AN INTEGRATED INFORMATION MODEL FOR CONSTRUCTION MATERIALS MANAGEMENT

LEILA MERAGHNI

B.Arch, M.Sc.

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

Vol. 2

Model Documentation

April 1997

AN INTEGRATED INFORMATION MODEL FOR CONSTRUCTION MATERIALS MANAGEMENT

LEILA MERAGHNI

B.Arch, M.Sc.

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

Vol. 2

Model Documentation

April 1997

A: ICMM- An Information Model

A.1. Classes Dictionary And Attributes

Class: **RESOURCES**

Description: All resources consumed and used during construction.
Superclass: None
Subclasses: Materials, Employees and Equipment
Attributes:

Reference Number
Description
Source
Type
Allocated budget

Class: **MATERIALS**

Description: Building materials used during construction.
Superclass: Resources
Subclasses: None
Attributes:

Name
Quantity required
Quantity in storage
Quantity used
Quantity to be delivered
Quantity wasted
Waste allowance
Unit load
Unit of measurement
Packaging
Storage recommendations
Volume of unit load
Number of packs in a unit load
Vulnerability to weather
Health and safety recommendations
Specification
SfB classification
Substitute materials
Handling recommendations
Storage address
Storage duration
Order lead on time
Frequency of delivery
Cost
Taxes

Class: **EMPLOYEES**

Description: People working for a contractor by contributing in the construction work, at head office and site
Superclass: Resources
Subclasses: Site Employees, Administration Employees
Attributes:

Name
Reference number
Job title
Address
Contact number
Projects involved in
Insurance number
Overtime
Bonus
Days absent from work
Access authorisation level

Class: **SITE EMPLOYEES**

Description: People working on site and involved in the construction process.
Superclass: Employees
Subclasses: Gang, Gang Responsible, Craftsman
Attributes:

Skills
Speciality
Worksections involved in
Grade
List of subordinates

Class: **GANG**

Description: Group of site employees working on a particular assignment or worksection
Superclass: Employees
Subclasses: None
Attributes:

Assignment
Worksections involved in
List of names
List of skills required
Reference number
Gang responsible reference number

Class: **GANG RESPONSIBLE**

Description: A site employee responsible for a gang.
Superclass: Employees
Subclasses: None
Attributes:

Reference number
Gang list reference numbers
Worksections involved in

Class: **LABOUR**

Description: A site employee with no specific craft
Superclass: Employees
Subclasses: None
Attributes:
Speciality
Worksections involved in

Class: **CRAFTSMAN**

Description: A site employee who is an expert in a building craft
Superclass: Employees
Subclasses: None
Attributes:
Speciality
Worksections involved in

Class: **ADMINISTRATION EMPLOYEES**

Description: People working in contractor's head office or site office.
Superclass: Employees
Subclasses: None
Attributes:
Department

Class: **EQUIPMENT**

Description: All and facilities required for the construction process.
Superclass: Resources
Subclasses: Plant, Scaffolding
Attributes:
Reference number
Description
Source
Allocated budget
Contract number

Class: **PLANT**

Description: Heavy machinery used during construction.
Superclass: Equipment
Subclasses: None
Attributes:
Name
Owner
Driver code number
Registration number
Age

Power
Capacity
Unit load
Output/day
Degrees of movement
Condition on 1st day
Travelling speed
Radius of movement
Slewing speed
Hoisting speed
Engine power
Telescoping
List of functions
Swing min
Swing max
Height max
Width max
Length max
Type (static / moveable)
Skills needed to operate it
Parking area code
Break down time
Idle time
Insurance number
Cost / hour
Cost of maintenance
Cost of transport
Fuel type used
Cost of fuel
Quantity of fuel / hour
Worksections involved in.

Class: **SCAFFOLDING**

Description: Structure used as an aid during construction.
Superclass: Equipment
Subclasses: None
Attributes:

Type
Material
Number of elements
Max height
Start date
Finish date
Worksections involved

Class: **AGENT**

Description: Persons or organisations involved in project
Superclass: None
Subclasses: Client, Contractor, Resource Suppliers
Attributes:

Name

Reference number
Contact person
Address
Projects list

Class: **CLIENT**

Description: Person or an organisation which pays for the project
Superclass: Agent
Subclasses: None
Attributes:

None

Class: **CONTRACTOR**

Description: Organisation responsible for building the project
Superclass: Agent
Subclasses: None
Attributes:

None

Class: **RESOURCE SUPPLIERS**

Description: Source of construction resources.
Superclass: Agent
Subclasses: Materials Suppliers, Subcontractors
Attributes:

Reference number
Name
Projects involved in
Address
Phone
Fax
Email
EDI
Contact person
Type (nominated / not)
Number of time working together
Account number
Frequency of payment
List of earlier payments dates
List of earlier amounts paid
Financial status
Quotation responses history
Retention

Class: **MATERIALS SUPPLIERS**

Description: All potential source of building materials.
Superclass: Resource Suppliers
Subclasses: None

Attributes:

List of products
List of orders
Branches
Stockists list
Certification body
Factory production
Remarks on delivery performance
Remarks on expediting
Remarks on accepting date changes
Remarks on cost
Remarks on quality of products
Remarks on service
Remarks on training
Remarks on testing
Remarks on product development
Parent organisation
Membership of trade organisation

.....

.....

Class: **SUBCONTRACTOR**

Description: / Required third party to contribute during construction. They usually provide site labour and or equipment
Superclass: Resource Suppliers
Subclasses: Labour Subcontractor, Equipment Subcontractor (Plant and Scaffolding)
Attributes:

Specialities
Start date
Finish date
Remarks on quality of work / service

.....

.....

Class: **DOCUMENTS**

Description: All information required for the realisation, planning and control of a construction project.
Superclass: None
Subclasses: Project Design Documents, Schedule Documents, Cost Documents, Materials Procurement Documents, Reports
Attributes:

Reference number
Date made
By

.....

.....

Class: **PROJECT DESIGN DOCUMENTS**

Description: Documents used for communicating a design in a written and graphic way.
Superclass: Documents
Subclasses: Site Layout Design, Facility Design Documents
Attributes:

Project name
By
Date made
Date approved

Date revised

Class: **FACILITY DESIGN DOCUMENTS**

Description: Source of construction resources.
Superclass: Project Design Documents
Subclasses: Design Drawings, Specifications, Architect's Notes
Attributes:
Facility name

Class: **DESIGN DRAWINGS**

Description: Graphical design information.
Superclass: Facility Design Documents
Subclasses: None
Attributes:
Title
Scale
Type

Class: **SPECIFICATIONS**

Description: Legal materials standards and required quality of materials to be used in a construction project.
Superclass: Facility Design Documents
Subclasses: None
Attributes:
Materials list
Specification list
Building elements list
Specifications of building elements list

Class: **ARCHITECT'S NOTES**

Description: Extra written design information
Superclass: Facility Design Documents
Subclasses: None
Attributes:
About
Notes

Class: **SCHEDULE DOCUMENT**

Description: Documents used for planning and scheduling construction activities.
Superclass: Documents
Subclasses: Programme of work, Resource schedule, Buying schedule, Delivery schedule
Attributes:
Start date
Finish date

Class: **PROGRAMME OF WORK**

Description: The plan and schedule of the construction activities in a building project.
Superclass: Schedule Documents
Subclasses: None
Attributes:
List of worksections
Sequence of work

Class: **RESOURCES SCHEDULE**

Description: Schedule of materials required during construction
Superclass: Schedule Documents
Subclasses: Materials Schedule, Labour Schedule, Equipment Schedule
Attributes:
Worksections involved

Class: **MATERIALS SCHEDULE**

Description: Schedule of materials required during construction
Superclass: Resources Schedule
Subclasses: None
Attributes:
Material name
List dates material is required
List quantity per date

Class: **EQUIPMENT SCHEDULE**

Description: Schedule of equipment required during construction
Superclass: Resources Schedule
Subclasses: None
Attributes:
Equipment type
List dates required
List usage time per date

Class: **LABOUR SCHEDULE**

Description: Schedule of site employees during construction
Superclass: Resources Schedule
Subclasses: None
Attributes:
List date labour required
List skills required
List number of labourers in gang

Class: **BUYING SCHEDULE**

Description: Schedule of materials purchasing
Superclass: Schedule Documents
Subclasses: None
Attributes:
List dates when to issue orders
List materials quantities and specification per order

.....
.....

Class: **DELIVERY SCHEDULE**

Description: List dates when deliveries are expected
Superclass: Schedule Documents
Subclasses: None
Attributes:
Number of days deliveries are expected
List time deliveries are expected
List where deliveries are to be distributed
List of deliveries

.....
.....

Class: **COST DOCUMENTS**

Description: Documents relating to the cost of the materials required and used in construction
Superclass: Documents
Subclasses: Invoice, Quotation, Payment, Estimating Documents
Attributes:
Date made
Reference number

.....
.....

Class: **INVOICE**

Description: List of goods supplied or work done, stating quantity and price.
Superclass: Cost Documents
Subclasses: None
Attributes:
Amount due
From
To
Payment method
Date for payment
Charges
Account number

.....
.....

Class: **QUOTATION**

Description: Result of an inquiry on materials or services cost.
Superclass: Cost Documents
Subclasses: None
Attributes:
Cost list
Materials list

Service included
Valid until
Minimum quantity
Discount

Class: **PAYMENT**

Description: Tender stage documents related to the cost of materials required for construction.
Superclass: Cost Documents
Subclasses: None
Attributes:

Invoice reference number
Amount paid

Class: **ESTIMATING DOCUMENTS**

Description: Tender stage documents related to the cost of materials required for construction.
Superclass: Cost Documents
Subclasses: None
Attributes:

None

Class: **REPORTS**

Description: Reports that are produced at defined interval for construction progress control.
Superclass: Documents
Subclasses: Progress Report, Delivery Distribution Report, Delivery Report, Retrieved Materials Report, Unusable Materials Report, List of Pickings, Delivered Materials Test Report, Unused Materials Report, Materials Reconciliation Report.
Attributes:

None

Class: **PROGRESS REPORT**

Description: Highlights work progress on site.
Superclass: Reports
Subclasses: None
Attributes:

Worksections involved
Project reference number
By
Facility reference number
Report result

Class: **DELIVERY DISTRIBUTION REPORT**

Description: Holds information on where materials should be distributed on site just after being delivered.
Superclass: Reports

Subclasses: None
Attributes:

Delivery list
Distribution zone list

Class: **DELIVERY REPORT**

Description: Feedback on deliveries as to their time of arrival, the quantities delivered, pendent orders, to the type of transport...etc.
Superclass: Reports
Subclasses: None
Attributes:

Time arrived
List of materials delivered
Remarks on delivered quantity
Remarks on quality
Remarks on transportation used
Remarks on offloading method used
Accepted (Yes / No)
Remarks on packaging
Remarks on delivery address

Class: **RETRIEVED MATERIALS REPORT**

Description: Feedback on the quantities of materials retrieved from the storage.
Superclass: Reports
Subclasses: None
Attributes:

List materials retrieved
Reference number of materials retriever
List of quantity retrieved
Address from which materials has been retrieved
Worksections involved
List of picking reference number
Date material retrieved

Class: **UNUSABLE MATERIALS REPORTS**

Description: Feedback on the quantities of unusable materials per worksection as well as reporting their defects and the cause of the defect.
Superclass: Reports
Subclasses: None
Attributes:

Material reference number
Quantity
Worksections involved
Nature of defect
Cause of defect

Class: **LIST OF PICKINGS**

Description: Lists required materials to be retrieved from storage per worksection. It will also give information on quantity, quality required as well as the addresses of the storage in question.

Superclass: Reports
Subclasses: None
Attributes:

Date of picking
Retriever reference number
Issued by
Retrieval date
Storage reference number
Plant required
List materials to pick
Quantity of materials to pick
Worksection involved

Class: **DELIVERED MATERIALS TEST REPORT**

Description: Highlights the result of a quality test of materials that were just delivered.
Superclass: Reports
Subclasses: None
Attributes:

List materials tested
List order number
List required specification
List test results
By

Class: **UNUSED MATERIALS REPORT**

Description: Lists materials in good condition but were not used because of over ordering, wrong specifications or to any other reason apart from those mounting to direct waste.
Superclass: Reports
Subclasses: None
Attributes:

Quantity
Worksection involved
Storage to which materials is put back
Materials reference number

Class: **MATERIALS RECONCILIATION REPORT**

Description: Highlights used, wasted, and disappeared materials per worksection
Superclass: Reports
Subclasses: None
Attributes:

Worksections involved
Materials involved
List of quantities ordered
List of quantities measured
List reconciliation results

Class: **MATERIALS PROCUREMENT DOCUMENTS**

Description: Documents used when procuring materials from suppliers
Superclass: Documents
Subclasses: Order, Requisition, Delivery Ticket
Attributes:

Reference number

Class: **ORDER**

Description: Formal request made by contractor to suppliers to provide materials or services.
Superclass: Materials Procurement Documents
Subclasses: None
Attributes:

List of materials
List of quantities
List of specifications
List of units
List of unit load
List of worksections involved
List supplier
Date made
Date of delivery
Date cancelled
Date changed
Date delayed to
Placed by
Order state
Gate reference number
List of packaging
Materials controller reference number
Date delivered
Means of ordering
Distribution area reference number
Storage section number
Expedition date
Expediting status
From requisition

Class: **REQUISITION**

Description: An internal formal request for materials. Issued by material scheduler and / or site manager
Superclass: Materials Procurement Documents
Subclasses: None
Attributes:

Materials list required
Packaging list required
Specification list required
Suppliers list
Delivery date list requited
Worksections list

Unit required
Unit load required
Issue against
Placed by

.....

.....

Class: **DELIVERY TICKET**

Description: Proof of delivery receipt
Superclass: Materials Procurement Documents
Subclasses: None
Attributes:

Ticket number
From
Date
List orders
List materials
Name of truck driver

.....

.....

Class: **SITE**

Description: Geographical location on which a construction project is to be carried out.
Superclass: None
Subclasses: None
Attributes:

Reference number
Address
Site office phone
Site office fax
Site office Email
Site manager's name
Access roads references
Access roads description
Type of site (restricted, open field, sloping, long and narrow)
Boundaries
Proximity to power lines
Type of soil

.....

.....

Class: **ACCESS ROADS**

Description: Temporary and permanent roads that are used during construction
Part Of: Site
Subclasses: None
Attributes:

Reference number
Width
Type of surfacing

.....

.....

Class: **GATES**

Description: Transport entrance to construction site.
Part Of: Site

Subclasses: None
Attributes:

Reference number
Address
Security men list
Size
List offloading areas connected to

.....

.....

Class: **OFFLOADING / CHECKING AREA**

Description: A site area where delivered are checked and offloaded.
Part Of: Site
Subclasses: None
Attributes:

Reference number
Size
Address

.....

.....

Class: **DISTRIBUTION ZONE**

Description: Site location where materials are stored for later usage or are immediately needed for building activity as soon as delivered.
Part Of: Site
Subclasses: Storage, Same Day Use Zone
Attributes:

Reference number
Address

.....

.....

Class: **STORAGE**

Description: Site location where materials are stored fro future use.
Superclass: Distribution Zone
Subclasses: Temporary storage (Offsite storage, Stockpiles), Permanent storage (Containers, Sheds)
Attributes:

Storage controller reference number
Type (permanent / temporary)
State
Hygrometry
Fire safety
Weather proofing
Size
Door size
Ventilation
Contact with ground
Material made from
Proximity to gates
Proximity to facility(ies)
List materials in it
List materials suitable to be stored in it
Capacity
Occupied space
Unoccupied space

Class: **SAME DAY USE ZONE**

Description: Site location where materials are needed as soon as they are delivered.
Superclass: Distribution Zone
Subclasses: Preparation Zone (Mixing, Fabrication, Assembly zones), Building Zone with its Built and Non Built Zones
Attributes:

None

Class: **PROJECT**

Description: Construction piece of work that needs skills, effort and careful planning, managed by a contractor and commissioned by a client.
Superclass: None
Subclasses: None
Attributes:

Reference number
Client name
Contract type
Start date
Finish date
Architect
Engineer
Quantity surveyor
Type of project
Capacity
Number of facilities
Facilities reference numbers list
Total area
Cost/m2
Tender cost
Address
List of materials

Class: **FACILITY**

Description: A building in a project.
Part Of: Site
Subclasses: None
Attributes:

Number of stories
Address on site
Type of structure

Class: **MATERIALS PROCUREMENT ELEMENTS**

Description: Elements through which materials are monitored for delivery and are delivered
Superclass: None
Subclasses: Expedition, Transport, Delivery
Attributes:

None

Class: **TRANSPORT**

Description: Means for carrying building materials from suppliers' premises to where they are required on site or offsite.
Superclass: Materials Procurement Elements
Subclasses: None
Attributes:

Type
Size
Gyration radius

Class: **EXPEDITION**

Description: Follow up of placed orders, to make sure that materials / services arrive when and where they are needed.
Superclass: Materials Procurement Elements
Subclasses: None
Attributes:

Order list
Orders expediting dates list
Expedition results list

Class: **DELIVERY**

Description: Act of bringing a complete order or a group of orders in the same time at a construction site. A part of an order will be considered as a completed order.
Superclass: Materials Procurement Elements
Subclasses: None
Attributes:

Reference number
Date
List of orders
Sequence of offloading
Sequence of storing
Plant to use
Delivery ticket number
Suppliers notes reference number
Remarks on delivery performance
Actual delivery date
Actual delivery time
Delivery truck number

Class: **WORKSECTION**

Description: A construction activity.
Superclass: None
Subclasses: None
Attributes:

Reference number

Subcontractor reference
Description
List of materials required
List of materials quantities
Early start date
Late start date
Early finish date
Late finish date
Number of labourers required
Skills required
Plant required
Time plant required
Level
Building elements involved
Space involved
Location of materials
Location of plant
Scaffolding?
Preceding worksections
Succeeding worksection
Lag
Status
Unit of measurement for costing

.....

.....

None is used when a class does not have its own attributes but inherits attributes from its parent class.

A.2. High Level Objects Of The Suggested Materials Management Information Model

RESOURCES
- MATERIALS
- EMPLOYEES
  - SITE EMPLOYEES
    - GANG
    - GANG RESPONSIBLE
    - CRAFTSMAN
  - ADMINISTRATION EMPLOYEES
- EQUIPMENT
  - PLANT
  - SCAFFOLDING
  - FACILITIES

AGENTS

CLIENT

CONTRACTOR

RESOURCE SUPPLIER
- MATERIALS SUPPLIER
- SUBCONTRACTOR
  - LABOUR SUBCONTRACTOR
  - EQUIPMENT SUBCONTRACTOR
    - PLANT SUBCONTRACTOR
    - SCAFFOLDING SUBCONTRACTOR
    - FACILITIES SUBCONTRACTOR

DOCUMENTS
- PROJECT DESIGN DOCUMENTS
  - SITE LAYOUT DESIGN
  - FACILITY DESIGN DOCUMENTS
    - DESIGN DRAWINGS
    - SPECIFICATIONS
    - ARCHITECT'S NOTES
- SCHEDULE DOCUMENTS
  - PROGRAM OF WORK
  - MATERIALS SCHEDULE
  - EQUIPMENT SCHEDULE
    - PLANT SCHEDULE
    - SCAFFOLDING SCHEDULE
    - FACILITIES SCHEDULE
  - LABOUR SCHEDULE
  - BUYING SCHEDULE
  - DELIVERY SCHEDULE
- COST DOCUMENTS
  - ESTIMATING DOCUMENTS
  - QUOTATION
  - INVOICE
  - PAYMENT
- MATERIALS PROCUREMENT DOCUMENTS
  - ORDER
  - REQUISITION
  - DELIVERY TICKET
- REPORTS
  - PROGRESS REPORT
  - SELECTED SUPPLIERS REPORT
  - SELECTION CRITERIA
  - DELIVERY REPORT
  - DELIVERED MATERIALS TEST REPORT
  - RETRIEVED MATERIALS REPORT
  - LIST OF PICKING
  - MATERIALS USAGE REPORT
  - MATERIALS RECONCILIATION REPORT

DELIVERY DISTRIBUTION REPORT

SITE

- ACCESS ROADS
- GATES
- SITE OFFICE
- OFFLOADING / CHECKING AREA
- DISTRIBUTION ZONES
  - STORAGE
    - TEMPORARY STORAGE
      - OFFSITE
      - STOCKPILES
    - PERMANENT STORAGE
      - CONTAINERS
      - SHEDS
  - SAME DAY USE ZONE
    - PREPARATION ZONE
      - MIXING
      - FABRICATION
      - ASSEMBLY
    - BUILDING ZONE
      - BUILT ZONE
      - NON-BUILT ZONE

PROJECT

- FACILITY

MATERIALS PROCUREMENT ELEMENTS

- EXPEDITION
- DELIVERY
- TRANSPORT

WORKSECTION

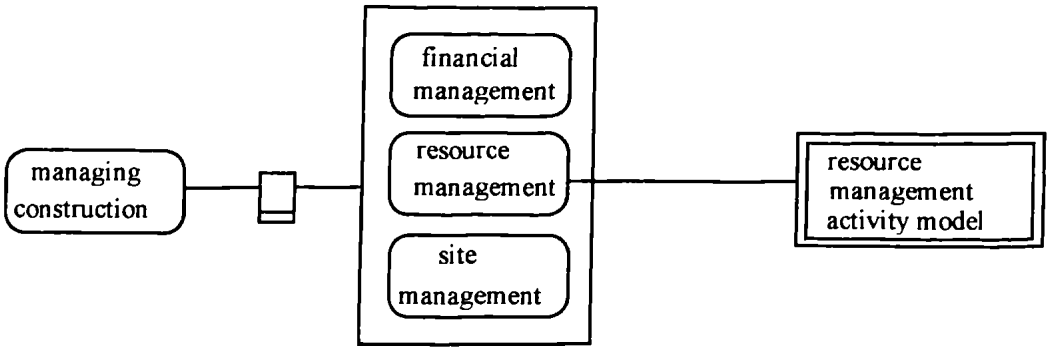
- WORKSECTION' S MATERIALS
- WORKSECTION' S SITE EMPLOYEES
- WORKSECTION' S EQUIPMENT

A.3. ICMM's Activity Hierarchy

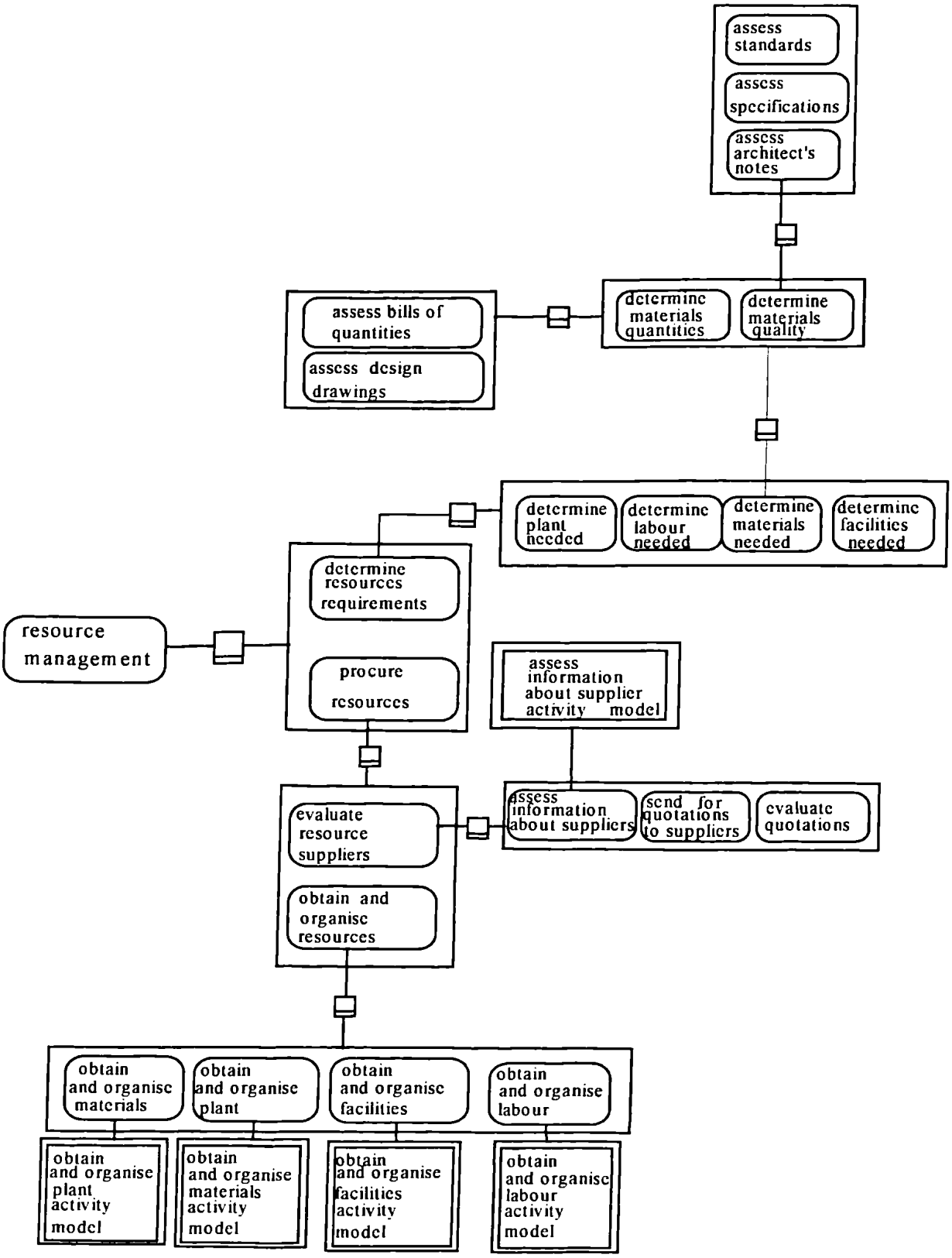
MANAGING CONSTRUCTION
FINANCIAL MANAGEMENT
SITE MANAGEMENT
RESOURCE MANAGEMENT
DETERMINE RESOURCES REQUIREMENTS
DETERMINE MATERIALS NEEDED
DETERMINE MATERIALS QUANTITIES
ASSESS DESIGN DRAWINGS
ASSESS BILLS OF QUANTITIES
DETERMINE MATERIALS QUALITY
ASSESS MATERIALS STANDARDS
ASSESS SPECIFICATIONS DOCUMENTS
ASSESS ARCHITECTS NOTES
DETERMINE PLANT NEEDED
DETERMINE LABOUR NEEDED
DETERMINE SUBCONTRACTORS NEEDED
DETERMINE FACILITIES NEEDED
PROCURE RESOURCES
EVALUATE RESOURCE SUPPLIERS
ASSESS INFORMATION ABOUT SUPPLIERS
OBTAIN DATA ON RESOURCE SUPPLIERS
OBTAIN DATA ON PARENT ORGANISATION
OBTAIN DATA ON AGENTS
OBTAIN DATA ON BRANCHES
OBTAIN ADDRESS AND POSTAL CODE
OBTAIN PHONE FAX/TELEX EDI
OBTAIN DATA ON HOME/EXPORT SALES OFFICES
OBTAIN MEMBERSHIP OF TRADE ORGANISATION
ASSESS SUPPLIER'S TYPE OF MANAGEMENT
IDENTIFY SELECTION CRITERIA
IDENTIFY TECHNICAL AND ADVISORY SUPPORT
OBTAIN INFORMATION ON TRAINING
OBTAIN INFORMATION ON TESTING
OBTAIN INFORMATION ON DEVELOPMENT
ASSESS SUPPLIER'S CODING SYSTEM
ASSESS SUPPLIER'S ASSURANCE ARRANGEMENT
OBTAIN DATA ON CERTIFICATION BODY
ASSESS SUPPLIER'S STOCK HOLDING
ASSESS SUPPLIERS' USE OF IT COMMUNICATION
ASSESS SUPPLIERS CAPABILITIES
ASSESS SUPPLIERS' EXPERIENCE WITH FIRM
ASSESS MANUFACTURING CAPACITY
ASSESS SUPPLIERS' SCOPE OF PRODUCTS
ASSESS SUPPLIERS' RELIABILITY
ASSESS USE IN OTHER CONTRACTS
ASSESS QUALITY OF DELIVERY
ASSESS FLEXIBILITY IN ACCEPTING ORDER DATES CHANGES
ASSESS PROMPTNESS IN RETURNING QUOTATIONS
ASSESS COMPETITIVENESS
ASSESS SUPPLIERS' FINANCIAL STATUS
ASSESS SUPPLIERS' PERSONNEL NUMBER
SELECT SUPPLIERS FOR QUOTATIONS
SEND FOR QUOTATIONS FROM SUPPLIERS
EVALUATE SUPPLIERS' QUOTATIONS
OBTAIN AND ORGANISE RESOURCES
OBTAIN AND ORGANISE MATERIALS
ORDER MATERIALS
STATE QUANTITY NEEDED
STATE SPECIFICATIONS
ESTABLISH DELIVERY ARRANGEMENT
PREPARE SITE FOR DELIVERY
PREPARE SITE ACCESS
PREPARE GROUND FOR DELIVERY
STONE STOCK PILES AREA
TIDY UP DELIVERY AREA
ASSESS TRANSPORT DIMENSIONS
ASSESS GROUND BEARING CAPACITY
PREPARE CONTROL/CHECK AREA
ESTABLISH POSSIBILITIES OF LIFTING AND STOCKING ON SITE
DEFINE PLANT CHARACTERISTICS
ASSESS PLANT UNIT LOAD
ASSESS PLANT TYPE
PREPARE LABOUR FOR DELIVERY
DETERMINE NUMBER OF LABOUR NEEDED
APPOINT QUALIFIED LABOURERS FOR CHECKING MATERIALS
ASSESS MATERIALS TO BE DELIVERED CHARACTERISTICS
ASSESS UNIT OF MEASUREMENT
ASSESS UNIT LOAD
ASSESS MATERIALS PACKAGING
ESTABLISH HANDLING METHOD
ASSESS MATERIALS CLASSIFICATION
ASSESS MATERIALS LABELLING
ESTABLISH DIMENSIONS OF STOCKPILES
PREPARE STORAGE AREAS
ASSESS WORKSECTIONS LOCATIONS
ESTABLISH DELIVERY SCHEDULE
STATE DELIVERY SITE ACCESS
EXPEDITE DELIVERY
DELIVER AND CHECK MATERIALS
OBTAIN DELIVERY TICKET

OBTAIN SUPPLIERS ADVICE NOTES
CHECK DELIVERY DATE , TIME AND ACCESS
CHECK QUANTITY DELIVERED
CHECK QUALITY DELIVERED
CHECK DELIVERY ARRANGEMENT
UPDATE SUPPLIERS PERFORMANCE FILE
DECIDE ON DELIVERY
ACCEPT DELIVERY
REJECT DELIVERY
HANDLE MATERIALS
ASSESS HANDLING PRECAUTIONS
ASSESS SUITABLE PLANT AND ALTERNATIVES
SELECT CREW SIZE FOR HANDLING
CHECK PLANT AVAILABILITY
LOCATE STORAGE WORKSECTION LOCATION
ASSESS WEIGH TO BE HANDLED
ASSESS METHODS OF OFFLOADING
STORE MATERIALS
ASSESS STORAGE SPECIFICATIONS
ASSESS MATERIALS USE IN SEQUENCE
ASSESS MATERIALS USE BY DATE
APPOINT PERSON RESPONSIBLE FOR DELIVERED MATERIALS
DETERMINE SUITABLE STORAGE
ASSESS MATERIALS VALUE
ASSESS MATERIALS VULNERABILITY
ASSESS MATERIALS VULNARABILITY TO THE WEATHER
ASSESS MATERIALS RISK OF THEFT LOSS
ASSESS MATERIALS RISK OF CONTAMINATION
ASSESS MATERIALS FRAGILITY
ASSESS MATERIALS FREQUENCY OF DELIVERY
ASSESS MATERIALS USE BY DATE
ASSESS MATERIALS PACKAGING
ASSESS WORKSECTIONS LOCATIONS
DISTRIBUTE DELIVERED MATERIALS
RETRIEVE AND INSTALL MATERIALS
REVIEW LABOUR SCHEDULE
DETERMINE REQUIRED NUMBER OF LABOUR
DETERMINE REQUIRED SKILLS
APPOINT RESPONSIBLE FOR MATERIALS
REVIEW PROGRAM OF WORK
REVIEW MATERIALS STOCK ON SITE
ASSESS QUANTITY PLANNED
ASSESS QUANTITY NEEDED
DEFINE A UNIT OF RETRIEVAL
PREPARE FOR MATERIALS RETRIEVAL
ASSESS WORK LOCATION
DEFINE INDOOR WORK
DEFINE OUTDOOR WORK
ASSESS POSSIBILITIES OF STACKING NEAR WORKSECTION
ASSESS MATERIALS LOCATION
ASSESS MATERIALS ONWARD HANDLING
ASSESS POSSIBILITIES OF MECHANICAL HANDLING
ASSESS DISTANCE TO STORAGE WORKSECTION
ASSESS POSSIBILITIES OF MANUAL HANDLING
ASSESS OBSTRUCTIONS ON SITE
ASSESS WAYS OF UNPACKING
PAY FOR MATERIALS SUPPLIERS
ESTABLISH CONDITIONS OF SALES
ESTABLISH WARRANTIES & GUARANTIES
ESTABLISH PRICES AND DISCOUNTS
ESTABLISH METHODS OF PAYMENT
ESTABLISH CREDIT TERMS
ESTABLISH HANDLING & DELIVERY CHARGES
ESTABLISH TAXES AND CUSTOMS
ESTABLISH FEES AND CHARGES
DETERMINE MATERIALS COSTS
COMPARE INVOICE WITH DELIVERY TICKET AND ORDER
OBTAIN AND ORGANISE LABOUR
OBTAIN AND ORGANISE PLANT
OBTAIN AND ORGANISE FACILITIES
OBTAIN AND ORGANISE SUBCONTRACTORS

