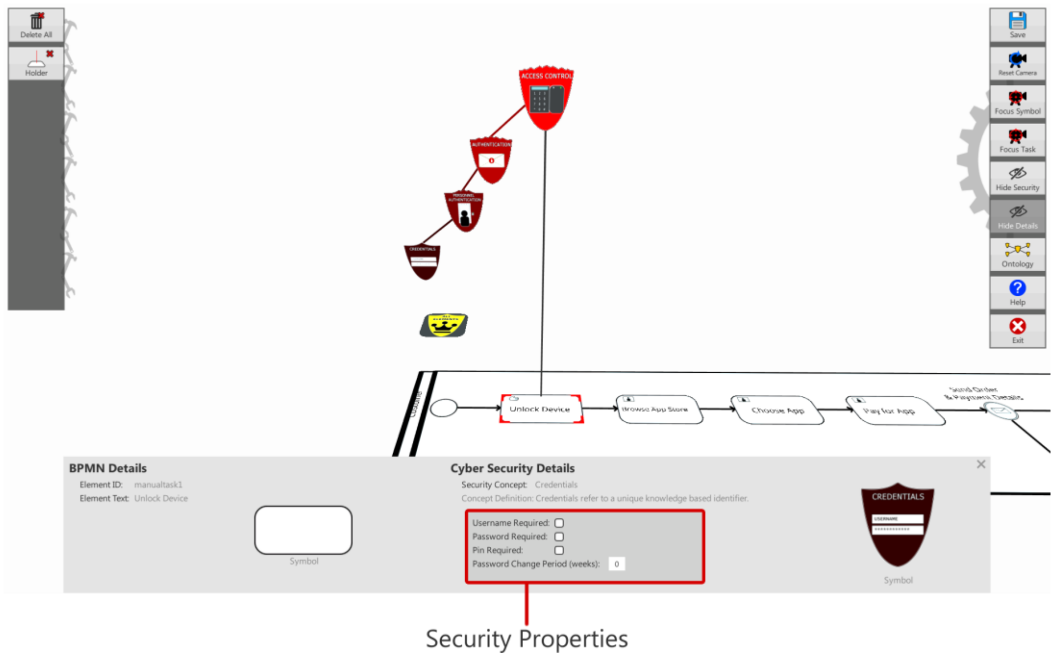


(Note: If you wish to close the hierarchy and return to any of the previous levels simply un-highlight the parent symbol by left clicking it again. You do not need to do this logically, if you want to close a full hierarchy from the lowest level simply left click the concept at the highest level. In the above example this would be done by left clicking Access Control.)

Specifying Properties

In BPMN 3D the user is able to specify properties for certain security requirements to further detail their needs.

1) This is accomplished by first highlighting the symbol you wish to specify properties for. If the respective symbol has properties they will appear in the detail bar as seen below.



BPMN Details

Element ID: manualtask1

Element Text: Unlock Device

Cyber Security Details

Security Concept: Credentials

Concept Definition: Credentials refer to a unique knowledge based identifier.

Username Required: ☐

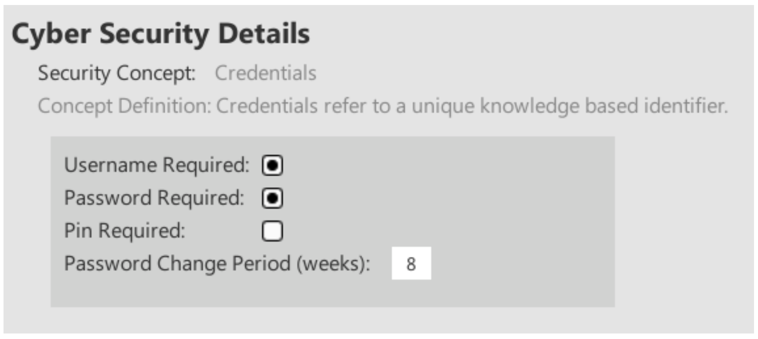
Password Required: ☐

Pin Required: ☐

Password Change Period (weeks): 0

*(Note: If you cannot see a details bar when you highlight a symbol, it is likely you have selected **Hide Details** in the **Settings** toolbar. If you press the **Show Details** button, the details bar will appear.)*

2) Next, you simply need to specify your properties. All of which consist of basic form input such as checkboxes, text boxes and dropdowns. It is just a case of specifying your requirements. In this case, we will say that *Credentials* require *username* and *password* with a *password change period* of *8 weeks*.



Cyber Security Details

Security Concept: Credentials

Concept Definition: Credentials refer to a unique knowledge based identifier.

Username Required: ☑

Password Required: ☑

Pin Required: ☐

Password Change Period (weeks): 8

All Elements

You will notice that on your diagram and in the policy creation there is a BPMN element called *All Elements*. This can be used to better manage the complexity of your diagrams.

ALL
ELEMENTS

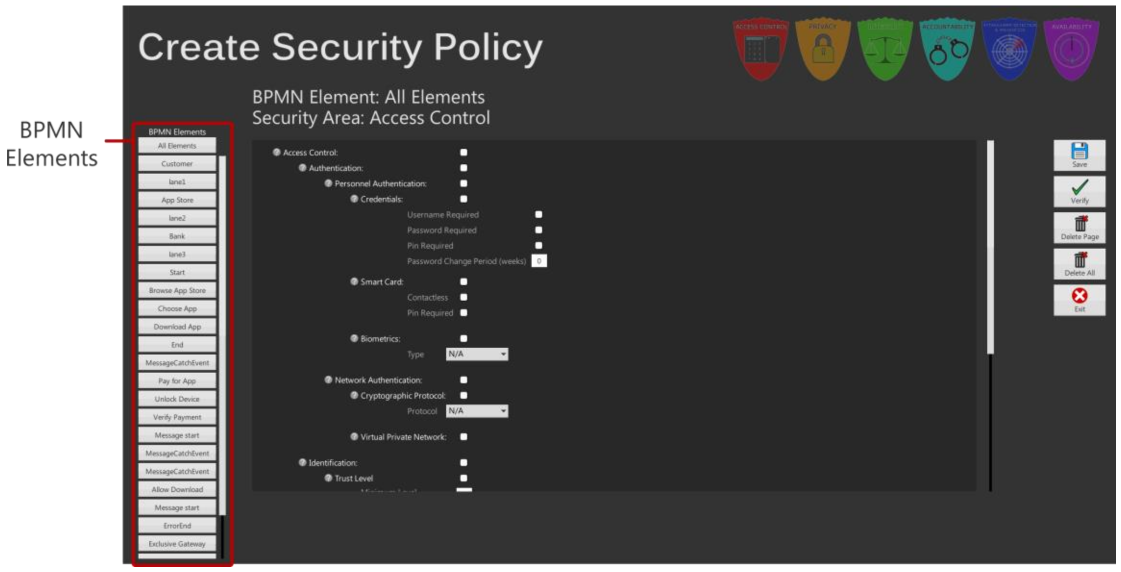
There may be instances where you want every element to have a certain security requirement, for example *Encryption*. Adding this requirement to every element not only increases the complexity of your diagram but also takes a long time. To overcome this issue, BPMN 3D added this element to act as a parent to the diagram itself.

BPMN 3D does not add the requirements on *All Elements* to every other individual element as this would unnecessarily increase complexity. Instead the user must work under the assumption that although each element may not have the requirement directly linked to it, if it is required on *All Elements* it is required on every other element too.

Create Security Policy

BPMN 3D also allows for security policy creation. Providing all the same functionality as the diagram editing only in a textual format. This allows for faster specification of security requirements and also acts as a customisable validation system for your diagram.

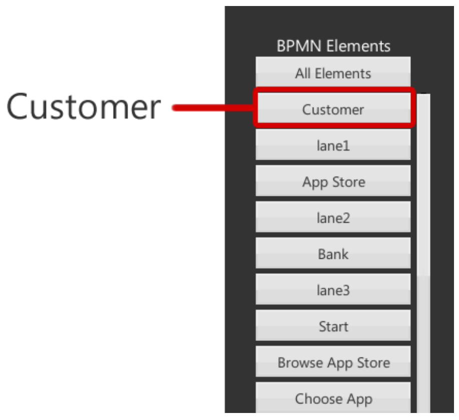
1) Creating a security policy for your diagram is very straightforward. Simply open **Create Security Policy** from the main menu and the BPMN elements from your diagram will be loaded in as seen below.



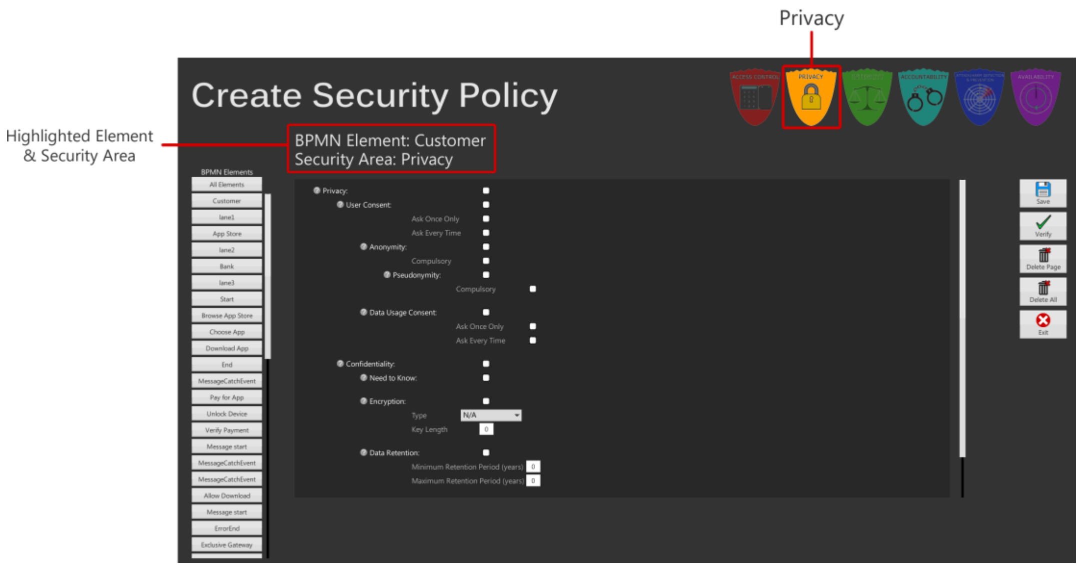
BPMN Element: All Elements
Security Area: Access Control

- Access Control:
  - Authentication:
    - Personnel Authentication:
      - Credentials:
        - Username Required
        - Password Required
        - Pin Required
        - Password Change Period (weeks) 0
      - Smart Card:
        - Contactless
        - Pin Required
      - Biometrics:
        - Type N/A
    - Network Authentication:
      - Cryptographic Protocol:
        - Protocol N/A
      - Virtual Private Network:
  - Identification:
    - Trust Level

2) You can then pick a BPMN element from the list and begin to specify security requirements for it. For instance, we will choose *Customer* and add *Privacy* security requirements. To select the *Customer* element, press the **Customer** button in the list of BPMN elements as seen below.



3) As we want to add *Privacy* security requirements we will also need to change the security area to *Privacy*. This can be done by pressing the **Privacy** button as seen below.



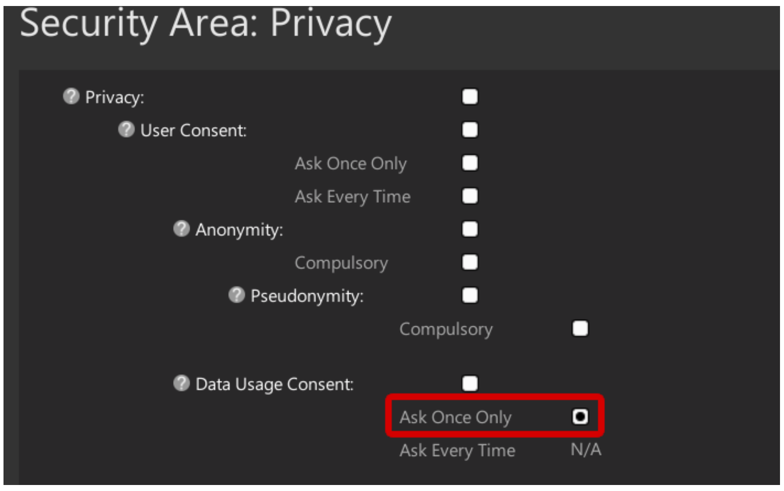
Create Security Policy

BPMN Element: Customer
Security Area: Privacy

Privacy:
User Consent:
Ask Once Only
Ask Every Time
Anonymity:
Compulsory
Pseudonymity:
Compulsory
Data Usage Consent:
Ask Once Only
Ask Every Time
Confidentiality:
Need to Know:
Encryption:
Type N/A
Key Length 0
Data Retention:
Minimum Retention Period (years) 0
Maximum Retention Period (years) 0

You will also notice that the currently highlighted BPMN element and security area will change respective to your choices. It is important you regularly check these to ensure you are working on the correct element.