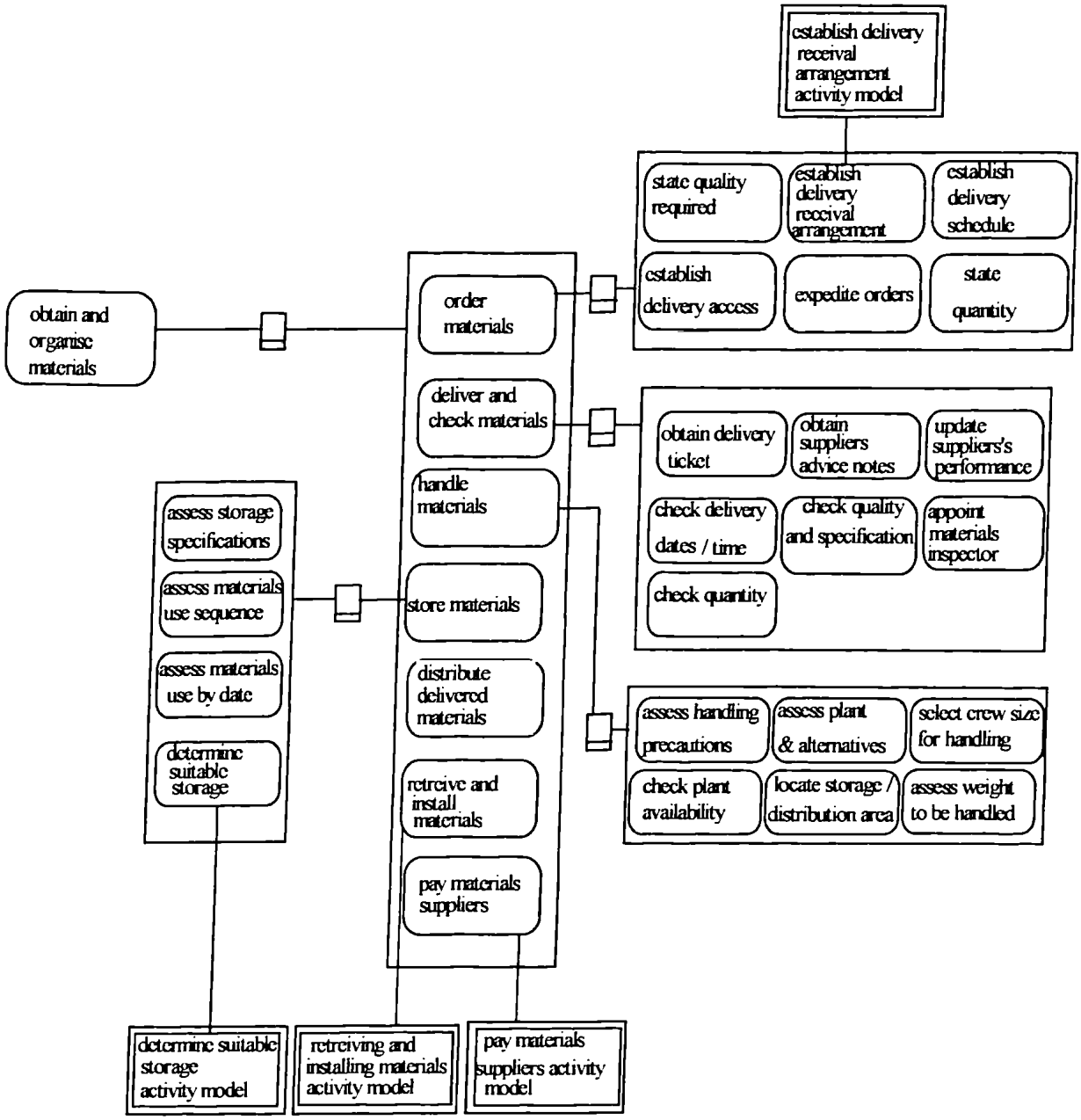
A.4. ICMM's Activity Models



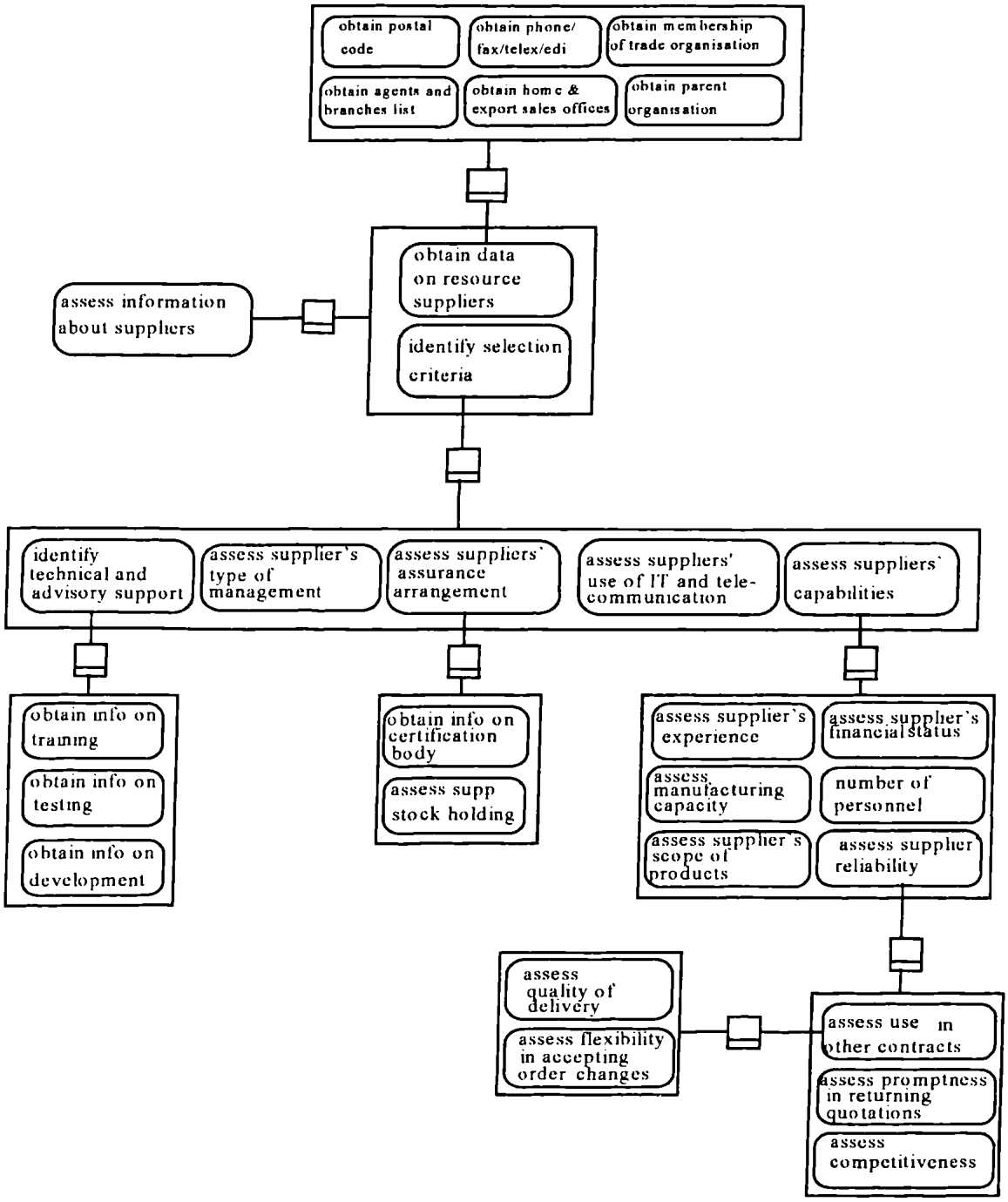
Managing construction activity model (Aouad et al 1994)



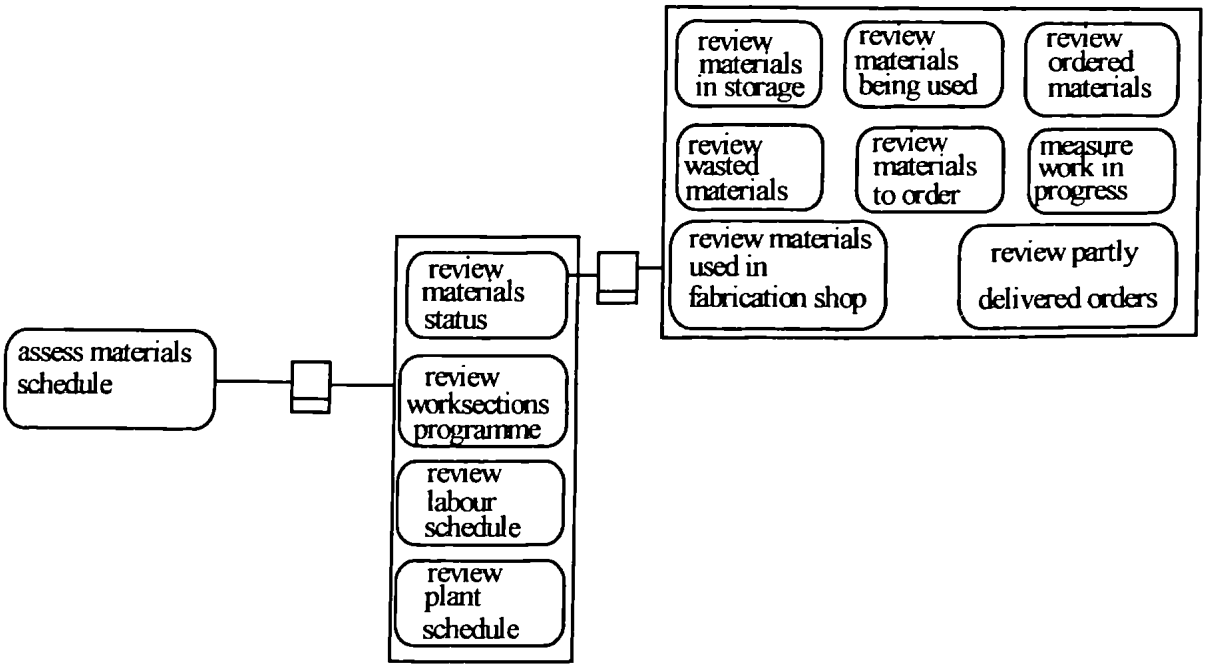
Resource management activity model



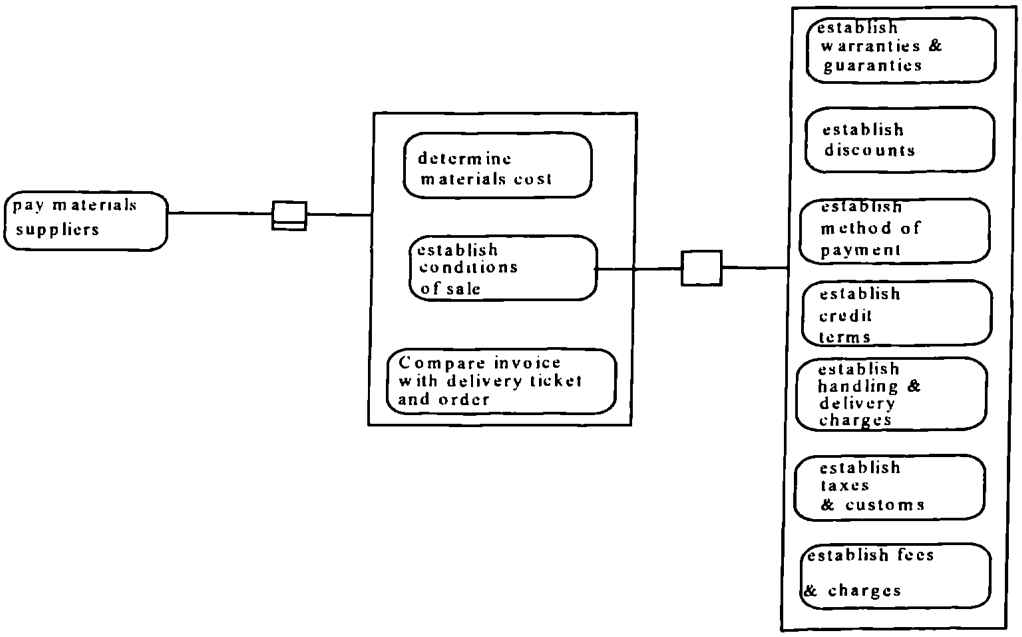
Obtain and organise materials activity model



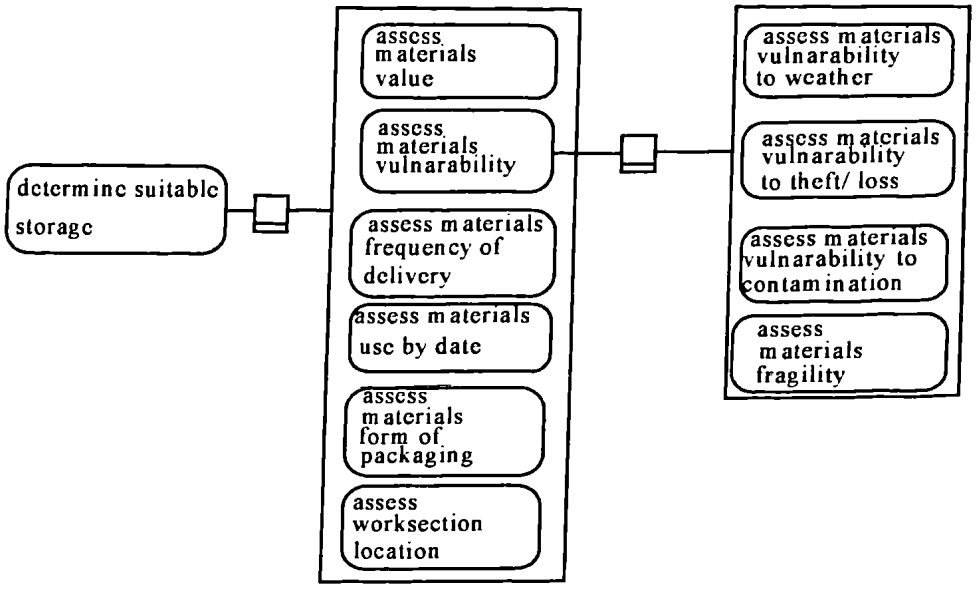
Assess information about suppliers activity model



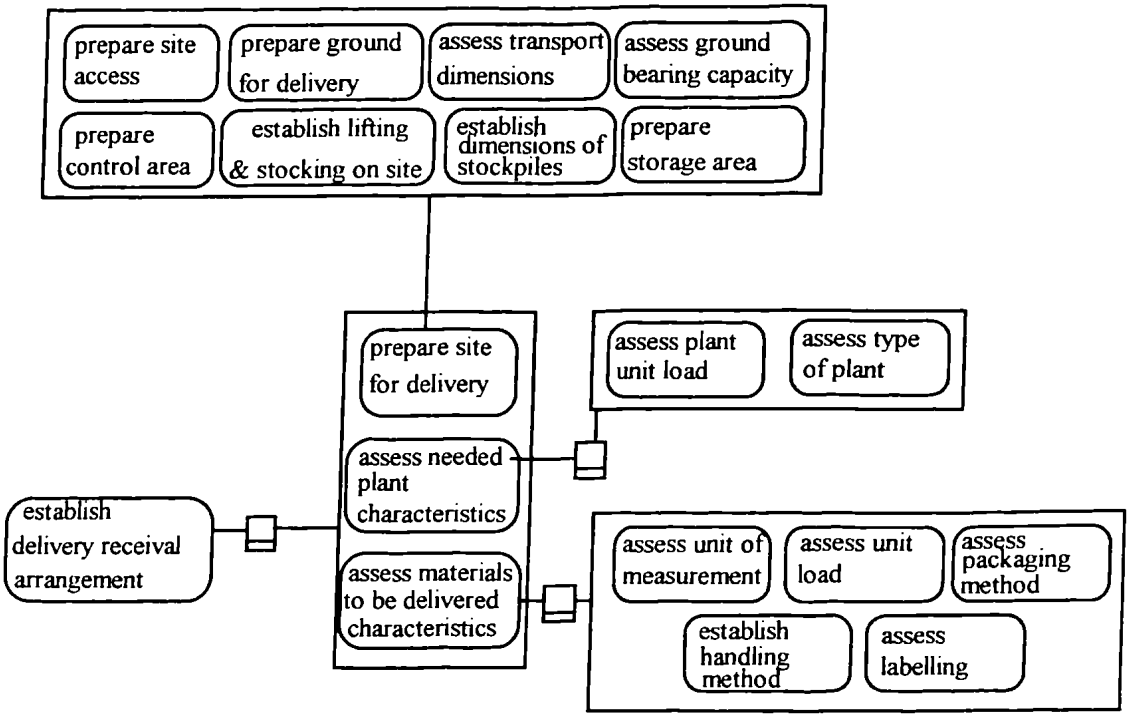
Assess materials schedule activity model



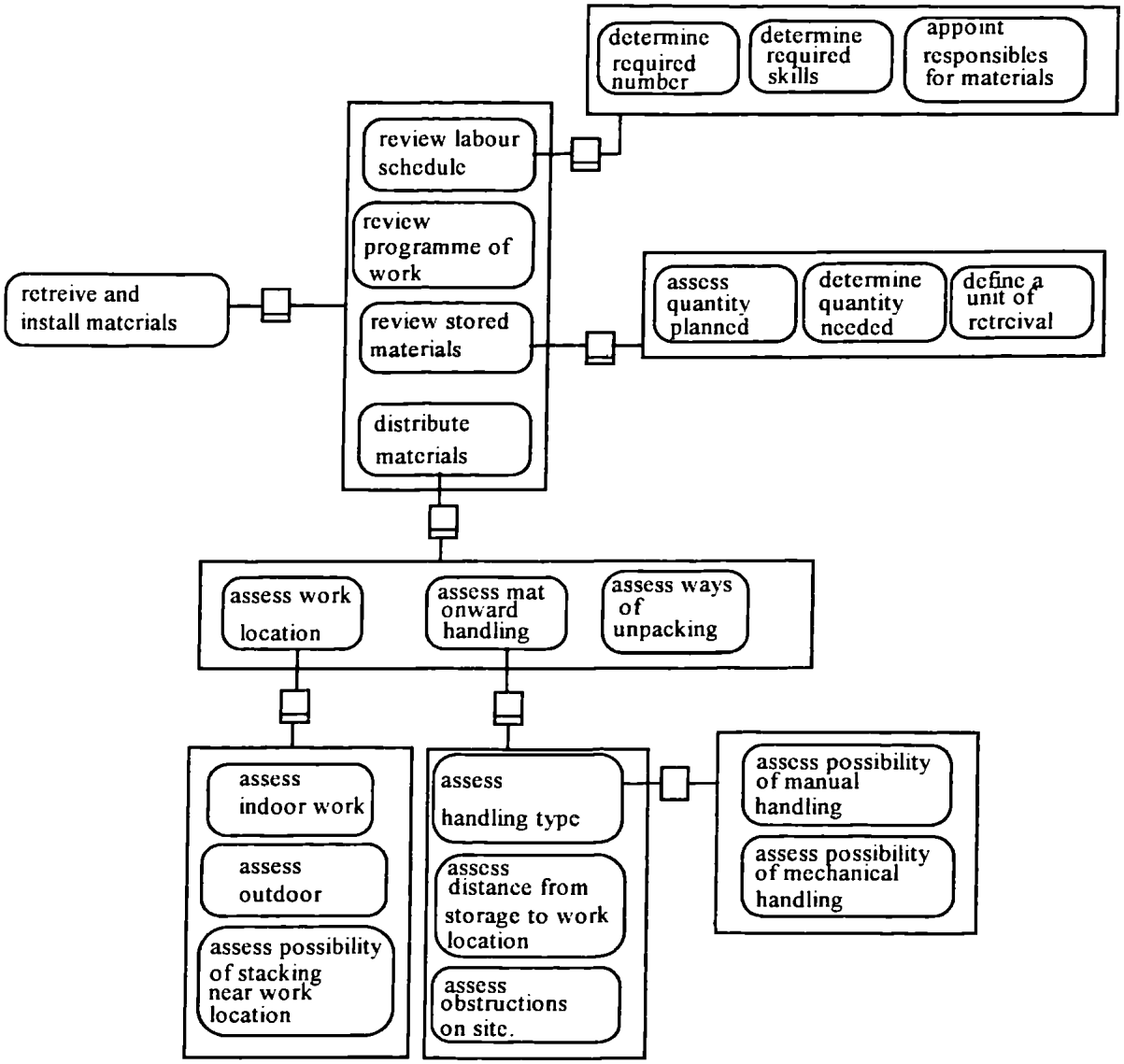
Pay suppliers activity model



Determine suitable storage activity model



Establish delivery receival arrangement activity model



Retrieve and install materials activity model

A.5. Classes' Responsibilities, Contracts and Collaborations

Classes	Collaborative Classes

Class: **Resources**
Superclass:
Type of object: Abstract
Subclasses: Materials, Finance, Employees, Equipment
Private responsibilities:
Interact with resource database
Know code number

Class: **Materials**
Superclass: Resource
Type of object: Abstract
Subclasses: Bulk, Uniquely identifiable, Both

Private responsibilities:

Know how much is in storage	Storage (49,50)
Know how much is required for project	Design Drawings(18)

Documents (15,16)

Public responsibilities:
1. Has knowledge of its description
Has knowledge of how it should be handled
Has knowledge of how it should be packaged
Has knowledge of how it should be stored
Has knowledge of its price
2. Commit and accept changes in database

Class: **Equipment**
Superclass: Resources
Type of object: Abstract
Subclasses: Plant, Scaffolding, Facilities
Private responsibilities:

Produce report on usage and availability	Equipment schedule (34)
Know its suitability for site	Site layout design (22,23,24)
Site (47)	
Know its suitability for worksection	Worksection (64)

Public responsibilities:
3. Know its characteristics
4. Interact with its database

Class: **Site Employees**
Superclass: Employees
Type of object: Abstract
Subclasses: Gang, Gang responsible, Craftsmen
Private responsibilities:

Know schedule of work	Labour schedule(33)

Public responsibilities:

5. Know its data, and skills
6. Commit and accept changes to site employees database

Class: **Agent**
Superclass:
Type of Object: Abstract
Subclasses: Contractor, Resource supplier, Client

Private responsibilities:

Commit changes and accept result in database	Selection Criteria (17)
Produce report on itself given a set of criteria	Resource Suppliers (8)
	Subcontractors (12)

Public responsibilities:
7. Know its data

Class: **Resource Supplier**
Superclass: Agent
Type of Object: Abstract
Subclasses: Material Supplier, Subcontractor

Private responsibilities:

Know in which project it is involved	Project (51)
Knows its products	

Public responsibilities:
8. Know its criteria
9. Know its bank account number
10. Know its method of payment
11. Know its code number

Class: Requisition
Superclass: Materials Procurement Document
Type of Object: Concrete
Subclasses:
Public responsibilities:
13. Provide data for use in orders
    Provide materials quantity
    Provide materials delivery date
    Provide materials type of packaging
    Provide materials specification
    Provide materials supplier (facultative)

Class: **Expedition**
Superclass: Materials Procurement Element
Type of Object: Concrete
Subclasses:
Private responsibilities:

Enquire about orders	Order(37)

Public responsibilities:
14. Know about orders changes

Class: **Progress Report**

Superclass: Report
Type of Object: Concrete
Subclasses:

Public responsibilities:

12. Get information on materials quantities used, wasted per date per worksection

Class: **Document**
Superclass:
Type of Object: Abstract
Subclasses: Project Design Document, Schedule document, Cost Document, Materials Procurement Documents, Delivery ticket
Private responsibilities:

Know data produced
Know document producer

Public responsibilities:

15. Know data it contains
16. Know its code number

Class: **Project Design Document**
Superclass: Document
Type of Object: Abstract
Subclasses: Facility Design Document, Site Layout Design
Private responsibilities:

Interact with CAD database

Class: **Selection criteria**
Superclass: Document
Type of Object: Concrete
Subclasses:
Private responsibilities:

Accept changes from project	Project (52)

Public responsibilities:

17. State criteria required from prospective agents, resource suppliers and subcontractors

Class: **Materials Reconciliation Report**
Superclass: Reports
Type of object: Concrete
Subclasses:
Private responsibilities:

Produce instances of itself	Worksection (58,59,61,62,63,68) Materials schedule (30) Orders (39) Progress Report (12)

Class: **Design Drawings**
Superclass: Project Design Document
Type of Object: Abstract
Subclasses: Design Drawing, Specifications, Architect's Notes
Private responsibilities:

Commit changes and accept result in database	Architect's notes (20)
Produce bills of materials with specifications	Specifications (19)

Public responsibilities:

18. Takeoff materials quantities

Class: **Specifications**
Superclass: Facility Design Document
Type of Object: Concrete
Subclasses:
Private responsibilities:
Commit and accept changes in specification database — Architect's notes (21)

Public responsibilities:
19. Interact with specifications database

Class: **Architect's notes**
Superclass: Facility Design Document
Type of Object: Concrete
Subclasses:
Public responsibilities:
20. Update facility Design Document (CAD format)
21. Update specifications database

Class: **Site Layout Design**
Superclass: Project Design Document
Type of Object: Concrete
Subclasses:
Public responsibilities:
22. Know where distribution zones, access roads, gates are
Know obstructions on site
Commit and accept changes in its drawing database
23. Know where mechanical circulation is
24. Know site subdivisions

Class: **Schedule Document**
Superclass: Documents
Type of Object: Abstract
Subclasses: Programme of work, Materials schedule, Buying schedule, Delivery schedule, Labour schedule, Equipment schedule.
Private responsibilities:
Know start / end dates

Class: **Programme of work**
Superclass: Schedule document
Type of Object: Concrete
Subclasses:

Private responsibilities:
Know worksection plan — Worksection(58,64,66)

Public responsibilities:
25. Update site employee schedule
26. Update Equipment schedule
27. Update Buying schedule
28. Update Materials schedule
29. Update Delivery schedule

Class: **Materials Schedule**
Superclass: Schedule Document
Type of Object: Concrete
Subclasses:
Private responsibilities:

Know when quantity and quality of materials needed / period of time	Programme of work (29) Design drawings (18) Worksection (58,59,64,66)
Open to programme of work changes	Programme of work (24)

Public responsibilities:
30. Make accessible details about quantities and quality needed per period of time

Class: **Buying Schedule**
Superclass: Schedule Document
Type of Object: Concrete
Subclasses:
Private responsibilities:

Know when to purchase materials	Worksection (66)
Commit and accept changes to itself	
Commit changes to orders database	Order (36)
Accept changes from programme of work	Programme of work (27)

Public responsibilities:
31. Know materials to buy

Class: **Delivery Schedule**
Superclass: Schedule Document
Type of Object: Concrete
Subclasses:
Private responsibilities:

Has information on deliveries	Delivery (54)
Know when deliveries are expected	Delivery (55)
Interact with delivery database	Delivery (53)
Know time, gate and transportation of deliveries	Delivery (56)
Accept changes from programme of work	Programme of work (29)

Public responsibilities:
32. Produce list of daily deliveries

Class: **Labour Schedule**
Superclass: Schedule Document
Type of Object: Concrete
Subclasses:
Private responsibilities:

Know , who, skills, and when site employees are required	Worksection (64,60)
Accept changes from programme of work	Programme of work (25)
Interact with site employee database	Site employee (6)

Public responsibilities:

33. Make available information on labour skills and schedules

Class: **Equipment Schedule**
Superclass: Schedule Document
Subclasses:

Private responsibilities:

Know which and when equipment is required for a time period	Worksection (64,65,66)
Interact with equipment database	Equipment (4)
Accept changes from programme of work	Programme of work(26)

Public responsibilities:

34. Produce report on usage and availability of equipment

Class: **Order**
Superclass: Materials Procurement Documents
Type of Object: Concrete
Subclasses:
Private responsibilities:

Know its status and produce report on orders problems	Delivery (57)
Expedite itself	Expedition (14)
Produce report on placed orders	
Know the date of order and of delivery	
Produce report on changed orders	
Knows how to print itself	
Knows what is being ordered	Requisition (12) Buying schedule (31)
Knows materials supplier	Materials supplier (11) Agents (7)
Knows if it has been paid for	Payment (44)
Knows worksections linked to it	Worksection (58)

Public responsibilities:

35. Know about its data
36. Update order database
37. Knows its delivery arrangement (time, quantity, quality, packaging, unit load, unit of measurement)
38. Knows its code number
39. Order maintenance
    - Produce report on order status
    - Hold information on what was delivered
    - Knows about pendent orders

Class: **Delivery Ticket**
Superclass: Materials Procurement Documents
Type of Object: Concrete
Subclasses:
Public responsibilities:

40. Know its code number
41. Know source and destination
42. Know order number
43. Know content of delivery

Class: **Cost Document**
Superclass: Documents
Type of Object: Abstract
Subclasses: Estimating Documents, Materials Suppliers Quotations, Invoice, Payment
Private responsibilities:
Know its code number
Know date of issue
Know date of receival
Know amount

Class: **Payment**
Superclass: Cost Documents
Subclasses:
Private responsibilities:

Get materials prices	Materials (1)
Decide on payment	
Document payment problems	
Check invoice against delivery ticket	Delivery ticket (40, 41, 42) Invoice (45, 46)
Check invoice against order	Order (35, 38) Invoice (45, 6146)
Pay supplier	Agents (7)
Acquire payment method	Materials supplier (9,10)
Acquire bank account number	Subcontractor (13, 14)

Public responsibilities:
44. Update order payment database

Class: **Invoice**
Superclass: Cost Documents
Type of Object: Concrete
Subclasses:
Public responsibilities:
45. Know order number in question
46. Know destination and source

Class: **Quotation**
Superclass: Cost Documents
Type of Object: Concrete
Subclasses:
Private responsibilities:

Know offers on materials	
Know supplier	Agents (7)
Know material in question and its quantity	
Know its issue date	
Know its period of validation	

Class: **Site**
Superclass:
Type of Object: Concrete

Subclasses:
Private responsibilities:
Know its characteristics (size, address, shape, orientation)
Public responsibilities:
47. Know site obstructions

Class: **Access roads**
Superclass: Site
Type of Object: Concrete
Subclasses:
Private responsibilities

Know its characteristics	Site layout design(23,24,22)
Know its location	
Know its dimensions, code number / name	
Know its obstructions	
Know its relative position to site (In / Out)	

Class: **Gate**
Superclass: Site
Type of Object: Concrete
Subclasses:
Private responsibilities:

Know its dimension, location and code number	Site design layout (23,24,22)
Know deliveries expected per day	Delivery schedule(32)

Class: **Offloading checking area**
Superclass: Site
Type of Object: Concrete
Subclasses:
Private responsibilities:

Responsible for acquiring delivery ticket	
Responsible for reading bar code data	
Know its characteristics	Site design layout (23,24,22)
Know its dimension / area	
Know its location	
Produce delivery report	Delivery (32), order (37)
Know what is expected in each delivery	
Know delivery's day, time of arrival, quantity, , packaging, unit load, transport	

Class: **Distribution Zone**
Superclass. Site
Type of Object: Abstract
Subclasses: Storage, Same Day Use Zone
Private responsibilities:

Know what is in it	
Know its location and position to worksection	Site design layout (23,24,22)
Update materials database	Materials (2)

Public responsibilities:
48. Know how much materials are in it

Class: **Storage**
Superclass: Distribution Zone
Type of Object: Abstract
Subclasses: Off site storage, In site storage
Private responsibilities:
Read bar code number
Know its physical condition
Know for which materials it is best suited (protection wise) Materials (1)

Public responsibilities:
49. Update itself when materials are retrieved
50. Produce report on retrieved materials (Who, when, why, quantity)

Class: **Same Day Use Zone**
Superclass: Distribution Zone
Type of Object: Abstract
Subclasses: Building Zone, Preparation Zone
Private responsibilities:
Private responsibilities:
Know its capacity at any time
Know what is in it
Know its location and position to worksection Site design layout (23,24,22)
Update materials database Materials (2)

Class: **Project**
Superclass:
Type of Object: Concrete
Subclasses:
Private responsibilities:
Interact with CAD drawing database

Public responsibilities:
51. Interact with project database
52. Hold materials suppliers selection criteria

Class: **Materials forwarding elements**
Superclass:
Type of Object: Abstract
Subclasses: Delivery, Transport, Expedition
Private responsibilities:
Responsible for dispatching orders to site

Class: **Transport**
Superclass: Materials forwarding element
Type of Object: Abstract
Subclasses:
Private responsibilities:
Know its suitability to site Site design layout (23,24,22)
Know which delivery for which transport Delivery (56)

Class: **Delivery**

Superclass: Materials forwarding element
Type of Object: Concrete
Subclasses:
Private responsibilities:

Know what is expected in it	Order (38)
Know its code	
Interact with order database	Order (36)
Know what is in it	
Know where to distribute delivered materials	Worksection (67), Distribution zone (48) Site design layout (23,24,22)

Public responsibilities:

53. Update delivery database
54. Update delivery schedule
55. Know its delivery date
56. Know its delivery arrangement
57. Know if it actually has been delivered

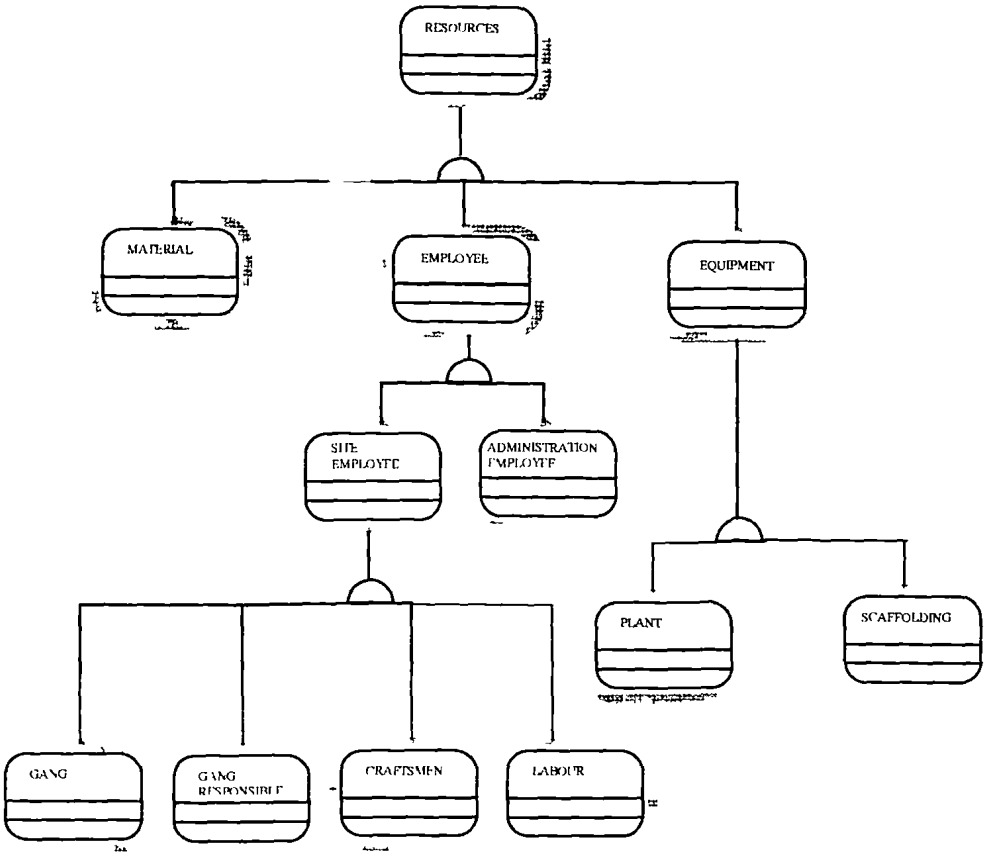
Class: **Worksection**
Superclass: Materials forwarding element
Type of Object: Concrete
Subclasses:
Private responsibilities:

Know worksection's responsible
Know site employee working on it

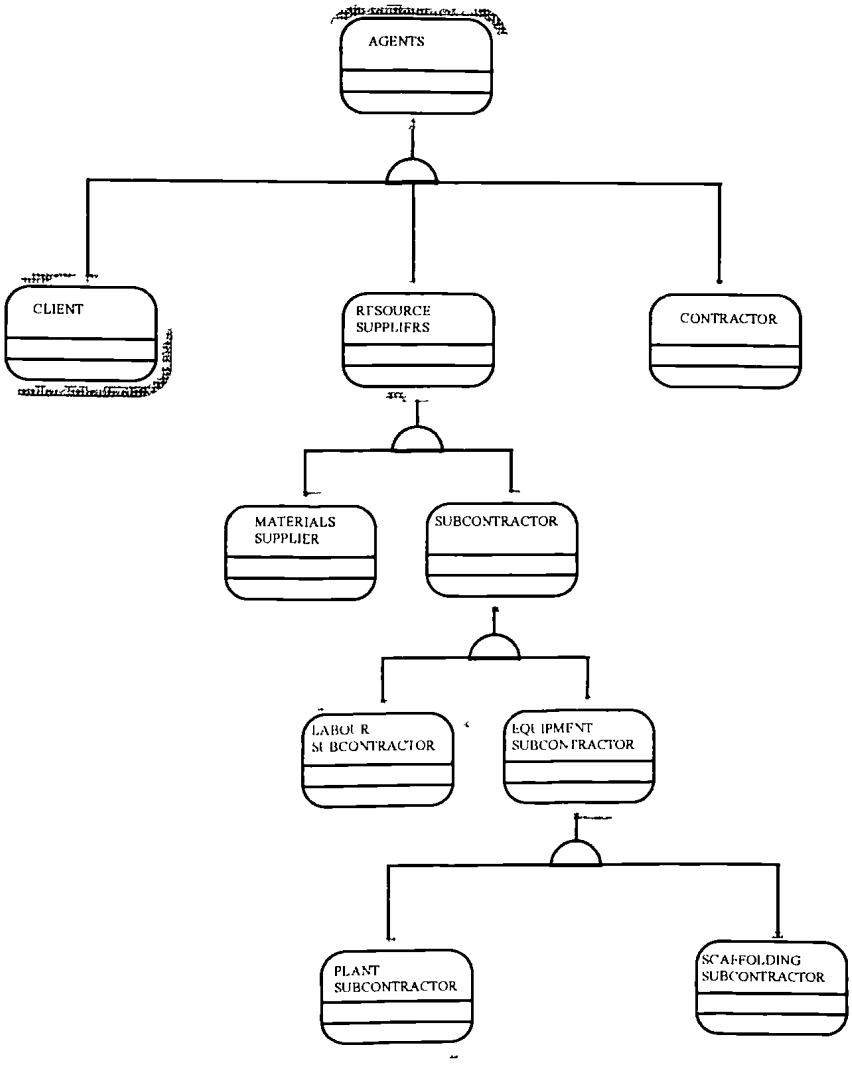
Public responsibilities:

58. Know its code number
59. Know materials needed, in quantity and quality
60. Know skills needed
61. Produce progress report
62. Know unused and unusable materials
63. Know quantity of materials used
64. Know its type of work
65. Know needed equipment
66. Responsible its time control
    - Know preceding and afterward worksection
    - Know start and end date
    - Update programme of work
67. Know its location
68. Update material database

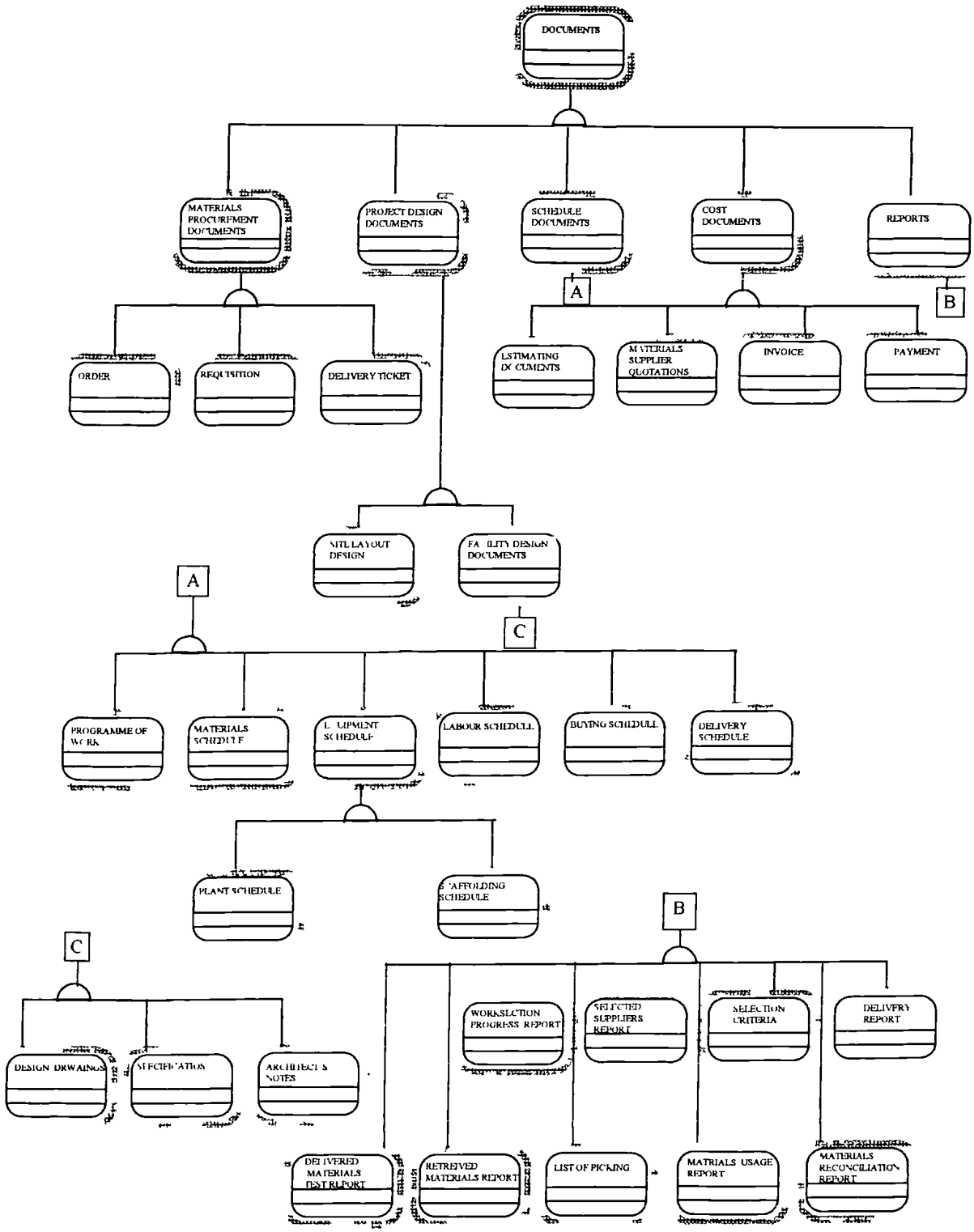
A.6. ICMM's Object Structures



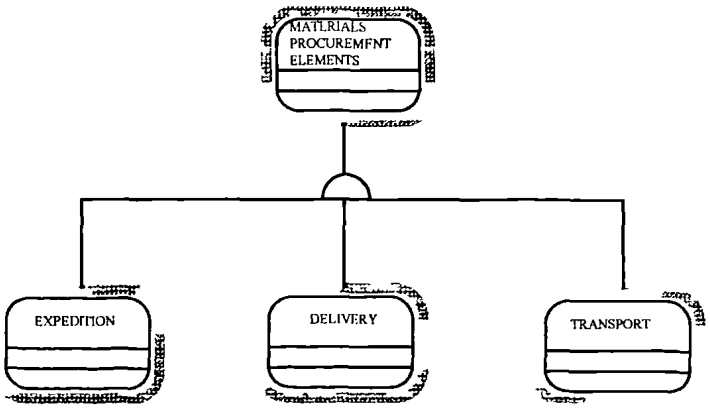
Resources structure layer



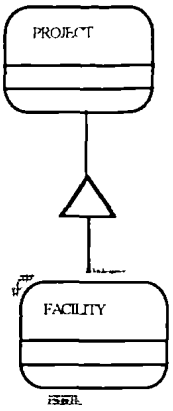
Agents structure layer



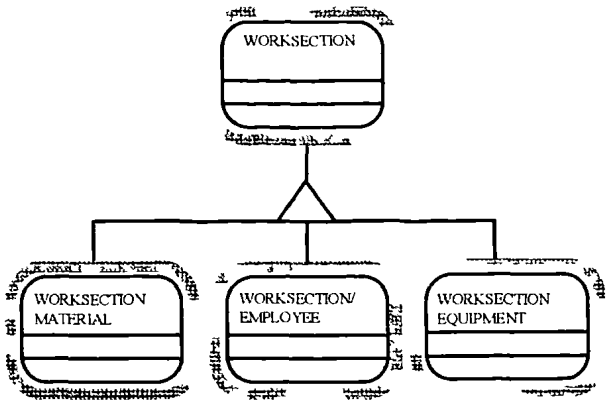
Documents structure layer



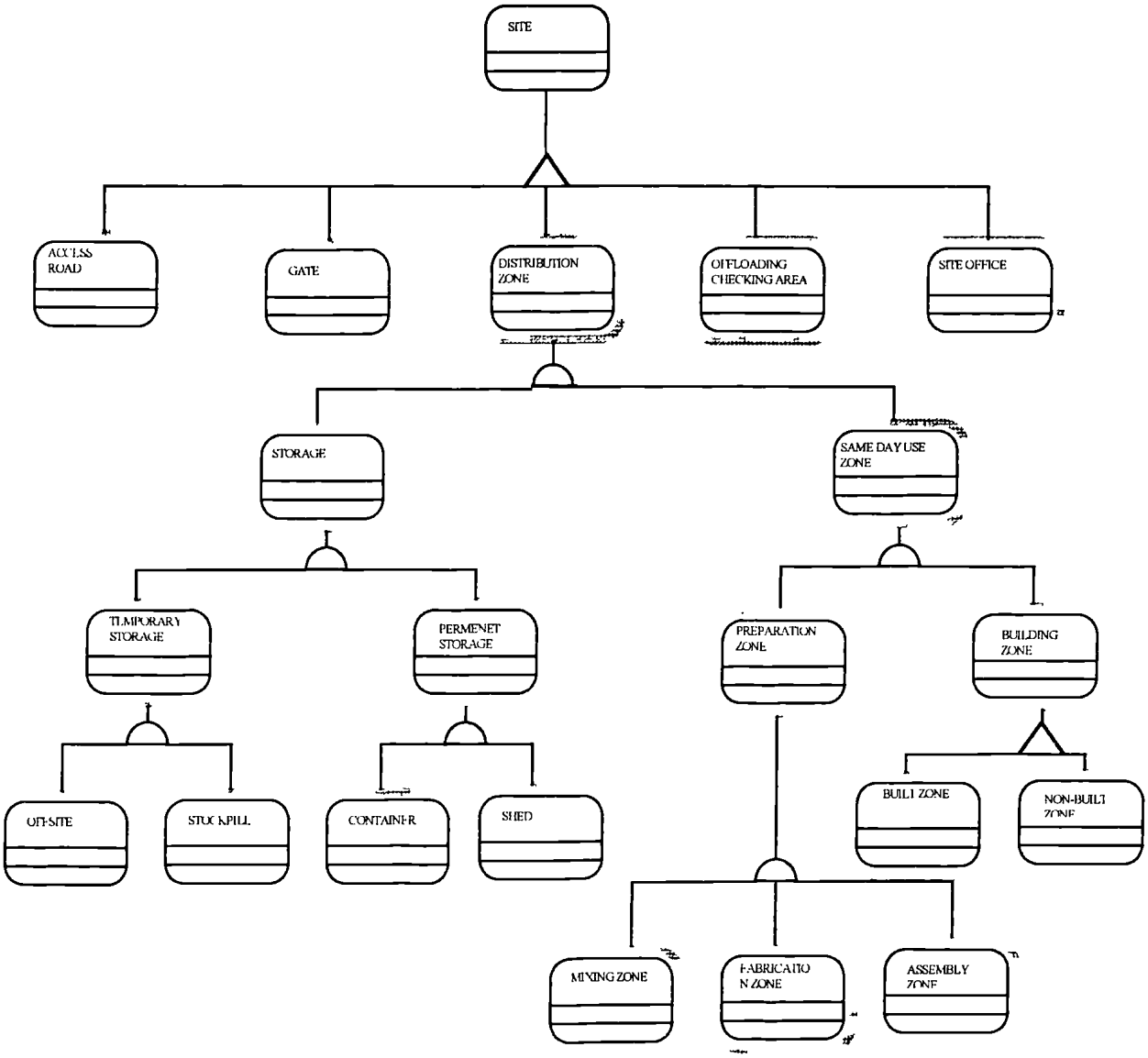
Materials procurement elements structure layer



ICMM's Project's Whole-Part structure

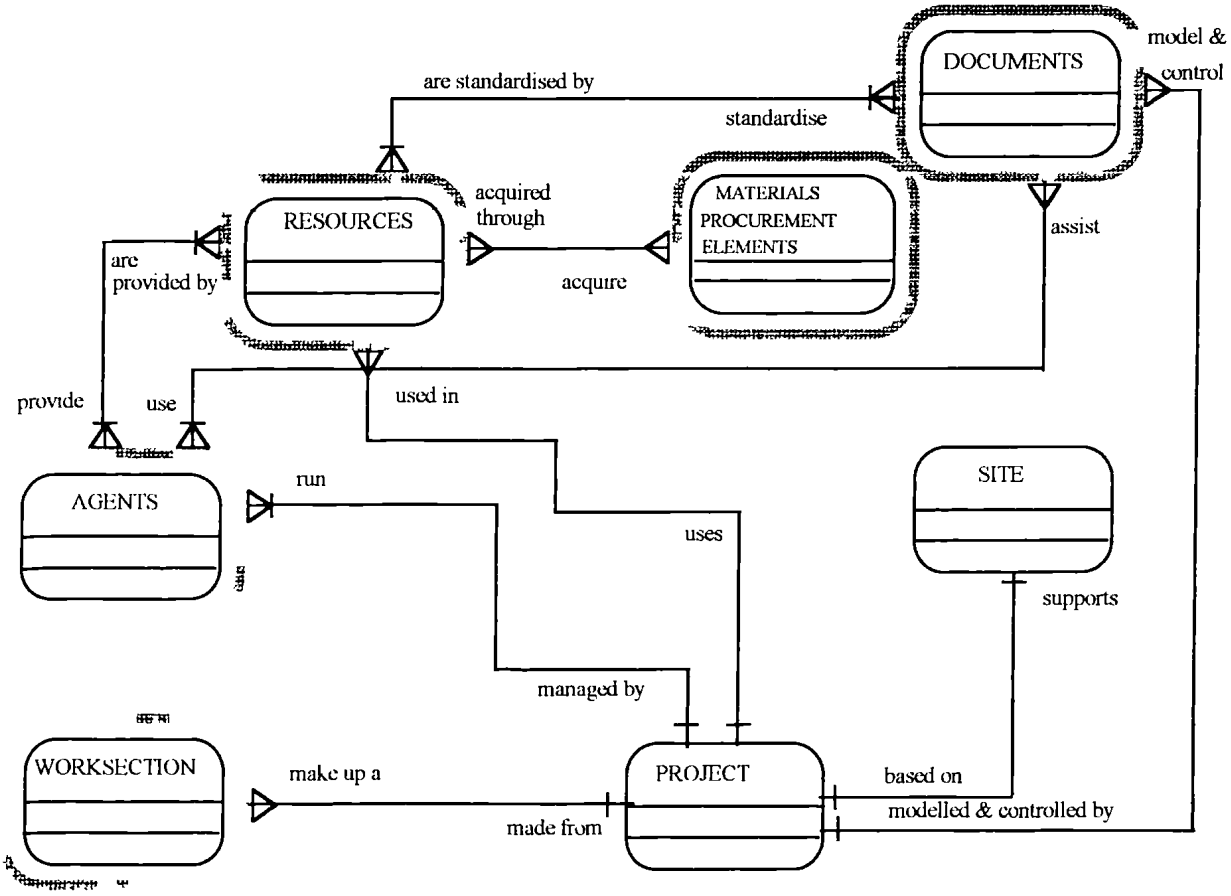


Worksection class and its parts



ICMM's Site's Whole-Part structure

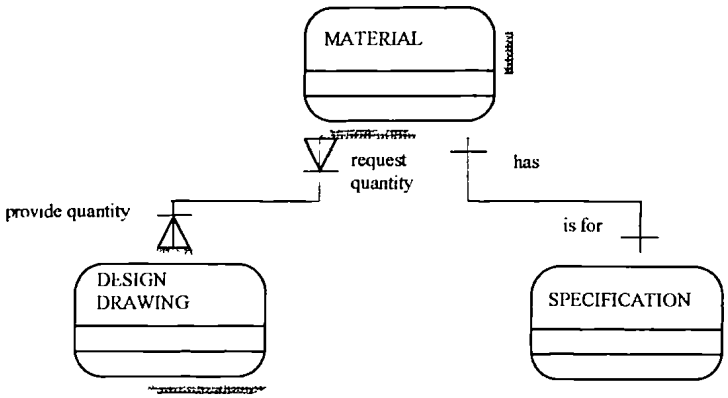
A.7. ICMM's Context Model, Subsystems Responsibilities, And Subsystems' Object Models



ICMM's object context model

1. Materials takeoff

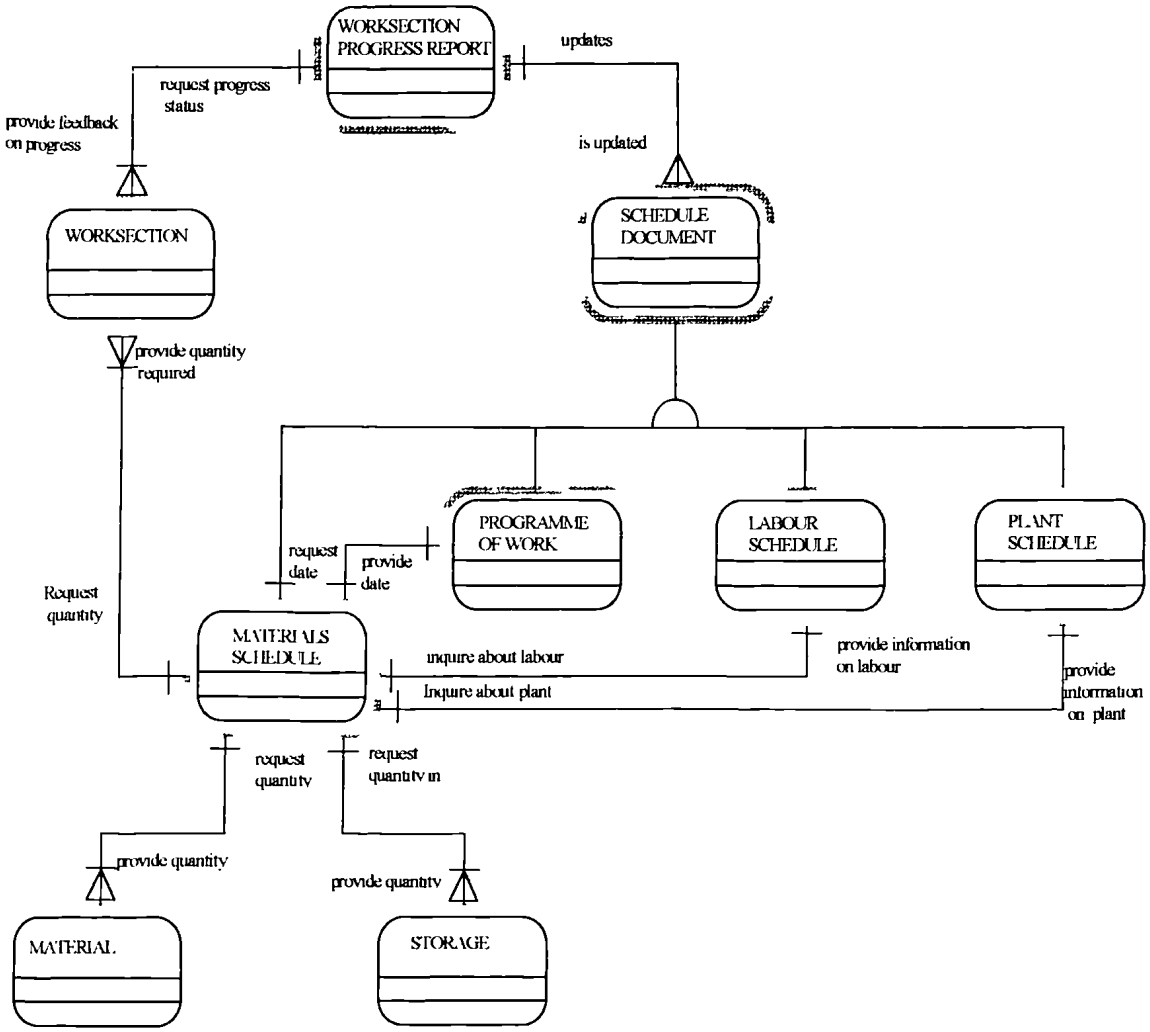
- Determine the required quantities of materials required with their corresponding specifications
- Quantities can be defined as the total required quantities for the project or per worksection.



Materials takeoff object model

2. Materials scheduling

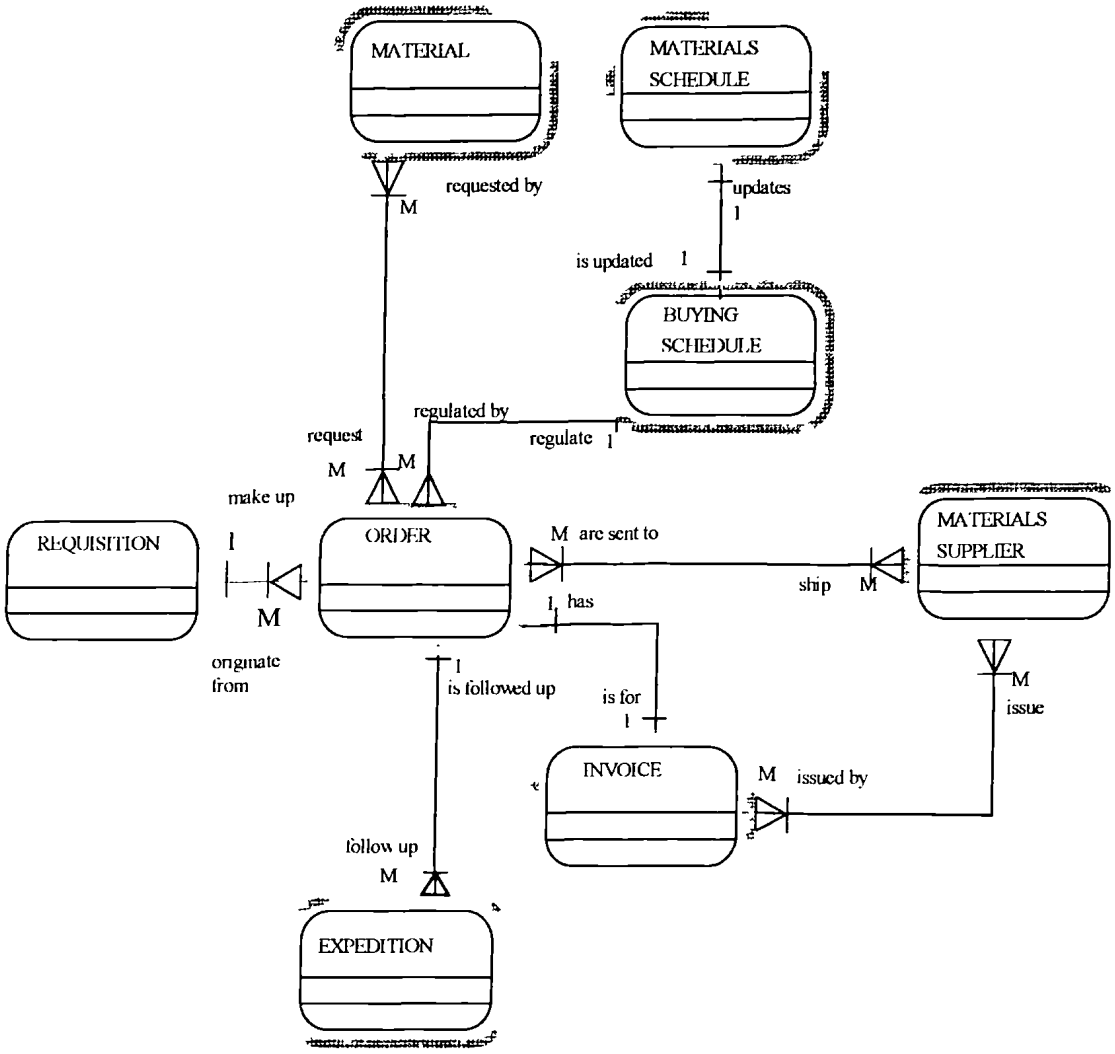
- Determine when, where and how much materials are required for the whole project, per day and per worksection.



Materials scheduling object model

3. Materials Ordering

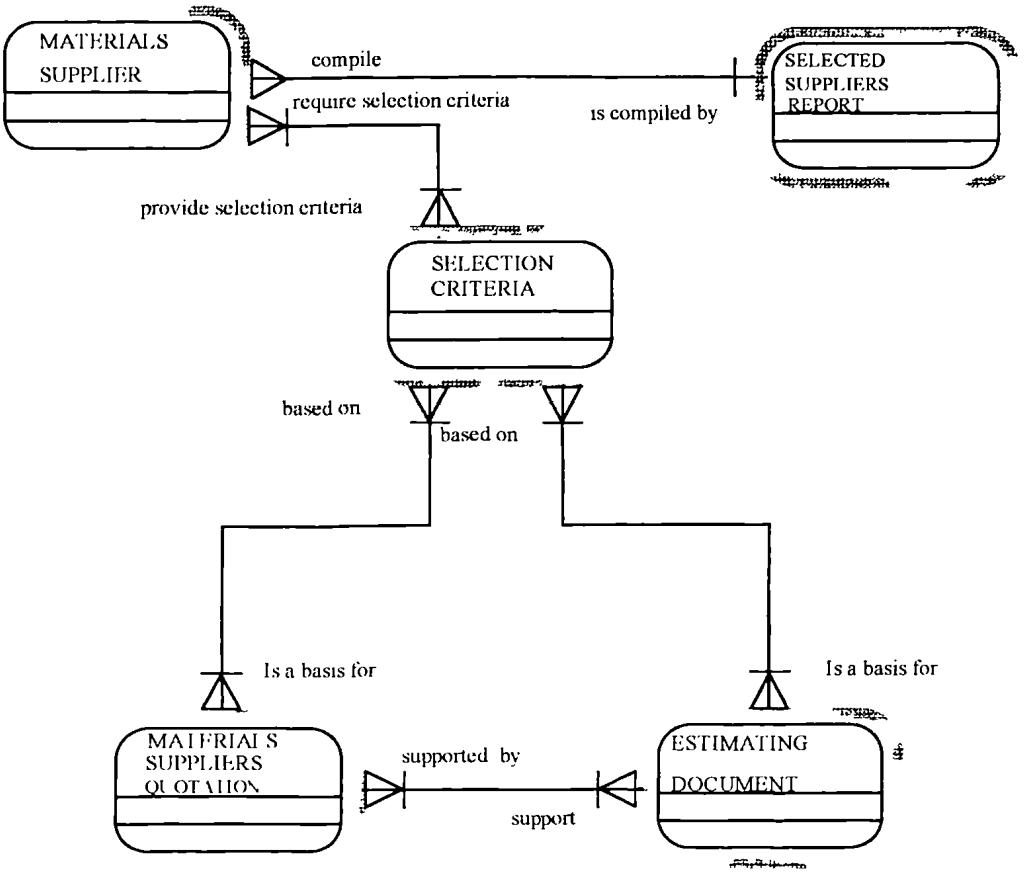
- Holds information on all required orders.
- Follows up order expeditions and maintains expeditions status.
- Produce order status.
- Manage requisitions
- Transform requisitions into orders
- Follows up orders invoice and maintains order invoices.
- Update buying schedule.
- Holds information on orders that have been sent out.
- Holds information on received orders and the materials in the orders.
- Holds information on pendent orders.



Materials ordering objet model

4. Supplier selection

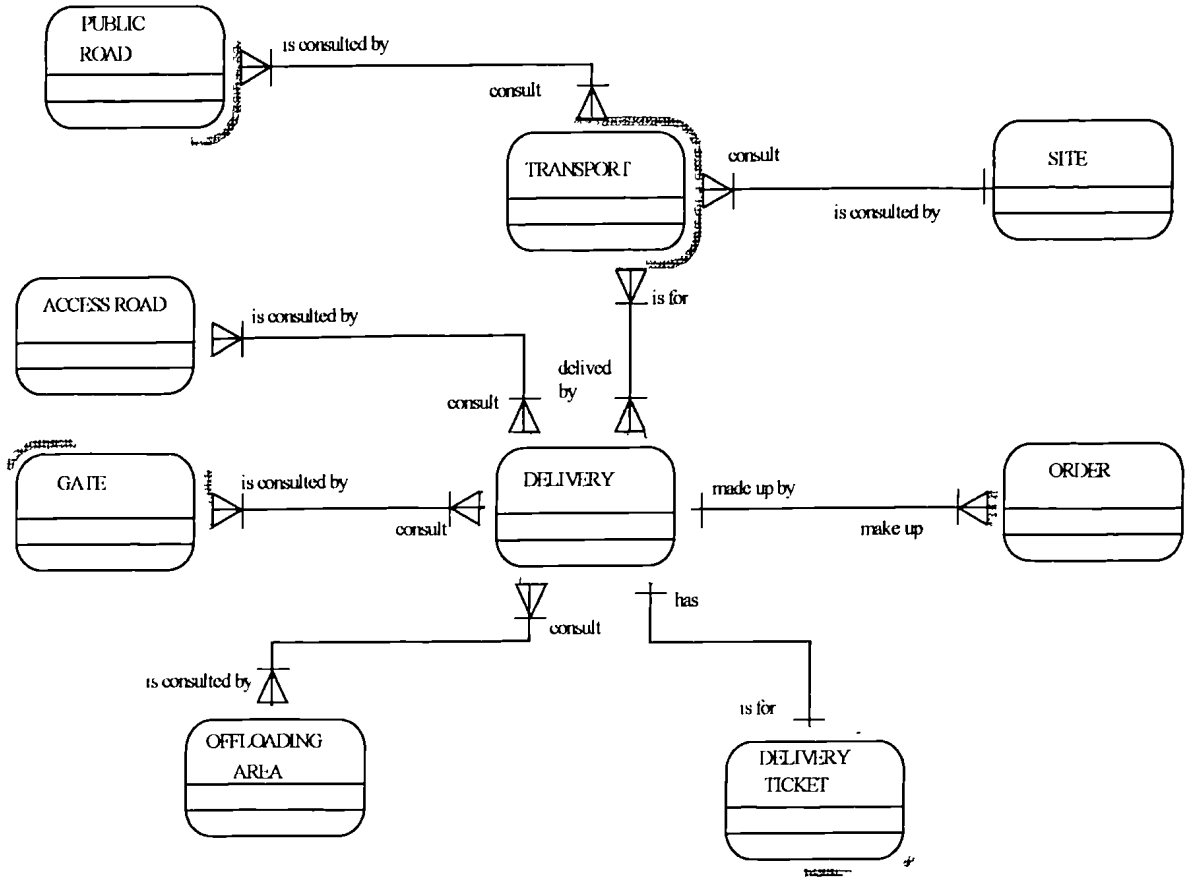
- Hold a master list of suppliers with all relevant information about them.
- Hold the company's suppliers selection criteria, which may change from a project to another.
- Produces report / list on selected suppliers for a particular project.
- Hold information on previous materials prices
- Hold information on received quotations for projects.
- Maintain quotation and estimating data



Supplier selection object model

5. Materials delivery

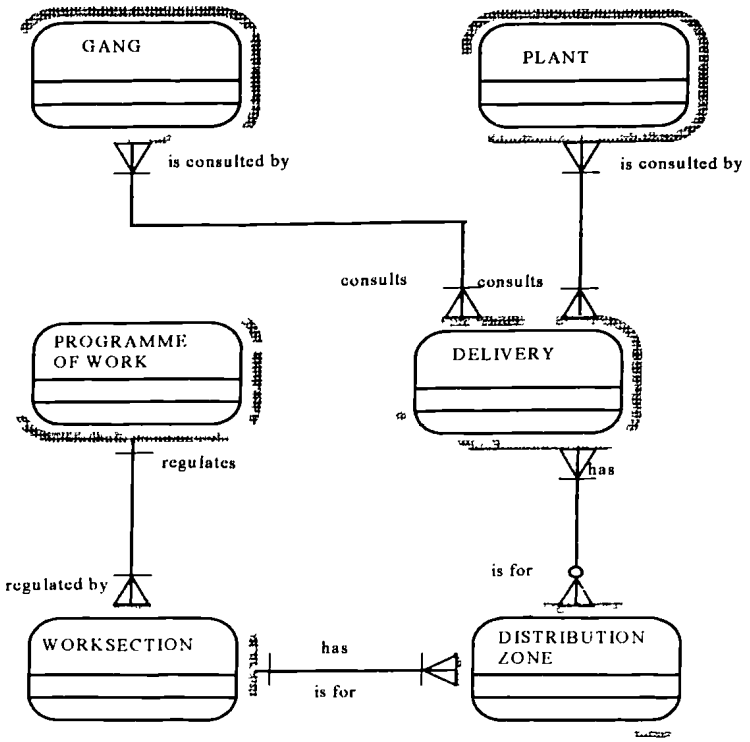
- Hold information on deliveries per day / gate / time.
- Check size of access roads, gates and highlights problems with delivery transport.
- Hold information on possibilities of offloading materials on site. This subsystem can be supported by an expert system for site layout design
- Hold information on the status of delivered materials on quantity and quality.
- Hold information on if materials arrived in the required packaging and the required unit load.
- Hold information on arrival time.
- Update materials suppliers performance.



Materials delivery object model

6. Materials delivery distribution

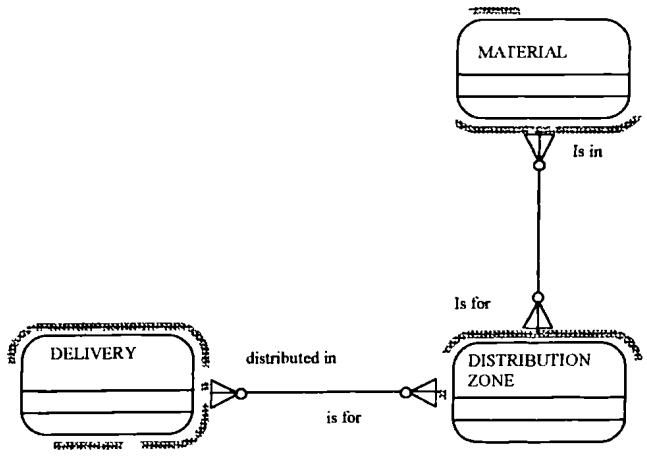
• Hold information on where delivered materials should be distributed on site.(may depend on worksection location, program of work, labour , plant, site...etc.)An expert system may be developed for this subsystem.



Materials delivery distribution object model

7. Materials storage

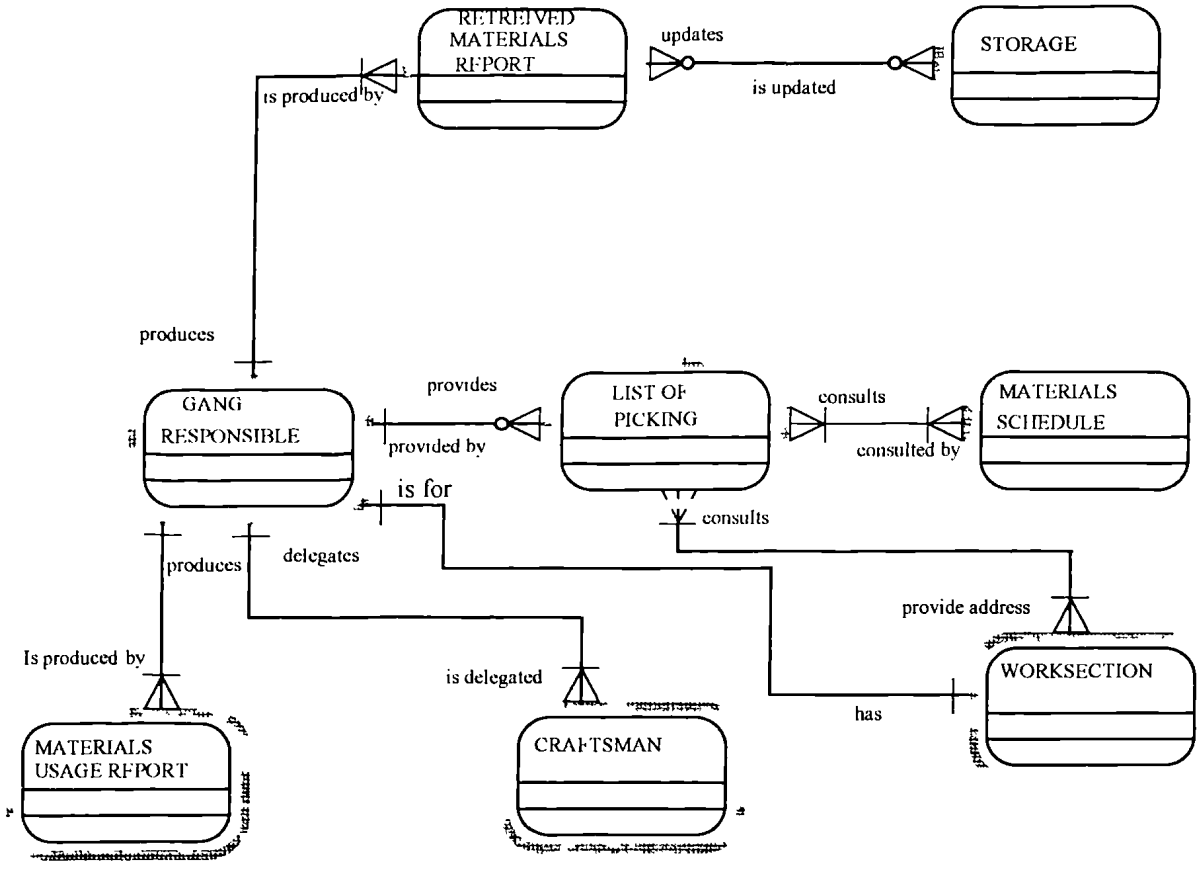
- Holds information on where materials are stored.
- Holds information on how much is stored in all and each storage.
- Updates the buying schedule.
- Holds information on how much room is left or occupied in all and each storage.
- Holds information on its physical condition and on the suitable materials to be stored in it
- Holds information on how close it is to the building zone.



Materials storage object model

8. Materials retrieving and installing

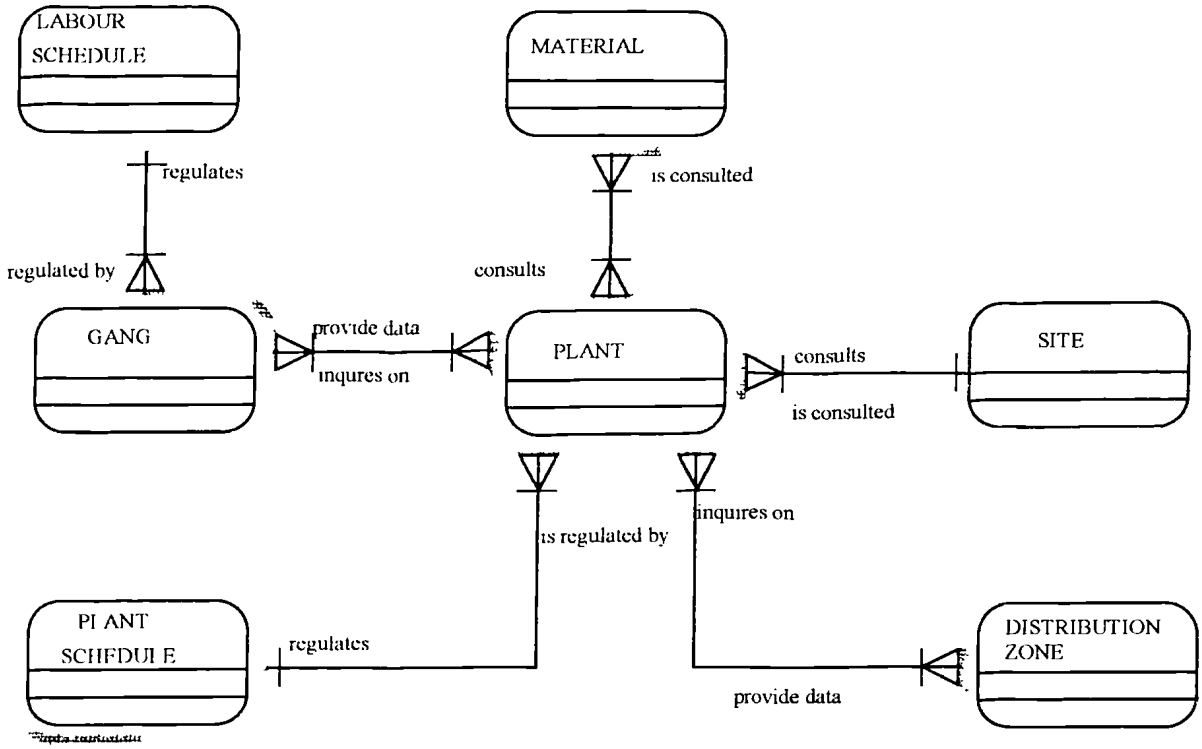
- Holds information on what materials are to be retrieved / worksection / day.
- Updates storage.
- Holds information on who retrieved what, how much and when per worksection / day.
- Holds information on materials that have been retrieved from storage to be used immediately on site or to be transferred to another location on site or to another site.
- Produces report on actual amount of materials that have been retrieved / day or as required.
- Holds information on plant or method to use to handle materials from the storage area to where they are needed for incorporation in the building work. This depends on many variables such as site obstructions, materials packaging, size, unit load, vulnerability, value. This subsystem could be supported or developed into an expert system.