4) With the *Privacy* security area selected you will see that the security requirements and their properties have changed. For this example we will specify that the *Customer* element will need to provide *Data Usage Consent, Ask Once Only*. The fastest way to specify this is to simply check the respective box in the form as seen below.

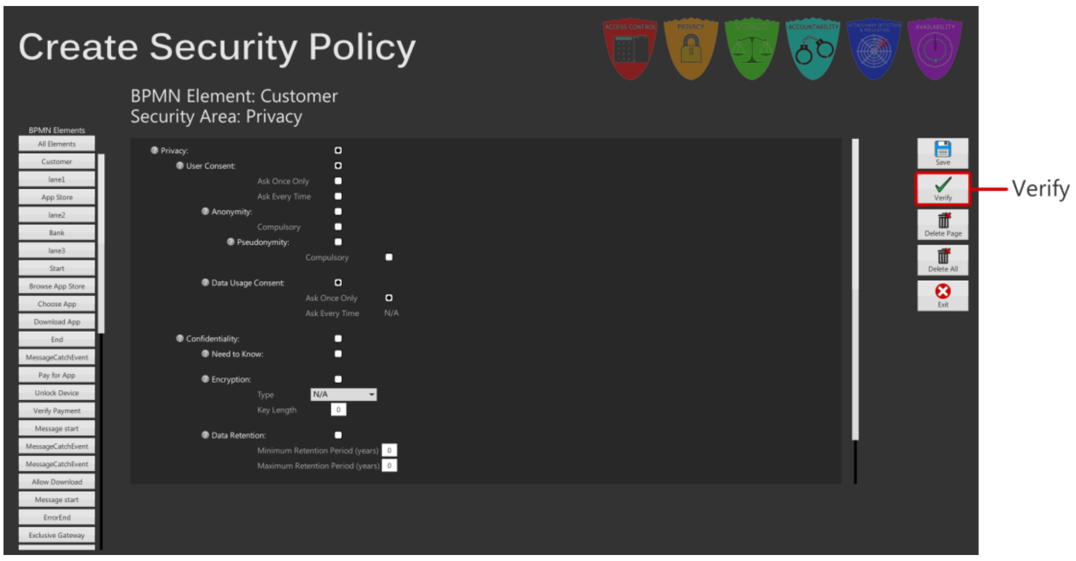


Security Area: Privacy

Privacy:
User Consent:
Ask Once Only
Ask Every Time
Anonymity:
Compulsory
Pseudonymity:
Compulsory
Data Usage Consent:
Ask Once Only ✓
Ask Every Time N/A

However, if we save the policy like this the application will throw several errors. This is because *Ask Once Only* is a property of a low level security requirement in which the parents have not been specified.

5) To overcome this issue you must either manually check all of the parent requirements to the highest level or you can simply press the **Verify** button and these will be checked for you.



Create Security Policy

BPMN Element: Customer
Security Area: Privacy

BPMN Elements: All Elements, Customer, lane1, App Store, lane2, Bank, lane3, Start, Browse App Store, Choose App, Download App, End, MessageCatchEvent, Pay for App, Unlock Device, Verify Payment, Message start, MessageCatchEvent, MessageCatchEvent, Allow Download, Message start, ErrorEnd, Exclusive Gateway

```
Privacy:
    User Consent:
            Ask Once Only
            Ask Every Time
        Anonymity:
            Compulsory
            Pseudonymity:
                Compulsory
        Data Usage Consent:
                Ask Once Only
                Ask Every Time      N/A
    Confidentiality:
        Need to Know:
        Encryption:
            Type        N/A
            Key Length  0
        Data Retention:
            Minimum Retention Period (years) 0
            Maximum Retention Period (years) 0
```

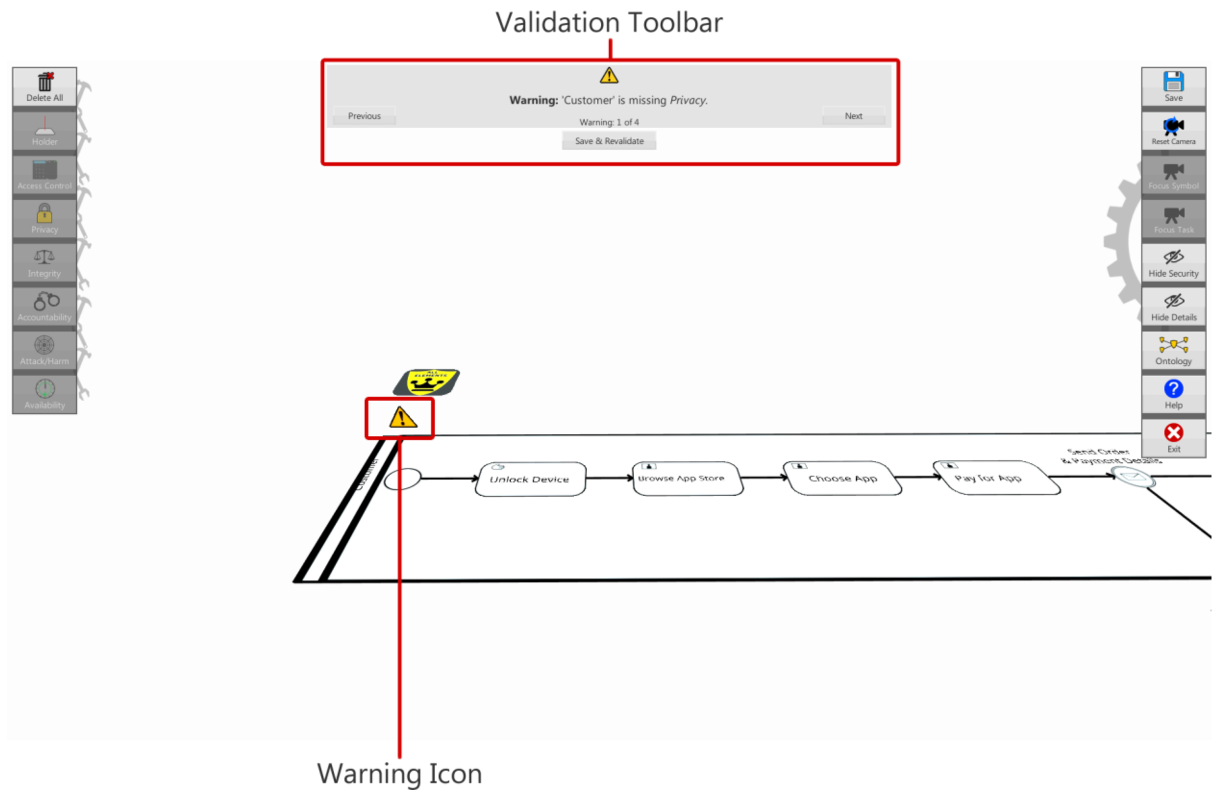
As seen, once the **Verify** button is pressed the application will autofill with the respective parent requirements.

*(Note: You will also see a **Delete Page** and **Delete All** button. Delete Page will delete all of the specified requirements in the current area where Delete All will delete every policy across every element.)*

Validate Security

Once you have added security requirements to your diagram and specified your own policy, you can use BPMN 3D's validation system.

1) If you navigate to **Validate Security** from the main menu you will be taking to a screen as seen below.

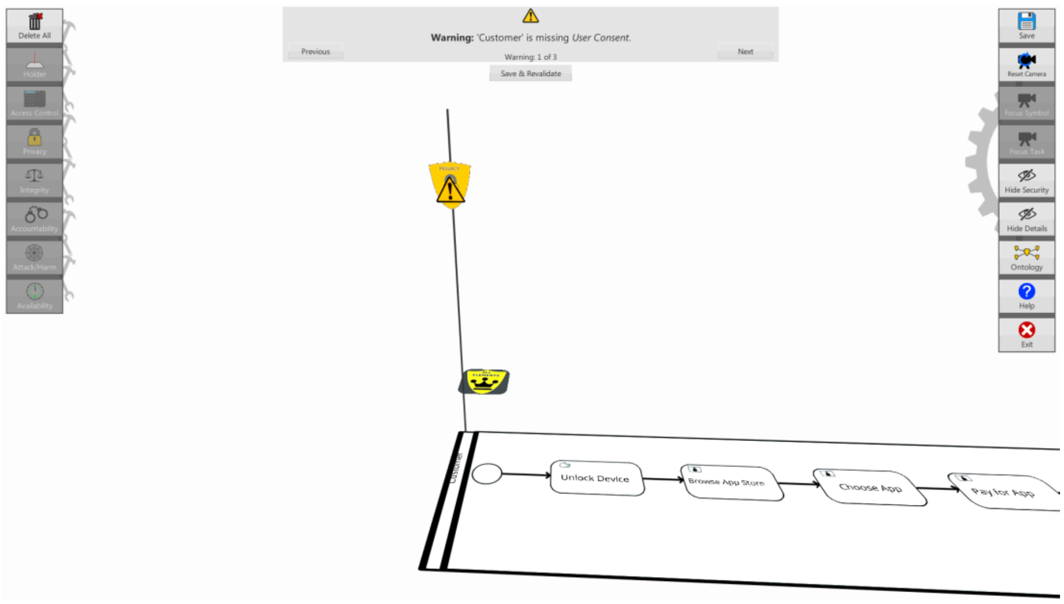


Warning: 'Customer' is missing *Privacy.*

Warning: 1 of 4

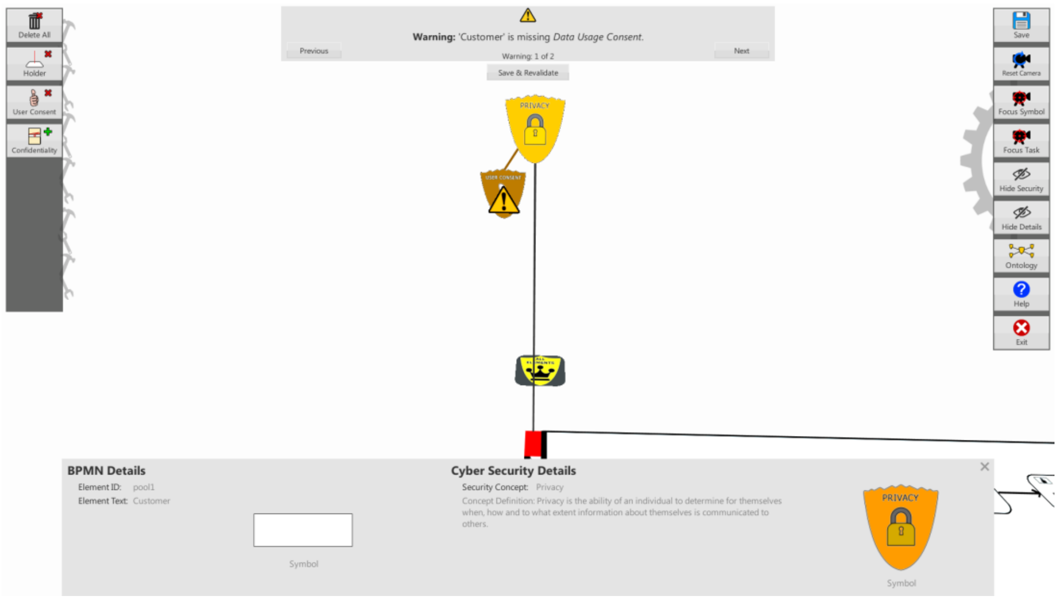
The *Validate Security* screen is very similar to *Add Security Requirements* featuring all the same functionality. The main difference being the *Validation* toolbar and the newly added warning icon.

2) The idea of *Validate Security* is to treat it like a code error system and logically work your way through the warnings. For instance, our first error specifies that "*Customer is missing Privacy*". As there is no parent symbol for *Privacy* the warning icon will hover over the BPMN element as a visual aid to notify us of the affected element. So let's add *Privacy* to *Customer,* then press **Save & Revalidate** to update the system.


Warning: 'Customer' is missing *User Consent*.

Warning: 1 of 3

3) As seen above, by adding *Privacy* to *Customer* we have fixed the validation warning, however we still have three warnings left. The next warning states "*Customer is missing User Consent*". You will notice that the warning icon has now moved to over the *Privacy* icon. This is because *Privacy* is a parent of *User Consent* and the icon is telling us the validation warning is to do with, or a child of *Privacy*. If we now add *User Consent* to *Customer* and revalidate the diagram the warning should disappear.


Warning: 'Customer' is missing *Data Usage Consent*.

Warning: 1 of 2

BPMN Details

Element ID: pool1

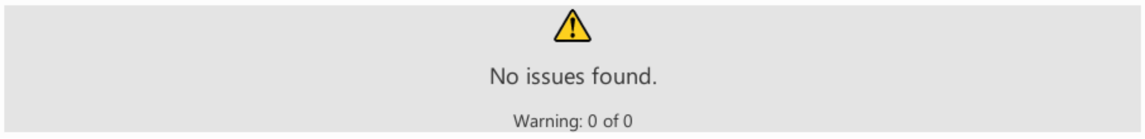
Element Text: Customer

Cyber Security Details

Security Concept: Privacy

Concept Definition: Privacy is the ability of an individual to determine for themselves when, how and to what extent information about themselves is communicated to others.

4) Throughout the previous steps we have successfully managed to fix two of our validation warnings. Continuing this process you will eventually have a fully validated and warning free diagram. The application will notify you of this with the below message.


No issues found.

Warning: 0 of 0

Troubleshooting

If for any reason you happen to experience any issues while using the application the first thing to do is delete all security requirements from your diagram and delete all policies. If you do this then save each respective file, the application should run as normal again.

However, if the application is still experiencing issues please contact Curtis Maines via c.l.maines@2011.ljmu.ac.uk.

2D versus 3D Experiment Results

Multimedia Computing Students:

			Education				
	Gender	Age Range	Current	Cyber Security	Art & Graphics	3D Application Usage	Learning Difficulty
MM5P1	Male	18-24	Undergrad #2	Undergrad #1	Undergrad #1	Everyday	None
MM5P2	Female	18-24	Undergrad #2	A Level	GCSE	1 a week	None
MM5P3	Female	18-24	Undergrad #2	GCSE	Undergrad #1	Never	None
MM5P4	Male	18-24	Undergrad #2	None	GCSE	Everyday	None
MM5P5	Male	35-44	Undergrad #2	None	Undergrad #1	1 a week	None
MM5P6	Female	35-44	Undergrad #2	None	A Level	1 a month	None
MM5P7	Male	18-24	Undergrad #2	None	None	Everyday	Dyslexia
MM5P8	Female	18-24	Undergrad #2	None	Undergrad #1	1 a month	None
MM5P9	Male	18-24	Undergrad #2	Undergrad #1	Undergrad #1	1 a week	None
MM5P10	Male	18-24	Undergrad #2	None	Undergrad #1	1 a week	None
MM5P11	Male	18-24	Undergrad #2	None	Undergrad #1	Everyday	None
MM5P12	Male	18-24	Undergrad #2	None	Undergrad #1	1 a week	None
MM5P13	Female	18-24	Undergrad #2	A Level	A Level	1 a week	None
MM5P14	Male	18-24	Undergrad #2	Undergrad #1	Undergrad #1	Everyday	None
MM5P15	Male	18-24	Undergrad #2	A Level	A Level	Everyday	None
MM5P16	Male	18-24	Undergrad #2	None	GCSE	Everyday	None
MM5P17	Male	18-24	Undergrad #2	None	Undergrad #1	Everyday	None
MM5P18	Male	18-24	Undergrad #2	None	Undergrad #1	Everyday	None
MM5P19	Male	18-24	Undergrad #2	Undergrad #1	Undergrad #1	Everyday	None
MM5P20	Male	18-24	Undergrad #2	Undergrad #1	Undergrad #1	Everyday	None
MM5P21	Male	18-24	Undergrad #2	Undergrad #1	Undergrad #1	Everyday	Dyslexia
MM5P22	Male	18-24	Undergrad #2	Undergrad #1	Undergrad #1	Everyday	None
MM5P23	Male	18-24	Undergrad #2	Undergrad #1	Undergrad #1	Everyday	None
MM5P24	Female	18-24	Undergrad #2	Undergrad #1	None	1 a week	None
MM5P25	Male	18-24	Undergrad #2	None	None		None
MM5P26	Female	18-24	Undergrad #2	None	A Level	Everyday	None
MM5P27	Male	18-24	Undergrad #2	None	Undergrad #2	1 a week	None
MM5P28	Male	18-24	Undergrad #2	None	GCSE	Everyday	None
MM5P29	Female	18-24	Undergrad #2	None	A Level	Never	None
MM5P30	Male	18-24	Undergrad #2	Undergrad #1	Undergrad #1	1 a week	None

	Correct (out of 9)		Wrong (out of 9)		Completion (%)		Accuracy %		Efficieny%	
	2D Simple	3D Simple	2D Simple	3D Simple	2D Simple	3D Simple	2D Simple	3D Simple	Completion	Accuracy
MM5P1	9	9	0	0	100	100	100	100	0	0
MM5P2	9	6	0	1	100	77.77777778	100	85.71428571	-22.22222222	-14.28571429
MM5P3	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE
MM5P4	9	9	0	0	100	100	100	100	0	0
MM5P5	9	9	0	0	100	100	100	100	0	0
MM5P6	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE
MM5P7	7	9	0	0	77.77777778	100	100	100	22.22222222	0
MM5P8	7	8	2	1	100	100	77.77777778	88.88888889	0	11.11111111
MM5P9	8	6	0	1	88.88888889	77.77777778	100	85.71428571	-11.11111111	-14.28571429
MM5P10	9	9	0	0	100	100	100	100	0	0
MM5P11	7	9	0	0	77.77777778	100	100	100	22.22222222	0
MM5P12	9	9	0	0	100	100	100	100	0	0
MM5P13	8	9	0	0	88.88888889	100	100	100	11.11111111	0
MM5P14	6	8	1	1	77.77777778	100	85.71428571	88.88888889	22.22222222	3.174603175
MM5P15	9	5	0	1	100	66.66666667	100	83.33333333	-33.33333333	-16.66666667
MM5P16	4	2	2	0	66.66666667	22.22222222	66.66666667	100	-44.44444444	33.33333333
MM5P17	7	9	0	0	77.77777778	100	100	100	22.22222222	0
MM5P18	9	2	0	1	100	33.33333333	100	66.66666667	-66.66666667	-33.33333333
MM5P19	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE
MM5P20	7	9	2	0	100	100	77.77777778	100	0	22.22222222
MM5P21	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE
MM5P22	9	9	0	0	100	100	100	100	0	0
MM5P23	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE
MM5P24	8	9	0	0	88.88888889	100	100	100	11.11111111	0
MM5P25	9	6	0	1	100	77.77777778	100	85.71428571	-22.22222222	-14.28571429
MM5P26	9	9	0	0	100	100	100	100	0	0
MM5P27	9	9	0	0	100	100	100	100	0	0
MM5P28	9	9	0	0	100	100	100	100	0	0
MM5P29	9	9	0	0	100	100	100	100	0	0
MM5P30	1	5	1	1	22.22222222	66.66666667	50	83.33333333	44.44444444	33.33333333
Average	7.84	7.68	0.32	0.32	90.67	88.89	94.32	94.73	-1.78	0.41

	Correct (out of 9)		Wrong (out of 9)		Completion (%)		Accuracy %		Improvement	
	2D Complex	3D Complex	2D Complex	3D Complex	2D Complex	3D Complex	2D Complex	3D Complex	Completion	Accuracy
MM5P1	3	8	3	1	66.66666667	100	50	88.88888889	33.33333333	38.88888889
MM5P2	4	9	5	0	100	100	44.44444444	100	0	55.55555556
MM5P3	1	9	6	0	77.77777778	100	14.28571429	100	22.22222222	85.71428571
MM5P4	4	9	5	0	100	100	44.44444444	100	0	55.55555556
MM5P5	4	9	2	0	66.66666667	100	66.66666667	100	33.33333333	33.33333333
MM5P6	1	4	8	4	100	88.88888889	11.11111111	50	-11.11111111	38.88888889
MM5P7	3	8	2	1	55.55555556	100	60	88.88888889	44.44444444	28.88888889
MM5P8	2	8	5	1	77.77777778	100	28.57142857	88.88888889	22.22222222	60.31746032
MM5P9	3	5	3	2	66.66666667	77.77777778	50	71.42857143	11.11111111	21.42857143
MM5P10	3	8	4	1	77.77777778	100	42.85714286	88.88888889	22.22222222	46.03174603
MM5P11	6	6	1	3	77.77777778	100	85.71428571	66.66666667	22.22222222	-19.04761905
MM5P12	9	9	0	0	100	100	100	100	0	0
MM5P13	1	4	8	5	100	100	11.11111111	44.44444444	0	33.33333333
MM5P14	2	8	6	1	88.88888889	100	25	88.88888889	11.11111111	63.88888889
MM5P15	3	9	4	0	77.77777778	100	42.85714286	100	22.22222222	57.14285714
MM5P16	2	7	3	0	55.55555556	77.77777778	40	100	22.22222222	60
MM5P17	2	9	7	0	100	100	22.22222222	100	0	77.77777778
MM5P18	3	7	3	1	66.66666667	88.88888889	50	87.5	22.22222222	37.5
MM5P19	1	8	8	1	100	100	11.11111111	88.88888889	0	77.77777778
MM5P20	6	8	1	1	77.77777778	100	85.71428571	88.88888889	22.22222222	3.174603175
MM5P21	3	9	3	0	66.66666667	100	50	100	33.33333333	50
MM5P22	6	9	3	0	100	100	66.66666667	100	0	33.33333333
MM5P23	5	9	1	0	66.66666667	100	83.33333333	100	33.33333333	16.66666667
MM5P24	4	9	5	0	100	100	44.44444444	100	0	55.55555556
MM5P25	6	9	1	0	77.77777778	100	85.71428571	100	22.22222222	14.28571429
MM5P26	7	9	2	0	100	100	77.77777778	100	0	22.22222222
MM5P27	0	7	9	2	100	100	0	77.77777778	0	77.77777778
MM5P28	6	9	3	0	100	100	66.66666667	100	0	33.33333333
MM5P29	6	9	3	0	100	100	66.66666667	100	0	33.33333333
MM5P30	0	4	5	4	55.55555556	88.88888889	0	50	33.33333333	50
Average	**3.53**	**7.83**	**3.97**	**0.93**	**83.33**	**97.41**	**47.58**	**89**	**14.07**	**41.42**

	2D Simple					3D Simple					
	Difficulty	Overwhelmed?	Method	Feelings		Difficulty	Overwhelmed?	Method	Feelings	Easier Extension	Preference
MM5P1	Easy	No	Very Good	Happy	MM5P1	Easy	No	Very Good	Happy	2D	2D
MM5P2	Easy	No	Good	Neutral	MM5P2	Difficult	No	Good	Neutral	2D	2D
MM5P3	NA	NA	NA	NA	MM5P3	NA	NA	NA	NA	NA	NA
MM5P4	Very Easy	No	Good	Happy	MM5P4	Easy	No	Good	Happy	2D	2D
MM5P5	Easy	No	Good	Neutral	MM5P5	Easy	No	Poor	Neutral	2D	2D
MM5P6	NA	NA	NA	NA	MM5P6	NA	NA	NA	NA	NA	NA
MM5P7	Very Easy	No	Good	Neutral	MM5P7	Easy	No	Good	Neutral	2D	2D
MM5P8	Easy	No	Poor	Neutral	MM5P8	Easy	No	Good	Neutral	3D	3D
MM5P9	Easy	Yes	Good	Happy	MM5P9	Difficult	Yes	Good	Frustrated	2D	2D
MM5P10	Easy	No	Good	Happy	MM5P10	Easy	No	Good	Neutral	2D	2D
MM5P11	Very Easy	No	Good	Neutral	MM5P11	Easy	No	Good	Happy	2D	2D
MM5P12	Very Easy	No	Good	Happy	MM5P12	Easy	No	Good	Happy	2D	2D
MM5P13	Very Easy	No	Very Poor	Frustrated	MM5P13	Very Easy	No	Very Good	Neutral	3D	3D
MM5P14	Very Easy	No	Good	Happy	MM5P14	Very Easy	No	Good	Happy	2D	2D
MM5P15	Easy	No	Poor	Happy	MM5P15	Difficult	No	Poor	Neutral	2D	2D
MM5P16	Easy	No	Good	Neutral	MM5P16	Difficult	No	Poor	Neutral	2D	3D
MM5P17	Easy	No	Good	Happy	MM5P17	Easy	No	Good	Happy	2D	2D
MM5P18	Very Easy	No	Good	Neutral	MM5P18	Difficult	Yes	Poor	Neutral	2D	2D
MM5P19	NA	NA	NA	NA	MM5P19	NA	NA	NA	NA	NA	NA
MM5P20	Very Easy	No	Good	Happy	MM5P20	Very Easy	No	Poor	Neutral	2D	2D
MM5P21	NA	NA	NA	NA	MM5P21	NA	NA	NA	NA	NA	NA
MM5P22	Very Easy	No	Good	Neutral	MM5P22	Very Easy	No	Good	Neutral	2D	2D
MM5P23	NA	NA	NA	NA	MM5P23	NA	NA	NA	NA	NA	NA
MM5P24	Easy	No	Poor	Neutral	MM5P24	Easy	No	Poor	Neutral	2D	2D
MM5P25	Very Easy	No	Good	Happy	MM5P25	Easy	Yes	Good	Neutral	2D	2D
MM5P26	Very Easy	No	Good	Happy	MM5P26	Very Easy	No	Good	Happy	2D	2D
MM5P27	Easy	No	Good	Neutral	MM5P27	Easy	No	Good	Neutral	3D	3D
MM5P28	Very Easy	No	Very Good	Neutral	MM5P28	Very Easy	No	Very Good	Neutral	2D	2D
MM5P29	Easy	No	Good	Happy	MM5P29	Easy	Yes	Good	Neutral	2D	2D
MM5P30	Difficult	No	Good	Neutral	MM5P30	Difficult	Yes	Good	Neutral	2D	2D