Materials retrieving and installing object model

10. Materials handling

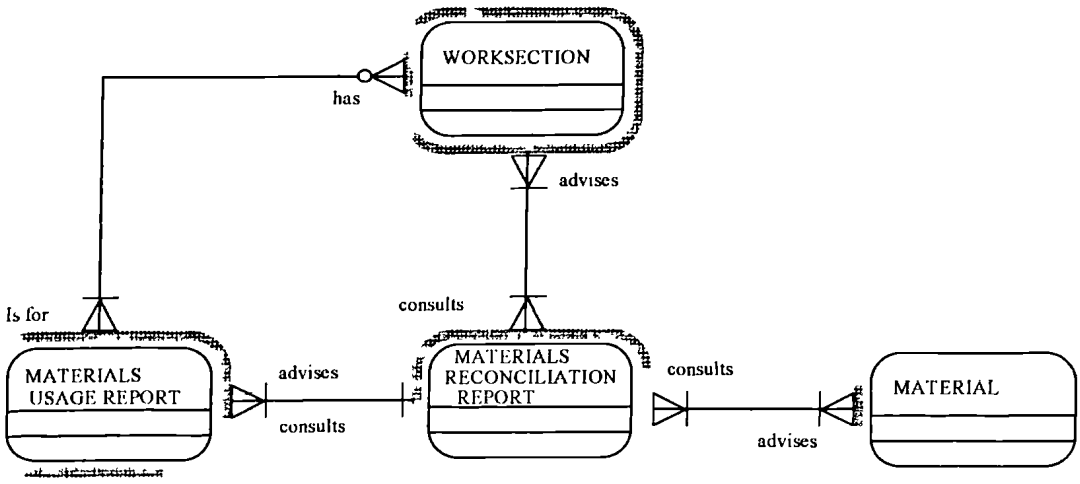
- Hold information on plant and methods of offloading materials from transport when delivered, and information and methods of onward handling of materials from storage to building zones. It will take into account materials, site obstructions, materials packaging, size, unit load, way of handling, vulnerability, value...etc. This subsystem can be supported with an expert system for materials handling.



Materials handling object model

10. Materials reconciliation

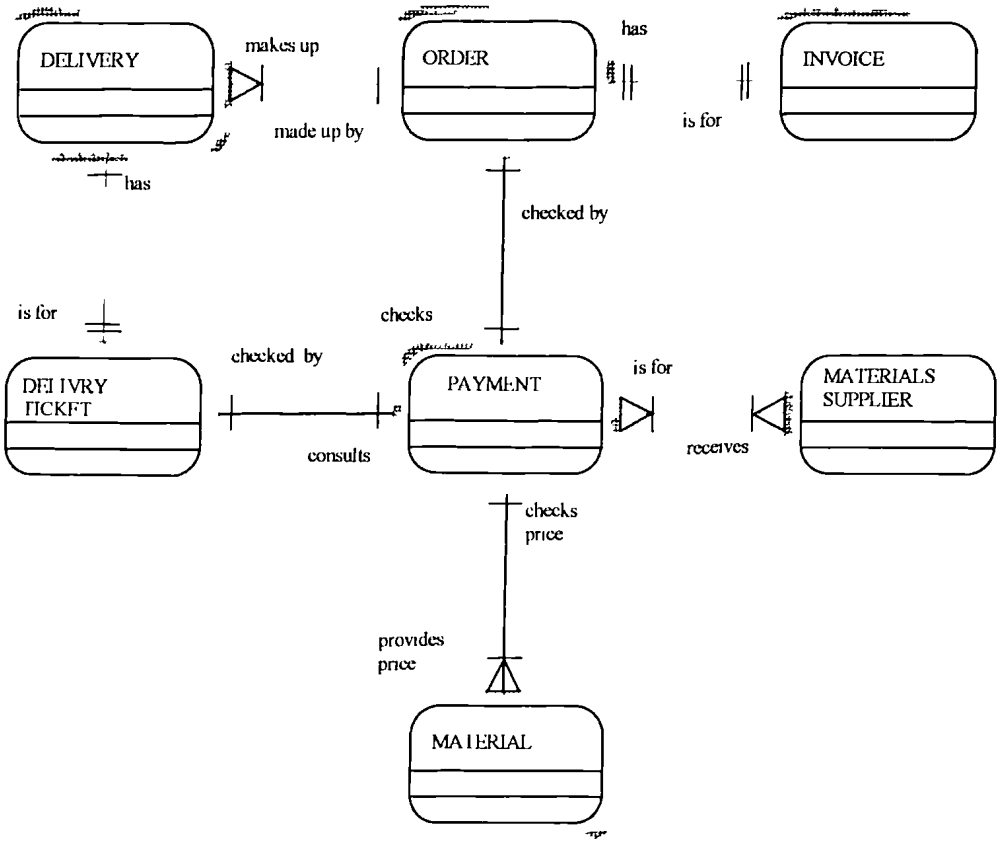
- Track down inconsistencies in the use of materials. Highlight how much materials has been used, wasted or has disappeared. It will also be able to track down the reasons and the responsible for materials wastage. (It will be linked to the class worksection, in which information about gangs, gang responsible and craftsmen are recorded).



Materials reconciliation object model

11. Payment

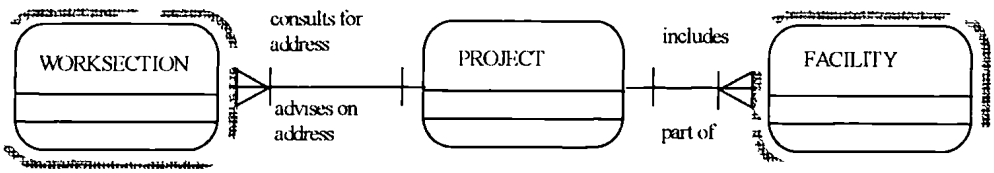
- Hold information on interim accounts (designer nominated supplier)
- Hold information on materials that have been paid or partly paid for.
- Hold information on claims and payments problems.
- Hold invoices
- Issue payments



Payment object model

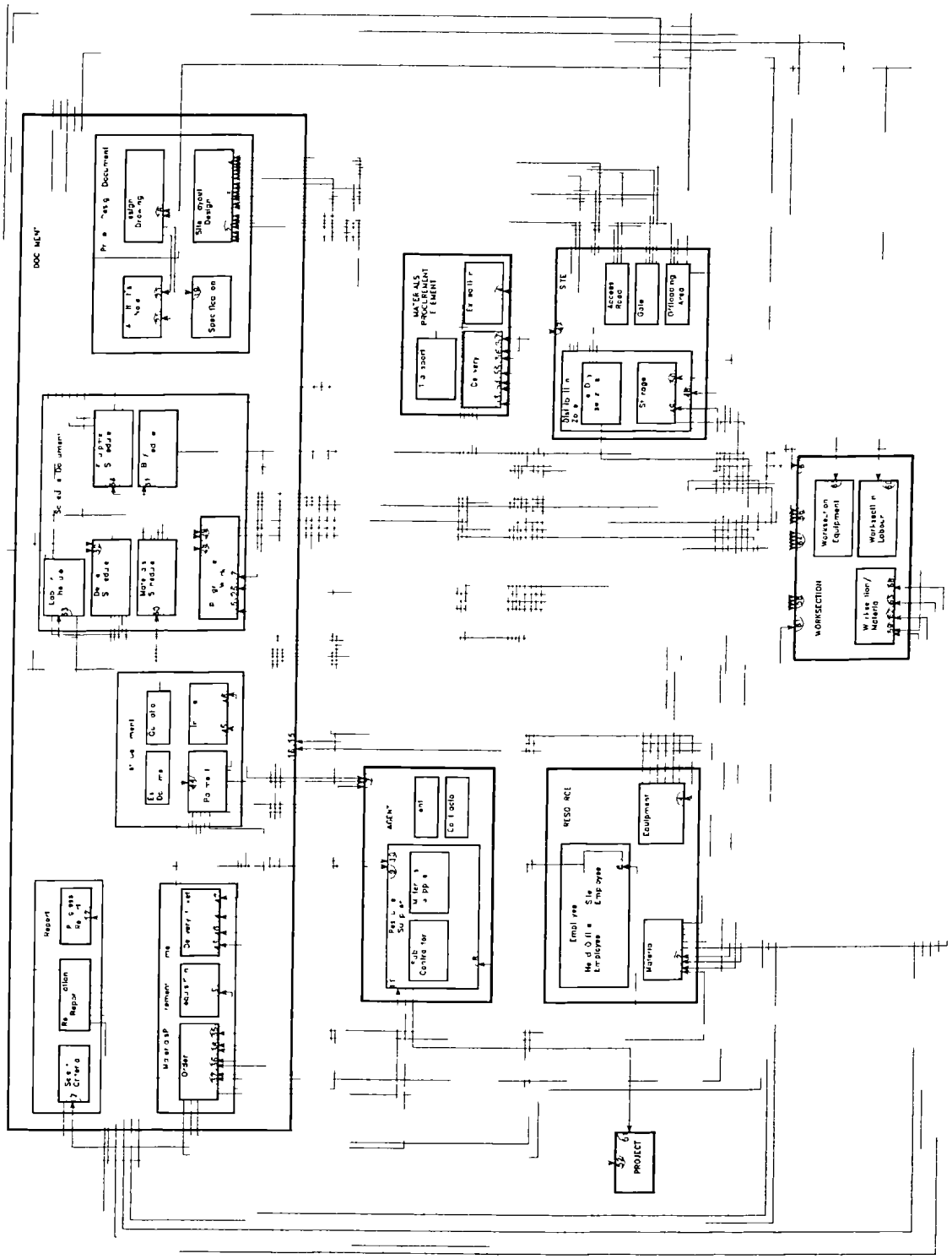
12. Construction

- Hold information on where construction work is to be performed on the designed project, through a link to the site layout design and a building product model.
- Give information on where to stock materials once work is inside the building (plastering internal walls), or how to get materials inside the building through the use of windows doors, stairs or service space ...etc.



Construction object model

A.8. ICMM's Collaboration Graph



ICMM's collaboration graph

B: ICMM-Model Implementation

B.1. Requirements Specification

The "Requisitions And Purchase Orders Management System" is designed to keep the user informed of project's requisitions and the progress of all current purchase orders. The system requires the integration between purchasing and planning activities. The objectives of the system are to ensure that ordered materials are delivered when and where needed, highlighting any problems that may arise, and informing the parties involved with any changes in delivery.
The system consists of two major subsystems:

1. Pre-Order subsystem

This subsystem's responsibility is to assist in tracking down the major milestones of materials orders, through materials schedule, and programme of work continuous monitoring. The subsystem holds information on how materials are to be delivered. This includes information on packaging, unit load, date, address, time of delivery, unit of measurement, specifications, and suppliers' names. The subsystem enables users to input materials requisitions and purchase orders. It also allows them to extract a list of orders that are to be delivered given a number of days. A link between the programme of work, and requisitions and purchase orders databases is a requisite for the success of the subsystem.

2. Post-Order subsystem

This subsystem is responsible for monitoring the progress of purchase orders and their delivery. It also reflects the latest information on orders' expeditions, highlighting any problems areas that might arise. This information is electronically distributed to the parties concerned, especially site managers, accountants, and purchasers to allow them to plan for corrective actions.

B.2. Requirements Model

The requirement model is developed to gain a better understanding of the system, and to analyse the requirements on it. To identify use cases, developers should look at each actor and investigate what that actor wants to do to the system. (Changing or inputting data into the systems must be done by authorises personnel only. However, data should be available to non-authorised personnel through read only files).

B.2.1. System's Actors

- Purchasing personnel
- Site managers
- Planners
- Accountants

Purchasing personnel

- Input new orders to the system
- Change existent orders in the system
- input requisition data into system before deciding on order strategy and call off schedule
- Input order details(specifications, quantity, packaging, transportation...)
- Update suppliers' performance file
- Update call off schedule if work on site is behind schedule
- Expedite orders and inform planners and site managers (to allow corrective actions..)
- Check claims for extra orders by site office

Planners

- Update programme of work if call off schedule is changed
- Inform purchasing personnel if any changes occur to the programme of work?

Site managers

- Produce delivery schedule (given conditions)
- Request order information
- Produce extra orders request
- Update daily / weekly programme of work

B.2.2. Use Cases

Use case 1: "Open prototype and select project"

1. User opens software prototype by clicking on its icon.
2. An introduction screen is presented to the user and prompts him/ her to login by typing his/ her username and password.
3. The "Project screen" appears with a single selection list box containing contractors projects list, catalogued by their name.
4. To select a project, the user double clicks on the project name.
5. The system loads materials and projects' data and programme of work from an external database and a project management environment respectively, into an object oriented development environment.
6. User may select a function from the menu bar as follows:

Items on menu bar	Selection
Data entry	Input requisition Input order
Transform	Requisition to order Change order Expedite order
View	Requisitions Orders Materials list Projects list
File	Save Exit Print
Communication	Send Phone Fax Email EDI

Alternative courses

1. User is not authorised to login
2. No data is loaded from external sources

Use case 2: Input a requisition into the system

1. User selects "Input requisition" from the "Data entry" menu on the menu bar.
2. The "Requisition screen" appears, with a selection box containing a list of potential materials that may be included in a requisition.
3. If the user chooses to change the list of materials to be included in a requisition, he/ she may click on the "Enter new requisition" button. The "Materials screen" is presented to the user, from which he / she may select a list of materials, and clicks on the "Change" button to move back to the "Requisition screen".
4. For each material selected in step 3, the user is required to following information.

- Quantity required
- Delivery date
- Type of packaging
- Specification code
- Supplier's name
- Unit load
- Unit of measurement
- Worksections involved

5. A material may be included several times in a requisition, if each time it is required it has a different specification code.
6. User clicks on the "Ok" button to close the data form.
7. To add the next material in the requisition, the material name is double clicked, and steps 4, 5 and 6 are repeated, until all the required material for that particular requisition have had their data specified.
8. The user may do the following:
   - Preview a requisition. This must be done before its is closed.
   - Close a requisition, which will give it a unique code number.
   - Cancel a requisition.
9. For reconciliation purposes, the user may select the reason behind raising a requisition from the following list:
   - Bills of quantities
   - Materials loss
   - Designer request
   - Site request
   - Drawings

Alternative courses:

1. A project has not been selected.
2. There are no material in the materials list.

Empty requisitions can neither be closed nor cancelled nor previewed.

Use case 3: Transform a requisition into order(s)

1. User selects "RequisitionToOrder" form " Transform" form the menu bar.
2. The "Requisition screen" is presented to the user, with a single list selection box containing the projects' requisitions list.
3. The user selects a requisition that he / she intends to transform into an order, by clicking on it once.
4. The user clicks the "Requisition to order?" button.
5. The system pop up a message asking the user to confirm the transformation.
6. If the user decides to cancel the process, he / she may select the "Cancel" button, the "Ok " button is selected otherwise.
7. An order is created with a unique code number. Each material in a requisition may be assigned a supplier that is to be confirmed by the buying department. Assuming, suppliers names entered when creating requisitions is approved by the buyer, the system will create as many orders from a requisition as there are suppliers in it. If two materials are required from the same supplier, only one order is made for them. If a third material in the same requisition is required from a different supplier then an order is created for it
8. At this screen, the user may click the "Back to previous screen " if he/ she wishes to open the "Requisition screen".

Alternative courses:

1. No project has been selected.
2. There are no requisitions for the selected project.
3. The selected requisition has already been transformed into an order.

Use case 4: Change an existent order

1. User selects "Change order" from "Transform" in the menu bar.
2. A window appears below the window and contains the project's list of orders.
3. The user may click on the order he / she wishes to change, and then click on the "Change selected order" button.
4. The system invites the user to enter the section number of the order he / she wishes to change.
5. At this stage only two attributes may be altered, and they are:
   - Quantity
   - Date of delivery
6. Once the new data is entered, the user clicks on the "Change" button.
7. The system will ask the user if the order's amendment number should be increased by one. If the user selects the "No" button, the order is closed. If he / she selects the 'Yes' button then the system check if the order has been sent to the supplier. If the order has not been sent to the supplier then the amendment number is not changed, otherwise it is increased by one.

Alternative courses:

1. No project has been selected.
2. There are no orders in the selected project.
3. User cancels process.
4. Order's new data is similar to the original one.

Use case 5: "Expedite order and communicate information"

1. The user selects "Expedite order" from "Transform" in the menu bar.
2. The "Expedition screen" is presented to the user, containing the list of the selected project's orders. Next to this list is a "radio button" selection box, with the following expedition options:
   - Not defined

- As planned
- Delayed
- Cancelled

3. An order may be selected by clicking on its name. The supplier of the selected order may be contacted by the user through a phone modem, and having clicked on the "Phone supplier" button. Information collected from the supplier allows the user to categorise the order's expedition status as specified in step 2.
4. In the case where an order is delayed or cancelled, a message is automatically sent to the parties involved to allow them to plan for corrective actions.
5. While an order is selected from the selected project's list of orders, the worksections that are dependent on that particular order are extracted from the project management environment and listed in a single selection list box.
6. The user may also extract the orders that are required to be delivered in a given number of days. He / she, enter a positive number in specially designed field, and click on the "Get the orders required within ...days" button. The corresponding orders are listed in a single list box, below the latter button.

Alternative courses:

1. No project has been selected.
2. There are no orders for the selected project.
3. Order has already been expedited.

Use case 6: "View orders"

1. The user selects the "Orders" from the "View" menu in the menu bar.
2. The "Order's data" screen is presented to the user, with a large display area in which order's data are displayed. Next to it is single list box, containing the selected projects' orders list.
3. To view an order's data, the order's name must by double clicked by the user.

4. The system pops up a message to the user informing him / her that data about the selected order's supplier details are being loaded from the external database.
5. Once suppliers data are loaded, the following appears on the display area:

- Project name
- Date order is made
- Project number
- From requisition number
- Order number
- Amendment number
- Supplier's name
- Number and street name
- Contact name
- Phone
- Fax
- Email address
- EDI address

The above is followed by a number of sections. Each section holds data about a particular material.

Alternative courses:

1. There are no orders to be viewed.
2. No project has been selected.

Use case 7: "View requisitions"

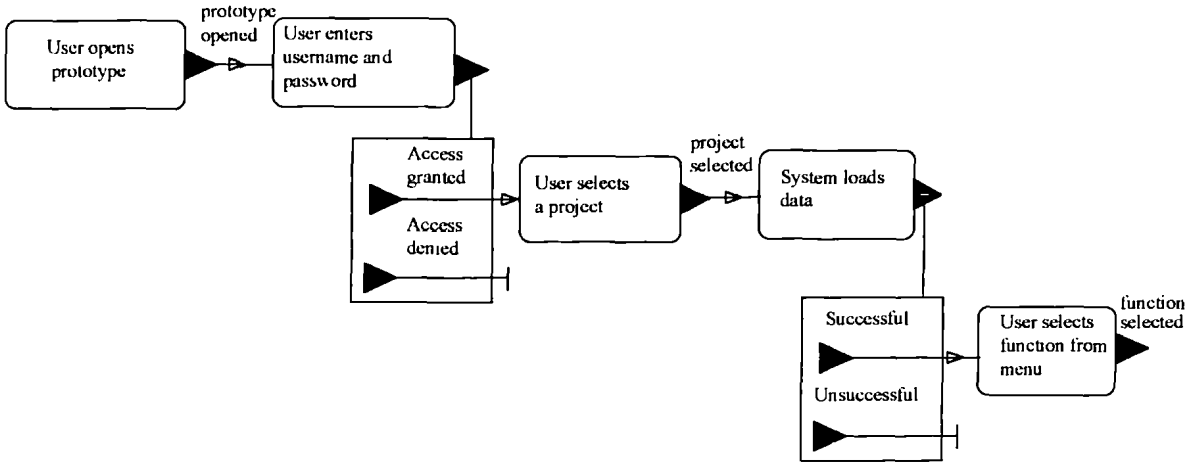
1. The user selects the "Requisitions" from the "View" menu in the menu bar.

2. The "Requisition's data" screen is presented to the user, with a large display area in which requisition's data are displayed. Next to it, is a single list box, containing the selected projects' requisitions list.
3. To view a requisition's data, the requisition's name must by double clicked by the user, and the data appears immediately in the display area.
4. The system display the following data in the display area:
    - Material name
    - Quantity required
    - Delivery date
    - Type of packaging
    - Specification code
    - Supplier's name
    - Unit load
    - Unit of measurement
    - Worksections involved
    - Project number
    - Requisition number
    - Requested by
    - Requested against
    - Date requisition is made

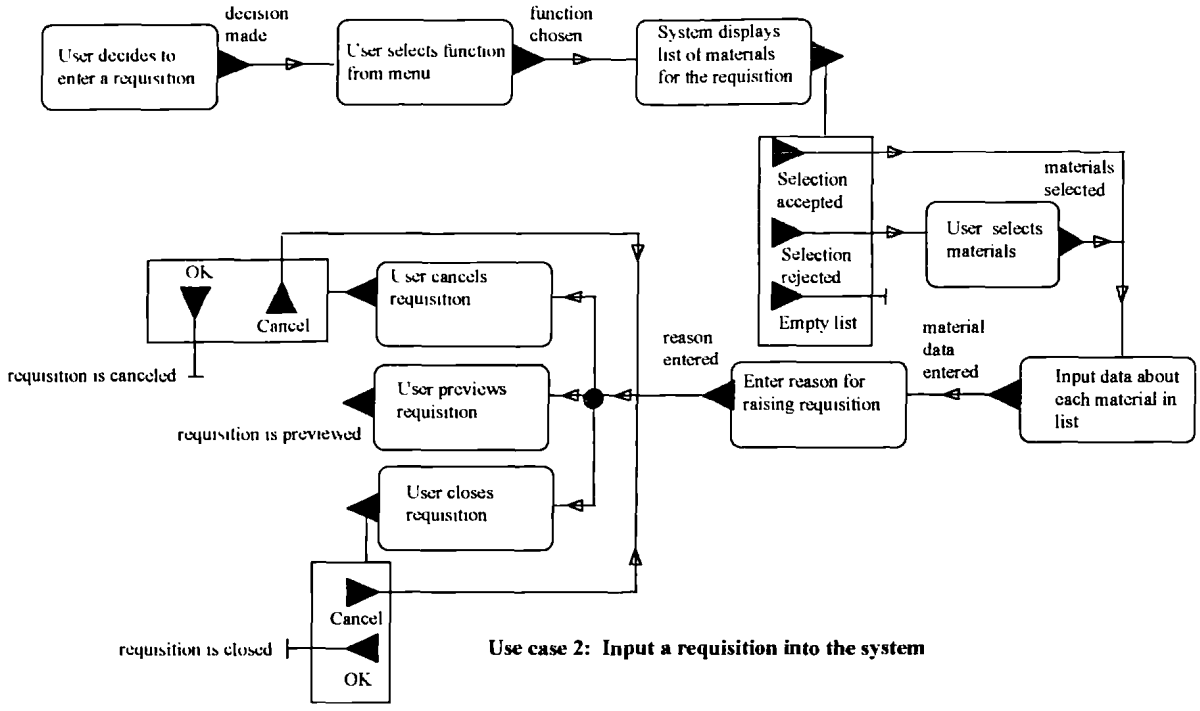
Alternative courses:

1. There are no requisitions to be viewed.
2. No project has been selected

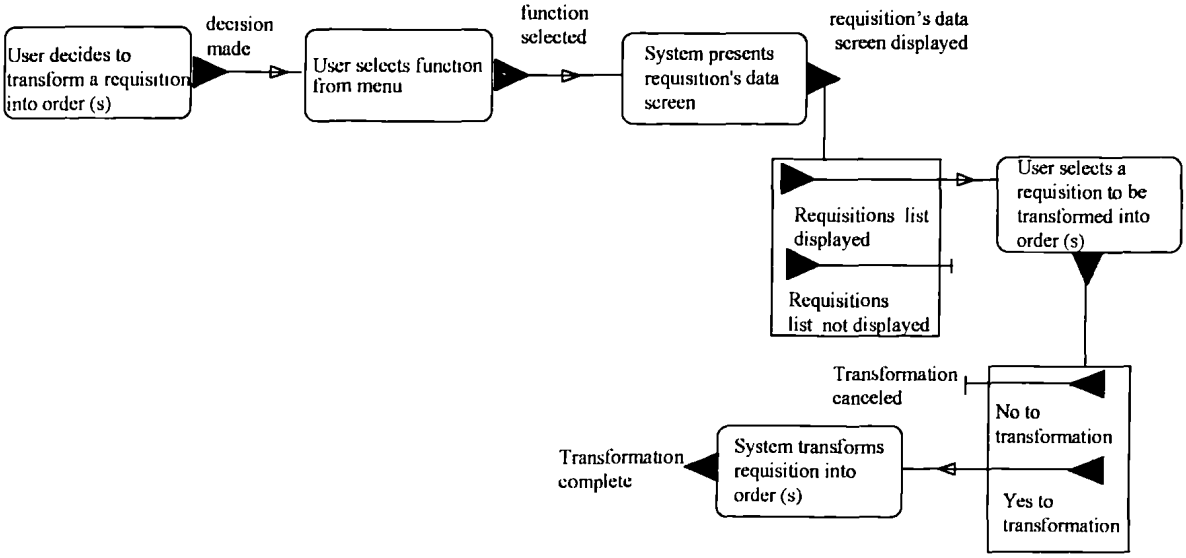
B.2.3. Use Cases Event Models



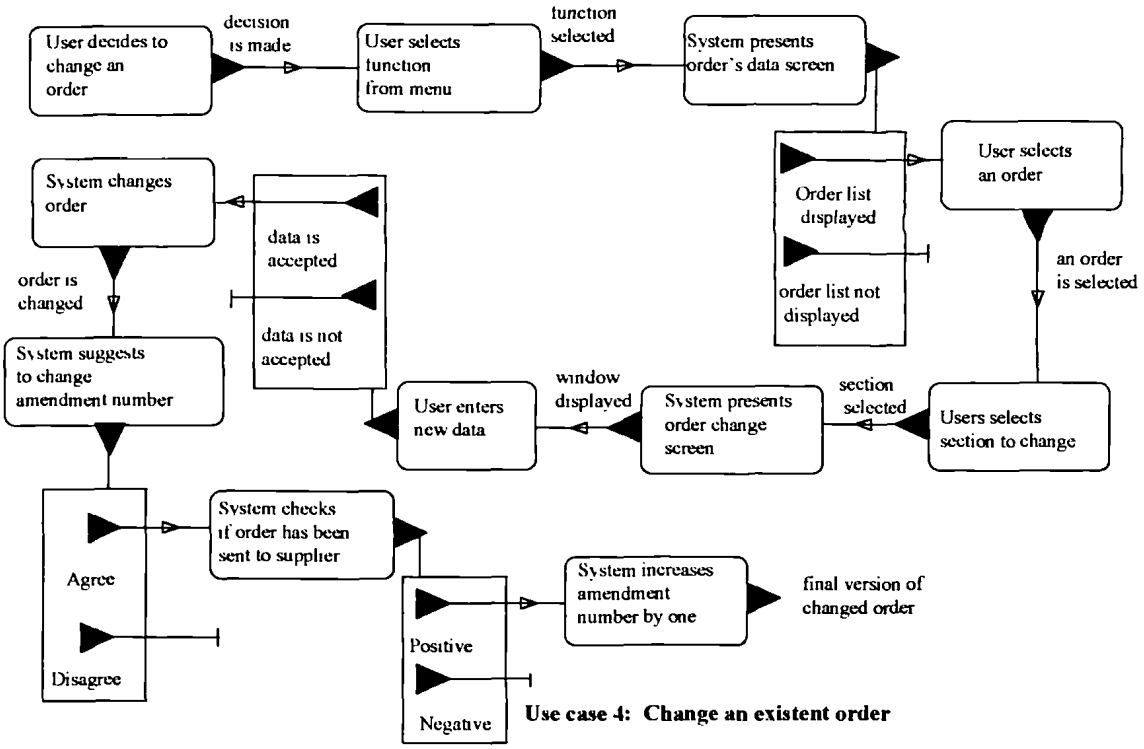
Use case 1: Open prototype and select a project



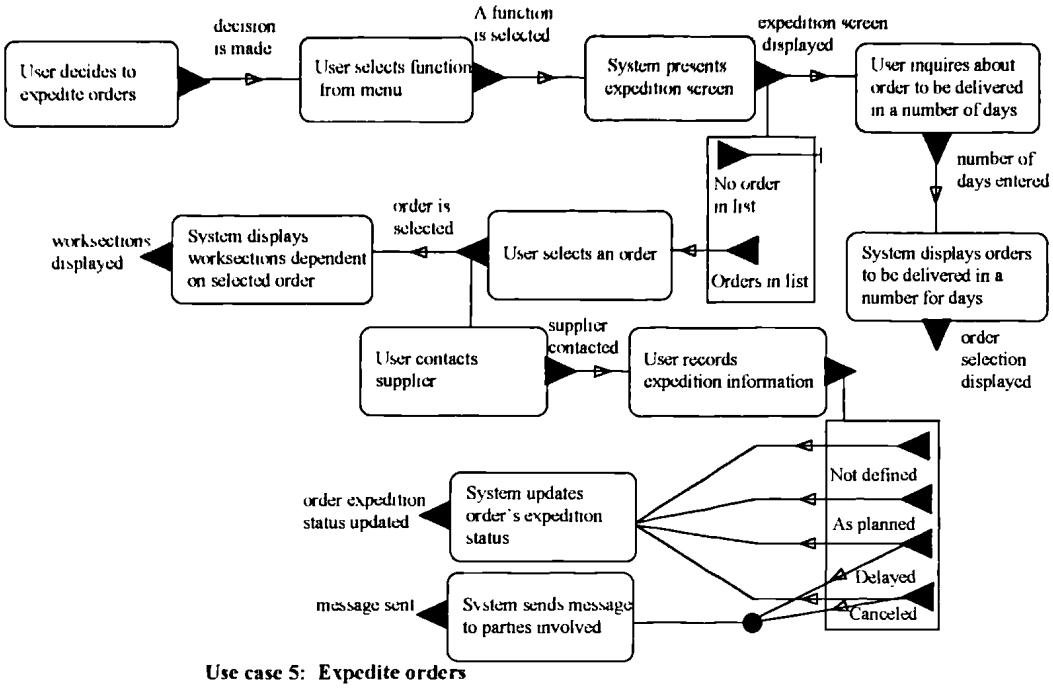
Use case 2: Input a requisition into the system



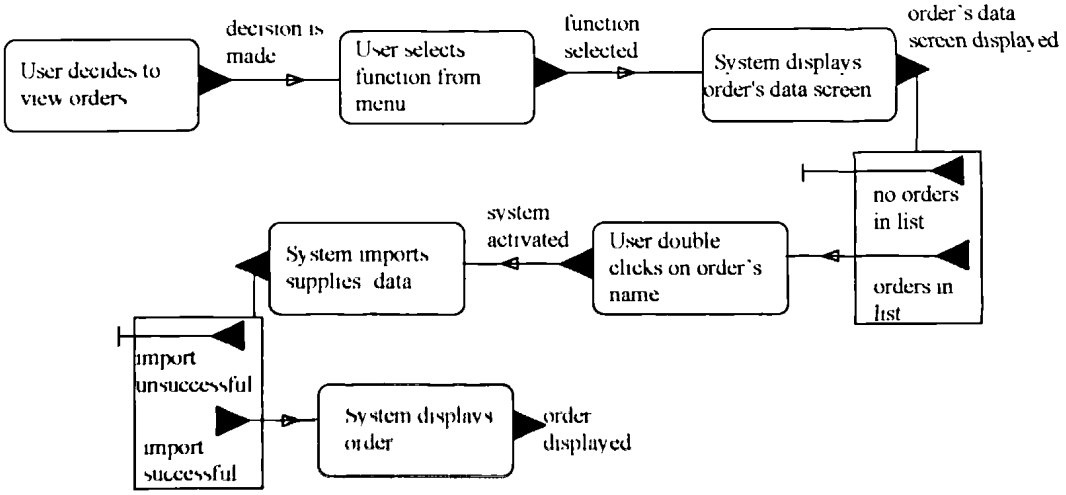
Use case 3: Transform a requisition into order (s)



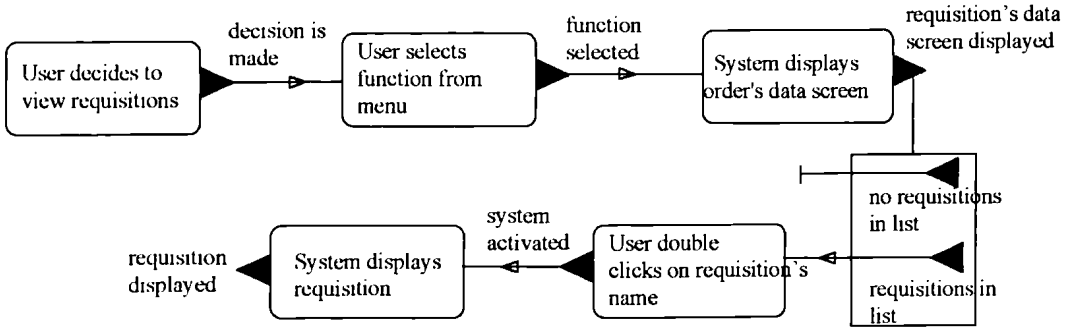
Use case 4: Change an existent order



Use case 5: Expedite orders



Use case 6: View orders



Use case 7: View requisitions

B.3. Classes, Their Reviewed Responsibilities, Contracts and Collaborations

Classes	Collaborative classes

Class: **Materials**
Superclass: Resource
Type of object: Abstract
Subclasses:

Private responsibilities:

Know how much is in storage	Storage (49,50)
Know how much is required for project	Design Drawings(18) Documents (15,16)
Update materials specification in requisitions	
Know if it is a formed material	
Interact, commit and accept changes in database	

Public responsibilities:

1. Has knowledge of its description
   - Has knowledge of how it should be handled
   - Has knowledge of how it should be packaged
   - Has knowledge of how it should be stored
   - Has knowledge of its price

Class: **Order**
Superclass: Materials Procurement Documents
Type of Object: Concrete
Subclasses:

Private responsibilities:

Transform a requisition into order (s) and make instances of itself	
Read itself	
Prompt user to change it	
Check changed data	
Send itself to supplier (Technology dependent)	
Respond to enquires about its delivery date	
Prompt user to expedite it	
Know its status and produce report on orders problems	Delivery (57)
Produce report on placed orders	
Know the date of order and of delivery	
Produce report on changed orders	
Know how to print itself	
Know what is being ordered	Requisition (12) Buying schedule (31)
Know materials supplier	Materials supplier (11) Agents (7)
Know if it has been paid for	Payment (44)
Know worksections linked to it	Worksection (58)
Make instances of itself	
Know expedition date	

Public responsibilities:

35. Know about its data

36. Update order database
37. Know its delivery arrangement (time, quantity, quality, packaging, unit load, unit of measurement)
38. Know its code number
39. Order maintenance
    Produce report on order status
    Hold information about what was delivered
    Knows about pendent orders
69. Hold list of projects' orders

Class: **Requisitions**
Superclass: Materials Procurement Document
Type of Object: Concrete
Subclasses:
Private responsibilities:
Make instances of itself
Cancel itself
Prompt user to enter a new requisition's data
Write itself in an ASCII file
Close itself once data entered
Preview itself before it is closed
Read previous instances of itself
Public responsibilities:
12. Know what materials are in it
13. Provide data for use in orders
    Provide materials quantity
    Provide materials delivery date
    Provide materials type of packaging
    Provide materials specification
    Provide materials supplier (facultative)
    Provide materials unit load
    Provide materials unit of measurement
70. Hold list of projects' requisitions

Class: **Suppliers**
Superclass: Agent
Type of Object: Abstract
Subclasses: Material Supplier, Subcontractor

Private responsibilities:
Know in which project it is involved — Project (51)
Know its products
Interact with supplier database
Public responsibilities:
8. Know its criteria
9. Know its bank account number
10. Know its method of payment
11. Know its code number

Class: **Worksections**
Superclass: Materials forwarding element
Type of Object: Concrete
Subclasses:
Private responsibilities:

Know worksection's responsible
Know site employee working on it
Make instances of itself
Interact with programme of work in a project management environment

Public responsibilities:

58. Know its code number
59. Know materials needed, in quantity and quality
60. Know skills needed
61. Produce progress report
62. Know unused and unusable materials
63. Know quantity of materials used
64. Know its type of work
65. Know needed equipment
66. Responsible for its time control
    Know preceding and afterward worksection
    Know start and end date
    Update programme of work
67. Know its location
68. Update material database
71. Hold list of projects' worksections

Class: **Projects**
Superclass:
Type of Object: Concrete
Subclasses:
Private responsibilities:
Interact with CAD drawing database

Get its orders list	Order (69)
Get its requisitions list	Requisition (70)
Get its worksections list	Worksection (71)

Public responsibilities:

51. Interact with project database
52. Hold materials suppliers selection criteria and selected project's suppliers list

NB:
Grey responsibilities text: responsibilities that are not relevant to this subsystem
Grey collaborative classes: classes that are not needed for this subsystem
Arial: New classes' responsibilities, and new collaborations

```
Select a requisition to be transformed into order (s)
    IF the requisition has not yet been transformed into order(s)
            Mark it as "Already transformed into order"
    ELSE
            Print screen message: "Sorry requisition has already been
            transformed into order(s)"
    IF "Dummy object" 's orders list >0
            Clear "Dummy object's orders list
    IF project's orders list >0 and <9
            Execute transformation version 1
    IF project's orders list >9 and <99
            Execute transformation version 2
    IF project's orders list >99 and <999
            Execute transformation version 3
Move newly made orders from "Dummy object" to the class "Order"
```

Transform requisition into order (s) 1st method's structure

```
Extract suppliers names into a list
    IF suppliers name list length = 1
            Create a new order code number
            Create an order as an instance of "Dummy object"
            Make a new instance of order

    ELSE
        Repeat for each supplier in the list
                Extract (Material name, quantity, delivery date,
                packaging, unit load, unit of measurement and
                specifications)
            Create a new order code number
            Create an order as an instance of "Dummy object"
            Make a new instance of order
```

Transform requisition into order (s) 2nd method's structure

For each order under "Dummy object"
Include supplier's name
Import suppliers details from the database
Import clauses from clauses database
Extract project details form the object "Project

Transform requisition into order (s) 3rd method's structure

B.4. Classes, Their Reviewed Attributes And Methods (Within Kappa-PC)

Notice

Kappa-PC does not accept spaces between letters, for this reason slot and method names are either made up by a number of words linked via a hyphen or a number of words each starting with a capital letter with no space in between.