	2D Complex					3D Complex					
	Difficulty	Overwhelmed?	Method	Feelings		Difficulty	Overwhelmed?	Method	Feelings	Easier Extension	Preference
MM5P1	Difficult	No	Poor	Frustrated	MM5P1	Easy	No	Good	Happy	3D	3D
MM5P2	Difficult	Yes	Good	Neutral	MM5P2	Difficult	Yes	Good	Neutral	2D	2D
MM5P3	Difficult	Yes	Poor	Neutral	MM5P3	Difficult	Yes	Poor	Neutral	3D	3D
MM5P4	Difficult	No	Good	Happy	MM5P4	Easy	No	Good	Happy	3D	3D
MM5P5	Difficult	Yes	Poor	Frustrated	MM5P5	Very Easy	No	Very Good	Happy	3D	3D
MM5P6	Difficult	Yes	Good	Frustrated	MM5P6	Difficult	Yes	Good	Neutral	3D	3D
MM5P7	Difficult	Yes	Poor	Frustrated	MM5P7	Easy	No	Very Good	Happy	3D	3D
MM5P8	Difficult	Yes	Poor	Frustrated	MM5P8	Very Easy	No	Good	Happy	3D	3D
MM5P9	Very Difficult	Yes	Good	Frustrated	MM5P9	Easy	No	Good	Neutral	3D	3D
MM5P10	Difficult	Yes	Good	Neutral	MM5P10	Very Easy	No	Good	Happy	3D	3D
MM5P11	Difficult	Yes	Good	Neutral	MM5P11	Difficult	Yes	Good	Neutral	2D	2D
MM5P12	Easy	Yes	Poor	Happy	MM5P12	Easy	No	Very Good	Happy	3D	3D
MM5P13	Very Difficult	Yes	Very Poor	Frustrated	MM5P13	Very Difficult	Yes	Very Poor	Frustrated	3D	3D
MM5P14	Easy	No	Poor	Happy	MM5P14	Easy	No	Good	Happy	3D	3D
MM5P15	Easy	No	Good	Happy	MM5P15	Easy	No	Good	Happy	3D	3D
MM5P16	Difficult	Yes	Poor	Neutral	MM5P16	Easy	No	Good	Neutral	3D	2D
MM5P17	Difficult	Yes	Poor	Neutral	MM5P17	Easy	No	Good	Happy	2D	2D
MM5P18	Difficult	Yes	Poor	Neutral	MM5P18	Easy	No	Good	Neutral	3D	3D
MM5P19	Difficult	Yes	Very Poor	Frustrated	MM5P19	Very Easy	No	Very Good	Neutral	2D	2D
MM5P20	Difficult	Yes	Very Poor	Frustrated	MM5P20	Easy	No	Good	Happy	3D	3D
MM5P21	Very Difficult	Yes	Very Poor	Frustrated	MM5P21	Very Easy	No	Very Good	Happy	3D	3D
MM5P22	Easy	Yes	Good	Neutral	MM5P22	Very Easy	No	Very Good	Happy	3D	3D
MM5P23	Difficult	Yes	Good	Happy	MM5P23	Very Easy	No	Very Good	Happy	3D	3D
MM5P24	Difficult	Yes	Very Poor	Neutral	MM5P24	Easy	No	Poor	Neutral	3D	3D
MM5P25	Very Difficult	Yes	Good	Frustrated	MM5P25	Easy	No	Good	Happy	3D	3D
MM5P26	Difficult	No	Poor	Neutral	MM5P26	Very Easy	No	Very Good	Happy	3D	3D
MM5P27	Difficult	Yes	Good	Neutral	MM5P27	Easy	No	Very Good	Neutral	3D	3D
MM5P28	Difficult	Yes	Poor	Happy	MM5P28	Very Easy	No	Very Good	Neutral	3D	3D
MM5P29	Difficult	Yes	Poor	Frustrated	MM5P29	Easy	No	Good	Neutral	3D	3D
MM5P30	Very Difficult	Yes	Good	Frustrated	MM5P30	Difficult	Yes	Good	Frustrated	2D	2D

Cyber Security Students:

			Education				
	Gender	Age Range	Current	Cyber Security	Art & Graphics	3D Application Usage	Learning Difficulty
S4P1	Male	18-24	Undergrad #1	None	None	1 a week	None
S4P2	Male	18-24	Undergrad #1	None	None	Everyday	None
S4P3	Male	18-24	Undergrad #1	A Level	GCSE	Everyday	Deaf/Hearing Impairment
S4P4	Male	18-24	Undergrad #1	None	GCSE	Everyday	None
S4P5	Male	18-24	Undergrad #1	A Level	GCSE	Never	None
S4P6	Male	18-24	Undergrad #1	None	None	Everyday	None
S4P7	Male	18-24	Undergrad #1	None	None	Everyday	None
S4P8	Male	18-24	Undergrad #1	None	None	Everyday	None
S4P9	Male	18-24	Undergrad #1	None	None	Everyday	None
S4P10	Male	18-24	Undergrad #1	None	None	1 a week	None
S4P11	Female	18-24	Undergrad #1	None	None	Everyday	None
S4P12	Male	18-24	Undergrad #1	None	None	Everyday	None
S4P13	Male	18-24	Undergrad #1	None	None	1 a week	None
S4P14	Male	18-24	Undergrad #1	None	None	Everyday	
S4P15	Male	18-24	Undergrad #1	A Level	None	1 a week	Other
S4P16	Male	18-24	Undergrad #1	None	None	1 a week	Dyslexia

	Correct (out of 9)		Wrong (out of 9)		Completion (%)		Accuracy %		Improvement	
	2D Simple	3D Simple	2D Simple	3D Simple	2D Simple	3D Simple	2D Simple	3D Simple	Completion	Accuracy
S4P1	3	8	0	1	33.33333333	100	100	88.88888889	66.66666667	-11.11111111
S4P2	9	9	0	0	100	100	100	100	0	0
S4P3	9	9	0	0	100	100	100	100	0	0
S4P4	9	9	0	0	100	100	100	100	0	0
S4P5	8	9	0	0	88.88888889	100	100	100	11.11111111	0
S4P6	9	9	0	0	100	100	100	100	0	0
S4P7	9	7	0	2	100	100	100	77.77777778	0	-22.22222222
S4P8	7	9	0	0	77.77777778	100	100	100	22.22222222	0
S4P9	7	9	0	0	77.77777778	100	100	100	22.22222222	0
S4P10	6	8	0	1	66.66666667	100	100	88.88888889	33.33333333	-11.11111111
S4P11	7	9	0	0	77.77777778	100	100	100	22.22222222	0
S4P12	7	9	2	0	100	100	77.77777778	100	0	28.57142857
S4P13	9	9	0	0	100	100	100	100	0	0
S4P14	7	8	0	1	77.77777778	100	100	88.88888889	22.22222222	-11.11111111
S4P15	2	5	0	0	22.22222222	55.55555556	100	100	33.33333333	0
S4P16	6	9	3	0	100	100	66.66666667	100	0	50
Average	**7.13**	**8.44**	**0.31**	**0.31**	**82.64**	**97.22**	**96.53**	**96.53**	**14.58**	**1.44**

	Correct (out of 9)		Wrong (out of 9)		Completion (%)		Accuracy %		Improvement	
	2D Complex	3D Complex	2D Complex	3D Complex	2D Complex	3D Complex	2D Complex	3D Complex	Completion	Accuracy
S4P1	2	7	7	2	100	100	22.22222222	77.77777778	0	55.55555556
S4P2	4	8	5	1	100	100	44.44444444	88.88888889	0	44.44444444
S4P3	4	8	5	1	100	100	44.44444444	88.88888889	0	44.44444444
S4P4	7	9	2	0	100	100	77.77777778	100	0	22.22222222
S4P5	4	9	3	0	77.77777778	100	57.14285714	100	22.22222222	42.85714286
S4P6	6	8	3	1	100	100	66.66666667	88.88888889	0	22.22222222
S4P7	0	9	7	0	77.77777778	100	0	100	22.22222222	100
S4P8	5	9	1	0	66.66666667	100	83.33333333	100	33.33333333	16.66666667
S4P9	5	7	2	2	77.77777778	100	71.42857143	77.77777778	22.22222222	6.349206349
S4P10	6	9	3	0	100	100	66.66666667	100	0	33.33333333
S4P11	5	5	2	2	77.77777778	77.77777778	71.42857143	71.42857143	0	0
S4P12	5	8	4	1	100	100	55.55555556	88.88888889	0	33.33333333
S4P13	6	9	1	0	77.77777778	100	85.71428571	100	22.22222222	14.28571429
S4P14	3	7	6	2	100	100	33.33333333	77.77777778	0	44.44444444
S4P15	0	5	6	1	66.66666667	66.66666667	0	83.33333333	0	83.33333333
S4P16	5	9	4	0	100	100	55.55555556	100	0	44.44444444
Average	**4.19**	**7.88**	**3.81**	**0.81**	**88.89**	**96.53**	**52.23**	**90.23**	**7.64**	**38**

	2D Simple			
	Difficulty	Overwhelmed?	Method	Feelings
S4P1	Very Easy	Yes	Good	Happy
S4P2	Very Easy	No	Good	Happy
S4P3	Very Easy	Yes	Good	Neutral
S4P4	Easy	No	Poor	Neutral
S4P5	Easy	No	Good	Neutral
S4P6	Very Easy	No	Good	Happy
S4P7	Easy	No	Good	Neutral
S4P8	Very Easy	No	Good	Happy
S4P9	Easy	No	Poor	Neutral
S4P10	Easy	No	Good	Neutral
S4P11	Easy	No	Good	Neutral
S4P12	Very Easy	No	Good	Happy
S4P13	Very Easy	No	Good	Happy
S4P14	Easy	No	Good	Neutral
S4P15	Difficult	Yes	Good	Frustrated
S4P16	Very Easy	No	Poor	Happy

	3D Simple					
	Difficulty	Overwhelmed?	Method	Feelings	Easier Extension	Preference
S4P1	Easy	Yes	Good	Happy	2D	3D
S4P2	Very Easy	No	Poor	Happy	2D	2D
S4P3	Very Easy	Yes	Very Good	Neutral	2D	2D
S4P4	Easy	No	Good	Neutral	2D	3D
S4P5	Easy	No	Good	Neutral	3D	3D
S4P6	Very Easy	No	Very Good	Happy	3D	3D
S4P7	Difficult	No	Good	Neutral	2D	2D
S4P8	Very Easy	No	Good	Happy	3D	3D
S4P9	Very Easy	No	Good	Happy	3D	3D
S4P10	Very Easy	No	Good	Neutral	3D	3D
S4P11	Difficult	Yes	Good	Neutral	2D	2D
S4P12	Easy	No	Good	Happy	2D	3D
S4P13	Easy	No	Good	Happy	3D	3D
S4P14	Easy	No	Good	Frustrated	3D	3D
S4P15	Difficult	Yes	Good	Neutral	2D	2D
S4P16	Very Easy	No	Good	Happy	2D	2D

	2D Complex			
	Difficulty	Overwhelmed?	Method	Feelings
S4P1	Easy		Good	Happy
S4P2	Difficult	No	Poor	Happy
S4P3	Difficult	Yes	Very Good	Neutral
S4P4	Difficult	Yes	Good	Neutral
S4P5	Difficult	No	Good	Neutral
S4P6	Difficult	Yes	Poor	Neutral
S4P7	Difficult	Yes	Poor	Neutral
S4P8	Difficult	Yes	Very Poor	Frustrated
S4P9	Difficult	Yes	Good	Happy
S4P10	Difficult	Yes	Good	Neutral
S4P11	Difficult	Yes	Poor	Frustrated
S4P12	Difficult	Yes	Good	Happy
S4P13	Difficult	Yes	Poor	Neutral
S4P14	Difficult	No	Poor	Frustrated
S4P15	Very Difficult	Yes	Poor	Frustrated
S4P16	Difficult	Yes	Poor	Frustrated

	3D Complex					
	Difficulty	Overwhelmed?	Method	Feelings	Easier Extension	Preference
S4P1	Very Easy	Yes	Good	Happy	2D	3D
S4P2	Very Easy	No	Good	Happy	3D	3D
S4P3	Easy	Yes	Very Good	Neutral	3D	3D
S4P4	Easy	No	Good	Happy	3D	3D
S4P5	Easy	No	Good	Neutral	3D	3D
S4P6	Easy	Yes	Good	Happy	3D	3D
S4P7	Easy	No	Very Good	Neutral	3D	3D
S4P8	Easy	No	Good	Happy	3D	3D
S4P9	Very Easy	No	Very Good	Happy	3D	3D
S4P10	Easy	Yes	Good	Neutral	3D	3D
S4P11	Difficult	Yes	Poor	Frustrated	3D	3D
S4P12	Easy	No	Good	Happy	3D	3D
S4P13	Easy	No	Very Good	Happy	3D	3D
S4P14	Difficult	Yes	Good	Happy	3D	3D
S4P15	Very Difficult	Yes	Poor	Frustrated	2D	2D
S4P16	Difficult	No	Good	Happy	3D	3D