Materials

Slots	Methods
Cost_Per_Unit	UpdateRequisitionSpecification (Updates a material's specification when it is being requested)
Material_Id_Name	
Material_Name	
Materials_Order_Lead_Time	Door_Detail (Prompts user to input more information about the Door being requested)
Materials_Packaging	
Material_Qty_In_Storage	
Material_Qty_Ordered	Door_Material (Prompts the user to input information about the material the door being requested is made from)
Material_Qty_Req	
Materrial_Qty_Used	
Material_Specif_Code	
Material_Substitute	
Material_Unit_Load	
Material_Vul_Weather	
Material_Waste_Allowance	
Nb_Pack_In_Unitload	
Plant_To_Use	
Taxes	
Unit_Measurement	

Suppliers

Slots	Methods
ContactName	GetSuppliers
EDIAddress	(Remote executes macros within Excel in
EMailAddress	order to search and retreive data about
Faxcimile	specific suppliers)
IDNumber	GetSuppliersDetails
ListProducts	(Remote gets suppliers' details from
Name	Excel)
PostalCode	
StreetName	
StreetNumber	
Telephone	
Town	

Projects

Slots	Methods
Client_Name	GetOrdersList
Finish_Date	(Groups orders that belong to a selected
OrdersList	project into a multiple slot in the instance
Project_Address	Project)
Project_ID	GetRequisitionsList
Project_Name	(Same as above but for requsitions)
RequisitionsList	GetWorksectionsList
Start_Date	(Same as above but for worksections)
WorksectionsList	UpdateRequisitionImage
	(Resets the graphical box holding the
	selected project's requisitions list)

Worksections

Slots	Methods
Activities_ID	GetSuperActivitiesID
Activity_ID	(Imports list of activities' code numbers
DDEItem	from CA-Superproject)
FinishDate	GetSuperFinishDate
MaterialsList	(Same as above but with finish dates)
Name	GetSuperNameList
ScheduleFileName	(Same as above but with worksection
StartDate	names)
SuperFinishDate	GetSuperStartDate
SuperNameList	(Same as above but with start dates)
SuperStartDate	LoadWorksectionList
WorksectionList	(Makes a list of allowable values for when
	selecting worksections when entering
	requisitions)
	MakeInstances
	(Makes instances of the class
	Worksections)

Requisitions

Slots	Methods
DateMade	InputMaterialsInformation
DeliveryDateRequested	(Prompts user to input information on
MareeialsListRequsted	requested materials)
PackagingRequestedList	UpdatePackagingRequested
QuantityRequested	(Updates list of packaging of the
ReferenceNumber	requisition being entered)
SpecificationRequested	UpdatesUppliersList

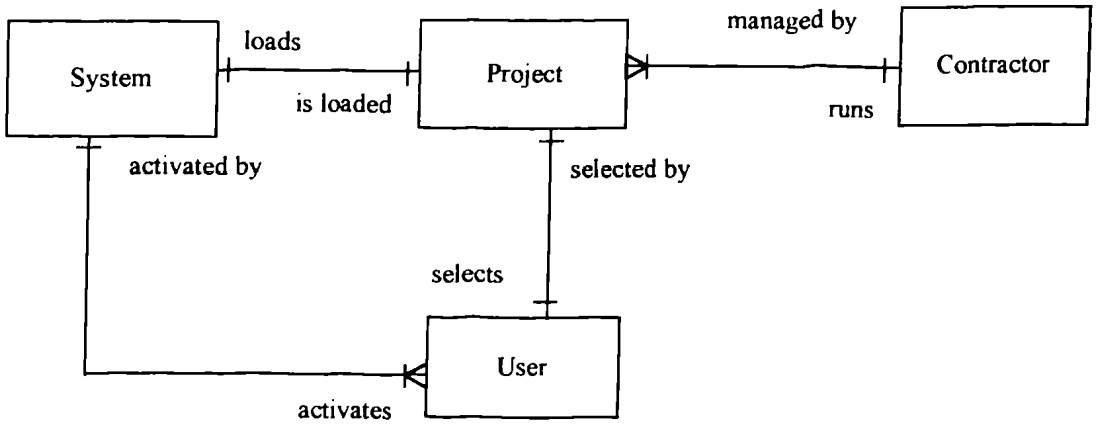
ToOrder?	(Updates list of supplier names held in the requisition being entered)
SuppliersList	
Unit	MakeInstance (Makes an instance of the requisition being made)
UnitLoadRequested	
WorksectionsList	

Orders

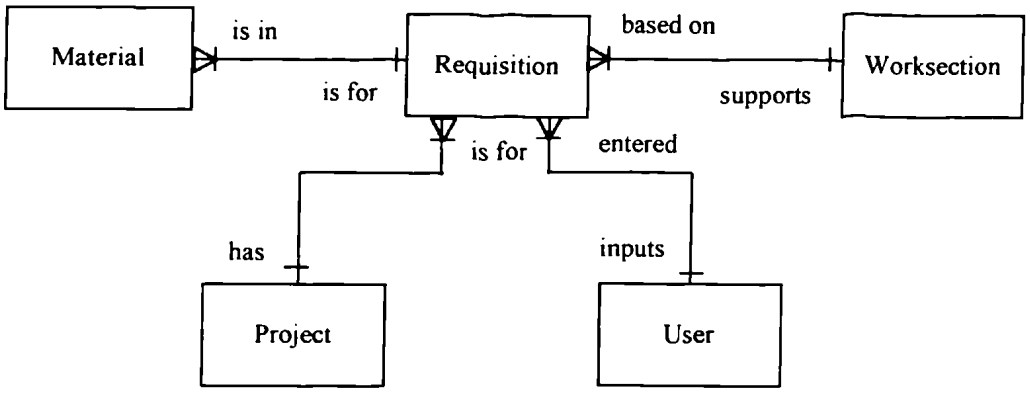
Slots	**Methods**
AmendmentNumber	AddInstances (Makes temporary instances of orders under the class Temp)
DeliveryDateRequested	
ExpeditionDate	
ExpeditionStatus	ClearInstances (Deletes temporary instances of orders)
Issue_Against	
MaterialsList	MakeInstances (Makes instances of orders)
OrderNumber	
PackagingRequestedList	ToInstances (Controls order numbers of orders instances)
QuantityRequested	
ReqNumber	
SentToSupplier?	
SpecificationRequested	
SupplierName	
Unit	
UnitLoadRequested	
ReferenceNumber	
DateMade	

B.5. Analysis Model

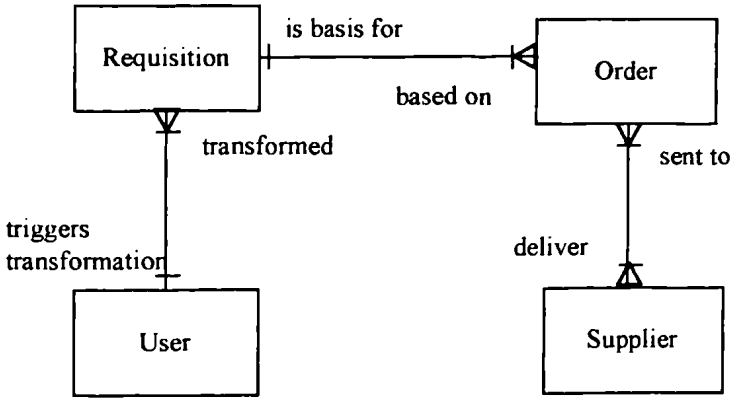
B.5.1 Use Cases Object Models



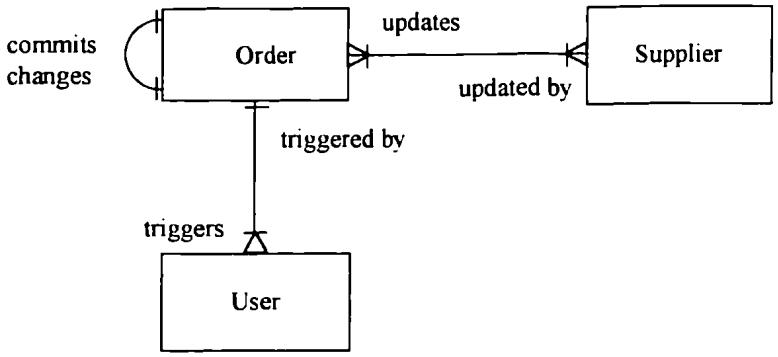
Use case 1: **Open prototype and select a project**



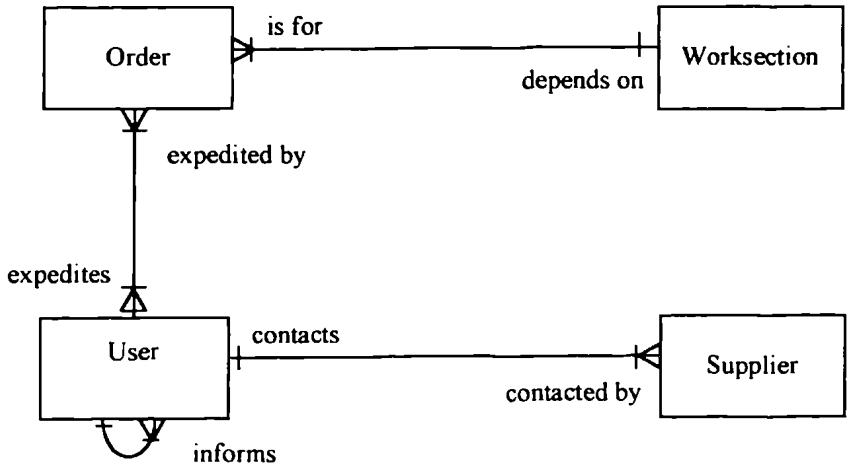
Use case 2: **Input a requisition into the system**



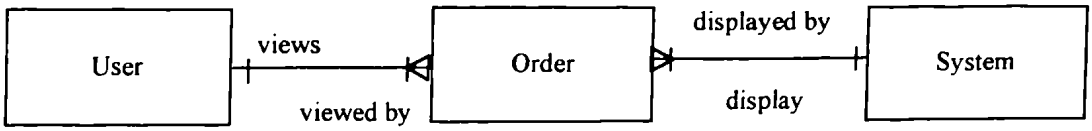
Use case 3: **Transform a requisition into order (s)**



Use case 4: **Change an existent order**



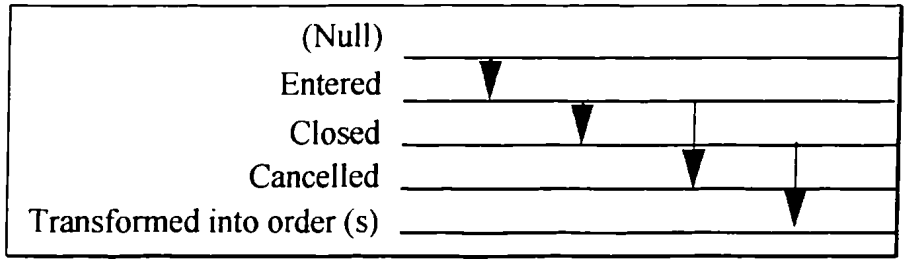
Use case 5: **Expedite orders**



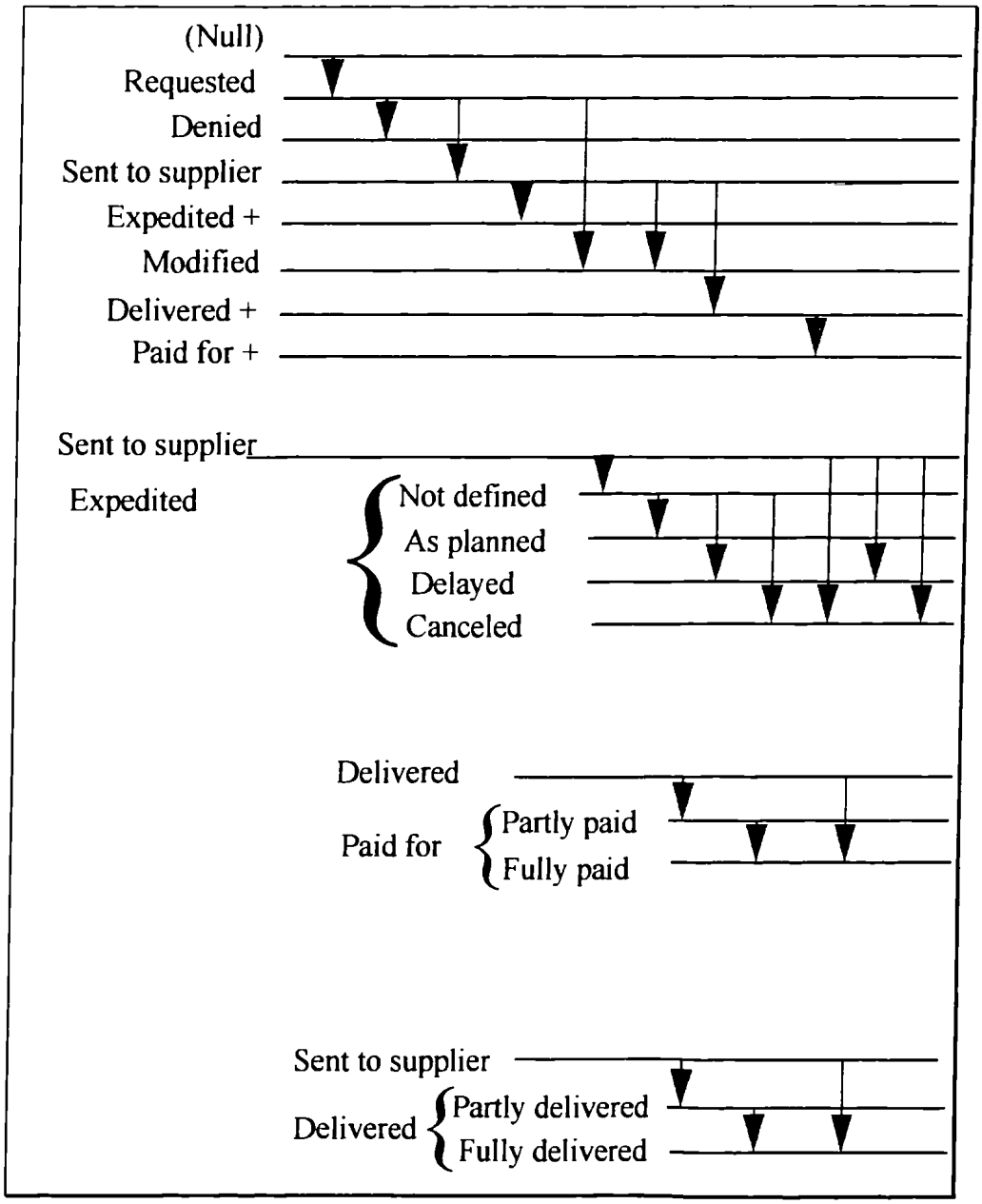
Use case 6: **View orders**

Use case 7: **View requisitions**

B.5.2. Transition Diagrams

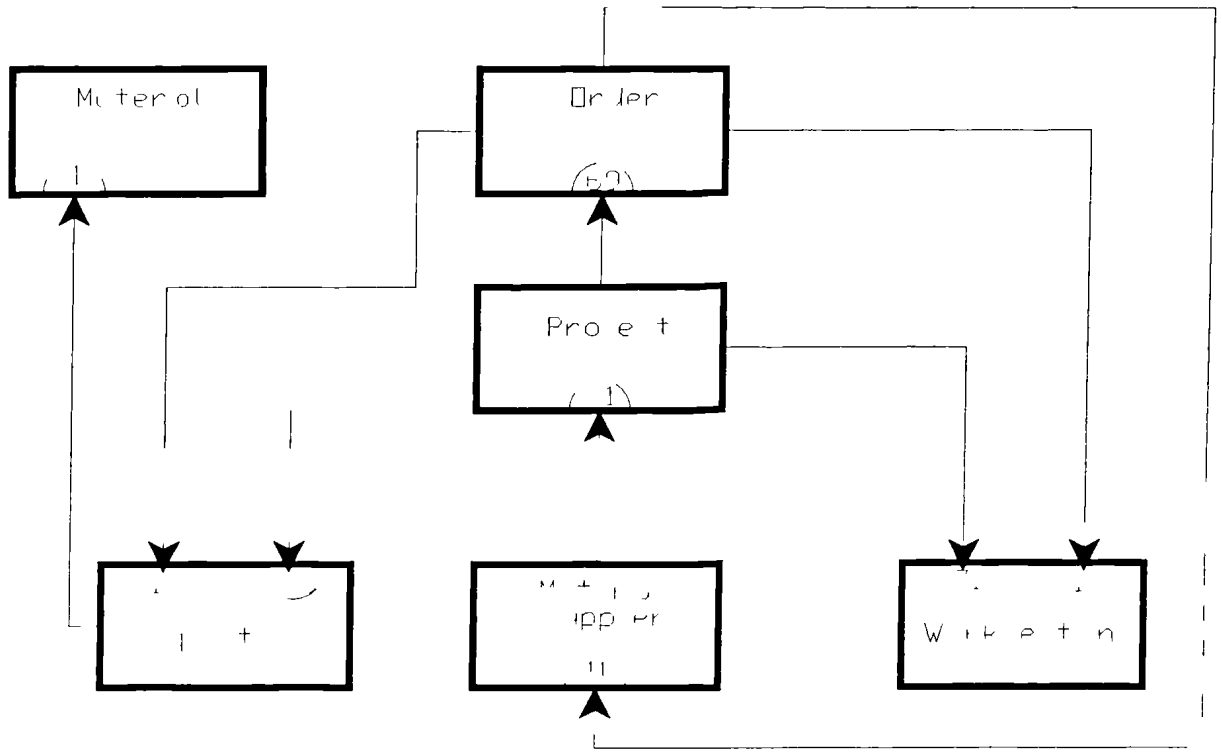


States and state transition of the object "Requisition"

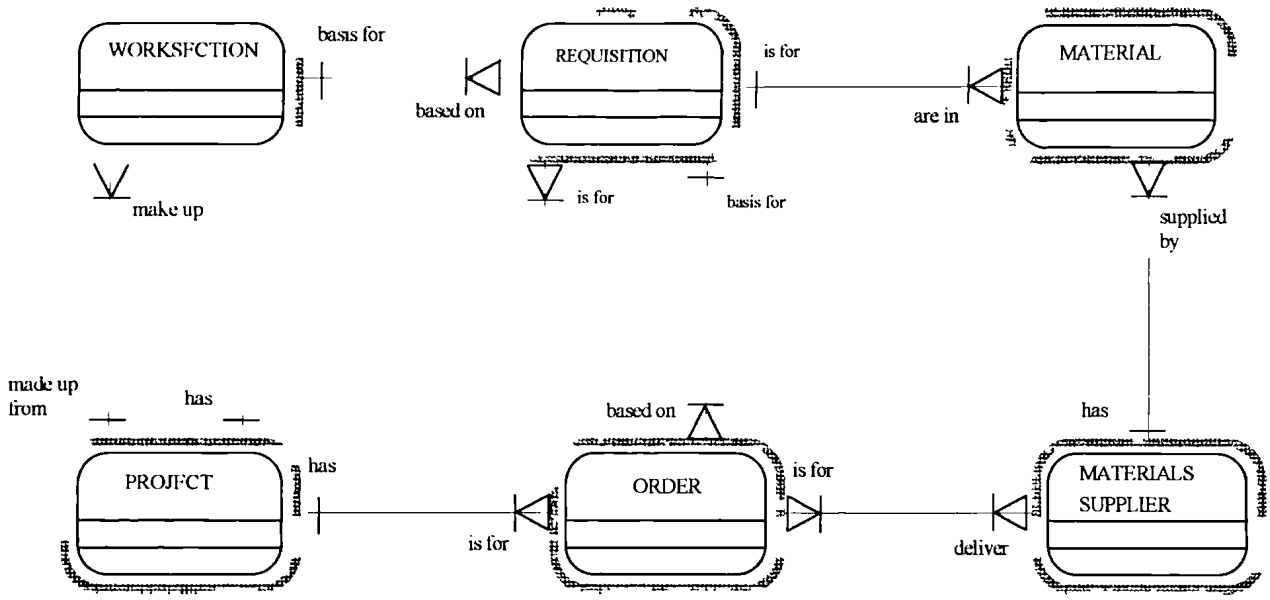


State and state transitions of the object "Order"

B.6. Prototype's Collaboration Graph And Object Model



Collaboration graph.



Object model of the prototype system

B.7. Prototype's Code (Kal)

```
/*************************************************************/
/**       ALL FUNCTIONS ARE SAVED BELOW          **/
/*************************************************************/

    /*************************************
     **** FUNCTION: RunApplication
     *************************************/
MakeFunction( RunApplication, [],

{{ForAll [ session|KSession ]
        HideWindow( session );
        IconifyWindow( KAPPA );
        HideWindow( BROWSER );
        HideWindow( KTOOLS );};

ShowWindow(SESSION);

} );


    /*************************************
     **** FUNCTION: ReadProjects
     *************************************/
MakeFunction( ReadProjects, [],
{
  { SendMessage( Global, Get_Excel_Data);
              ClearList( SLB1:AllowableValues );
              ResetImage( SLB1 );
              };

  SetWindowTitle( Session1, "Double click on Applicant Name to Select." );
  } );

    /*************************************
     **** FUNCTION: LoadExcelProgram
     *************************************/
MakeFunction( LoadExcelProgram, [],
 {
  CatchError( Execute( Excel, Projects.xls ), PostMessage( "Could not load Excel spreadsheet, check DOS
path" ) );
  While (WaitForInput( ))
     TRUE;
  } );

    /*************************************
     **** FUNCTION: LoadingProjects
     *************************************/
MakeFunction( LoadingProjects, [],
  {
  PostBusy( ON, "Loading ", image:Value, "'s data." );
  If ( Global:Database #= Excel )
```

```
    Then {
       Let [row GetElemPos( image:AllowableValues, image:Value )]
          RemoteGet( FormatValue( "R%dC1:R%dC22", row, row ),
             Excel, projects.xls, Global, TempValues );
       Let [max LengthList( Global:Slots )]
          For i From 1 To max
             Do SetValue( Applicant, GetNthElem( Global:Slots,
                          i ), GetNthElem( Global:TempValues,
                               i ) );
       };
   PostBusy( OFF );
   } );

    /*************************************
     **** FUNCTION: GetExcelDate
     *************************************/
MakeFunction( GetExcelDate, [],
{RemoteGet("r2c5:r7c5",excel,projects.xls,SLB1,AllowableValues);
PostMessage("Remote Get DONE");} );

    /*************************************
     **** FUNCTION: NamesInListBox
     *************************************/
MakeFunction( NamesInListBox, [],

AppendToList (SingleListBox1:AllowableValues, SLB1:AllowableValues) );

    /*************************************
     **** FUNCTION: LoadProjectData
     *************************************/
MakeFunction( LoadProjectData, [],
CatchError( {ReclaimStringSpace();
   PostBusy( ON, "Loading ", SingleListBox1_6:Value, "'s data." );
          Let [row GetElemPos( SingleListBox1_6:AllowableValues, SingleListBox1_6:Value )]
             RemoteGet( FormatValue( "R%dC1:R%dC22", row,
                      row ), Excel, projects.xls,
                Global, TempValues );
           Let [max LengthList( Global:Slots )]
             For i From 1 To max
                Do SetValue( Project, GetNthElem( Global:Slots,
                             i ),
                   GetNthElem( Global:TempValues, i ) );
PostMessage("Finished loading "#SingleListBox1_6:Value, "'s data" );
ClearList(Project:RequisitionsList);
ClearList(Project, OrdersList);
ResetValue(SingleListBox3, AllowableValues);
SendMessage(Project,GetRequisitionsList);
{Let [x GetValue(Project:Project_ID)#.PJ]
RemoteExecute( "SPJPreferenceLoad ( c:\spjwin3\Order.SPJ )",SPJWIN,x);
SendMessage(Worksections, MakeInstances); };
SendMessage(Project, GetOrdersList);
ClearList (Worksections:WorksectionList);
SendMessage(Worksections,LoadWorksectionList);
PostBusy(OFF);
TodayDate(); },
```

```
PostMessage("Only Project_1 has a corresponding schedule file.
More will be implemented for the rest of projects in the near future")) );


    /*************************************
     **** FUNCTION: InputReqData
     *************************************/
MakeFunction( InputReqData, [],
{
ShowWindow(Session2);
SetWindowTitle(Session2,SingleListBox1_6:Value);
HideWindow(Session1);
ShowWindow(Session2);} );

    /*************************************
     **** FUNCTION: Date_Time
     *************************************/
MakeFunction( Date_Time, [],
RemoteGet( "r10c1",excel,
 projects.xls, Global, Date_Time) );

    /*************************************
     **** FUNCTION: GetMaterialFromDatabase
     *************************************/
MakeFunction( GetMaterialFromDatabase, [],
{
ResetValue(Global, MaterialTemp);
RemoteGet("r2c8:r9c8", excel,
projects.xls, Global, MaterialTemp);
SendMessage(Global,InputMaterialListInBox);
} );

    /*************************************
     **** FUNCTION: time
     *************************************/
MakeFunction( time, [],
RemoteGet("Sheet2!r8c2", excel,projects.xls,Global,time) );

    /*************************************
     **** FUNCTION: ToSession3
     *************************************/
MakeFunction( ToSession3, [],
{HideWindow(Session2);
ShowWindow(Session3);
MaximizeWindow(Session3);
SetValue(SingleListBox2:AllowableValues, MultipleListBox1:Value);
ResetImage(SingleListBox2);
ClearList(Requisition:DeliveryDateRequested);
ClearList(Requisition:MaterialListRequested);
ClearList(Requisition:PackagingRequestedList);
ClearList(Requisition:QuantityRequested);
ClearList(Requisition:SpecificationRequested);
ClearList(Requisition:SuppliersList);
ClearList(Requisition:UnitLoadRequested);
ClearList(Global:MaterialsRequested);
```

```
ClearList(MultipleListBox3,AllowableValues);
ResetImage(MultipleListBox3);} );

/************************************
     **** FUNCTION: InputRequisitionData
     ************************************/
MakeFunction( InputRequisitionData, [],
SendMessage(Requisitions, InputMaterialInfo) );

    /************************************
     **** FUNCTION: AppendToRequisition
     ************************************/
MakeFunction( AppendToRequisition, [],
AppendToList(Requisition, QuantityRequested, Global:QuantityRequested) );

    /************************************
     **** FUNCTION: MaterialTitle1
     ************************************/
MakeFunction( MaterialTitle1, [],
{SetForwardChainMode(BREADTHFIRST, NOIGNORE);
ForwardChain ([NOASSERT]);
SetWindowTitle(Session3, "You are now requesting :"#SingleListBox2:Value);
RemoteExecute(FormatValue("[run(\"projects.xls!ToSuppliers\")]"),excel, Projects.xls);
RemoteExecute(FormatValue("[run(\"projects.xls!extractnames\")]"),excel, Projects.xls);
ClearList(Global:SupplierList);
ResetSlotOption(Requisition,SupplierList,ALLOWABLE_VALUES);
AppendToList(MultipleListBox3:AllowableValues, GetValue(SingleListBox2:Value));
ResetImage(MultipleListBox3);
RemoteGet("r118c8:r124c8", Excel, "[projects.xls]Suppliers", Global, SupplierList);
SendMessage(Requisition,InputMaterialInfo);
SetValue(Requisition:MaterialListRequested, MultipleListBox3:AllowableValues);
If(SingleListBox2:Value #= Windows)
  Then
    {SendMessage(Materials, Window_Detail);
     SendMessage(Materials, Window_Material);};
If (SingleListBox2:Value #= Doors)
  Then
    {SendMessage(Materials, Door_Detail);
    SendMessage(Materials, Window_Material);};
} ),

    /************************************
     **** FUNCTION: CancelPresentRequisition
     ************************************/
MakeFunction( CancelPresentRequisition, [],
{
Let[ answer
PostMenu("Are you sure you want to cancel present requisition?",
"Yes", "No") ]
If
answer #= Yes
Then
{ClearList( Requisition:DeliveryDateRequested);
    ClearList( Requisition:MaterialListRequested);
    ClearList(Requisition:PackagingRequestedList);
```

```
    ClearList(Requisition:QuantityRequested);
    ClearList(Requisition:SpecificationRequested);
    ClearList(Requisition:UnitLoadRequested);
    ClearList(Requisition:SuppliersList);
    ResetValue(Requisition:ReferenceNb);
    ResetValue(Requisition:DateMade);}
Else
ShowWindow(Session3);
MaximizeWindow(Session3);
});

    /*************************************
     **** FUNCTION: EnterNewRequisition
     *************************************/
MakeFunction( EnterNewRequisition, [],
{ShowWindow(Session2);
HideWindow(Session3);} );

    /*************************************
     **** FUNCTION: WriteRequisition
     *************************************/
MakeFunction( WriteRequisition, [],
CatchError(
{ClearTranscriptImage(Transcript1);
OpenWriteFile(Requi.wri);
WriteLine(FormatValue("\n\t\t\t\t\t\t\t\t%s",Date( )));
WriteLine(
FormatValue("\n\t%s\t\t%s\t%s\t%s", "Project name: ",GetValue(SingleListBox1_6:Value), "Project
Number: ", GetValue(Project:Project_ID)));
WriteLine(FormatValue("\n\t%s", " Requisition Nb:"));
WriteLine(FormatValue("\n\t%s", " Requested By:"));
WriteLine(FormatValue("\n\t%s\t%s", " Requested against:", GetValue(SingleListBox3_2:Value)));

WriteLine(
FormatValue("\n\t%s",
"==========================================================================
==========="));
Let [x LengthList(Requisition:MaterialListRequested)]
For i From 1 To x Do
{WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Material Name", GetNthElem(Requisition:MaterialListRequested, i)));
WriteLine(
FormatValue("\n\t\t%s\t%d", "Quantity Requested", GetNthElem(Requisition:QuantityRequested, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Type Of Packaging", GetNthElem(Requisition:PackagingRequestedList, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Delivery Date",GetNthElem(Requisition:DeliveryDateRequested, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Specification Code",GetNthElem(Requisition:SpecificationRequested, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Supplier's name",GetNthElem(Requisition:SuppliersList, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Unit Load",GetNthElem(Requisition:UnitLoadRequested, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t\t%s", "Unit", GetNthElem(Requisition:Unit, i)));
```

```
WriteLine(
FormatValue("\n\t\t%s", "Cost Per Unit"));
WriteLine(
FormatValue("\n\t\t%s", "Total Cost"));
WriteLine(
FormatValue("\n\t%s",
"=============================================================================
==========================="));
};
CloseWriteFile();
DisplayFile(Transcript1, Requi.wri);},
{CloseWriteFile();
PostMessage("You have tried either to view an empty or incomplete requisition,
or you are trying to view a cancelled / closed requisition.
Try again");
ShowWindow(Session3);
MaximizeWindow(Session3),} ) ),

    /*************************************
     **** FUNCTION: CloseRequisition
     *************************************/
MakeFunction( CloseRequisition, [],
{
Let[ answer
PostMenu("Are you sure you want to close this requisition?",
"Yes", "No") ]
If
answer #= Yes
Then
Let [t  LengthList(Requisition:DeliveryDateRequested)]
If t == 0
Then PostMessage("Can not close.No data is available in this requisition. Enter data and
then close.")
Else
{Let [x LengthList(Project:RequisitionsList)]
{Let [y (x+1)]
  If y < 10
Then
  {{MakeInstance("Req_"#GetValue(Project:Project_ID)#_#y, Projects_Requisitions);
  Let [a SubString("Req_"#GetValue(Project:Project_ID)#_#y, 1,11)]
    {SetValue(a:DeliveryDateRequested,GetValue( Requisition:DeliveryDateRequested));
    SetValue(a:MaterialListRequested,GetValue( Requisition:MaterialListRequested));
    SetValue(a:PackagingRequestedList, GetValue(Requisition:PackagingRequestedList));
    SetValue(a:QuantityRequested, GetValue(Requisition:QuantityRequested));
    SetValue(a:SpecificationRequested, GetValue(Requisition:SpecificationRequested));
    SetValue(a:UnitLoadRequested, GetValue(Requisition:UnitLoadRequested));
    SetValue(a:SuppliersList, GetValue(Requisition:SuppliersList));
    SetValue(a:Unit, GetValue(Requisition:Unit));
    SetValue(a:Worksection, GetValue(Requisition:Worksection));
    SetValue(a:ToOrder?, N);
    SetValue(a:ReferenceNb, a);
    SetValue(a:DateMade, Date());
Let [z SubString("Req_"#GetValue(Project:Project_ID)#_#y,1,11)]
{AppendToList(Project:RequisitionsList, z);
AppendToList (Projects_Requisitions:RequisitionsList, z);};};},}
```

```
Else Close1();
SetValue(UsedRequisition:DeliveryDateRequested,GetValue(
Requisition:DeliveryDateRequested));
      SetValue(UsedRequisition:MaterialListRequested,GetValue(
Requisition:MaterialListRequested));
      SetValue(UsedRequisition:PackagingRequestedList,
GetValue(Requisition:PackagingRequestedList));
      SetValue(UsedRequisition:QuantityRequested,
GetValue(Requisition:QuantityRequested));
      SetValue(UsedRequisition:SpecificationRequested,
GetValue(Requisition:SpecificationRequested));
      SetValue(UsedRequisition:UnitLoadRequested,
GetValue(Requisition:UnitLoadRequested));
      SetValue(UsedRequisition:SuppliersList, GetValue(Requisition:SuppliersList));
      SetValue(UsedRequisition:ReferenceNb, GetValue(Requisition:ReferenceNb));
      SetValue(UsedRequisition:DateMade, Date());
SetValue(UsedRequisition:Worksection, GetValue(Global:Worksection));

SetValue(UsedRequisition:Unit, GetValue(Requisition:Unit));
SetValue(UsedRequisition:ToOrder?, N);

     ClearList( Requisition:DeliveryDateRequested);
     ClearList( Requisition:MaterialListRequested);
     ClearList(Requisition:PackagingRequestedList);
     ClearList(Requisition:QuantityRequested);
     ClearList(Requisition:SpecificationRequested);
     ClearList(Requisition:UnitLoadRequested);
     ClearList(Requisition:SuppliersList);
     ResetValue(Requisition:ReferenceNb);
     ResetValue(Requisition:DateMade);
     ClearList(Requisition:Unit);
 ClearList(Requisition:Worksection);
Let [answer1
PostMenu("Do you want to enter another requisition?", "Yes", "No")]
If
answer1 #= Yes
Then
ShowWindow(Session2);};};} );

    /*************************************
     **** FUNCTION: BackSession3
     *************************************/
MakeFunction( BackSession3, [],
{HideWindow(Session7);
ShowWindow(Session3);
MaximizeWindow(Session3);} );



    /*************************************
     **** FUNCTION: ShowSession1
     *************************************/
MakeFunction( ShowSession1, [],
{{ForAll [session|KSession]
```

```
SetMenuBar(session);};
If
{Member?(Global:NAMES_ALLOWABLE , GetValue(Global:NAME)) And
Member?(Global:USER_ALLOWABLE , GetValue(Global:USERNAME)) And
GetElemPos(Global:NAMES_ALLOWABLE, GetValue(Global:NAME)) ==
GetElemPos(Global:USER_ALLOWABLE, GetValue(Global:USERNAME)) }
Then
{HideWindow(SESSION);
ShowWindow(Session1);
ProjectMenu1();
ProjectMenu2();
 IconifyWindow( KAPPA );
          HideWindow( BROWSER );
          HideWindow( KTOOLS );}
Else
{ Beep();
Wait(0.3);
Beep();
PostMessage("Sorry, wrong username or password");};
} );

    /*************************************
     **** FUNCTION: UpdateRequisition
     *************************************/
MakeFunction( UpdateRequisition, [],
{Let [x LengthList(Project:RequisitionsList)]
Let [y (x+1)]
  {MakeInstance("Req_"#GetValue(Project:Project_ID)#_#y, Requisitions);
  Let [a SubString("Req_"#GetValue(Project:Project_ID)#_#y, 1,11)]
    {SetValue(a:MaterialListRequested,GetValue( Requisition:MaterialListRequested));
    SetValue(a:PakagingListRequested, GetValue(Requisition:PakagingListRequested));
    SetValue(a:QuantityRequested, GetValue(Requisition:QuantityRequested));
    SetValue(a:SpecificationRequestedList, GetValue(Requisition:SpecificationRequestedList));
    SetValue(a:SupplierList,GetValue( Requisition:SupplierList));
    SetValue(a:UnitLoadRequested, GetValue(Requisition:UnitLoadRequested));};};
Let [z SubString("Req_"#GetValue(Project:Project_ID)#_#y,1,11)]
{AppendToList(Project:RequisitionsList, z);
AppendToList (Projects_Requisitions:RequisitionsList, z)};} );

    /*************************************
     **** FUNCTION: PreviewRequisition
     *************************************/
MakeFunction( PreviewRequisition, [],
{ShowWindow(Session7);
MaximizeWindow(Session7) ;
ClearList(Requisition:MaterialListRequested);
SetValue(Requisition:MaterialListRequested,
SingleListBox2:AllowableValues);} );

    /*************************************
     **** FUNCTION: MakeMenu
     *************************************/
MakeFunction( MakeMenu, [],
SetMenuBar(Session6,Menu1) );
```

```
/************************************
     **** FUNCTION: ReadOldRequisitions
     ************************************/
MakeFunction( ReadOldRequisitions, [],
CatchError({
Let[y GetValue(SingleListBox1:Value)]
{ClearTranscriptImage(Transcript3);
OpenWriteFile(Oldrequi.wri);
WriteLine(FormatValue("\n\t\t\t\t\t\t\t%s",GetValue(y:DateMade)));
WriteLine(
FormatValue("\n\t%s\t\t%s", "Project name: ",GetValue(SingleListBox1_6:Value)));
WriteLine(
FormatValue("\n\t%s\t\t%s","Project Number: ", SubString( y, 5,9)));
WriteLine(FormatValue("\n\t%s\t\t%s", " Requisition Nb:", GetValue( y:ReferenceNb)));
WriteLine(FormatValue("\n\t%s", " Requested By:"));
WriteLine(FormatValue("\n\t%s\t%s", " Requested against:", GetValue(SingleListBox3_2:Value)));
WriteLine(
FormatValue("\n\t%s",
"==========================================================================
==========="));
{Let [x
LengthList(y:MaterialListRequested)]
For i From 1 To x Do
{WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Material Name", GetNthElem(y:MaterialListRequested, i)));
WriteLine(
FormatValue("\n\t\t%s\t%d", "Quantity Requested", GetNthElem(y:QuantityRequested, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Type Of Packaging", GetNthElem(y:PackagingRequestedList, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Delivery Date",GetNthElem(y:DeliveryDateRequested, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Specification Code",GetNthElem(y:SpecificationRequested, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Supplier's name",GetNthElem(y.SuppliersList, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Unit Load",GetNthElem(y:UnitLoadRequested, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t\t%s", "Unit",GetNthElem(y:Unit, i) ));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Worksection",GetNthElem(y:Worksection, i) ));
WriteLine(
FormatValue("\n\t\t%s", "Cost Per Unit"));
WriteLine(
FormatValue("\n\t\t%s", "Total Cost"));
WriteLine(
FormatValue("\n\t%s",
"==========================================================================
==========="));
};};
CloseWriteFile();
DisplayFile(Transcript3, Oldrequi.wri);
};}, {CloseWriteFile();
PostMessage("No requisition has been selected.  Select a requisition and double click on its name to select
it.");}) );
```

```
    /*************************************
     **** FUNCTION: Menu11
     *************************************/
MakeFunction( Menu11, [],
SetMenuBar(Session11,Post) );

    /*************************************
     **** FUNCTION: Read_Requisitions
     *************************************/
MakeFunction( Read_Requisitions, [],
{ResetSlotOption(SingleListBox1, Value, ALLOWABLE_VALUES);
SetValue(SingleListBox1: AllowableValues, Project:RequisitionsList);
ResetImage(SingleListBox1)} );

    /*************************************
     **** FUNCTION: View_Existent_Requisitions
     *************************************/
MakeFunction( View_Existent_Requisitions, [],
Let [answer
PostMenu("Please select one of the following",
"Input new requisitions", "View old requisitions")]
If answer #= "Input new requisitions"
Then ShowWindow(Session2)
Else
{Read_Requisitions();
ShowWindow(Session10);} );

    /*************************************
     **** FUNCTION: ToSession10
     *************************************/
MakeFunction( ToSession10, [],
{HideWindow(Session3);
ShowWindow(Session10);
MaximizeWindow(Session10);} );

    /*************************************
     **** FUNCTION: PreviousScreen
     *************************************/
MakeFunction( PreviousScreen, [],
{   ShowWindow(Session3);
MaximizeWindow(Session3);
HideWindow(Session10)} );


/*************************************
     **** FUNCTION: ProjectsScreen
     *************************************/
MakeFunction( ProjectsScreen, [],
{
ShowWindow(Session1);
MaximizeWindow(Session1);
HideWindow(Session10);} );

    /*************************************
```

```
     **** FUNCTION: ReadRequisitions
     ************************************/
MakeFunction( ReadRequisitions, [],
{ClearList(SingleListBox1:AllowableValues);
AppendToList(SingleListBox1, AllowableValues, GetValue(Project, RequisitionsList));
ResetImage(SingleListBox1);} );


    /************************************
     **** FUNCTION: Delete
     ************************************/
MakeFunction( Delete, [],
{ForAll [ x|Worksections] DeleteInstance(x);
ClearList(Worksections:SuperNameList);
ClearList(Worksections:SuperStartDate);
ClearList(Worksections:SuperFinishDate);
ResetValue(Global:Temp1);
ClearList(Worksections:Activities_ID);} );

    /************************************
     **** FUNCTION: ShowSession12
     ************************************/
MakeFunction( ShowSession12, [],
{ShowWindow(Session12);
PositionWindow(Session12, 1, 1, 640, 350) ;
HideWindow(Session10);
ClearList(SingleListBox3:AllowableValues);
AppendToList(SingleListBox3:AllowableValues , GetValue(Project:OrdersList)) ;
ResetImage(SingleListBox3);
} );

    /************************************
     **** FUNCTION: RequiToOrder
     ************************************/
MakeFunction( RequiToOrder, [],
If SingleListBox1:Value:ToOrder? #= Y
Then PostMessage ("This requisition has already been changed into order")
Else
{
Let [Answer PostMenu("Are you sure you want to transform this requisition into an order?", "Yes", "No")]
If Answer #= Yes Then
{SendMessage(Orders,MakeInstances);
PostMessage("Requisition: "#GetValue(SingleListBox1:Value)#" Has been transformed
into order");};} );

    /************************************
     **** FUNCTION: ReadOrder
     ************************************/
MakeFunction( ReadOrder, [],
CatchError({
SetValue(Global:SupplierName, SingleListBox3:Value:SupplierName);
Let[y GetValue(SingleListBox3:Value)]
{ClearTranscriptImage(Transcript4);
OpenWriteFile(Order.wri);
WriteLine(FormatValue("\n\t\t\t\t\t\t\t\t%s",Date()));
```

```
WriteLine(
FormatValue("\n\t%s\t\t%s", "Project name: ",GetValue(SingleListBox1_6:Value)));
WriteLine(
FormatValue("\n\t%s\t\t%s","Project Number: ", SubString( y, 5,9)));
WriteLine(FormatValue("\n\t%s\t%s", " From Requisition Nb:", GetValue( y:ReqNumber)));
WriteLine(
FormatValue("\n\t%s\t\t%s", " Order Number:", GetValue( y:OrderNumber)));
WriteLine(
FormatValue("\n\t%s\t\t%s", " Amendment Nb:", GetValue( y:AmmendmentNumber)));
WriteLine(FormatValue("\n\t%s", " Requested By:"));
WriteLine(
FormatValue("\n\t%s\t\t%s", "Supplier's name",GetValue(y:SupplierName)));
WriteLine(
FormatValue("\n\t%s\t\t%s", "Contact name",GetValue(Supplier:ContactName)));
WriteLine(
FormatValue("\n\t%s\t%s\b%s","Street Nb and
Name",GetValue(Supplier:StreetNumber),GetValue(Supplier:StreetName) ));
WriteLine(
FormatValue("\n\t%s\t\t\t%s", "Town",GetValue(Supplier:Town)));
WriteLine(
FormatValue("\n\t%s\t\t%s", "Postal Code",GetValue(Supplier:PostalCode)));
WriteLine(
FormatValue("\n\t%s\t\t%s", "Telephone Number",GetValue(Supplier:Telephone)));
WriteLine(
FormatValue("\n\t%s\t\t%s", "Faxcimile Number",GetValue(Supplier:Faxcimile)));
WriteLine(
FormatValue("\n\t%s\t\t%s", "Email Address",GetValue(Supplier:EmailAddress)));
WriteLine(
FormatValue("\n\t%s\t\t\t%s", "EDI",GetValue(Supplier:EDIAddress)));
WriteLine(
FormatValue("\n\t%s",
"=============================================================================
==========================="));
{Let [x
LengthList(y:MaterialsList)]
For i From 1 To x Do
{WriteLine(FormatValue("\n\t%s\t\t%s", "Section Number", i));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Material Name", GetNthElem(y:MaterialsList, i)));
WriteLine(
FormatValue("\n\t\t%s\t%d", "Quantity Requested", GetNthElem(y:QuantityRequested, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Type Of Packaging", GetNthElem(y:PackagingRequestedList, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Delivery Date",GetNthElem(y:DeliveryDateRequested, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Specification Code",GetNthElem(y:SpecificationRequested, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t%s", "Unit Load",GetNthElem(y:UnitLoadRequested, i)));
WriteLine(
FormatValue("\n\t\t%s\t\t\t%s", "Unit",GetNthElem(y:Unit, i) ));
WriteLine(
FormatValue("\n\t\t%s", "Cost Per Unit"));
WriteLine(
FormatValue("\n\t\t%s", "Total Cost"));
```

```
WriteLine(
FormatValue("\n\t%s",
"==========================================================================
====================="));
};};
WriteLine(
FormatValue("\n\t\t%s", "Total Order Cost"));
CloseWriteFile();
DisplayFile(Transcript4, Order.wri);};
SetValue(Global:SupplierName, SingleListBox3:Value:SupplierName);},
{CloseWriteFile();
PostMessage("An error has occured.
The order you have selected did not have all required data.  Please try again");}) );

    /*************************************
     **** FUNCTION: ReadOrder1
     *************************************/
MakeFunction( ReadOrder1, [],
If Not(SingleListBox3:Value:SupplierName #= Global:SupplierName)
Then
SendMessage(Suppliers, PresentOrder1())
Else
ReadOrder() );

    /*************************************
     **** FUNCTION: PresentOrder
     *************************************/
MakeFunction( PresentOrder, [],
{ReadOrder1();
ReadOrder();} );

    /*************************************
     **** FUNCTION: ChangeOrder
     *************************************/
MakeFunction( ChangeOrder, [],
{PostInputForm("Change order",Global,
SectionNumber,"Please input the section number you want to change") ;
ShowWindow(Session13);
PositionWindow(Session13, 0,350,640,140);
} );


  /*************************************
    **** FUNCTION: CheckNewValues
    *************************************/
MakeFunction( CheckNewValues, [],
{Let [r SingleListBox3:Value]
Let [p r:AmmendmentNumber]
Let [q r:SentToSupplier?]
{If GetNthElem(r:QuantityRequested, Global:SectionNumber) ==Global:NewQuantity
Then PostMessage ("New quantity is equal to original quantity. Kappa will ignore new quantity")
Else SetNthElem(r:QuantityRequested, Global:SectionNumber, Global:NewQuantity) ;
{If GetNthElem(r:DeliveryDateRequested, Global:SectionNumber) #= Global:NewDate
Then PostMessage ("New date is equal to original date. Kappa will ignore new date")
Else SetNthElem(r:DeliveryDateRequested, Global:SectionNumber, Global:NewDate) ;
```

```
};
{Let [Answer PostMenu("Change order done.
Do you want kappa to update order ammendment number?", "Yes", "No")]
If {Answer #= Yes And SingleListBox3:Value:SentToSupplier? #= Y}
Then {SetValue(SingleListBox3:Value:AmmendmentNumber,
(SingleListBox3:Value:AmmendmentNumber+1));
PostMessage("Done");}
Else{
If {Answer #= Yes And SingleListBox3:Value:SentToSupplier? #= N}
Then PostMessage("Order has not been sent yet to supplier. Kappa will not update ammendment number");
 };};};
PresentOrder();} );

    /************************************
     **** FUNCTION: CheckDate
     ************************************/
MakeFunction( CheckDate, [],
Let [x Global:DeliveryDateRequested]
Let [y StringLength(x)]
If y >6 Or y<6
Then PostMessage("Sorry, you enterd a wrong date")
Else
{
Let [p SubString(x, 1, 2)]
Let [q SubString(x, 3,4)]
Let [r SubString(x,5,6)]

If
{p < 1 Or p > 31 Or q < 1 Or q > 12 Or r < 94 Or r > 99}
Then PostMessage("Sorry you have entered a wrong date");} );

    /************************************
     **** FUNCTION: CheckDate1
     ************************************/
MakeFunction( CheckDate1, [],
{Let [x Global:NewDate]
Let [y StringLength(x)]
If y >6 Or y<6
Then PostMessage("Sorry, you enterd a wrong date")
Else
{
Let [p SubString(x, 1, 2)]
Let [q SubString(x, 3,4)]
Let [r SubString(x,5,6)]

If
{p < 1 Or p > 31 Or q < 1 Or q > 12 Or r < 94 Or r > 99}
Then PostMessage("Sorry you have entered a wrong date")
};} );

    /************************************
     **** FUNCTION: ChangeDate
     ************************************/
MakeFunction( ChangeDate, [],
Let [r SingleListBox3:Value]
```

```
Let [p r:AmmendmentNumber]
Let [q r:SentToSupplier?]
If GetNthElem(r:DeliveryDateRequested, Global:SectionNumber) #=
  Global:NewDate
Then PostMessage ("New date is equal to original date. Kappa will ignore new date")
Else { SetNthElem(r:DeliveryDateRequested, Global:SectionNumber, Global:NewDate) ;
If q #= Y
Then {
SetValue (r:AmmendmentNumber, (p+1));
PostMessage("Order ammendment number has been updated");
};} );

    /*************************************
     **** FUNCTION: CheckOrderChange
     *************************************/
MakeFunction( CheckOrderChange, [],
{SetValue(Global:DateCheck, CheckDate1());
If Global:DateCheck #= FALSE
Then CheckNewValues();} );

    /*************************************
     **** FUNCTION: SendOrder
     *************************************/
MakeFunction( SendOrder, [],
{Let [r SingleListBox3:Value]
SetValue(r:SentToSupplier?, Y);
PostMessage("Order number: "#GetValue(SingleListBox3:Value)#", has been market as sent to supplier");} );

    /*************************************
     **** FUNCTION: MakeMenu1
     *************************************/
MakeFunction( MakeMenu1, [],
  SetMenuBar( Session2, Post ) );

    /*************************************
     **** FUNCTION: MakeMenu2
     *************************************/
MakeFunction( MakeMenu2, [],
  SetMenuBar( Session2, Top, Post ) );

    /*************************************
     **** FUNCTION: MakeFileMenu
     *************************************/
MakeFunction( MakeFileMenu, [],
  SetMenuBar( Session2, File, FileSelection ) );

    /*************************************
     **** FUNCTION: MakeFileMenu1
     *************************************/
MakeFunction( MakeFileMenu1, [],
  SetMenuBar( Session2, File, FileSelection ) );

    /*************************************
     **** FUNCTION: MakeFileMenu2
     *************************************/
```

```
MakeFunction( MakeFileMenu2, [],
  SetMenuBar( Session2, File, FileSelection ) );

    /*************************************
     **** FUNCTION: MakeFileMenu3
     *************************************/
MakeFunction( MakeFileMenu3, [],
  SetMenuBar( Session2, FileSelection, file ) );

    /*************************************
     **** FUNCTION: MakeTransformMenu
     *************************************/
MakeFunction( MakeTransformMenu, [],
  SetMenuBar( Session2, TransformSelection, Transform ) );

    /*************************************
     **** FUNCTION: MakeDataEntryMenu
     *************************************/
MakeFunction( MakeDataEntryMenu, [],
  SetMenuBar( Session2, DataEntrySelection, DataEntry ) );

    /*************************************
     **** FUNCTION: MakeViewMenu
     *************************************/
MakeFunction( MakeViewMenu, [],
  SetMenuBar( Session2, ViewSelection, View ) );

    /*************************************
     **** FUNCTION: MakeCommunicationMenu
     *************************************/
MakeFunction( MakeCommunicationMenu, [],
  SetMenuBar( Session2, CommunicationSelection, Communication ) );

    /*************************************
     **** FUNCTION: ProjectMenu
     *************************************/
MakeFunction( ProjectMenu, [],
SetMenuBar(Session2, ProjectMenuSelection, ProjectMenu) );

    /*************************************
     **** FUNCTION: UpdateSentToSupplier
     *************************************/
MakeFunction( UpdateSentToSupplier, [],
Let [Answer  PostMenu("Are you sure you want to send order to supplier", "Yes", "No")]
If Answer #= Yes
Then {
SetValue(SingleListBox3:Value:SentToSupplier?, Y);
    PostMessage("Order is considered as sent to supplier");} );

    /*************************************
     **** FUNCTION: Expedition
     *************************************/
MakeFunction( Expedition, [],
{RemoteGet( "r10c1",excel,projects.xls, Global, Date_Time);
ShowWindow(Session14);
```

```
PositionWindow(Session14, 0,0,640,350);
SetValue(SingleListBox4:AllowableValues, GetValue(Project,OrdersList));
ResetImage(SingleListBox4);
SetValue(Global:Box4Name, GetValue(SingleListBox4:Value));
SetValue(RadioButtonGroup1:AllowableValues,
GetSlotOption( GetValue(Global:Box4Name),ExpeditionStatus,ALLOWABLE_VALUES));
} );

    /*************************************
     **** FUNCTION: CheckExpedition
     *************************************/
MakeFunction( CheckExpedition, [],
{SetValue(Edit4:Owner, GetValue(SingleListBox4:Value));
SetValue(Edit4:OwnerSlot,ExpeditionDate);
SetValue (SingleListBox4:Value:ExpeditionDate,
GetNthElem( SingleListBox4:Value:DeliveryDateRequested,1));
ResetImage(Edit4);
{
ClearList(MultipleListBox4:AllowableValues);
ForAll [x|Worksections]
Let [aa SubString (x:StartDate, 1,2)]
Let [bb SubString (x:StartDate, 4,5)]
Let [cc SubString (x:StartDate, 7,8)]
Let [dd aa#bb#cc]
If dd #= GetNthElem( SingleListBox4:Value:DeliveryDateRequested,1)
Then AppendToList (MultipleListBox4:AllowableValues, x);
ResetImage(MultipleListBox4);
};
SetValue(Global:Expedition, SingleListBox4:Value:ExpeditionStatus);
DeliveryDate();
DaysBetweenDates();
ResetImage(Edit10);
CompareDates();
ResetImage(Edit10);
} );


 /*************************************
     **** FUNCTION: PhoneSupplier
     *************************************/
MakeFunction( PhoneSupplier, [],
PostMessage("Dialling, Please wait.") );

    /*************************************
     **** FUNCTION: MessageToConcerned
     *************************************/
MakeFunction( MessageToConcerned, [],
PostMessage
("This message has been sent to parties involved.  "
#"**The following project :"#GetValue(Project:Project_ID)#"will have delays in materials supply for the
worksections listed on the screen"
#".  Plan for corrective actions"#"**") );

    /*************************************
     **** FUNCTION: ProjectMenu1
```

```
    ************************************/
MakeFunction( ProjectMenu1, [],
{SetMenuBar(Session12,ProjectSelection);
SetMenuBar(Session13,ProjectSelection);
SetMenuBar(Session14,ProjectSelection);
} );

    /************************************
     **** FUNCTION: ProjectMenu2
     ************************************/
MakeFunction( ProjectMenu2, [],
{SetMenuBar(Session1,ProjectSelection2);
SetMenuBar(Session2,ProjectSelection2);
SetMenuBar(Session3,ProjectSelection2);
SetMenuBar(Session10,ProjectSelection2);} );

    /************************************
     **** FUNCTION: Exit1
     ************************************/
MakeFunction( Exit1, [],
 Let [x PostMenu( "Exit to:", "KAPPA-PC", "MS-Windows", CANCEL )]
     {
     If ( x #= "KAPPA-PC" )
      Then {
          ForAll [ session|KSession ]
               HideWindow( session );
          ShowWindow( KAPPA );
          ShowWindow( BROWSER );
          ShowWindow( KTOOLS );
          };
     If ( x #= "MS-Windows" )
      Then
{ForAll [ session|KSession ]
               HideWindow( session );
          IconifyWindow( KAPPA );
          HideWindow( BROWSER );
          HideWindow( KTOOLS );
ShowWindow(SESSION);
Exit( );};} );

    /************************************
     **** FUNCTION: NotImplemented
     ************************************/
MakeFunction( NotImplemented, [],
PostMessage("Not Implemented Yet!") );


    /************************************
     **** FUNCTION: TodayDate
     ************************************/
MakeFunction( TodayDate, [],
SetValue(TodayDate:Date, Date()) );

    /************************************
     **** FUNCTION: DeliveryDate
```

```
     ************************************/
MakeFunction( DeliveryDate, [],
{
SetValue(DeliveryDate:Day, SubString(Edit4:Value, 1,2));
SetValue(DeliveryDate:Month, SubString(Edit4:Value, 3,4));
SetValue(DeliveryDate:KappaYear, SubString(Edit4:Value,5,6));
SetValue(DeliveryDate:Year, GetValue(Date:PreYear)#GetValue(DeliveryDate:KappaYear));} );

    /************************************
     **** FUNCTION: TodayCalendar
     ************************************/
MakeFunction( TodayCalendar, [],
{
  ClearTranscriptImage(Transcript5 );
  SetWindowTitle(Session15, "This Month Calendar");
  If ( Number?( TodayDate:Month ) And
       Number?( TodayDate:Year ) )
   Then DisplayCalendar( TodayDate:Month, TodayDate:Year, Transcript5 )
   Else PostMessage( FormatValue( "The month and year should be numbers,\nthe year in
its full form (e.g. 1991)" ) );
 ShowWindow(Session15); } );

    /************************************
     **** FUNCTION: DisplayCalendar
     ************************************/
MakeFunction( DisplayCalendar, [month year transcript],
{
  PostBusy( ON, "Formatting Calendar ..." );
  DisplayText( transcript,
    CalendarString( month, year ) );
  PostBusy( OFF );
  } );




    /************************************
     **** FUNCTION: DeliveryCalendar
     ************************************/
MakeFunction( DeliveryCalendar, [],
{
  ClearTranscriptImage(Transcript6 );
  SetWindowTitle(Session16, "Delivery Month Calendar");
  If ( Number?( DeliveryDate:Month ) And
       Number?( DeliveryDate:Year ) )
   Then DisplayCalendar( DeliveryDate:Month, DeliveryDate:Year, Transcript6 )
   Else PostMessage( FormatValue( "The month and year should be numbers,\nthe year in
its full form (e.g. 1991)" ) );
ShowWindow(Session16); } );