Business Studies Students:

			Education				
	Gender	Age Range	Current	Cyber Security	Art & Graphics	3D Application Usage	Learning Difficulty
B6P1	Male	25-34	PhD	None	None	Never	None
B6P2	Male	18-24	Undergrad #3	None	None	Never	None
B6P3	Female	18-24	Undergrad #3	None	None	Never	None
B6P4	Female	18-24	Undergrad #3	None	A Level	Never	None
B6P5	Female	18-24	Undergrad #3	None	None	Never	None
B6P6	Male	18-24	Undergrad #3	None	None	1 a week	None
B6P7	Male	18-24	Undergrad #3	None	None	1 a month	None
B6P8	Male	18-24	Undergrad #3	None	GCSE	Everyday	None
B6P9	Male	18-24	Undergrad #3	None	None	1 a week	None
B6P10	Male	18-24	Undergrad #3	None	None	Never	None
B6P11	Female	18-24	Undergrad #3	None	GCSE	1 a month	None
B6P12	Male	18-24	Undergrad #3	None	None	1 a week	None
B6P13	Female	18-24	Undergrad #3	None	GCSE	Never	None

	Correct (out of 9)		Wrong (out of 9)		Completion (%)		Accuracy %		Improvement	
	2D Simple	3D Simple	2D Simple	3D Simple	2D Simple	3D Simple	2D Simple	3D Simple	Completion	Accuracy
B6P1	9	8	0	1	100	100	100	88.88888889	0	-11.11111111
B6P2	5	8	0	1	55.55555556	100	100	88.88888889	44.44444444	-11.11111111
B6P3	2	6	0	3	22.22222222	100	100	66.66666667	77.77777778	-33.33333333
B6P4	1	8	0	1	11.11111111	100	100	88.88888889	88.88888889	-11.11111111
B6P5	6	8	0	1	66.66666667	100	100	88.88888889	33.33333333	-11.11111111
B6P6	5	8	0	1	55.55555556	100	100	88.88888889	44.44444444	-11.11111111
B6P7	9	9	0	0	100	100	100	100	0	0
B6P8	6	7	0	2	66.66666667	100	100	77.77777778	33.33333333	-22.22222222
B6P9	7	9	2	0	100	100	77.77777778	100	0	22.22222222
B6P10	7	8	2	1	100	100	77.77777778	88.88888889	0	11.11111111
B6P11	9	8	0	1	100	100	100	88.88888889	0	-11.11111111
B6P12	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE	DID NOT COMPLETE
B6P13	1	2	1	0	22.22222222	22.22222222	50	100	0	50
Average	**5.58**	**7.42**	**0.42**	**1**	**66.67**	**93.52**	**92.13**	**88.89**	**26.85**	**-3.24**

	Correct (out of 9)		Wrong (out of 9)		Completion (%)		Accuracy %		Improvement	
	2D Complex	3D Complex	2D Complex	3D Complex	2D Complex	3D Complex	2D Complex	3D Complex	Completion	Accuracy
B6P1	2	5	5	4	77.77777778	100	28.57142857	55.55555556	22.22222222	26.98412698
B6P2	4	8	3	1	77.77777778	100	57.14285714	88.88888889	22.22222222	31.74603175
B6P3	1	7	8	2	100	100	11.11111111	77.77777778	0	66.66666667
B6P4	7	9	2	0	100	100	77.77777778	100	0	22.22222222
B6P5	5	9	3	0	88.88888889	100	62.5	100	11.11111111	37.5
B6P6	4	9	5	0	100	100	44.44444444	100	0	55.55555556
B6P7	6	8	3	1	100	100	66.66666667	88.88888889	0	22.22222222
B6P8	4	9	2	0	66.66666667	100	66.66666667	100	33.33333333	33.33333333
B6P9	6	9	3	0	100	100	66.66666667	100	0	33.33333333
B6P10	5	9	4	0	100	100	55.55555556	100	0	44.44444444
B6P11	4	9	5	0	100	100	44.44444444	100	0	55.55555556
B6P12	6	9	2	0	88.88888889	100	75	100	11.11111111	25
B6P13	3	8	6	1	100	100	33.33333333	88.88888889	0	55.55555556
Average	**4.58**	**8.58**	**3.83**	**0.42**	**93.52**	**100**	**55.11**	**95.37**	**6.48**	**40.26**

	2D Simple			
	Difficulty	Overwhelmed?	Method	Feelings
B6P1	Easy	No	Good	Happy
B6P2	Easy	No	Good	Neutral
B6P3	Easy	No	Good	Neutral
B6P4				
B6P5	Easy	No	Good	Neutral
B6P6	Difficult	Yes	Poor	Neutral
B6P7	Easy	No	Good	Neutral
B6P8	Easy	Yes	Good	Happy
B6P9	Very Easy	No	Good	Happy
B6P10	Easy	No	Good	Happy
B6P11	Very Easy	No	Good	Neutral
B6P12				
B6P13				

	3D Simple					
	Difficulty	Overwhelmed?	Method	Feelings	Easier Extension	Preference
B6P1	Easy	No	Good	Happy	3D	3D
B6P2	Very Easy	No	Very Good	Happy	3D	3D
B6P3	Easy	No	Good	Neutral	3D	3D
B6P4	Easy	No	Good	Neutral	2D	2D
B6P5	Easy	No	Good	Happy	3D	3D
B6P6	Very Easy	No	Good	Happy	3D	3D
B6P7	Very Easy	No	Good	Happy	3D	3D
B6P8	Very Easy	No	Very Good	Happy	3D	3D
B6P9	Very Easy	No	Good	Happy	3D	3D
B6P10	Very Easy	No	Good	Happy	2D	2D
B6P11	Very Easy	No	Good	Neutral	2D	3D
B6P12						
B6P13						

	2D Complex			
	Difficulty	Overwhelmed?	Method	Feelings
B6P1	Very Difficult	Yes	Very Poor	Frustrated
B6P2	Very Difficult	Yes	Poor	Frustrated
B6P3	Very Difficult	No	Poor	Neutral
B6P4	Difficult	Yes	Good	Neutral
B6P5	Difficult	Yes	Good	Neutral
B6P6	Difficult	Yes	Poor	Frustrated
B6P7	Very Difficult	Yes	Poor	Frustrated
B6P8	Difficult	Yes	Poor	Neutral
B6P9	Easy	Yes	Poor	Neutral
B6P10	Difficult	Yes	Good	Happy
B6P11	Difficult	Yes	Good	Frustrated
B6P12	Very Difficult	Yes	Poor	Frustrated
B6P13				

	3D Complex					
	Difficulty	Overwhelmed?	Method	Feelings	Easier Extension	Preference
B6P1	Difficult	No	Good	Happy	3D	3D
B6P2	Very Easy	No	Very Good	Happy	3D	3D
B6P3	Easy	No	Good	Happy	3D	3D
B6P4	Easy	No	Good	Neutral	3D	3D
B6P5	Easy	Yes	Good	Neutral	3D	3D
B6P6	Easy	Yes	Good	Happy	3D	3D
B6P7	Easy	No	Good	Neutral	3D	3D
B6P8	Very Easy	No	Very Good	Happy	3D	3D
B6P9	Easy	No	Good	Happy	3D	3D
B6P10	Easy	No	Very Good	Happy	3D	3D
B6P11	Difficult	No	Good	Happy	2D	3D
B6P12	Easy	No	Good	Happy	3D	3D
B6P13	Easy	Yes	Good	Happy	3D	3D

Product Design Level 4 Students:

			Education				
	Gender	Age Range	Current	Cyber Security	Art & Graphics	3D Application Usage	Learning Difficulty
PD4P1	Female	18-24	Undergrad #1	GCSE	A Level	1 a week	Dyslexia
PD4P2	Female	18-24	Undergrad #1	A Level	A Level	1 a week	None
PD4P3	Female	18-24	Undergrad #1	None	A Level	1 a week	None
PD4P4	Female	18-24	Undergrad #1	None	A Level	Everyday	
PD4P5	Female	18-24	Undergrad #1	None	A Level	Everyday	None
PD4P6	Male	18-24	Undergrad #1	None	GCSE	Everyday	None
PD4P7	Male	18-24	Undergrad #1	A Level	A Level	Everyday	None
PD4P8	Male	18-24	Undergrad #1	A Level	A Level	1 a week	None
PD4P9	Male	18-24	Undergrad #1	None	GCSE	Never	None
PD4P10	Male	18-24	Undergrad #1	None	A Level	Everyday	None
PD4P11	Male	18-24	Undergrad #1	A Level	A Level	Everyday	None
PD4P12	Male	18-24	Undergrad #1	None	A Level	Everyday	None
PD4P13	Male	18-24	Undergrad #1	A Level	A Level	Everyday	None
PD4P14	Female	18-24	Undergrad #1	None	A Level	Everyday	None
PD4P15	Female	18-24	Undergrad #1	None	GCSE	1 a week	None
PD4P16	Female	18-24	Undergrad #1	A Level	A Level	Everyday	None
PD4P17	Female	18-24	Undergrad #1	A Level	A Level	Everyday	None
PD4P18	Male	18-24	Undergrad #1	GCSE	GCSE	Everyday	None

	Correct (out of 9)		Wrong (out of 9)		Completion (%)		Accuracy %		Improvement	
	2D Simple	3D Simple	2D Simple	3D Simple	2D Simple	3D Simple	2D Simple	3D Simple	Completion	Accuracy
PD4P1	6	6	3	3	100	100	66.66666667	66.66666667	0	0
PD4P2	6	9	3	0	100	100	66.66666667	100	0	33.33333333
PD4P3	7	9	2	0	100	100	77.77777778	100	0	22.22222222
PD4P4	8	8	1	1	100	100	88.88888889	88.88888889	0	0
PD4P5	9	8	0	1	100	100	100	88.88888889	0	-11.11111111
PD4P6	2	9	0	0	22.22222222	100	100	100	77.77777778	0
PD4P7	8	9	1	0	100	100	88.88888889	100	0	11.11111111
PD4P8	7	9	2	0	100	100	77.77777778	100	0	22.22222222
PD4P9	9	9	0	0	100	100	100	100	0	0
PD4P10	9	9	0	0	100	100	100	100	0	0
PD4P11	7	9	2	0	100	100	77.77777778	100	0	22.22222222
PD4P12	6	7	1	2	77.77777778	100	85.71428571	77.77777778	22.22222222	-7.936507937
PD4P13	6	9	0	0	66.66666667	100	100	100	33.33333333	0
PD4P14	5	9	4	0	100	100	55.55555556	100	0	44.44444444
PD4P15	8	9	1	0	100	100	88.88888889	100	0	11.11111111
PD4P16	6	9	0	0	66.66666667	100	100	100	33.33333333	0
PD4P17	5	8	0	1	55.55555556	100	100	88.88888889	44.44444444	-11.11111111
PD4P18	7	7	2	2	100	100	77.77777778	77.77777778	0	0
Average	**6.72**	**8.44**	**1.22**	**0.56**	**88.27**	**100**	**86.24**	**93.83**	**11.73**	**7.58**

	Correct (out of 9)		Wrong (out of 9)		Completion (%)		Accuracy %		Improvement	
	2D Complex	3D Complex	2D Complex	3D Complex	2D Complex	3D Complex	2D Complex	3D Complex	Completion	Accuracy
PD4P1	2	6	7	3	100	100	22.22222222	66.66666667	0	44.44444444
PD4P2	2	9	5	0	77.77777778	100	28.57142857	100	22.22222222	71.42857143
PD4P3	4	9	3	0	77.77777778	100	57.14285714	100	22.22222222	42.85714286
PD4P4	8	8	1	1	100	100	88.88888889	88.88888889	0	0
PD4P5	6	9	3	0	100	100	66.66666667	100	0	33.33333333
PD4P6	5	9	4	0	100	100	55.55555556	100	0	44.44444444
PD4P7	5	7	2	2	77.77777778	100	71.42857143	77.77777778	22.22222222	6.349206349
PD4P8	7	9	2	0	100	100	77.77777778	100	0	22.22222222
PD4P9	4	9	2	0	66.66666667	100	66.66666667	100	33.33333333	33.33333333
PD4P10	3	9	6	0	100	100	33.33333333	100	0	66.66666667
PD4P11	3	9	3	0	66.66666667	100	50	100	33.33333333	50
PD4P12	5	9	4	0	100	100	55.55555556	100	0	44.44444444
PD4P13	1	7	4	2	55.55555556	100	20	77.77777778	44.44444444	57.77777778
PD4P14	4	9	5	0	100	100	44.44444444	100	0	55.55555556
PD4P15	5	9	1	0	66.66666667	100	83.33333333	100	33.33333333	16.66666667
PD4P16	6	9	2	0	88.88888889	100	75	100	11.11111111	25
PD4P17	4	9	4	0	88.88888889	100	50	100	11.11111111	50
PD4P18	0	5	9	4	100	100	0	55.55555556	0	55.55555556
Average	**4.11**	**8.33**	**3.72**	**0.67**	**87.04**	**100**	**52.59**	**92.59**	**12.96**	**40**

	2D Simple					3D Simple					
	Difficulty	Overwhelmed?	Method	Feelings		Difficulty	Overwhelmed?	Method	Feelings	Easier Extension	Preference
PD4P1	Very Easy	Yes	Good	Happy	PD4P1	Easy	No	Very Good	Happy	3D	3D
PD4P2	Easy	No	Good	Neutral	PD4P2	Easy	No	Good	Neutral	3D	3D
PD4P3	Easy	Yes	Poor	Frustrated	PD4P3	Easy	No	Poor	Neutral	2D	2D
PD4P4	Very Easy	No	Good	Happy	PD4P4	Very Easy	No	Very Good	Happy	3D	3D
PD4P5	Easy	No	Good	Happy	PD4P5	Easy	No	Good	Happy	3D	3D
PD4P6	Easy	Yes	Good	Neutral	PD4P6	Very Easy	No	Good	Neutral	3D	3D
PD4P7	Very Easy	No	Good	Happy	PD4P7	Easy	No	Good	Happy	2D	2D
PD4P8	Easy	No	Good	Neutral	PD4P8	Very Easy	No	Very Good	Happy	3D	3D
PD4P9	Easy	No	Good	Happy	PD4P9	Very Easy	No	Good	Happy	3D	3D
PD4P10	Easy	No	Good	Happy	PD4P10	Easy	No	Good	Happy	2D	3D
PD4P11	Easy	No	Very Good	Happy	PD4P11	Easy	No	Very Good	Happy	3D	3D
PD4P12	Easy	No	Good	Neutral	PD4P12	Very Easy	No	Good	Neutral	3D	3D
PD4P13	Easy	No	Good	Neutral	PD4P13	Easy	No	Good		3D	3D
PD4P14	Easy	No	Good	Happy	PD4P14	Easy	No	Good	Happy	3D	3D
PD4P15	Very Easy	No	Poor	Happy	PD4P15	Very Easy	No	Good	Happy	2D	2D
PD4P16					PD4P16	Easy	No	Good	Neutral	3D	3D
PD4P17	Easy	No	Good	Neutral	PD4P17	Easy	No	Good	Neutral	3D	3D
PD4P18	Easy	No	Good	Neutral	PD4P18	Easy	No	Good	Neutral		

	2D Complex					3D Complex					
	Difficulty	Overwhelmed?	Method	Feelings		Difficulty	Overwhelmed?	Method	Feelings	Easier Extension	Preference
PD4P1	Very Difficult	Yes	Very Poor	Frustrated	PD4P1	Difficult	No	Poor	Frustrated	3D	3D
PD4P2	Very Difficult	Yes	Good	Frustrated	PD4P2	Easy	No	Good	Happy	3D	3D
PD4P3	Difficult	Yes	Very Poor	Frustrated	PD4P3	Easy	No	Good	Neutral	3D	3D
PD4P4	Difficult	Yes	Good	Happy	PD4P4	Easy	No	Good	Happy	3D	3D
PD4P5	Difficult	Yes	Good	Neutral	PD4P5	Easy	Yes	Good	Neutral	3D	3D
PD4P6	Easy	Yes	Good	Neutral	PD4P6	Very Easy	No	Very Good	Happy	3D	3D
PD4P7	Easy	No	Good	Happy	PD4P7	Very Easy	No	Good	Happy	3D	3D
PD4P8	Difficult	Yes	Very Good	Neutral	PD4P8	Very Easy	No	Very Good	Happy	3D	3D
PD4P9	Difficult	Yes	Very Poor	Frustrated	PD4P9	Easy	No	Good	Happy	3D	3D
PD4P10	Difficult	No	Poor	Happy	PD4P10	Difficult	No	Poor	Happy	3D	3D
PD4P11	Difficult	Yes	Poor	Frustrated	PD4P11	Easy	No	Good	Happy	3D	3D
PD4P12	Difficult	Yes	Good	Neutral	PD4P12	Easy	No	Good	Neutral	3D	3D
PD4P13	Difficult	No	Very Poor	Frustrated	PD4P13						
PD4P14	Easy	No	Good	Happy	PD4P14	Easy	No	Good	Happy		
PD4P15	Difficult	Yes	Good	Frustrated	PD4P15	Easy	No	Very Good	Happy	2D	2D
PD4P16	Difficult	Yes	Poor	Frustrated	PD4P16	Very Easy	No	Very Good	Happy		
PD4P17	Difficult	Yes	Poor	Frustrated	PD4P17	Easy	No	Good	Happy	3D	3D
PD4P18	Easy	No	Good	Happy	PD4P18	Easy	No	Good	Neutral		

Product Design Level 5 Students:

			Education				
	Gender	Age Range	Current	Cyber Security	Art & Graphics	3D Application Usage	Learning Difficulty
PD5P1	Male	18-24	Undergrad #2	GCSE			
PD5P2	Male	18-24	Undergrad #2	GCSE	Undergrad #1	Everyday	None
PD5P3	Male	18-24	Undergrad #2		None	Everyday	None
PD5P4	Male	18-24	Undergrad #2	None	GCSE	1 a week	None
PD5P5	Male	18-24	Undergrad #2	None	None	Everyday	None
PD5P6	Male	18-24	Undergrad #2	None	Undergrad #2	1 a week	None
PD5P7	Male	18-24	Undergrad #2	None	None	Everyday	None
PD5P8	Male	18-24	Undergrad #2	None	Undergrad #1	Everyday	None
PD5P9	Male	18-24	Undergrad #2	GCSE	A Level	Everyday	None
PD5P10	Male	18-24	Undergrad #2	GCSE	Undergrad #1	Everyday	Dyslexia
PD5P11	Male	18-24	Undergrad #2	A Level	A Level	Everyday	None
PD5P12	Male	18-24	Undergrad #2	GCSE	A Level	Everyday	

	Correct (out of 9)		Wrong (out of 9)		Completion (%)		Accuracy %		Improvement	
	2D Simple	3D Simple	2D Simple	3D Simple	2D Simple	3D Simple	2D Simple	3D Simple	Completion	Accuracy
PD5P1	7	9	2	0	100	100	77.77777778	100	0	22.22222222
PD5P2	8	9	1	0	100	100	88.88888889	100	0	11.11111111
PD5P3	8	8	0	1	88.88888889	100	100	88.88888889	11.11111111	-11.11111111
PD5P4	8	7	1	2	100	100	88.88888889	77.77777778	0	-11.11111111
PD5P5	2	9	0	0	22.22222222	100	100	100	77.77777778	0
PD5P6	8	9	1	0	100	100	88.88888889	100	0	11.11111111
PD5P7	6	6	3	1	100	77.77777778	66.66666667	85.71428571	-22.22222222	19.04761905
PD5P8	8	9	1	0	100	100	88.88888889	100	0	11.11111111
PD5P9	6	9	1	0	77.77777778	100	85.71428571	100	22.22222222	14.28571429
PD5P10	7	9	2	0	100	100	77.77777778	100	0	22.22222222
PD5P11	0	9	4	0	44.44444444	100	0	100	55.55555556	100
PD5P12	9	9	0	0	100	100	100	100	0	0
Average	**6.42**	**8.5**	**1.33**	**0.33**	**86.11**	**98.15**	**80.29**	**96.03**	**12.04**	**15.74**

	Correct (out of 9)		Wrong (out of 9)		Completion (%)		Accuracy %		Improvement	
	2D Complex	3D Complex	2D Complex	3D Complex	2D Complex	3D Complex	2D Complex	3D Complex	Completion	Accuracy
PD5P1	4	5	5	4	100	100	44.44444444	55.55555556	0	11.11111111
PD5P2	3	8	6	1	100	100	33.33333333	88.88888889	0	55.55555556
PD5P3	2	8	4	1	66.66666667	100	33.33333333	88.88888889	33.33333333	55.55555556
PD5P4	4	7	3	2	77.77777778	100	57.14285714	77.77777778	22.22222222	20.63492063
PD5P5	4	5	3	2	77.77777778	77.77777778	57.14285714	71.42857143	0	14.28571429
PD5P6	7	7	2	2	100	100	77.77777778	77.77777778	0	0
PD5P7	5	9	4	0	100	100	55.55555556	100	0	44.44444444
PD5P8	2	4	4	3	66.66666667	77.77777778	33.33333333	57.14285714	11.11111111	23.80952381
PD5P9	3	7	3	2	66.66666667	100	50	77.77777778	33.33333333	27.77777778
PD5P10	2	6	3	2	55.55555556	88.88888889	40	75	33.33333333	35
PD5P11	4	8	5	1	100	100	44.44444444	88.88888889	0	44.44444444
PD5P12	4	9	3	0	77.77777778	100	57.14285714	100	22.22222222	42.85714286
Average	**3.67**	**6.92**	**3.75**	**1.67**	**82.41**	**95.37**	**48.64**	**79.93**	**12.96**	**31.29**

	2D Simple					3D Simple					
	Difficulty	Overwhelmed?	Method	Feelings		Difficulty	Overwhelmed?	Method	Feelings	Easier Extension	Preference
PD5P1	Easy	No	Poor	Neutral	PD5P1						
PD5P2	Very Easy	No	Good	Neutral	PD5P2	Easy	Yes	Poor	Neutral	2D	2D
PD5P3	Easy	No	Good	Neutral	PD5P3	Very Easy	No	Very Good	Happy	3D	3D
PD5P4	Easy	No	Good	Neutral	PD5P4	Easy	No	Good	Neutral	2D	2D
PD5P5	Very Easy	No	Good	Neutral	PD5P5	Very Easy	No	Very Good	Happy	3D	3D
PD5P6	Very Easy	No	Good	Neutral	PD5P6	Easy	No	Poor	Neutral	2D	2D
PD5P7	Easy	No	Very Good	Happy	PD5P7	Easy	No	Good	Happy	2D	2D
PD5P8	Easy	No	Good	Neutral	PD5P8	Difficult	Yes	Poor	Neutral	2D	2D
PD5P9	Very Easy	No	Good	Happy	PD5P9	Easy	No	Good	Happy	2D	2D
PD5P10	Easy	No	Good	Happy	PD5P10	Easy	No	Good	Happy	2D	3D
PD5P11	Difficult	Yes	Very Good	Neutral	PD5P11	Easy	Yes	Good	Happy	3D	3D
PD5P12	Easy	No	Good	Neutral	PD5P12	Very Easy	No	Good	Neutral	3D	3D

	2D Complex					3D Complex					
	Difficulty	Overwhelmed?	Method	Feelings		Difficulty	Overwhelmed?	Method	Feelings	Easier Extension	Preference
PD5P1					PD5P1	Very Easy	No		Neutral	3D	3D
PD5P2	Difficult	Yes	Very Poor	Frustrated	PD5P2	Difficult	Yes	Poor	Neutral	3D	3D
PD5P3	Very Difficult	Yes	Good	Frustrated	PD5P3	Easy	No	Good	Happy	3D	3D
PD5P4	Difficult	Yes	Poor	Frustrated	PD5P4	Easy	No	Good	Happy	3D	3D
PD5P5	Difficult	Yes	Poor	Frustrated	PD5P5	Difficult	Yes	Very Poor	Frustrated	2D	2D
PD5P6	Very Difficult	Yes	Very Poor	Neutral	PD5P6	Easy	No	Very Good	Neutral	3D	3D
PD5P7	Easy	No	Good	Happy	PD5P7	Easy	No	Good	Happy	3D	3D
PD5P8	Very Difficult	Yes	Very Poor	Frustrated	PD5P8	Easy	No	Good	Happy	3D	3D
PD5P9	Very Difficult	Yes	Poor	Frustrated	PD5P9	Easy	No	Good	Happy	2D	2D
PD5P10	Difficult	No	Poor	Frustrated	PD5P10	Easy	No	Good	Neutral	3D	3D
PD5P11	Difficult	Yes	Good	Neutral	PD5P11	Easy	No	Good	Happy	3D	3D
PD5P12	Difficult	Yes	Poor	Neutral	PD5P12	Easy	No	Good	Neutral	3D	3D

Media Production Students:

			Education				
	Gender	Age Range	Current	Cyber Security	Art & Graphics	3D Application Usage	Learning Difficulty
MP6P1	Female	18-24	Undergrad #3	None	A Level	Never	None
MP6P2	Male	18-24	Undergrad #3	A Level	Undergrad #2	1 a week	None
MP6P3	Female	18-24	Undergrad #3	None	Undergrad #3	Never	None
MP6P4	Female	18-24	Undergrad #3	None	Undergrad #3	Never	None
MP6P5	Male	18-24	Undergrad #3	None	None	1 a week	None
MP6P6	Male	18-24	Undergrad #3	None	None	1 a week	None
MP6P7	Male	18-24	Undergrad #3	None	Undergrad #2	1 a week	Dyslexia
MP6P8	Male	18-24	Undergrad #3	GCSE	Undergrad #2	Everyday	None
MP6P9	Male	18-24	Undergrad #3	None	Undergrad #2	Everyday	None
MP6P10	Female	18-24	Undergrad #3	None	Undergrad #3	Never	None
MP6P11	Female	18-24	Undergrad #3	None	Undergrad #3	1 a week	None