    /************************************
     **** FUNCTION: GetOrders
     ************************************/
MakeFunction( GetOrders, [],
```

```
{
ClearList (Global:OrdersList);
Let [x LengthList(Project:OrdersList)]
For i From 1 To x Do
{SetValue (Global:Order, GetNthElem(Project:OrdersList,i));
SetValue(DeliveryDate:Day, SubString(Global:Order:ExpeditionDate, 1,2));
SetValue(DeliveryDate:Month, SubString(Global:Order:ExpeditionDate, 3,4));
SetValue(DeliveryDate:KappaYear, SubString(Global:Order:ExpeditionDate,5,6));
SetValue(DeliveryDate:Year, GetValue(Date:PreYear)#GetValue(DeliveryDate:KappaYear));
DaysBetweenDates();
If DateGlobal:DaysBetween == DateGlobal:DifDaysReq
Then
AppendToList (Global:OrdersList,Global:Order);
};
If LengthList(Global:OrdersList) > 0
Then
{ClearList(SingleListBox5:AllowableValues);
AppendToList(SingleListBox5:AllowableValues, GetValue(Global:OrdersList));
ResetImage(SingleListBox5);
MaximizeWindow(Session14);};
If LengthList(Global:OrdersList) == 0
Then
{
ClearList(SingleListBox5:AllowableValues);
ResetImage(SingleListBox5);
PostMessage("There are no orders for this project that are due within: "
#GetValue(DateGlobal:DifDaysReq));};} );

    /*************************************
     **** FUNCTION: PresentOrder1
     *************************************/
MakeFunction( PresentOrder1, [],
{PostBusy( ON, "Kappa, importing data. Please wait") ;
SetValue(Global:SupplierName, SingleListBox3:Value:SupplierName);
SendMessage(Global,GetSuppliers);
SendMessage(Suppliers, GetSuppliersDetails);
ReadOrder();
PostBusy(OFF);} );

    /*************************************
     **** FUNCTION: Close1
     *************************************/
MakeFunction( Close1, [],
{Let [x LengthList(Project:RequisitionsList)]
Let [y (x+1)]
If {y <99 And y > 9 }Then
{
{MakeInstance("Req_"#GetValue(Project:Project_ID)#_#y, Projects_Requisitions);
  Let [a SubString("Req_"#GetValue(Project:Project_ID)#_#y, 1,12)]
    {SetValue(a:DeliveryDateRequested,GetValue( Requisition:DeliveryDateRequested));
    SetValue(a:MaterialListRequested,GetValue( Requisition:MaterialListRequested));
    SetValue(a:PackagingRequestedList, GetValue(Requisition:PackagingRequestedList));
    SetValue(a:QuantityRequested, GetValue(Requisition:QuantityRequested));
    SetValue(a:SpecificationRequested, GetValue(Requisition:SpecificationRequested));
    SetValue(a:UnitLoadRequested, GetValue(Requisition:UnitLoadRequested));
```

```
    SetValue(a:SuppliersList, GetValue(Requisition:SuppliersList));
    SetValue(a:Unit, GetValue(Requisition:Unit));
SetValue(a:Worksection, GetValue(Requisition:Worksection));
SetValue(a:ToOrder?, N);
SetValue(a:ReferenceNb, a);
    SetValue(a:DateMade, Date());
Let [z SubString("Req_"#GetValue(Project:Project_ID)#_#y,1,12)]
{AppendToList(Project:RequisitionsList, z);
AppendToList (Projects_Requisitions:RequisitionsList, z);};};};};} );

    /*************************************
     **** FUNCTION: ToInstances1
     *************************************/
MakeFunction( ToInstances1, [],
{Let [r GetValue(SingleListBox1,Value)]
Let [x LengthList(r:MaterialListRequested)]
Let [y LengthList(Project:OrdersList)]
{For i From 1 To 1 Do
  If {(y+1) > 9 And (y+1) < 99 } Then
  { MakeInstance(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,12),Temp);
AppendToList                (SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,12),MaterialsList,
GetNthElem(r:MaterialListRequested,i));
AppendToList(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,12),QuantityRequested,GetNthEl
em(r:QuantityRequested,i));
AppendToList(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,12),PackagingRequestedList,Get
NthElem(r:PackagingRequestedList,i));
AppendToList(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,12),DeliveryDateRequested,Get
NthElem(r:DeliveryDateRequested,i));
AppendToList(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,12),SpecificationRequested,GetN
thElem(r:SpecificationRequested,i));
AppendToList(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,12),UnitLoadRequested,GetNthE
lem(r:UnitLoadRequested,i));
AppendToList(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,12),Unit,GetNthElem(r:Unit,i);
SetValue(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,12),ReqNumber,
GetValue(SingleListBox1:Value));
SetValue(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,12),OrderNumber,SubString("Ord_"#
GetValue(Project:Project_ID)#_#(y+1),1,12));
AppendToList(Project:OrdersList, SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,12));
SetValue(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,12),SupplierName,
GetNthElem(r:SuppliersList,i));
GetInstanceList (Temp,Temp:InstanceList);}};} );

    /*************************************
     **** FUNCTION: MaterScr
     *************************************/
MakeFunction( MaterScr, [],
Execute("c:\kappa\helpfile\materscr.doc") );

    /*************************************
     **** FUNCTION: ProjSrn
     *************************************/
MakeFunction( ProjSrn, [],
Execute("c:\kappa\helpfile\projsrn.doc") );

    /*************************************
```

```
        **** FUNCTION: ExpdScr
        ************************************/
    MakeFunction( ExpdScr, [],
    Execute("c:\kappa\helpfile\expdscr.doc") );

        /************************************
        **** FUNCTION: OrdrScrn
        ************************************/
    MakeFunction( OrdrScrn, [],
    Execute("c:\kappa\helpfile\ordrscrn.doc") );

        /************************************
        **** FUNCTION: ChangeSc
        ************************************/
    MakeFunction( ChangeSc, [],
    Execute("c:\kappa\helpfile\changesc.doc") );

        /************************************
        **** FUNCTION: RequScrn
        ************************************/
    MakeFunction( RequScrn, [],
    Execute("c:\kappa\helpfile\requscrn.doc") );

        /************************************
        **** FUNCTION: RequDate
        ************************************/
    MakeFunction( RequDate, [],
    Execute("c:\kappa\helpfile\requdata.doc") );



  /*************************************************************/
  /**       ALL CLASSES ARE SAVED BELOW          **/
  /*************************************************************/


      /************************************
      **** CLASS: NewMenu
      ************************************/
  MakeClass( NewMenu, Menu );


      /************************************
      **** CLASS: Suppliers
      ************************************/
  MakeClass( Suppliers, Root );

  /************** METHOD: GetSuppliersDetails  **************/
  MakeMethod( Suppliers, GetSuppliersDetails, [],
  {RemoteGet("r54c1", Excel, "[projects.xls]Sheet3", Supplier, StreetNumber);
  RemoteGet("r54c2", Excel, "[projects.xls]Sheet3", Supplier, StreetName);
  RemoteGet("r54c3", Excel, "[projects.xls]Sheet3", Supplier, Town);
  RemoteGet("r54c4", Excel, "[projects.xls]Sheet3", Supplier, PostalCode);
  RemoteGet("r54c5", Excel, "[projects.xls]Sheet3", Supplier, ContactName);
```

```
RemoteGet("r54c6", Excel, "[projects.xls]Sheet3", Supplier, Telephone);
RemoteGet("r54c7", Excel, "[projects.xls]Sheet3", Supplier, Faxcimile);
RemoteGet("r54c8", Excel, "[projects.xls]Sheet3", Supplier, IDNumber);
SetValue(Supplier:Name, Global:SupplierName);

} );

/************** METHOD: GetSuppliers  **************/
MakeMethod( Suppliers, GetSuppliers, [],
{
RemoteExecute(FormatValue("[run(\"projects.xls!ToSheet3\")]"),excel,
Projects.xls);
RemoteExecute(FormatValue("[run(\"projects.xls!SuppAddress\")]"),excel,
Projects.xls);
RemoteExecute(FormatValue("[run(\"projects.xls!Supp1\")]"),excel,
Projects.xls);
} );
MakeSlot( Suppliers:Name );
SetSlotOption( Suppliers:Name, IMAGE, Edit8 );
MakeSlot( Suppliers:Telephone );
MakeSlot( Suppliers:Fascimile );
MakeSlot( Suppliers:List_Products );
MakeSlot( Suppliers:ContactName );
MakeSlot( Suppliers:IDNumber );
MakeSlot( Suppliers:Suppliers_List );
SetSlotOption( Suppliers:Suppliers_List, MULTIPLE );
SetValue( Suppliers:Suppliers_List, sup_01, sup_02, sup_05, sup_10, sup_12, sup_13, sup_14,);
MakeSlot( Suppliers:StreetNumber );
MakeSlot( Suppliers:StreetName );
MakeSlot( Suppliers:Town );
MakeSlot( Suppliers:PostalCode );
MakeSlot( Suppliers:Faxcimile );
MakeSlot( Suppliers:EmailAddress );
Suppliers:EmailAddress = X;
MakeSlot( Suppliers:EDIAddress );
Suppliers:EDIAddress = X;

    /*************************************
     **** CLASS: Materials
     *************************************/
MakeClass( Materials, Root );

/************** METHOD: Window_Detail  **************/
MakeMethod( Materials, Window_Detail, [],
PostInputForm("Please input window's dimensions in mm", Global, WinHeight, "Please input window's height
required",
 Global, WinWidth, "Please input window's width required",
Global, WinDepth, "Please input window's Depth required"
) );

/************** METHOD: Door_Detail  **************/
MakeMethod( Materials, Door_Detail, [],
PostInputForm("Please input Door's dimensions in mm", Global, DoorHeight, "Please input Door's height
required",
 Global, DoorWidth, "Please input Door's width required",
```

```
 Global, DoorDepth, "Please input Door's Depth required"
) );

/************** METHOD: Door_Material  **************/
MakeMethod( Materials, Door_Material, [],
PostInputForm(" Door's material", Materials, DoorWinMaterials,"Please select door's material") );

/************** METHOD: Window_Material  **************/
MakeMethod( Materials, Window_Material, [],
PostInputForm(" Window's material",
Materials, DoorWinMaterials,
"Please select window's material") );

/************** METHOD: UpdateRequisitionSpecification  **************/
MakeMethod( Materials, UpdateRequisitionSpecification, [],
AppendToList(Requisition:SpecificationRequested, Materials:Material_Specification_Code) );
MakeSlot( Materials:Material_Name );
MakeSlot( Materials:Material_ID_Number );
MakeSlot( Materials:Material_Packaging );
SetSlotOption( Materials:Material_Packaging, MULTIPLE );
ClearList( Materials:Material_Packaging );
MakeSlot( Materials:Material_Unit_Load );
MakeSlot( Materials:Material_Specification_Code );
SetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES, Solid, Cellular, Hollow,
Common, Facing, Architectural_masonry, Insulating, Special_Face_Blocks );
Materials:Material_Specification_Code = Cellular;
SetSlotOption( Materials:Material_Specification_Code, AFTER_CHANGE, UpdateRequisitionSpecification
);
MakeSlot( Materials:Unit_Measurement );
MakeSlot( Materials:Cost_Per_Unit );
MakeSlot( Materials:Material_Qty_Req );
MakeSlot( Materials:Material_Qty_In_Storage );
MakeSlot( Materials:Material_Qty_Used );
MakeSlot( Materials:Material_Storage_Rec );
MakeSlot( Materials:Material_Vol_Unit );
MakeSlot( Materials:Nb_Pack_In_UnitLoad );
MakeSlot( Materials:Material_Vul_Weather );
MakeSlot( Materials:Material_Waste_Allowance );
MakeSlot( Materials:Material_Substitutes );
MakeSlot( Materials:Material_Storage_Duration );
MakeSlot( Materials:Material_Order_Lead_Time );
MakeSlot( Materials:Taxes );
MakeSlot( Materials:Plant_To_Use );
MakeSlot( Materials:Material_Storage_Ground_Type );
MakeSlot( Materials:Material_Qty_Ordered );
MakeSlot( Materials:DoorWinMaterials );
SetSlotOption( Materials:DoorWinMaterials, ALLOWABLE_VALUES, Wood, Aluminium, Plastic_U_PVC,
Galvanised_Steel );
Materials:DoorWinMaterials = Wood;

    /************************************
     **** CLASS: Projects
     ************************************/
MakeClass( Projects, Root );
```

```
/************** METHOD: GetRequisitionsList  **************/
MakeMethod( Projects, GetRequisitionsList, [],
{ForAll [Req|Projects_Requisitions]
Let [ R SubString(GetValue(Req:ReferenceNb), 5, 9)]
If R #= GetValue(Project:Project_ID)
Then AppendToList ( Project:RequisitionsList, GetValue(Req:ReferenceNb))} );

/************** METHOD: UpdateRequisitionsImage  **************/
MakeMethod( Projects, UpdateRequisitionsImage, [],
ReadRequisitions() );

/************** METHOD: GetWorksectionsList  **************/
MakeMethod( Projects, GetWorksectionsList, [],
{ForAll [Work|Worksections]
Let [ R SubString(GetValue(Work:Name), 1, 5)]
If R #= GetValue(Project:Project_ID)
Then AppendToList ( Project:WorksectionsList, GetValue(Work:Name))} );

/************** METHOD: GetOrdersList  **************/
MakeMethod( Projects, GetOrdersList, [],
{ForAll [Ord|Projects_Orders]
Let [ R SubString(GetValue(Ord:OrderNumber), 5, 9)]
If R #= GetValue(Project:Project_ID)
Then AppendToList ( Project:OrdersList, GetValue(Ord:OrderNumber))} );
MakeSlot( Projects:Project_ID );
MakeSlot( Projects:Project_Name );
MakeSlot( Projects:Client_Name );
MakeSlot( Projects:Project_Address );
MakeSlot( Projects:Start_Date );
MakeSlot( Projects:Finish_Date );
MakeSlot( Projects:RequisitionsList );
SetSlotOption( Projects:RequisitionsList, MULTIPLE );
ClearList( Projects:RequisitionsList );
SetSlotOption( Projects:RequisitionsList, AFTER_CHANGE, UpdateRequisitionsImage );
MakeSlot( Projects:OrdersList );
SetSlotOption( Projects:OrdersList, MULTIPLE );
ClearList( Projects:OrdersList );
MakeSlot( Projects:WorksectionsList );
SetSlotOption( Projects:WorksectionsList, MULTIPLE );
ClearList( Projects:WorksectionsList );

    /************************************
     **** CLASS: Worksections
     ************************************/
MakeClass( Worksections, Root );

/************** METHOD: GetSuperNameList  **************/
MakeMethod( Worksections, GetSuperNameList, [],


{SetValue(Worksections, ScheduleFileName,  GetValue(Project:Project_ID)#".pj");
RemoteGet(SPJDDE_1, SPJWIN, GetValue(Worksections, ScheduleFileName),
 Worksections, SuperNameList);} );

/************** METHOD: GetSuperStartDate  **************/
```

```
MakeMethod( Worksections, GetSuperStartDate, [],
CatchError(

{SetValue(Worksections, ScheduleFileName,  GetValue(Project:Project_ID)#".pj");
RemoteGet(SPJDDE_2, SPJWIN, GetValue(Worksections, ScheduleFileName),
 Worksections, SuperStartDate);},
PostMessage("At the moment, this prototype deals with only one file from
Superproject. More files will be introduced in the near future")) );

/************** METHOD: GetSuperFinishDate  **************/
MakeMethod( Worksections, GetSuperFinishDate, [],
CatchError(

{SetValue(Worksections, ScheduleFileName,  GetValue(Project:Project_ID)#".pj");
RemoteGet(SPJDDE_3, SPJWIN, GetValue(Worksections, ScheduleFileName),
 Worksections, SuperFinishDate);},
PostMessage("At the moment, this prototype deals with only one file from
Superproject. More files will be introduced in the near future")) );

/************** METHOD: MakeInstances  **************/
MakeMethod( Worksections, MakeInstances, [],
{
PostBusy(ON, "Kappa importing data from Superproject. Please wait");
ForAll [d|Worksections] DeleteInstance(d);
SendMessage(Worksections,GetSuperNameList);
SendMessage(Worksections,GetSuperStartDate);
SendMessage(Worksections,GetSuperFinishDate);
SendMessage(Worksections,GetSuperActivity_ID);
Let [x LengthList(Worksections:SuperNameList)]
For i From 1 To x Do
   Let [y GetNthElem(Worksections:SuperNameList, i)]
     If StringLength (y) > 31
      Then
         Let [t  SubString (y, 1, 25)]
          {Let [u StringLength(t)]
           For r From 1 To u Do
              Let [p SubString(t,r,r)]
              {If Member?(Global:Alpha, p)
              Then SetValue(Global:Temp1, GetValue(Global:Temp1)#SubString(t,r,r));};
              MakeInstance(GetValue(Project:Project_ID)#_#GetValue(Global:Temp1),Worksections);
                              SetValue(GetValue(Project:Project_ID)#_#GetValue(Global:Temp1),StartDate,
GetNthElem(Worksections:SuperStartDate, i));
                             SetValue(GetValue(Project:Project_ID)#_#GetValue(Global:Temp1),FinishDate,
GetNthElem(Worksections:SuperFinishDate, i));
SetValue(GetValue(Project:Project_ID)#_#GetValue(Global:Temp1),Activity_ID,
 GetNthElem(Worksections:Activities_ID, i));
Let [ j (i+ 4)]
SetValue(Worksections:DDEItem, GetValue(Worksections:DDEItem)#j);
RemoteGet(GetValue(Worksections:DDEItem),SPJWIN, Worksections:ScheduleFileName,
 GetValue(Project:Project_ID)#_#GetValue(Global:Temp1), MaterialsList);
SetValue(Worksections,DDEItem, SPJDDE_);
SetValue(GetValue(Project:Project_ID)#_#GetValue(Global:Temp1), Name,
GetValue(Project:Project_ID)#_#GetValue(Global:Temp1));
ResetValue(Global:Temp1);};
PostMessage("Data loading completed");
```

```
PostBusy(OFF);
} );

/************** METHOD: GetSuperActivity_ID  **************/
MakeMethod( Worksections, GetSuperActivity_ID, [],
CatchError(

{SetValue(Worksections, ScheduleFileName,  GetValue(Project:Project_ID)#".pj");
RemoteGet(SPJDDE_31, SPJWIN, GetValue(Worksections, ScheduleFileName),
 Worksections, Activities_ID);},
PostMessage("At the moment, this prototype deals with only one file from
Superproject. More files will be introduced in the near future")) );

/************** METHOD: LoadWorksectionList  **************/
MakeMethod( Worksections, LoadWorksectionList, [],
{ForAll [x|Worksections]
AppendToList(Worksections:WorksectionList, x);
SetSlotOption(Global:Worksection, ALLOWABLE_VALUES,
   GetValue (Worksections, WorksectionList));} );
MakeSlot( Worksections:Name );
MakeSlot( Worksections:SuperNameList );
SetSlotOption( Worksections:SuperNameList, INHERIT, FALSE );
SetSlotOption( Worksections:SuperNameList, MULTIPLE );
SetValue( Worksections:SuperNameList, "Foundations ", "Columns ", "Beams  ", "Roof ", "External Walls ",
"Roof Covering ", "Internal Walls ", "Doors ", "Windows" );
MakeSlot( Worksections:StartDate );
MakeSlot( Worksections:FinishDate );
MakeSlot( Worksections:MaterialsList );
SetSlotOption( Worksections:MaterialsList, MULTIPLE );
ClearList( Worksections:MaterialsList );
MakeSlot( Worksections:SuperStartDate );
SetSlotOption( Worksections:SuperStartDate, INHERIT, FALSE );
SetSlotOption( Worksections:SuperStartDate, MULTIPLE );
SetValue( Worksections:SuperStartDate, "30-04-96", "14-05-96", "14-05-96", "22-05-96", "22-05-96", "27-
05-96", "29-05-96", "05-06-96", "05-06-96" );
MakeSlot( Worksections:SuperFinishDate );
SetSlotOption( Worksections:SuperFinishDate, INHERIT, FALSE );
SetSlotOption( Worksections:SuperFinishDate, MULTIPLE );
SetValue( Worksections:SuperFinishDate, "13-05-96", "17-05-96", "21-05-96", "28-05-96", "30-05-96", "30-
05-96", "07-06-96", "10-06-96", "10-06-96" );
MakeSlot( Worksections:ScheduleFileName );
SetSlotOption( Worksections:ScheduleFileName, INHERIT, FALSE );
Worksections:ScheduleFileName = Pro_1.pj;
MakeSlot( Worksections:list );
SetSlotOption( Worksections:list, INHERIT, FALSE );
SetSlotOption( Worksections:list, MULTIPLE );
SetValue( Worksections:list, "Foundations", "Columns ", "Beams ", "Roof  ", "External_Walls ",
"Roof_Covering ", "Internal_Walls  ", "Doors  ", "Windows  ", "Foundations  ", "Columns  ", "Beams
", "Roof  ", "External_Walls  ", "Roof_Covering ", "Internal_Walls  ", "Doors ", "Windows" );
MakeSlot( Worksections:DDEItem );
SetSlotOption( Worksections:DDEItem, INHERIT, FALSE );
Worksections:DDEItem = SPJDDE_;
MakeSlot( Worksections:Activities_ID );
SetSlotOption( Worksections:Activities_ID, INHERIT, FALSE );
SetSlotOption( Worksections:Activities_ID, MULTIPLE );
```

```
SetValue( Worksections:Activities_ID, "  001", "  002", "  003", "  004", "  005", "  006", "  007", "
008", "  009" );
MakeSlot( Worksections:Activity_ID );
MakeSlot( Worksections:WorksectionList );
SetSlotOption( Worksections:WorksectionList, MULTIPLE );
SetValue( Worksections:WorksectionList, Pro_1_Foundations, Pro_1_Columns, Pro_1_Beams, Pro_1_Roof,
Pro_1_ExternalWalls, Pro_1_RoofCovering, Pro_1_InternalWalls, Pro_1_Doors, Pro_1_Windows );

    /*************************************
     **** CLASS: Mat_Proc_Documents
     *************************************/
MakeClass( Mat_Proc_Documents, Root );

    /*************************************
     **** CLASS: Requisitions
     *************************************/
MakeClass( Requisitions, Mat_Proc_Documents );

/************** METHOD: InputMaterialInfo  **************/
MakeMethod( Requisitions, InputMaterialInfo, [],
PostInputForm("Information on materials requested: "#SingleListBox2:Value,
Global, QuantityRequested,"Enter quantity requested",
Global, DeliveryDateRequested, "Enter Desired Delivery date",
Requisition, PackagingRequested, " Enter Type of packaging requested",
Materials, Material_Specification_Code, "Enter Specification code requested",
Requisition, SupplierList, "Enter name of prefered supplier (Facultatif)",
Global, UnitLoadRequested, "Enter unit load requested",
Global, Unit, "Enter unit used",
Global, Worksection, "Select worksection"
) );

/************** METHOD: UpdatePackagingRequestedList  **************/
MakeMethod( Requisitions, UpdatePackagingRequestedList, [],
AppendToList(Requisition:PackagingRequestedList, Requisition:PackagingRequested) );


/************** METHOD: UpdateSuppliersList  **************/
MakeMethod( Requisitions, UpdateSuppliersList, [],
AppendToList(Requisition:SuppliersList,
Requisition:SupplierList) );
MakeSlot( Requisitions:QuantityRequested );
SetSlotOption( Requisitions:QuantityRequested, MULTIPLE );
ClearList( Requisitions:QuantityRequested );
MakeSlot( Requisitions:PackagingRequested );
SetSlotOption( Requisitions:PackagingRequested, AFTER_CHANGE, UpdatePackagingRequestedList );
MakeSlot( Requisitions:UnitLoadRequested );
SetSlotOption( Requisitions:UnitLoadRequested, MULTIPLE );
ClearList( Requisitions:UnitLoadRequested );
MakeSlot( Requisitions:DeliveryDateRequested );
SetSlotOption( Requisitions:DeliveryDateRequested, MULTIPLE );
ClearList( Requisitions:DeliveryDateRequested );
MakeSlot( Requisitions:SpecificationRequested );
SetSlotOption( Requisitions:SpecificationRequested, MULTIPLE );
ClearList( Requisitions:SpecificationRequested );
MakeSlot( Requisitions:SupplierList );
```

```
SetSlotOption( Requisitions:SupplierList, AFTER_CHANGE, UpdateSuppliersList );
MakeSlot( Requisitions:MaterialListRequested );
SetSlotOption( Requisitions:MaterialListRequested, MULTIPLE );
ClearList( Requisitions:MaterialListRequested );
MakeSlot( Requisitions:PackagingRequestedList );
SetSlotOption( Requisitions:PackagingRequestedList, MULTIPLE );
ClearList( Requisitions:PackagingRequestedList );
MakeSlot( Requisitions:ReferenceNb );
MakeSlot( Requisitions:SuppliersList );
SetSlotOption( Requisitions:SuppliersList, MULTIPLE );
ClearList( Requisitions:SuppliersList );
MakeSlot( Requisitions:DateMade );
MakeSlot( Requisitions:Unit );
SetSlotOption( Requisitions:Unit, MULTIPLE );
ClearList( Requisitions:Unit );
MakeSlot( Requisitions:ToOrder? );
SetSlotOption( Requisitions:ToOrder?, ALLOWABLE_VALUES, Y, N );
MakeSlot( Requisitions:Worksection );
SetSlotOption( Requisitions:Worksection, MULTIPLE );
ClearList( Requisitions:Worksection );



/*************************************
     **** CLASS: Projects_Requisitions
     *************************************/
MakeClass( Projects_Requisitions, Requisitions );
MakeSlot( Projects_Requisitions:FileName );
MakeSlot( Projects_Requisitions:RequisitionsList );
SetSlotOption( Projects_Requisitions:RequisitionsList, INHERIT, FALSE );
SetSlotOption( Projects_Requisitions:RequisitionsList, MULTIPLE );
SetValue( Projects_Requisitions:RequisitionsList, Req_Pro_1_1, Req_Pro_1_2, Req_Pro_1_3,
Req_Pro_1_4);

    /*************************************
     **** CLASS: Orders
     *************************************/
MakeClass( Orders, Mat_Proc_Documents );

/************** METHOD: ToInstances  **************/
MakeMethod( Orders, ToInstances, [],
{Let [r GetValue(SingleListBox1,Value)]
Let [x LengthList(r:MaterialListRequested)]
Let [y LengthList(Project:OrdersList)]
{For i From 1 To 1 Do
  If (y+1) < 10 Then
  { MakeInstance(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,11),Temp);
AppendToList (SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,11),MaterialsList,
GetNthElem(r:MaterialListRequested,i));
AppendToList(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,11),QuantityRequested,GetNthEl
em(r:QuantityRequested,i));
AppendToList(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,11),PackagingRequestedList,Get
NthElem(r:PackagingRequestedList,i));
```

```
AppendToList(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,11),DeliveryDateRequested,Get
NthElem(r:DeliveryDateRequested,i));
AppendToList(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,11),SpecificationRequested,GetN
thElem(r:SpecificationRequested,i));
AppendToList(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,11),UnitLoadRequested,GetNthE
lem(r:UnitLoadRequested,i));
AppendToList(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,11),Unit,GetNthElem(r:Unit,i);
SetValue(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,11),ReqNumber,
GetValue(SingleListBox1:Value));
SetValue(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,11),OrderNumber,SubString("Ord_"#
GetValue(Project:Project_ID)#_#(y+1),1,11));
AppendToList(Project:OrdersList, SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,11));
SetValue(SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,11),SupplierName,
GetNthElem(r:SuppliersList,i));
GetInstanceList (Temp,Temp:InstanceList);}
Else ToInstances1();};} );

/************** METHOD: MakeInstances  **************/
MakeMethod( Orders, MakeInstances, [],
{
SetValue(SingleListBox1:Value:ToOrder?,Y);
SendMessage(Orders,ClearInstances);
SendMessage(Orders,ToInstances);
{Let [ y LengthList(Project:OrdersList)]
If y > 9 And y <99 Then
SendMessage(Orders, AddInstances1)   Else
SendMessage(Orders, AddInstances);};
ForAll [x|Temp] MoveInstance(x, Projects_Orders); } );

/************** METHOD: AddInstances  **************/
MakeMethod( Orders, AddInstances, [],
{Let [r GetValue(SingleListBox1:Value)]
Let [ x LengthList(r:MaterialListRequested)]
For i From 2 To x Do
{SetValue (Global:a,GetNthElem(r:SuppliersList,i));
SetValue(Global:b,SelectList(Temp:InstanceList,t,
            t:SupplierName #= GetValue(Global:a)));
If Null?(Global:b)
Then
{Let [y LengthList(Project:OrdersList)]
SetValue(Global:c, SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,11));
MakeInstance (GetValue(Global:c),Temp);
AppendToList(Project:OrdersList, GetValue(Global:c));
AppendToList(Temp,InstanceList, GetValue(Global:c));
SetValue (GetValue(Global:c),SupplierName,GetValue(Global:a));
AppendToList(GetValue(Global:c),MaterialsList,GetNthElem(r:MaterialListRequested,i));
AppendToList(GetValue(Global:c),QuantityRequested,GetNthElem(r:QuantityRequested,i));
AppendToList(GetValue(Global:c),PackagingRequestedList,GetNthElem(r:PackagingRequestedList,i));
AppendToList(GetValue(Global:c),DeliveryDateRequested,GetNthElem(r:DeliveryDateRequested,i));
AppendToList(GetValue(Global:c),SpecificationRequested,GetNthElem(r:SpecificationRequested,i));
AppendToList(GetValue(Global:c),UnitLoadRequested,GetNthElem(r:UnitLoadRequested,i));
AppendToList(GetValue(Global:c),Unit,GetNthElem(r:Unit,i));
SetValue(GetValue(Global:c), ReqNumber, GetValue(SingleListBox1:Value));
SetValue(GetValue(Global:c), OrderNumber, GetValue(Global:c));
}
```

```
Else
{If
Global:a #= Global:b:SupplierName
Then
{AppendToList(Global:b:MaterialsList, GetNthElem(r:MaterialListRequested,i));
AppendToList(Global:b:QuantityRequested,GetNthElem(r:QuantityRequested,i));
AppendToList(Global:b:PackagingRequestedList,GetNthElem(r:PackagingRequestedList,i));
AppendToList(Global:b:DeliveryDateRequested,GetNthElem(r:DeliveryDateRequested,i));
AppendToList(Global:b:SpecificationRequested,GetNthElem(r:SpecificationRequested,i));
AppendToList(Global:b:UnitLoadRequested,GetNthElem(r:UnitLoadRequested,i));
AppendToList(Global:b:Unit,GetNthElem(r:Unit,i));
SetValue(Global:b:ReqNumber, GetValue(SingleListBox1:Value));
SetValue(Global:b:OrderNumber, GetValue(Global:b));};};};} );

/************** METHOD: ClearInstances  **************/
MakeMethod( Orders, ClearInstances, [],
{{ForAll [x|Temp]DeleteInstance(x);};
ClearList(Temp:InstanceList);} );

/************** METHOD: UpdateExpeditionDate  **************/
MakeMethod( Orders, UpdateExpeditionDate, [],
SetValue(Self:ExpeditionDate, GetNthElem(Self:DeliveryDateRequested, 1)) );

/************** METHOD: AddInstances1  **************/
MakeMethod( Orders, AddInstances1, [],
{Let [r GetValue(SingleListBox1:Value)]
Let [ x LengthList(r:MaterialListRequested)]
For i From 2 To x Do
{SetValue (Global:a,GetNthElem(r:SuppliersList,i));
SetValue(Global:b,SelectList(Temp:InstanceList,t,
                t:SupplierName #= GetValue(Global:a)));
If Null?(Global:b)
Then
{Let [y LengthList(Project:OrdersList)]
SetValue(Global:c, SubString("Ord_"#GetValue(Project:Project_ID)#_#(y+1),1,12));
MakeInstance (GetValue(Global:c),Temp);
AppendToList(Project:OrdersList, GetValue(Global:c));
AppendToList(Temp,InstanceList, GetValue(Global:c));
SetValue (GetValue(Global:c),SupplierName,GetValue(Global:a));
AppendToList(GetValue(Global:c),MaterialsList,GetNthElem(r:MaterialListRequested,i));
AppendToList(GetValue(Global:c),QuantityRequested,GetNthElem(r:QuantityRequested,i));
AppendToList(GetValue(Global:c),PackagingRequestedList,GetNthElem(r:PackagingRequestedList,i));
AppendToList(GetValue(Global:c),DeliveryDateRequested,GetNthElem(r:DeliveryDateRequested,i));
AppendToList(GetValue(Global:c),SpecificationRequested,GetNthElem(r:SpecificationRequested,i));
AppendToList(GetValue(Global:c),UnitLoadRequested,GetNthElem(r:UnitLoadRequested,i));
AppendToList(GetValue(Global:c),Unit,GetNthElem(r:Unit,i));
SetValue(GetValue(Global:c), ReqNumber, GetValue(SingleListBox1:Value));
SetValue(GetValue(Global:c), OrderNumber, GetValue(Global:c));}
Else
{If
Global:a #= Global:b:SupplierName
Then
{AppendToList(Global:b:MaterialsList, GetNthElem(r:MaterialListRequested,i));
AppendToList(Global:b:QuantityRequested,GetNthElem(r:QuantityRequested,i));
AppendToList(Global:b:PackagingRequestedList,GetNthElem(r:PackagingRequestedList,i));
```

```
AppendToList(Global:b:DeliveryDateRequested,GetNthElem(r:DeliveryDateRequested,i));
AppendToList(Global:b:SpecificationRequested,GetNthElem(r:SpecificationRequested,i));
AppendToList(Global:b:UnitLoadRequested,GetNthElem(r:UnitLoadRequested,i));
AppendToList(Global:b:Unit,GetNthElem(r:Unit,i));
SetValue(Global:b:ReqNumber, GetValue(SingleListBox1:Value));
SetValue(Global:b:OrderNumber, GetValue(Global:b));};};};} );
MakeSlot( Orders:OrderNumber );
MakeSlot( Orders:Order_Date );
MakeSlot( Orders:DeliveryDateRequested );
SetSlotOption( Orders:DeliveryDateRequested, MULTIPLE );
ClearList( Orders:DeliveryDateRequested );
SetSlotOption( Orders:DeliveryDateRequested, AFTER_CHANGE, UpdateExpeditionDate );
MakeSlot( Orders:Issue_Against );
SetSlotOption( Orders:Issue_Against, ALLOWABLE_VALUES, Drawings, Site_Request, Material_Loss,
Bills_of_Quantities, Designer_Request );
MakeSlot( Orders:MaterialsList );
SetSlotOption( Orders:MaterialsList, MULTIPLE );
ClearList( Orders:MaterialsList );
MakeSlot( Orders:SupplierName );
MakeSlot( Orders:QuantityRequested );
SetSlotOption( Orders:QuantityRequested, MULTIPLE );
ClearList( Orders:QuantityRequested );
MakeSlot( Orders:PackagingRequestedList );
SetSlotOption( Orders:PackagingRequestedList, MULTIPLE );
ClearList( Orders:PackagingRequestedList );
MakeSlot( Orders:SpecificationRequested );
SetSlotOption( Orders:SpecificationRequested, MULTIPLE );
ClearList( Orders:SpecificationRequested );
MakeSlot( Orders:Unit );
SetSlotOption( Orders:Unit, MULTIPLE );
ClearList( Orders:Unit );
MakeSlot( Orders:UnitLoadRequested );
SetSlotOption( Orders:UnitLoadRequested, MULTIPLE );
ClearList( Orders:UnitLoadRequested );
MakeSlot( Orders:ReqNumber );
MakeSlot( Orders:AmmendmentNumber );
SetSlotOption( Orders:AmmendmentNumber, VALUE_TYPE, NUMBER );
SetSlotOption( Orders:AmmendmentNumber, MINIMUM_VALUE, 0 );
SetSlotOption( Orders:AmmendmentNumber, MAXIMUM_VALUE, 20 );
Orders:AmmendmentNumber = 0;
MakeSlot( Orders:SentToSupplier? );
SetSlotOption( Orders:SentToSupplier?, ALLOWABLE_VALUES, Y, N );
Orders:SentToSupplier? = N;
MakeSlot( Orders:ExpeditionStatus );
SetSlotOption( Orders:ExpeditionStatus, ALLOWABLE_VALUES, NotDefined, AsPlanned, Delayed,
Cancelled );
Orders:ExpeditionStatus = NotDefined;
MakeSlot( Orders:ExpeditionDate );

    /*************************************
     **** CLASS: Projects_Orders
     *************************************/
MakeClass( Projects_Orders, Orders );
MakeSlot( Projects_Orders:Order_Item_Code );
MakeSlot( Projects_Orders:Quantity );
```

```
/*************************************
 **** CLASS: Temp
 *************************************/
MakeClass( Temp, Orders );
MakeSlot( Temp:InstanceList );
SetSlotOption( Temp:InstanceList, MULTIPLE );
SetValue( Temp:InstanceList, Ord_Pro_1_19 );

/*************************************
 **** CLASS: Date
 *************************************/
MakeClass( Date, Root );
MakeSlot( Date:Day );
SetSlotOption( Date:Day, VALUE_TYPE, NUMBER );
MakeSlot( Date:Month );
SetSlotOption( Date:Month, VALUE_TYPE, NUMBER );
MakeSlot( Date:Year );
SetSlotOption( Date:Year, VALUE_TYPE, NUMBER );
MakeSlot( Date:KappaYear );
SetSlotOption( Date:KappaYear, VALUE_TYPE, NUMBER );
MakeSlot( Date:PreYear );
SetSlotOption( Date:PreYear, INHERIT, FALSE );
SetSlotOption( Date:PreYear, VALUE_TYPE, NUMBER );
Date:PreYear = 19;

/*************************************************************/
/**       ALL INSTANCES ARE SAVED BELOW          **/
/*************************************************************/

/************** METHOD: Get_Excel_Data  **************/
MakeMethod( Global, Get_Excel_Data, [],
{
ClearList(SLB1:AllowableValues);
ResetImage(SLB1);
CatchError(RemoteGet("R2C5:R7C5",Excel,PROJECTS.xls,SLB1,
AllowableValues),
{
LoadExcelProgram( );
CatchError (RemoteGet("R2C6:R7C6", Excel, PROJECTS.xls,
SLB1, AllowableValues),
NoActionTaken);
});
ResetImage(SLB1);
} );

/************** METHOD: Read_Projects  **************/
MakeMethod( Global, Read_Projects, [],
 {
   If KnownValue?( Self:slot )
    Then {
       If ( Self:slot #= Excel )
            Then SendMessage( Self, Get_Excel_Data )
            Else {
               PostMessage( "Interface to ", Self:slot,
```

```
                " not implemented in this demo.  See configuration demo for additional interfaces." );
            ClearList( SLB1:AllowableValues );
            ResetImage( SLB1 );
            };
        };
    SetWindowTitle( Session1, "Double click on Applicant Name to Select." );
    } );

 /************** METHOD: InputMaterialListInBox  **************/
 MakeMethod( Global, InputMaterialListInBox, [],
 SetValue(MultipleListBox1:AllowableValues, Global:MaterialTemp) );

 /************** METHOD: UpdateQuantityRequested  **************/
 MakeMethod( Global, UpdateQuantityRequested, [],
 AppendToList(Requisition:QuantityRequested, Global:QuantityRequested) );

 /************** METHOD: UpdateDeliveryDateRequestedList  **************/
 MakeMethod( Global, UpdateDeliveryDateRequestedList, [],
 {AppendToList(Requisition:DeliveryDateRequested, Global:DeliveryDateRequested);
 CheckDate();} );

 /************** METHOD: UpdatePackagingRequestedList  **************/
 MakeMethod( Global, UpdatePackagingRequestedList, [],
 AppendToList(Requisition:QuantityRequestedList, Requisition:PackagingRequested) );

 /************** METHOD: UpdateSpecificaRequestedList  **************/
 MakeMethod( Global, UpdateSpecificaRequestedList, [],
 AppendToList(Requisition:SpecificationRequested, Global:SpecificationRequested) );

 /************** METHOD: UpdateSupplierPreffered  **************/
 MakeMethod( Global, UpdateSupplierPreffered, [],

 SetSlotOption(Requisition,SupplierList, ALLOWABLE_VALUES, Global:SupplierList) );

 /************** METHOD: UpdateUnitLoadRequested  **************/
 MakeMethod( Global, UpdateUnitLoadRequested, [],
 AppendToList(Requisition:UnitLoadRequested, Global:UnitLoadRequested) );

 /************** METHOD: MaintainReqMaterials  **************/
 MakeMethod( Global, MaintainReqMaterials, [],
 Let [p LengthList(Global:MaterialsRequested)]
 If p >0
 Then
 {For b From 1 To p Do
 If
 GetNthElem(Global:MaterialsRequested, b) #= GetNthElem(Global:MaterialsRequested, (b+1))
 Then
 RemoveNthElem(Global:MaterialsRequested, (b+1)) ;} );

/************** METHOD: UpdateUnit  **************/
MakeMethod( Global, UpdateUnit, [],
AppendToList(Requisition:Unit,
Global:Unit) );

/************** METHOD: CheckSectionNumber  **************/
```

```
MakeMethod( Global, CheckSectionNumber, [],
{SetValue(Global:SectionTest, Null?(Global:SectionNumber));

If {Global:SectionNumber <=0 Or
GetValue(Global:SectionTest) #= TRUE}
Then { PostMessage(
"Section numbers must be greater than 0.Try again");
ResetValue(Global:SectionNumber);};
} );

/************** METHOD: UpdateExpeditionStatus  **************/
MakeMethod( Global, UpdateExpeditionStatus, [],
{SetValue(SingleListBox4:Value:ExpeditionStatus, Global:Expedition);

If  Global:Expedition #= Cancelled Or
 Global:Expedition #= Delayed

Then PostMessage("The selected order is "#GetValue(Global:Expedition)#".
Send a message to the parties involved");} );

/************** METHOD: GetSuppliers  **************/
MakeMethod( Global, GetSuppliers, [],
SendMessage(Suppliers,GetSuppliers) );

/************** METHOD: UpdateWorksection  **************/
MakeMethod( Global, UpdateWorksection, [],
AppendToList(Requisition:Worksection, Global:Worksection) );
MakeSlot( Global:Temp );
SetSlotOption( Global:Temp, MULTIPLE );
MakeSlot( Global:Temp2 );
SetSlotOption( Global:Temp2, MULTIPLE );
ClearList( Global:Temp2 );
MakeSlot( Global:Temp1 );
MakeSlot( Global:I );
SetSlotOption( Global:I, MULTIPLE );
MakeSlot( Global:Database );
Global:Database = Excel;
MakeSlot( Global:Fileds );
SetSlotOption( Global:Fileds, MULTIPLE );
SetValue( Global:Fileds, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, X, Y, Z );
MakeSlot( Global:Slots );
SetSlotOption( Global:Slots, MULTIPLE );
SetValue( Global:Slots, Client_Name, Finish_Date, Project_Address, Project_ID, Project_Name, Start_Date
);
MakeSlot( Global:TempValues );
SetSlotOption( Global:TempValues, MULTIPLE );
SetValue( Global:TempValues, "John Smith", "12/01/96", "25 Liverpool Road", Pro_1, "House Restoration",
"12/12/93", NULL, Material_Name, Materials_ID_number, MaterialsPackaging, Material_Unit_Load,
Material_Specification_Code, Unit_Measurement, Cost_PerUnit, Material_Vol_Unit, Nb_Pack_In_UnitLoad,
Material_Vul_Weather, Materials_Waste_Allowance, Materials_Substitutes, Material_Storage_Duration,
Materials_Order_Lead_Time, Taxes );
MakeSlot( Global:Xlist );
SetSlotOption( Global:Xlist, MULTIPLE );
ClearList( Global:Xlist );
MakeSlot( Global:XText );
```

```
SetSlotOption( Global:XText, MULTIPLE );
ClearList( Global:XText );
MakeSlot( Global:Date_Time );
Global:Date_Time = "06/11/96";
SetSlotOption( Global:Date_Time, IMAGE, Edit3 );
MakeSlot( Global:MaterialTemp );
SetSlotOption( Global:MaterialTemp, MULTIPLE );
SetValue( Global:MaterialTemp, Bricks, Concrete, Wood, Blocks, Cement, Sand, Doors, Windows );
MakeSlot( Global:time );
Global:time = NULL;
MakeSlot( Global:TempImageValue );
MakeSlot( Global:QuantityRequested );
Global:QuantityRequested = 1200;
SetSlotOption( Global:QuantityRequested, AFTER_CHANGE, UpdateQuantityRequested );
MakeSlot( Global:PackagingRequested );
MakeSlot( Global:DeliveryDateRequested );
Global:DeliveryDateRequested = 121196;
SetSlotOption( Global:DeliveryDateRequested, AFTER_CHANGE, UpdateDeliveryDateRequestedList );
MakeSlot( Global:SpecificationRequested );
Global:SpecificationRequested = 54;
SetSlotOption( Global:SpecificationRequested, AFTER_CHANGE, UpdateSpecificaRequestedList );
MakeSlot( Global:SupplierList );
SetSlotOption( Global:SupplierList, MULTIPLE );
SetValue( Global:SupplierList, "Brick Suppliers", "Smith and Sons", "M&G BrickCutters", "Sellite Blocks
Ltd", "Tarmac Bricks and tiles", "Redland Bricks", "Sand Masters" );
SetSlotOption( Global:SupplierList, AFTER_CHANGE, UpdateSupplierPreffered );
MakeSlot( Global:UnitLoadRequested );
Global:UnitLoadRequested = x;
SetSlotOption( Global:UnitLoadRequested, AFTER_CHANGE, UpdateUnitLoadRequested );
MakeSlot( Global:WinHeight );
Global:WinHeight = 12;
MakeSlot( Global:WinDepth );
Global:WinDepth = 465;
MakeSlot( Global:WinWidth );
Global:WinWidth = 45;
MakeSlot( Global:DoorHeight );
Global:DoorHeight = 67;
MakeSlot( Global:DoorWidth );
Global:DoorWidth = 7;
MakeSlot( Global:DoorDepth );
Global:DoorDepth = 12;
MakeSlot( Global:DoorMaterial );
MakeSlot( Global:Loto );
SetSlotOption( Global:Loto, MULTIPLE );
SetValue( Global:Loto, 44, 36, 23, 26, 2, 07 );
SetSlotOption( Global:Loto, IMAGE, MultipleListBox2 );
MakeSlot( Global:MaterialsRequested );
SetSlotOption( Global:MaterialsRequested, MULTIPLE );
ClearList( Global:MaterialsRequested );
SetSlotOption( Global:MaterialsRequested, IMAGE, MultipleListBox3 );
MakeSlot( Global:NAME );
SetSlotOption( Global:NAME, ALLOWABLE_VALUES, LEILA, ANDY, DAVID, SOADY, JAMES );
Global:NAME = LEILA;
SetSlotOption( Global:NAME, IMAGE, Edit7 );
MakeSlot( Global:NAMES_ALLOWABLE );
```

```
SetSlotOption( Global:NAMES_ALLOWABLE, MULTIPLE );
SetValue( Global:NAMES_ALLOWABLE, LEILA, ANDY, DAVID, SOADY, JAMES );
MakeSlot( Global:numbers );
SetSlotOption( Global:numbers, MULTIPLE );
SetValue( Global:numbers, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
25, 26, 27, 28, 29 );
MakeSlot( Global:Names );
SetSlotOption( Global:Names, MULTIPLE );
SetValue( Global:Names, Pro_4.pj );
MakeSlot( Global:Alpha );
SetSlotOption( Global:Alpha, MULTIPLE );
SetValue( Global:Alpha, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, _, a, b,
c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z );
MakeSlot( Global:USERNAME );
SetSlotOption( Global:USERNAME, ALLOWABLE_VALUES, BLTLEILA, BLTANDY, BLTDAVID,
BLTSOADY, BLTJAMES );
Global:USERNAME = BLTLEILA;
SetSlotOption( Global:USERNAME, IMAGE, Edit9 );
MakeSlot( Global:USER_ALLOWABLE );
SetSlotOption( Global:USER_ALLOWABLE, MULTIPLE );
SetValue( Global:USER_ALLOWABLE, BLTLEILA, LTANDY, BLTDAVID, BLTSOADY, BLTJAMES,
BLT );
MakeSlot( Global:Unit );
Global:Unit = 1;
SetSlotOption( Global:Unit, AFTER_CHANGE, UpdateUnit );
MakeSlot( Global:a );
Global:a = "Brick Suppliers";
MakeSlot( Global:b );
Global:b = Ord_Pro_1_19;
MakeSlot( Global:c );
Global:c = Ord_Pro_1_17;
MakeSlot( Global:SupplierName );
Global:SupplierName = "Brick Suppliers";
SetSlotOption( Global:SupplierName, AFTER_CHANGE, GetSuppliers );
MakeSlot( Global:SectionNumber );
SetSlotOption( Global:SectionNumber, VALUE_TYPE, NUMBER );
Global:SectionNumber = 1;
MakeSlot( Global:SectionTest );
Global:SectionTest = TRUE;
MakeSlot( Global:NewQuantity );
SetSlotOption( Global.NewQuantity, VALUE_TYPE, NUMBER );
Global:NewQuantity = 28000;
SetSlotOption( Global:NewQuantity, IMAGE, Edit11 );
MakeSlot( Global:NewDate );
Global:NewDate = 130996;
SetSlotOption( Global:NewDate, IMAGE, Edit14 );
MakeSlot( Global:DateCheck );
Global:DateCheck = FALSE;
MakeSlot( Global:Box4Name );
Global:Box4Name = Ord_Pro_1_19;
MakeSlot( Global:Expedition );
SetSlotOption( Global:Expedition, ALLOWABLE_VALUES, NotDefined, AsPlanned, Delayed, Cancelled );
Global:Expedition = NotDefined;
SetSlotOption( Global:Expedition, AFTER_CHANGE, UpdateExpeditionStatus );
SetSlotOption( Global:Expedition, IMAGE, RadioButtonGroup1 );
```

```
MakeSlot( Global:ExpeditionDate );
Global:ExpeditionDate = 150296;
MakeSlot( Global:WorksectionsOrders );
SetSlotOption( Global:WorksectionsOrders, MULTIPLE );
ClearList( Global:WorksectionsOrders );
SetSlotOption( Global:WorksectionsOrders, IMAGE, MultipleListBox4 );
MakeSlot( Global:Order );
Global:Order = Ord_Pro_1_19;
MakeSlot( Global:OrdersList );
SetSlotOption( Global:OrdersList, MULTIPLE );
SetValue( Global:OrdersList, Ord_Pro_1_19 );
MakeSlot( Global:OrderList );
SetSlotOption( Global:OrderList, MULTIPLE );
SetValue( Global:OrderList, Ord_Pro_1_4, Ord_Pro_1_5, Ord_Pro_1_6,);
MakeSlot( Global:A1 );
Global:A1 = Pro_2.PJ;
MakeSlot( Global:Worksection );
SetSlotOption( Global:Worksection, ALLOWABLE_VALUES, Pro_1_Foundations, Pro_1_Columns,
Pro_1_Beams, Pro_1_Roof, Pro_1_ExternalWalls, Pro_1_RoofCovering, Pro_1_InternalWalls,
Pro_1_Doors, Pro_1_Windows );
Global:Worksection = Pro_1_ExternalWalls;
SetSlotOption( Global:Worksection, AFTER_CHANGE, UpdateWorksection );

    /************************************
     **** INSTANCE: SESSION
     ************************************/
SESSION:X = 110;
SESSION:Y = -13;
SESSION:Title = SESSION;
SESSION:SessionNumber = 0;
SESSION:Width = 397;
SESSION:Height = 477;
SESSION:Menu = FALSE;
SESSION:Visible = TRUE;
SESSION:State = NORM;
SetValue( SESSION:BackgroundColor, 192, 192, 192 );
SESSION:Titlebar = FALSE;
SESSION:Sizebox = FALSE;
ResetWindow( SESSION );

    /************************************
     **** INSTANCE: Supplier
     ************************************/
MakeInstance( Supplier, Suppliers );
Supplier:StreetNumber = 15;
Supplier:StreetName = "Stanly Road";
Supplier:Town = Liverpool;
Supplier:PostalCode = "L5 3UG";
Supplier:ContactName = "Ms Smith";
Supplier:Telephone = "(0151)233 0568";
Supplier:Faxcimile = "(0151)233 0568";
Supplier:IDNumber = sup_01;
Supplier:Name = "Brick Suppliers";

    /************************************
```

```
    **** INSTANCE: Order
    ************************************/
MakeInstance( Order, Orders );
MakeSlot( Order:Issue_Against );
SetSlotOption( Order:Issue_Against, ALLOWABLE_VALUES, Drawings, Site_Request, Material_Loss,
Bills_of_Quantities, Designer_Request );
Order:Issue_Against = Material_Loss;
SetSlotOption( Order:Issue_Against, IMAGE, SingleListBox3_0, SingleListBox3_2 );
MakeSlot( Order:SupplierList );
SetSlotOption( Order:SupplierList, MULTIPLE );
ClearList( Order:SupplierList );
ClearList( Order:MaterialsList );

    /************************************
    **** INSTANCE: Material
    ************************************/
MakeInstance( Material, Materials );

    /************************************
    **** INSTANCE: Requisition
    ************************************/
MakeInstance( Requisition, Requisitions );
MakeSlot( Requisition:PackagingRequested );
SetSlotOption( Requisition:PackagingRequested, ALLOWABLE_VALUES, Packaged, Loose, Palleted );
Requisition:PackagingRequested = Packaged;
SetSlotOption( Requisition:PackagingRequested, AFTER_CHANGE, UpdatePackagingRequestedList );
MakeSlot( Requisition:QuantityRequested );
SetSlotOption( Requisition:QuantityRequested, MULTIPLE );
ClearList( Requisition:QuantityRequested );
MakeSlot( Requisition:DeliveryDateRequested );
SetSlotOption( Requisition:DeliveryDateRequested, MULTIPLE );
ClearList( Requisition:DeliveryDateRequested );
MakeSlot( Requisition:UnitLoadRequested );
SetSlotOption( Requisition:UnitLoadRequested, MULTIPLE );
ClearList( Requisition:UnitLoadRequested );
MakeSlot( Requisition:SpecificationRequested );
SetSlotOption( Requisition:SpecificationRequested, MULTIPLE );
ClearList( Requisition:SpecificationRequested );
MakeSlot( Requisition:PackagingRequestedList );
SetSlotOption( Requisition:PackagingRequestedList, MULTIPLE );
ClearList( Requisition:PackagingRequestedList );
MakeSlot( Requisition:MaterialListRequested );
SetSlotOption( Requisition:MaterialListRequested, MULTIPLE );
ClearList( Requisition:MaterialListRequested );
MakeSlot( Requisition:SupplierList );
SetSlotOption( Requisition:SupplierList, ALLOWABLE_VALUES, "Brick Suppliers", "Smith and Sons",
"M&G BrickCutters", "Sellite Blocks Ltd", "Tarmac Bricks and tiles", "Redland Bricks", "Sand Masters" );
Requisition:SupplierList = "Brick Suppliers";
SetSlotOption( Requisition:SupplierList, AFTER_CHANGE, UpdateSuppliersList );
MakeSlot( Requisition:SuppliersList );
SetSlotOption( Requisition:SuppliersList, MULTIPLE );
ClearList( Requisition:SuppliersList );
MakeSlot( Requisition:Unit );
SetSlotOption( Requisition:Unit, MULTIPLE );
ClearList( Requisition:Unit );
```

```
MakeSlot( Requisition:Worksection );
SetSlotOption( Requisition:Worksection, MULTIPLE );
ClearList( Requisition:Worksection );

    /*************************************
     **** INSTANCE: Project
     *************************************/
MakeInstance( Project, Projects );
Project:Client_Name = "John Smith";
SetSlotOption( Project:Client_Name, IMAGE, Edit2 );
Project:Project_Address = "25 Liverpool Road";
Project:Project_ID = Pro_1;
Project:Project_Name = "House Restoration";
SetSlotOption( Project:Project_Name, IMAGE, Edit1 );
Project:Start_Date = "12/12/93";
Project:Finish_Date = "12/01/96";
SetValue( Project:RequisitionsList, Req_Pro_1_1, Req_Pro_1_2, Req_Pro_1_3, Req_Pro_1_4,
Req_Pro_1_5);
ClearList( Project:WorksectionsList );
SetValue( Project:OrdersList, Ord_Pro_1_1, Ord_Pro_1_2, Ord_Pro_1_3, Ord_Pro_1_4, Ord_Pro_1_5,
Ord_Pro_1_6, Ord_Pro_1_7, Ord_Pro_1_8);


    /*************************************
     **** INSTANCE: Req_Pro_1_1
     *************************************/
MakeInstance( Req_Pro_1_1, Projects_Requisitions );
SetValue( Req_Pro_1_1:DeliveryDateRequested, 050396 );
SetValue( Req_Pro_1_1:MaterialListRequested, Concrete );
SetValue( Req_Pro_1_1:PackagingRequestedList, Bulk, Bulk );
SetValue( Req_Pro_1_1:QuantityRequested, 7, 9 );
SetValue( Req_Pro_1_1:SpecificationRequested, q4_Reinforced_Concrete, q4_Reinforced_Concrete );
SetValue( Req_Pro_1_1:UnitLoadRequested, x, x );
SetValue( Req_Pro_1_1:SuppliersList, "Carter Concrete", "Carter Concrete" );
SetValue( Req_Pro_1_1:Unit, m3, m3 );
Req_Pro_1_1:ToOrder? = Y;
Req_Pro_1_1:ReferenceNb = Req_Pro_1_1;
Req_Pro_1_1:DateMade = "14/2/96";


/*************************************
     **** INSTANCE: EXCEL
     *************************************/
MakeInstance( EXCEL, DDEService );
EXCEL:Executable = "C:\MSOFFICE\EXCEL\EXCEL.EXE";
EXCEL:Service = EXCEL;

    /*************************************
     **** INSTANCE: projects
     *************************************/
MakeInstance( projects, DDEService );

    /*************************************
     **** INSTANCE: projects.xls
     *************************************/
```

```
MakeInstance( projects.xls, DDEService );

    /*************************************
     **** INSTANCE: KAPPA
     *************************************/
MakeInstance( KAPPA, DDEService );
KAPPA:Executable = "C:\KAPPA\KAPPA.EXE";
KAPPA:Service = KAPPA;



/*************************************
    **** INSTANCE: Req_Pro_1_2
    *************************************/
MakeInstance( Req_Pro_1_2, Projects_Requisitions );
SetValue( Req_Pro_1_2:DeliveryDateRequested, 120396 );
SetValue( Req_Pro_1_2:MaterialListRequested, Concrete );
SetValue( Req_Pro_1_2:PackagingRequestedList, Bulk );
SetValue( Req_Pro_1_2:QuantityRequested, 6 );
SetValue( Req_Pro_1_2:SpecificationRequested, q4_Reinforced_Concrete );
SetValue( Req_Pro_1_2:UnitLoadRequested, x );
SetValue( Req_Pro_1_2:SuppliersList, "Carter Concrete" );
SetValue( Req_Pro_1_2:Unit, m3 );
Req_Pro_1_2:ToOrder? = Y;
Req_Pro_1_2:ReferenceNb = Req_Pro_1_2;
Req_Pro_1_2:DateMade = "14/2/96";

    /*************************************
     **** INSTANCE: UsedRequisition
     *************************************/
MakeInstance( UsedRequisition, Requisitions );
MakeSlot( UsedRequisition:DeliveryDateRequested );
SetSlotOption( UsedRequisition:DeliveryDateRequested, MULTIPLE );
SetValue( UsedRequisition:DeliveryDateRequested, 121196 );
MakeSlot( UsedRequisition:MaterialListRequested );
SetSlotOption( UsedRequisition:MaterialListRequested, MULTIPLE );
SetValue( UsedRequisition:MaterialListRequested, Blocks );
MakeSlot( UsedRequisition:PackagingRequestedList );
SetSlotOption( UsedRequisition:PackagingRequestedList, MULTIPLE );
SetValue( UsedRequisition:PackagingRequestedList, Packaged );
MakeSlot( UsedRequisition:QuantityRequested );
SetSlotOption( UsedRequisition:QuantityRequested, MULTIPLE );
SetValue( UsedRequisition:QuantityRequested, 1200 );
MakeSlot( UsedRequisition:SpecificationRequested );
SetSlotOption( UsedRequisition:SpecificationRequested, MULTIPLE );
SetValue( UsedRequisition:SpecificationRequested, Cellular );
MakeSlot( UsedRequisition:UnitLoadRequested );
SetSlotOption( UsedRequisition:UnitLoadRequested, MULTIPLE );
SetValue( UsedRequisition:UnitLoadRequested, x );
MakeSlot( UsedRequisition:SuppliersList );
SetSlotOption( UsedRequisition:SuppliersList, MULTIPLE );
SetValue( UsedRequisition:SuppliersList, "Brick Suppliers" );
MakeSlot( UsedRequisition:ReferenceNb );
UsedRequisition:ReferenceNb = NULL;
```

```
MakeSlot( UsedRequisition:DateMade );
UsedRequisition:DateMade = "24/11/96";
MakeSlot( UsedRequisition:Unit );
SetSlotOption( UsedRequisition:Unit, MULTIPLE );
SetValue( UsedRequisition:Unit, 1 );
UsedRequisition:ToOrder? = N;
SetValue( UsedRequisition:Worksection, Pro_1_ExternalWalls );



    /************************************
     **** INSTANCE: FileSelection
     ************************************/
MakeInstance( FileSelection, NewMenu );
SetValue( FileSelection:Choices, File );

    /************************************
     **** INSTANCE: File
     ************************************/
MakeInstance( File, NewMenu );


/************** METHOD: Save  **************/
MakeMethod( File, Save, [],
Exit() );

/************** METHOD: Exit  **************/
MakeMethod( File, Exit, [],
Exit1() );

/************** METHOD: Print  **************/
MakeMethod( File, Print, [],
PostMessage("Not Implemented Yet!") );
SetValue( File:Choices, Save, Exit, Print );

/************************************
     **** INSTANCE: SPJWIN
     ************************************/
MakeInstance( SPJWIN, DDEService );
SPJWIN:Executable = "C:\SPJWIN3\SPJWIN.EXE";
SPJWIN:Service = SPJWIN;

       /************************************
     **** INSTANCE: TransformSelection
     ************************************/
MakeInstance( TransformSelection, NewMenu );
SetValue( TransformSelection:Choices, Transform );

    /************************************
     **** INSTANCE: Transform
     ************************************/
MakeInstance( Transform, NewMenu );

/************** METHOD: ExpediteOrder  **************/
```

```
MakeMethod( Transform, ExpediteOrder, [],
Expedition() );

/************** METHOD: RequisitionToOrder  **************/
MakeMethod( Transform, RequisitionToOrder, [],
{ShowWindow(Session10);
MaximizeWindow(Session10);} );

/************** METHOD: ChangeOrder  **************/
MakeMethod( Transform, ChangeOrder, [],
{ShowWindow(Session12);
ShowWindow(Session13);} );
SetValue( Transform:Choices, RequisitionToOrder, ChangeOrder, ExpediteOrder );

    /*************************************
     **** INSTANCE: DataEntrySelection
     *************************************/
MakeInstance( DataEntrySelection, NewMenu );
SetValue( DataEntrySelection:Choices, DataEntry );

    /*************************************
     **** INSTANCE: DataEntry
     *************************************/
MakeInstance( DataEntry, NewMenu );

/************** METHOD: Requisition  **************/
MakeMethod( DataEntry, Requisition, [],
{ShowWindow(Session3);
MaximizeWindow(Session3);} );

/************** METHOD: Order  **************/
MakeMethod( DataEntry, Order, [],
{ShowWindow(Session10);
MaximizeWindow(Session10);} );
SetValue( DataEntry:Choices, Requisition, Order );

    /*************************************
     **** INSTANCE: ViewSelection
     *************************************/
MakeInstance( ViewSelection, NewMenu );
SetValue( ViewSelection:Choices, View );

    /*************************************
     **** INSTANCE: View
     *************************************/
MakeInstance( View, NewMenu );

/************** METHOD: Requisitions  **************/
MakeMethod( View, Requisitions, [],
{ShowWindow(Session10);
MaximizeWindow(Session10);} );

/************** METHOD: Orders  **************/
MakeMethod( View, Orders, [],
{ShowWindow(Session12);
```

```
PositionWindow(Session12, 0,0,640,350);

ClearList(SingleListBox3:AllowableValues);
AppendToList(SingleListBox3:AllowableValues , GetValue(Project:OrdersList)) ;
ResetImage(SingleListBox3); } );

/************** METHOD: Materials_List  **************/
MakeMethod( View, Materials_List, [],
{ShowWindow(Session2);
MaximizeWindow(Session2);} );

/************** METHOD: Projects_List  **************/
MakeMethod( View, Projects_List, [],
{ShowWindow(Session1);
MaximizeWindow(Session1);} );
SetValue( View:Choices, Requisitions, Orders, Materials_List, Projects_List );

    /*************************************
     **** INSTANCE: CommunicationSelection
     *************************************/
MakeInstance( CommunicationSelection, NewMenu );
SetValue( CommunicationSelection:Choices, Communication );

    /*************************************
     **** INSTANCE: Communication
     *************************************/
MakeInstance( Communication, NewMenu );

/************** METHOD: Send?  **************/
MakeMethod( Communication, Send?, [],
NotImplemented() );

/************** METHOD: Phone  **************/
MakeMethod( Communication, Phone, [],
NotImplemented() );

/************** METHOD: Fax  **************/
MakeMethod( Communication, Fax, [],
NotImplemented() );

/************** METHOD: EMail  **************/
MakeMethod( Communication, EMail, [],
NotImplemented() );

/************** METHOD: EDI  **************/
MakeMethod( Communication, EDI, [],
NotImplemented() );

/************** METHOD: Post  **************/
MakeMethod( Communication, Post, [],
NotImplemented() );
SetValue( Communication:Choices, Send?, Phone, Fax, EMail, EDI, Post );

    /*************************************
     **** INSTANCE: ProjectMenuSelection
```

```
     ************************************/
MakeInstance( ProjectMenuSelection, NewMenu );
SetValue( ProjectMenuSelection:Choices, ProjectSelection );

    /************************************
     **** INSTANCE: ProjectSelection
     ************************************/
MakeInstance( ProjectSelection, NewMenu );
SetValue( ProjectSelection:Choices, File, Transform, DataEntry, View, Communication, Help );

/************************************
     **** INSTANCE: Help
     ************************************/
MakeInstance( Help, NewMenu );

/************** METHOD: AboutPrototype  **************/
MakeMethod( Help, AboutPrototype, [],
NotImplemented() );

/************** METHOD: SearchForHelpOn  **************/
MakeMethod( Help, SearchForHelpOn, [],
NotImplemented() );
SetValue( Help:Choices, AboutPrototype, SearchForHelpOn );

    /************************************
     **** INSTANCE: HelpSelection
     ************************************/
MakeInstance( HelpSelection, NewMenu );

/************** METHOD: Help  **************/
MakeMethod( HelpSelection, Help, [],
NotImplemented() );
SetValue( HelpSelection:Choices, Help );

    /************************************
     **** INSTANCE: ProjectSelection2
     ************************************/
MakeInstance( ProjectSelection2, NewMenu );

/************** METHOD: Help  **************/
MakeMethod( ProjectSelection2, Help, [],
NotImplemented() );
SetValue( ProjectSelection2:Choices, File, Transform, DataEntry, View, Help );

    /************************************
     **** INSTANCE: Ord_Pro_1_1
     ************************************/
MakeInstance( Ord_Pro_1_1, Projects_Orders );
SetValue( Ord_Pro_1_1:MaterialsList, Concrete );
SetValue( Ord_Pro_1_1:QuantityRequested, 7 );
SetValue( Ord_Pro_1_1:PackagingRequestedList, Bulk );
SetValue( Ord_Pro_1_1:DeliveryDateRequested, 050396 );
Ord_Pro_1_1:ExpeditionDate = 050396;
SetValue( Ord_Pro_1_1:SpecificationRequested, q4_Reinforced_Concrete );
SetValue( Ord_Pro_1_1:UnitLoadRequested, x );
```

```
SetValue( Ord_Pro_1_1:Unit, m3 );
Ord_Pro_1_1:ReqNumber = Req_Pro_1_1;
Ord_Pro_1_1:OrderNumber = Ord_Pro_1_1;
Ord_Pro_1_1:SupplierName = "Carter Concrete";
Ord_Pro_1_1:ExpeditionStatus = NotDefined;

    /************************************
     **** INSTANCE: Ord_Pro_1_2
     ************************************/
MakeInstance( Ord_Pro_1_2, Projects_Orders );
SetValue( Ord_Pro_1_2:MaterialsList, Concrete );
SetValue( Ord_Pro_1_2:QuantityRequested, 6 );
SetValue( Ord_Pro_1_2:PackagingRequestedList, Bulk );
SetValue( Ord_Pro_1_2:DeliveryDateRequested, 120396 );
Ord_Pro_1_2:ExpeditionDate = 120396;
SetValue( Ord_Pro_1_2:SpecificationRequested, q4_Reinforced_Concrete );
SetValue( Ord_Pro_1_2:UnitLoadRequested, x );
SetValue( Ord_Pro_1_2:Unit, m3 );
Ord_Pro_1_2:ReqNumber = Req_Pro_1_2;
Ord_Pro_1_2:OrderNumber = Ord_Pro_1_2;
Ord_Pro_1_2:SupplierName = "Carter Concrete";
Ord_Pro_1_2:ExpeditionStatus = AsPlanned;

    /************************************
     **** INSTANCE: Req_Pro_1_3
     ************************************/
MakeInstance( Req_Pro_1_3, Projects_Requisitions );
SetValue( Req_Pro_1_3:DeliveryDateRequested, 250396 );
SetValue( Req_Pro_1_3:MaterialListRequested, Blocks );
SetValue( Req_Pro_1_3:PackagingRequestedList, Packaged );
SetValue( Req_Pro_1_3:QuantityRequested, 13000 );
SetValue( Req_Pro_1_3:SpecificationRequested, Solid );
SetValue( Req_Pro_1_3:UnitLoadRequested, x );
SetValue( Req_Pro_1_3:SuppliersList, "Brick Suppliers" );
SetValue( Req_Pro_1_3:Unit, 1 );
Req_Pro_1_3:ToOrder? = Y;
Req_Pro_1_3:ReferenceNb = Req_Pro_1_3;
Req_Pro_1_3:DateMade = "26/02/96";

    /************************************
     **** INSTANCE: Ord_Pro_1_3
     ************************************/
MakeInstance( Ord_Pro_1_3, Projects_Orders );
SetValue( Ord_Pro_1_3:MaterialsList, Blocks );
SetValue( Ord_Pro_1_3:QuantityRequested, 13000 );
SetValue( Ord_Pro_1_3:PackagingRequestedList, Packaged );
SetValue( Ord_Pro_1_3:DeliveryDateRequested, 250396 );
Ord_Pro_1_3:ExpeditionDate = 250396;
SetValue( Ord_Pro_1_3:SpecificationRequested, Solid );
SetValue( Ord_Pro_1_3:UnitLoadRequested, x );
SetValue( Ord_Pro_1_3:Unit, 1 );
Ord_Pro_1_3:ReqNumber = Req_Pro_1_3;
Ord_Pro_1_3:OrderNumber = Ord_Pro_1_3;
Ord_Pro_1_3:SupplierName = "Brick Suppliers";
Ord_Pro_1_3:ExpeditionStatus = Delayed;
```

```
    /************************************
     **** INSTANCE: Pro_1_Foundations
     ************************************/
MakeInstance( Pro_1_Foundations, Worksections );
Pro_1_Foundations:StartDate = "30-04-96";
Pro_1_Foundations:FinishDate = "13-05-96";
Pro_1_Foundations:Activity_ID = "  001";
SetValue(                    Pro_1_Foundations:MaterialsList,                    "q4_Reinf_Con
" );
Pro_1_Foundations:Name = Pro_1_Foundations;

    /************************************
     **** INSTANCE: Pro_1_Columns
     ************************************/
MakeInstance( Pro_1_Columns, Worksections );
Pro_1_Columns:StartDate = "14-05-96";
Pro_1_Columns:FinishDate = "17-05-96";
Pro_1_Columns:Activity_ID = "  002";
SetValue(                    Pro_1_Columns:MaterialsList,                    "q4_Reinf_Con
" );
Pro_1_Columns:Name = Pro_1_Columns;


    /************************************
     **** INSTANCE: TodayDate
     ************************************/
MakeInstance( TodayDate, Date );

/************** METHOD: FormatTodayDate  **************/
MakeMethod( TodayDate, FormatTodayDate, [],
{
SetValue(Self:Day, SubString(Self:Date, 1,2));
SetValue(Self:Month, SubString(Self:Date, 4,5));
SetValue(Self:KappaYear,  SubString(Self:Date,7,8));
SetValue(Self:Year, GetValue(Date:PreYear)#GetValue(Self:KappaYear));
} );
TodayDate:Day = 24;
TodayDate:Month = 11;
TodayDate:Year = 1996;
MakeSlot( TodayDate:Date );
TodayDate:Date = "24/11/96";
SetSlotOption( TodayDate:Date, AFTER_CHANGE, FormatTodayDate );
TodayDate:KappaYear = 96;

    /************************************
     **** INSTANCE: DeliveryDate
     ************************************/
MakeInstance( DeliveryDate, Date );
DeliveryDate:Day = 12;
DeliveryDate:Month = 11;
DeliveryDate:Year = 1996;
DeliveryDate:KappaYear = 96;

    /************************************
```

```
      **** INSTANCE: Pro_1_Beams
      ************************************/
MakeInstance( Pro_1_Beams, Worksections );
Pro_1_Beams:StartDate = "14-05-96";
Pro_1_Beams:FinishDate = "21-05-96";
Pro_1_Beams:Activity_ID = "  003";
SetValue( Pro_1_Beams:MaterialsList, "q4_Reinf_Con
" );
Pro_1_Beams:Name = Pro_1_Beams;

    /************************************
      **** INSTANCE: Pro_1_Roof
      ************************************/
MakeInstance( Pro_1_Roof, Worksections );
Pro_1_Roof:StartDate = "22-05-96";
Pro_1_Roof:FinishDate = "28-05-96";
Pro_1_Roof:Activity_ID = "  004";
SetValue( Pro_1_Roof:MaterialsList, "i_Timber
" );
Pro_1_Roof:Name = Pro_1_Roof;

    /************************************
      **** INSTANCE: Pro_1_ExternalWalls
      ************************************/
MakeInstance( Pro_1_ExternalWalls, Worksections );
Pro_1_ExternalWalls:StartDate = "22-05-96";
Pro_1_ExternalWalls:FinishDate = "30-05-96";
Pro_1_ExternalWalls:Activity_ID = "  005";
SetValue( Pro_1_ExternalWalls:MaterialsList, "Facing_Brick
", "Common_Block " );
Pro_1_ExternalWalls:Name = Pro_1_ExternalWalls;

    /************************************
      **** INSTANCE: Pro_1_RoofCovering
      ************************************/
MakeInstance( Pro_1_RoofCovering, Worksections );
Pro_1_RoofCovering:StartDate = "27-05-96";
Pro_1_RoofCovering:FinishDate = "30-05-96";
Pro_1_RoofCovering:Activity_ID = "  006";
SetValue( Pro_1_RoofCovering:MaterialsList, "Slates
" );
Pro_1_RoofCovering:Name = Pro_1_RoofCovering;

    /************************************
      **** INSTANCE: Pro_1_InternalWalls
      ************************************/
MakeInstance( Pro_1_InternalWalls, Worksections );
Pro_1_InternalWalls:StartDate = "29-05-96";
Pro_1_InternalWalls:FinishDate = "07-06-96";
Pro_1_InternalWalls:Activity_ID = "  007";
SetValue( Pro_1_InternalWalls:MaterialsList, "Common_Block
" );
Pro_1_InternalWalls:Name = Pro_1  InternalWalls;

    /************************************
```

```
     **** INSTANCE: Pro_1_Doors
     ************************************/
MakeInstance( Pro_1_Doors, Worksections );
Pro_1_Doors:StartDate = "05-06-96";
Pro_1_Doors:FinishDate = "10-06-96";
Pro_1_Doors:Activity_ID = "  008";
SetValue( Pro_1_Doors:MaterialsList, "622_Hinged
" );
Pro_1_Doors:Name = Pro_1_Doors;

    /************************************
     **** INSTANCE: Pro_1_Windows
     ************************************/
MakeInstance( Pro_1_Windows, Worksections );
Pro_1_Windows:StartDate = "05-06-96";
Pro_1_Windows:FinishDate = "10-06-96";
Pro_1_Windows:Activity_ID = "  009";
SetValue( Pro_1_Windows:MaterialsList, "611_Fixed
" );
Pro_1_Windows:Name = Pro_1_Windows;

/************************************
     **** INSTANCE: SLB1
     ************************************/
MakeInstance( SLB1, SingleListBox );
SLB1:X = 100;
SLB1:Y = 100;
SLB1:Width = 150;
SLB1:Height = 120;
SLB1:SessionNumber = 0;
SLB1:TabStop = FALSE;
SetSlotOption( SLB1:AllowableValues, MULTIPLE );
ClearList( SLB1:AllowableValues );
SetSlotOption( SLB1:AllowableValues, IMAGE, SingleListBox1 );
SLB1:Action = LoadProjectData;
ResetImage( SLB1 );



/*************************************************************/
/**      ALL RULES ARE SAVED BELOW          **/
/*************************************************************/

    /************************************
     **** RULE: PackagingRule1
     ************************************/
MakeRule( PackagingRule1, [],
  {
  SingleListBox2:Value #= Sheet_Timber Or SingleListBox2:Value
    #= Steel_Reinforcemnet Or SingleListBox2:Value #= Bricks
    Or SingleListBox2:Value #= Blocks Or SingleListBox2:Value
    #= Doors Or SingleListBox2:Value #= Windows;
  },
  {
  ResetSlotOption( Requisition:PackagingRequested, ALLOWABLE_VALUES );
```

```
  SetSlotOption( Requisition:PackagingRequested, ALLOWABLE_VALUES,
    Packaged, Loose, Palleted );
  } );

    /*************************************
     **** RULE: PackagingRule2
     *************************************/
MakeRule( PackagingRule2, [],
  {
  SingleListBox2:Value #= Sand Or SingleListBox2:Value #= Gravel
    Or SingleListBox2:Value #= Hardcore Or SingleListBox2:Value
    #= Ashes Or SingleListBox2:Value #= Spoil Or SingleListBox2:Value
    #= Topsoil Or SingleListBox2:Value #= Concrete;
  },
  {
  ResetSlotOption( Requisition:PackagingRequested, ALLOWABLE_VALUES );
  SetSlotOption( Requisition:PackagingRequested, ALLOWABLE_VALUES,
    Bulk );
  } );

    /*************************************
     **** RULE: PackagingRule3
     *************************************/
MakeRule( PackagingRule3, [],
  {
  SingleListBox2:Value #= Cement Or SingleListBox2:Value #=
    Plaster;
  },
  {
  ResetSlotOption( Requisition:PackagingRequested, ALLOWABLE_VALUES );
  SetSlotOption( Requisition:PackagingRequested, ALLOWABLE_VALUES,
    Bulk, Bagged, Palleted );
  } );

    /*************************************
     **** RULE: PackagingRule4
     *************************************/
MakeRule( PackagingRule4, [],
  {
  SingleListBox2:Value #= Turves Or SingleListBox2:Value #=
    Structural_Timber Or SingleListBox2:Value #= Trusses Or
    SingleListBox2:Value #= Metal_Windows Or SingleListBox2:Value
    #= Steel_Section Or SingleListBox2:Value #= Sanitary_Fittings
    Or SingleListBox2:Value #= Kitchen_Fittings Or SingleListBox2:Value
    #= Paving_Slabs Or SingleListBox2:Value #= Scaffolding
    Or SingleListBox2:Value #= Radiators;
  },
  {
  ResetSlotOption( Requisition:PackagingRequested, ALLOWABLE_VALUES );
  SetSlotOption( Requisition:PackagingRequested, ALLOWABLE_VALUES,
    Loose );
  } );

    /*************************************
     **** RULE: PackagingRule5
```

```
    ************************************/
MakeRule( PackagingRule5, [],
  {
  SingleListBox2:Value #= Joinery_Sections Or SingleListBox2:Value
    #= Service_Pipes Or SingleListBox2:Value #= Ironmongery
    Or SingleListBox2:Value #= Scaffolding_Fittings;
  },
  {
  ResetSlotOption( Requisition:PackagingRequested, ALLOWABLE_VALUES );
  SetSlotOption( Requisition:PackagingRequested, ALLOWABLE_VALUES,
    Packaged, Loose );
  } );

    /************************************
     **** RULE: PackagingRule6
     ************************************/
MakeRule( PackagingRule6, [],
  {
  SingleListBox2:Value #= Nails Or SingleListBox2:Value #= Screws
    Or SingleListBox2:Value #= Ceiling_Tiles Or SingleListBox2:Value
    #= Floor_Tiles Or SingleListBox2:Value #= Electrical_Fittings;
  },
  {
  ResetSlotOption( Requisition:PackagingRequested, ALLOWABLE_VALUES );
  SetSlotOption( Requisition:PackagingRequested, ALLOWABLE_VALUES,
    Packaged );
  } );

    /************************************
     **** RULE: PackagingRule7
     ************************************/
MakeRule( PackagingRule7, [],
  {
  SingleListBox2:Value #= Precast_Concrete Or SingleListBox2:Value
    #= Plasterboard Or SingleListBox2:Value #= Drainage_Pipes
    Or SingleListBox2:Value #= Curbs;
  },
  {
  ResetSlotOption( Requisition:PackagingRequested, ALLOWABLE_VALUES );
  SetSlotOption( Requisition:PackagingRequested, ALLOWABLE_VALUES,
    Palleted, Loose );
  } );

    /************************************
     **** RULE: MasonryBS
     ************************************/
MakeRule( MasonryBS, [],
  {
  SingleListBox2:Value #= Bricks;
  },
  {
  ResetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES );
  SetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES,
    Common_Bricks_BS3921_Cat_M, Second_Hard_Stock_Bricks_BS3921_Cat_M,
    Engineering_Bricks_BS3921_Cat_F, Facing_Bricks_BS3921_Cat_F );
```

```
    } );
      /************************************
       **** RULE: CementSfb
       ************************************/
  MakeRule( CementSfb, [],
    SingleListBox2:Value #= Cement,
    {
    ResetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES );
    SetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES,
      q2_Portland_Cement, q2_Rapid_Hardening_Cement, q2_Low_Heat_Cement,
      q2_Sulphate_Resisting_Cement, q2_White_Cement, q2_Coloured_Cement,
      q2_Blast_Furnace, q2_Slag, q2_Supersulphated, q2_High_Alumina,
      q2_Masonnery_Cement, q2_Oil_Well_Cement, q2_Cement_Sand_Mix,
      q3_Lime_Cement );
    } );

      /************************************
       **** RULE: ConcreteSfb
       ************************************/
  MakeRule( ConcreteSfb, [],
    SingleListBox2:Value #= Concrete,
    {
    ResetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES );
    SetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES,
      q4_All_In_Aggregate_Concrete, q4_Heavy_Concrete, q4_Plain_Concrete,
      q4_Mass_Concrete, q4_Reinforced_Concrete, q5_Terrazzo,
      q5_Granolithic, q6_Lightweight_Celluar_Concrete, q7_Lightweight_Aggregate_Concrete );
    } );

      /************************************
       **** RULE: SandSfb
       ************************************/
  MakeRule( SandSfb, [],
    SingleListBox2:Value #= Sand,
    ResetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES ) );

      /************************************
       **** RULE: WoodSfb
       ************************************/
  MakeRule( WoodSfb, [],
    SingleListBox2:Value #= Wood,
    {
    ResetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES );
    SetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES,
      i_Timber, i_Fir, i_Hemlock, i_Spruce, i_Whitewood, i_Larch,
      i_Cedar, i_Pine, i_Redwood, i_Walnut, i_Beech, i_Oak, i_Elm,
      i_Iroko, i_Teak, i_Mahogany, i_Sapele, i_Plywood, i_Wood_Veneers );
    } );

      /************************************
       **** RULE: WindowsSfb
       ************************************/
  MakeRule( WindowsSfb, [],
    SingleListBox2:Value #= Windows,
    {
```

```
    ResetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES );
    SetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES,
      611_Fixed, 612_Hinged, 613_Pivoting_Projected, 613_Pivoting_Louvred,
      614_Horizontal_Folding, 614_Horizontal_Sliding, 615_Vertical_Sliding,
      617_Inward_Opening, 617_Outward_Opening );
    } );

     /************************************
      **** RULE: DoorsSfb
      ************************************/
 MakeRule( DoorsSfb, [],
    SingleListBox2:Value #= Doors,
    {
    ResetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES );
    SetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES,
      622_Hinged, 623_Pivoting, 623_Louvered, 623_Projected,
      623_Revolving, 624_Sliding, 624_Folding, 624_Telescoping,
      627_Inward_Opening, 627_Outward_Opening, 628_Fire_Check,
      628_Flush, 628_Sectional, 628_Panelled, 628_Braced, 628_Framed,
      628_Casement, 628_Felexible, 628_Matchboarded );
    } );

     /************************************
      **** RULE: BlockSfb
      ************************************/
MakeRule( BlockSfb, [],
   {
   SingleListBox2:Value #= Blocks;
   },
   {
   ResetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES );
   SetSlotOption( Materials:Material_Specification_Code, ALLOWABLE_VALUES,
     Solid, Cellular, Hollow, Common, Facing, Architectural_masonry,
     Insulating, Special_Face_Blocks );
   } );
```