	Correct (out of 9)		Wrong (out of 9)		Completion (%)		Accuracy %		Improvement	
	2D Simple	3D Simple	2D Simple	3D Simple	2D Simple	3D Simple	2D Simple	3D Simple	Completion	Accuracy
MP6P1	4	4	1	2	55.55555556	66.66666667	80	66.66666667	11.11111111	-13.33333333
MP6P2	7	7	0	2	77.77777778	100	100	77.77777778	22.22222222	-22.22222222
MP6P3	7	8	0	1	77.77777778	100	100	88.88888889	22.22222222	-11.11111111
MP6P4	9	9	0	0	100	100	100	100	0	0
MP6P5	5	9	0	0	55.55555556	100	100	100	44.44444444	0
MP6P6	9	9	0	0	100	100	100	100	0	0
MP6P7	8	7	1	2	100	100	88.88888889	77.77777778	0	-11.11111111
MP6P8	9	7	0	0	100	77.77777778	100	100	-22.22222222	0
MP6P9	7	9	2	0	100	100	77.77777778	100	0	22.22222222
MP6P10	7	9	2	0	100	100	77.77777778	100	0	22.22222222
MP6P11	9	8	0	1	100	100	100	88.88888889	0	-11.11111111
Average	**7.36**	**7.82**	**0.55**	**0.73**	**87.88**	**94.95**	**93.13**	**90.91**	**7.07**	**-2.22**

	Correct (out of 9)		Wrong (out of 9)		Completion (%)		Accuracy %		Improvement	
	2D Complex	3D Complex	2D Complex	3D Complex	2D Complex	3D Complex	2D Complex	3D Complex	Completion	Accuracy
MP6P1	2	2	6	4	88.88888889	66.66666667	25	33.33333333	-22.22222222	8.333333333
MP6P2	4	7	4	2	88.88888889	100	50	77.77777778	11.11111111	27.77777778
MP6P3	4	8	3	1	77.77777778	100	57.14285714	88.88888889	22.22222222	31.74603175
MP6P4	6	7	0	2	66.66666667	100	100	77.77777778	33.33333333	-22.22222222
MP6P5	4	6	2	1	66.66666667	77.77777778	66.66666667	85.71428571	11.11111111	19.04761905
MP6P6	8	9	1	0	100	100	88.88888889	100	0	11.11111111
MP6P7	0	5	5	4	55.55555556	100	0	55.55555556	44.44444444	55.55555556
MP6P8	4	9	2	0	66.66666667	100	66.66666667	100	33.33333333	33.33333333
MP6P9	8	7	1	2	100	100	88.88888889	77.77777778	0	-11.11111111
MP6P10	7	9	2	0	100	100	77.77777778	100	0	22.22222222
MP6P11	9	9	0	0	100	100	100	100	0	0
Average	**5.09**	**7.09**	**2.36**	**1.45**	**82.83**	**94.95**	**65.55**	**81.53**	**12.12**	**15.98**

	2D Simple			
	Difficulty	Overwhelmed?	Method	Feelings
MP6P1	Easy	Yes	Good	Neutral
MP6P2	Easy	No	Good	Happy
MP6P3	Easy	No	Good	Happy
MP6P4	Very Easy	No	Good	Happy
MP6P5	Easy	Yes	Good	Neutral
MP6P6	Very Easy	No	Good	Happy
hearing/red&green/MP6P7	Difficult	Yes	Very Good	Happy
MP6P8	Very Easy	No	Good	Happy
MP6P9	Easy	No	Poor	Neutral
MP6P10	Very Easy	No	Very Good	Happy
MP6P11	Very Easy	No	Very Good	Happy

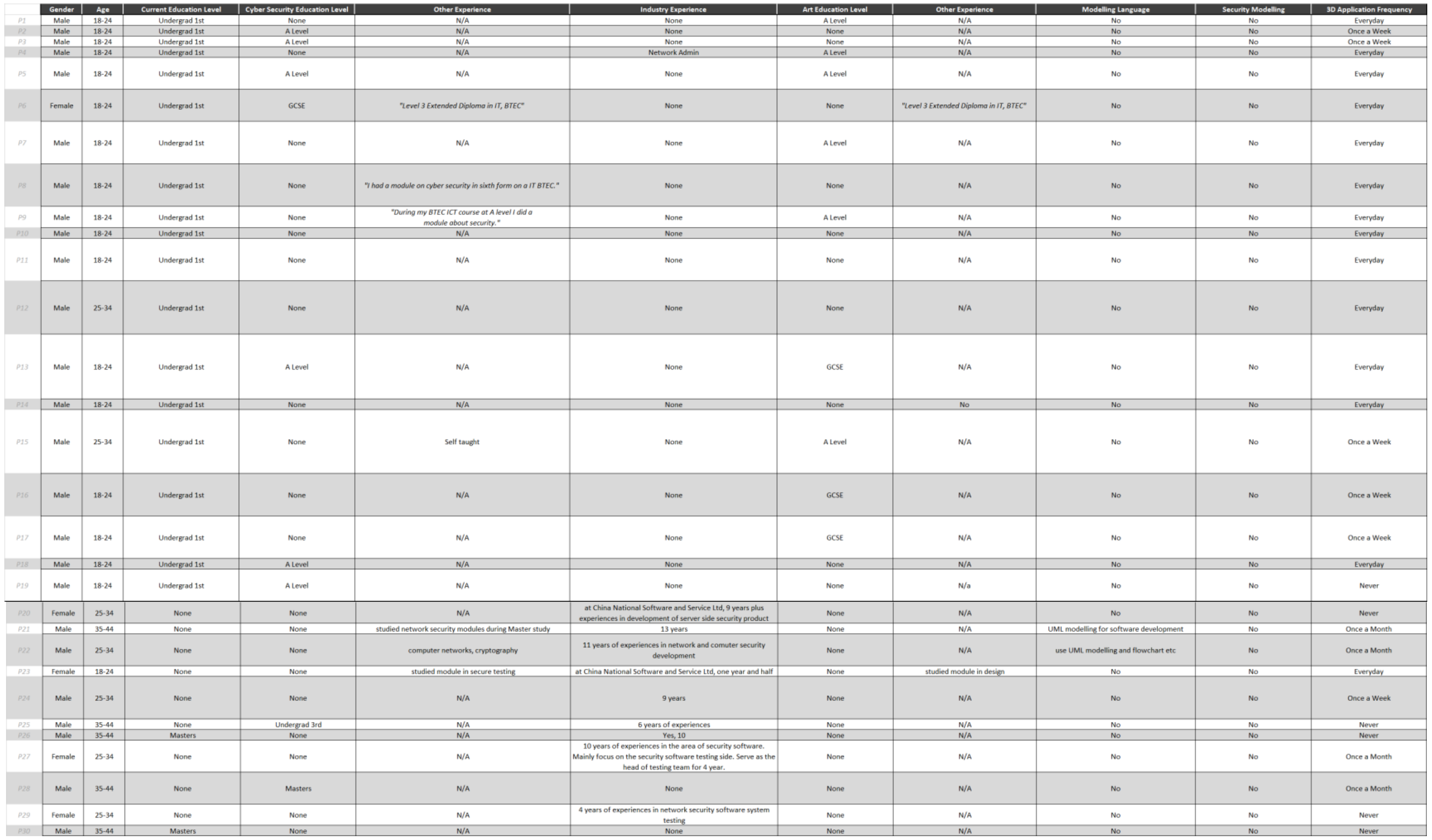
	3D Simple					
	Difficulty	Overwhelmed?	Method	Feelings	Easier Extension	Preference
MP6P1	Easy	Yes	Poor	Frustrated	2D	2D
MP6P2	Easy	No	Good	Happy	3D	2D
MP6P3	Easy	No	Good	Neutral	2D	3D
MP6P4	Easy	Yes	Poor	Happy	2D	2D
MP6P5	Very Easy	No	Good	Neutral	3D	3D
MP6P6	Easy	No	Poor	Neutral	2D	2D
MP6P7	Very Difficult	Yes	Poor	Neutral	2D	2D
MP6P8	Easy	No	Good	Neutral	2D	2D
MP6P9	Easy	Yes	Poor	Neutral	2D	3D
MP6P10	Very Easy	No	Good	Happy	2D	2D
MP6P11	Very Easy	Yes	Very Good	Happy	2D	2D

	2D Complex			
	Difficulty	Overwhelmed?	Method	Feelings
MP6P1	Difficult	Yes	Poor	Neutral
MP6P2	Difficult	Yes	Good	Happy
MP6P3	Difficult	Yes	Good	Neutral
MP6P4	Very Difficult	Yes	Good	Frustrated
MP6P5	Difficult	Yes	Poor	Frustrated
MP6P6	Difficult	Yes	Very Poor	Neutral
MP6P7	Very Difficult	Yes	Very Poor	Frustrated
MP6P8	Difficult	Yes	Good	Neutral
MP6P9	Very Difficult	Yes	Very Poor	Frustrated
MP6P10	Easy	Yes	Good	Neutral
MP6P11	Difficult	Yes	Poor	Frustrated

	3D Complex					
	Difficulty	Overwhelmed?	Method	Feelings	Easier Extension	Preference
MP6P1	Easy	Yes	Good	Neutral	2D	2D
MP6P2	Very Easy	No	Very Good	Happy	3D	3D
MP6P3	Easy	No	Good	Happy	3D	3D
MP6P4	Very Easy	No	Good	Happy	3D	3D
MP6P5	Difficult	Yes	Good	Frustrated	3D	3D
MP6P6	Easy	No	Good	Happy	3D	3D
MP6P7	Difficult	Yes	Poor	Neutral	3D	3D
MP6P8	Very Easy	No	Very Good	Happy	3D	3D
MP6P9	Difficult	Yes	Good	Frustrated	3D	3D
MP6P10	Easy	No	Very Good	Happy	3D	3D
MP6P11	Easy	Yes	Good	Happy	3D	3D

BPMN 3D Experiment Results

Background Questionnaire:



	Gender	Age	Current Education Level	Cyber Security Education Level	Other Experience	Industry Experience	Art Education Level	Other Experience	Modelling Language	Security Modelling	3D Application Frequency
P1	Male	18-24	Undergrad 1st	None	N/A	None	A Level	N/A	No	No	Everyday
P2	Male	18-24	Undergrad 1st	A Level	N/A	None	None	N/A	No	No	Once a Week
P3	Male	18-24	Undergrad 1st	A Level	N/A	None	None	N/A	No	No	Once a Week
P4	Male	18-24	Undergrad 1st	None	N/A	Network Admin	A Level	N/A	No	No	Everyday
P5	Male	18-24	Undergrad 1st	A Level	N/A	None	A Level	N/A	No	No	Everyday
P6	Female	18-24	Undergrad 1st	GCSE	*"Level 3 Extended Diploma in IT, BTEC"*	None	None	*"Level 3 Extended Diploma in IT, BTEC"*	No	No	Everyday
P7	Male	18-24	Undergrad 1st	None	N/A	None	A Level	N/A	No	No	Everyday
P8	Male	18-24	Undergrad 1st	None	*"I had a module on cyber security in sixth form on a IT BTEC."*	None	None	N/A	No	No	Everyday
P9	Male	18-24	Undergrad 1st	None	*"During my BTEC ICT course at A level I did a module about security."*	None	A Level	N/A	No	No	Everyday
P10	Male	18-24	Undergrad 1st	None	N/A	None	None	N/A	No	No	Everyday
P11	Male	18-24	Undergrad 1st	None	N/A	None	None	N/A	No	No	Everyday
P12	Male	25-34	Undergrad 1st	None	N/A	None	None	N/A	No	No	Everyday
P13	Male	18-24	Undergrad 1st	A Level	N/A	None	GCSE	N/A	No	No	Everyday
P14	Male	18-24	Undergrad 1st	None	N/A	None	None	No	No	No	Everyday
P15	Male	25-34	Undergrad 1st	None	Self taught	None	A Level	N/A	No	No	Once a Week
P16	Male	18-24	Undergrad 1st	None	N/A	None	GCSE	N/A	No	No	Once a Week
P17	Male	18-24	Undergrad 1st	None	N/A	None	GCSE	N/A	No	No	Once a Week
P18	Male	18-24	Undergrad 1st	A Level	N/A	None	None	N/A	No	No	Everyday
P19	Male	18-24	Undergrad 1st	A Level	N/A	None	None	N/a	No	No	Never
P20	Female	25-34	None	None	N/A	at China National Software and Service Ltd, 9 years plus experiences in development of server side security product	None	N/A	No	No	Never
P21	Male	35-44	None	None	studied network security modules during Master study	13 years	None	N/A	UML modelling for software development	No	Once a Month
P22	Male	25-34	None	None	computer networks, cryptography	11 years of experiences in network and comuter security development	None	N/A	use UML modelling and flowchart etc	No	Once a Month
P23	Female	18-24	None	None	studied module in secure testing	at China National Software and Service Ltd, one year and half	None	studied module in design	No	No	Everyday
P24	Male	25-34	None	None	N/A	9 years	None	N/A	No	No	Once a Week
P25	Male	35-44	None	Undergrad 3rd	N/A	6 years of experiences	None	N/A	No	No	Never
P26	Male	35-44	Masters	None	N/A	Yes, 10	None	N/A	No	No	Never
P27	Female	25-34	None	None	N/A	10 years of experiences in the area of security software. Mainly focus on the security software testing side. Serve as the head of testing team for 4 year.	None	N/A	No	No	Once a Month
P28	Male	35-44	None	Masters	N/A	None	None	N/A	No	No	Once a Month
P29	Female	25-34	None	None	N/A	4 years of experiences in network security software system testing	None	N/A	No	No	Never
P30	Male	35-44	Masters	None	N/A	None	None	N/A	No	No	Never

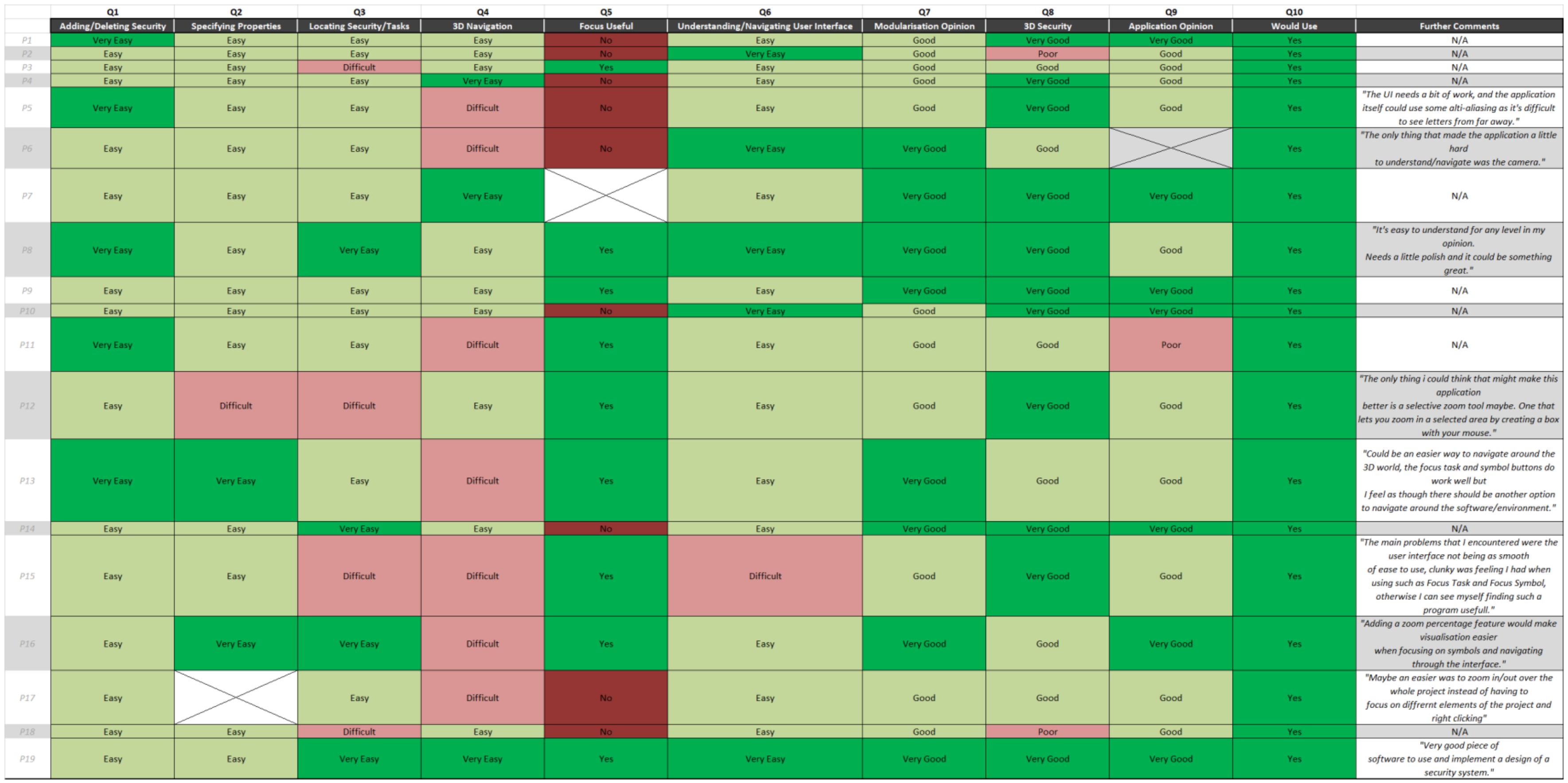
Undergraduate results:

	Start Time	End Time	Total Time	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Grade	Avg. Time per Security (s)	
P1	11:07	11:17	00:10	0	1	0.5	0	1	0	0	35.71%	66.67 s	
P2	11:11	11:22	00:11	1	1	1	1	0.5	1	1	92.86%	73.33 s	
P3	11:06	11:21	00:15	1	1	1	1	1	1	1	100.00%	100.00 s	
P4				1	1	1	1	1	1	1	100.00%	0.00 s	Didn't record time
P5	11:15	11:24	00:09	0	0	1	0	1	0	0	*Anomaly*	*Anomaly*	Didn't include any low level detail. Possibly didn't realise they had to. Set to anomaly
P6	11:09	11:24	00:15	1	1	1	1	1	1	1	100.00%	100.00 s	
P7	11:21	11:27	00:06	1	0	0.5	1	0.5	1	1	71.43%	40.00 s	
P8	11:18	11:25	00:07										Didn't save diagram
P9	11:05	11:25	00:20	1	1	1	1	1	1	1	100.00%	133.33 s	
P10	11:16	11:26	00:10	0	1	1	1	1	1	1	85.71%	66.67 s	
P11	11:17	11:27	00:10	1	1	0.5	1	0.5	1	1	85.71%	66.67 s	
P12	11:11	11:20	00:09	1	1	1	1	0.5	1	1	92.86%	60.00 s	
P13	11:27	11:35	00:08	0	0	1	1	1	1	1	71.43%	53.33 s	
P14	11:18	11:42	00:24	1	1	1	0	1	1	1	85.71%	160.00 s	
P15	11:28	11:35	00:07	0	1	1	0	1	1	1	71.43%	46.67 s	
P16	11:27	11:35	00:08	1	1	1	1	1	1	1	100.00%	53.33 s	
P17	11:24	11:37	00:13	1	1	0.5	1	0	0	0	50.00%	86.67 s	
P18	11:31	11:44	00:13									86.67 s	Didn't save diagram
P19				1	1	1	1	1	0	0	71.43%		Didn't record time

Expert results:

P20	10:47	11:01	00:14									93.33 s	— Didn't save diagram
P21	10:34	10:50	00:16	0	1	1	1	1	1	1	85.71%	106.67 s	
P22	10:43	11:03	00:20	1	1	1	1	0.5	0	1	78.57%	133.33 s	
P23	10:35	10:47	00:12	0	1	1	1	1	1	1	85.71%	80.00 s	
P24	10:30	10:39	00:09	0	0	1	1	1	0	0	42.86%	60.00 s	
P25	10:06	10:30	00:24	0	1	0.5	1	1	1	1	78.57%	160.00 s	
P26	10:11	10:28	00:17	1	0	1	1	1	1	1	85.71%	113.33 s	
P27	10:46	11:01	00:15	0	1	0.5	1	0.5	0	1	57.14%	100.00 s	
P28	10:26	10:35	00:09	0	1	0.5	0	0	0	0	21.43%	60.00 s	
P29	10:30	10:52	00:22	0	1	0.5	1	1	1	1	78.57%	146.67 s	
P30			00:00								0.00%	0.00 s	— Didn't save diagram

Undergrads Post Experiment Questionnaire:



	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	
	Adding/Deleting Security	Specifying Properties	Locating Security/Tasks	3D Navigation	Focus Useful	Understanding/Navigating User Interface	Modularisation Opinion	3D Security	Application Opinion	Would Use	Further Comments
P1	Very Easy	Easy	Easy	Easy	No	Easy	Good	Very Good	Very Good	Yes	N/A
P2	Easy	Easy	Easy	Easy	No	Very Easy	Good	Poor	Good	Yes	N/A
P3	Easy	Easy	Difficult	Easy	Yes	Easy	Good	Good	Good	Yes	N/A
P4	Easy	Easy	Easy	Very Easy	No	Easy	Good	Very Good	Good	Yes	N/A
P5	Very Easy	Easy	Easy	Difficult	No	Easy	Good	Very Good	Good	Yes	*"The UI needs a bit of work, and the application itself could use some alti-aliasing as it's difficult to see letters from far away."*
P6	Easy	Easy	Easy	Difficult	No	Very Easy	Very Good	Good		Yes	*"The only thing that made the application a little hard to understand/navigate was the camera."*
P7	Easy	Easy	Easy	Very Easy		Easy	Very Good	Very Good	Very Good	Yes	N/A
P8	Very Easy	Easy	Very Easy	Easy	Yes	Very Easy	Very Good	Very Good	Good	Yes	*"It's easy to understand for any level in my opinion. Needs a little polish and it could be something great."*
P9	Easy	Easy	Easy	Easy	Yes	Easy	Very Good	Very Good	Very Good	Yes	N/A
P10	Easy	Easy	Easy	Easy	No	Very Easy	Good	Very Good	Very Good	Yes	N/A
P11	Very Easy	Easy	Easy	Difficult	Yes	Easy	Good	Good	Poor	Yes	N/A
P12	Easy	Difficult	Difficult	Easy	Yes	Easy	Good	Very Good	Good	Yes	*"The only thing i could think that might make this application better is a selective zoom tool maybe. One that lets you zoom in a selected area by creating a box with your mouse."*
P13	Very Easy	Very Easy	Easy	Difficult	Yes	Easy	Very Good	Good	Good	Yes	*"Could be an easier way to navigate around the 3D world, the focus task and symbol buttons do work well but I feel as though there should be another option to navigate around the software/environment."*
P14	Easy	Easy	Very Easy	Easy	No	Easy	Very Good	Very Good	Very Good	Yes	N/A
P15	Easy	Easy	Difficult	Difficult	Yes	Difficult	Good	Very Good	Good	Yes	*"The main problems that I encountered were the user interface not being as smooth of ease to use, clunky was feeling I had when using such as Focus Task and Focus Symbol, otherwise I can see myself finding such a program usefull."*
P16	Easy	Very Easy	Very Easy	Difficult	Yes	Easy	Very Good	Good	Very Good	Yes	*"Adding a zoom percentage feature would make visualisation easier when focusing on symbols and navigating through the interface."*
P17	Easy		Easy	Difficult	No	Easy	Good	Good	Good	Yes	*"Maybe an easier was to zoom in/out over the whole project instead of having to focus on diffrernt elements of the project and right clicking"*
P18	Easy	Easy	Difficult	Easy	No	Easy	Good	Poor	Good	Yes	N/A
P19	Easy	Easy	Very Easy	Very Easy	Yes	Very Easy	Very Good	Very Good	Very Good	Yes	*"Very good piece of software to use and implement a design of a security system."*

Experts Post Experiment Questionnaire:

P20	Very Easy	Very Easy	Very Easy	Very Difficult	Yes	Very Difficult	Good	Good	Good	Yes	N/A
P21	Very Easy	Very Easy	Easy	Difficult	Yes	Easy	Very Good	Very Good	Good	Yes	N/A
P22	Easy	Easy	Easy	Difficult	No	Easy	Good	Good	Poor	Yes	Maybe the cameca can be fixed at 45 degree angle and use the scroll wheel of the mouse to zoom in and out
P23		Easy	Easy	Easy	Yes	Difficult	Very Good	Good	Good	Yes	N/A
P24	Very Easy	Very Easy	Easy	Easy	Yes	Very Easy	Very Good	Good	Good	Yes	suggest moving the camera to newly added element so that less effort and time is needed to locate the element.
P25		Difficult	Difficult	Difficult	Yes	Easy	Good	Good	Good	Yes	N/A
P26	Easy	Easy	Difficult	Easy	No	Easy	Good	Good	Poor	No	N/A
P27	Easy	Easy	Easy	Easy	Yes	Very Easy	Good	Very Good	Good	Yes	off button of the keyboard and mouse didn’t work properly.
P28	Very Easy	Easy	Easy	Easy	Yes	Easy	Good	Very Good	Good		the software should be made to automatically adjust to different screen size. Some inforamiton cannot be displayed properly
P29		Difficult	Easy	Very Easy	Yes	Easy	Good	Poor	Very Good	Yes	N/A
P30											