CAPE-OPEN

Delivering the power of component software
and open standard interfaces
in Computer-Aided Process Engineering

Open Interface Specification:

Planning and Scheduling Interface

www.colan.org

ARCHIVAL INFORMATION

Filename	Planning and Scheduling Interface Specification.doc
Authors	CO-LaN consortium
Status	Public
Date	August 2003
Version	version 2
Number of pages	109
Versioning	version 2, reviewed by Jean-Pierre Belaud, August 2003
	version 1, December 2001
Additional material	
Web location	www.colan.org
Implementation specifications version	CAPE-OPENv1-0-0.idl (CORBA) CAPE-OPENv1-0-0.zip and CAPE-OPENv1-0-0.tlb (COM)
Comments	

IMPORTANT NOTICES

Disclaimer of Warranty

CO-LaN documents and publications include software in the form of *sample code.* Any such software described or provided by CO-LaN --- in whatever form --- is provided "as-is" without warranty of any kind. CO-LaN and its partners and suppliers disclaim any warranties including without limitation an implied warrant or fitness for a particular purpose. The entire risk arising out of the use or performance of any sample code --- or any other software described by the CAPE-OPEN Laboratories Network --- remains with you.

Copyright © 2003 CO-LaN and/or suppliers. All rights are reserved unless specifically stated otherwise.

CO-LaN is a non for profit organization established under French law of 1901.

Trademark Usage

Many of the designations used by manufacturers and seller to distinguish their products are claimed as trademarks. Where those designations appear in CO-LaN publications, and the authors are aware of a trademark claim, the designations have been printed in caps or initial caps.

Microsoft, Microsoft Word, Visual Basic, Visual Basic for Applications, Internet Explorer, Windows and Windows NT are registered trademarks and ActiveX is a trademark of Microsoft Corporation.

Netscape Navigator is a registered trademark of Netscape Corporation.

Adobe Acrobat is a registered trademark of Adobe Corporation.

SUMMARY

This document, which was produced by the Global Cape Open (GCO) work-package 5.3 (Part B, Scheduling and Planning), describes the Interface Specifications for the Scheduling and Planning Package of the Global Cape Open Interface System.

The document starts with a text description of the requirements identified for a scheduling and planning component. Then is expressed in Unified Modelling Language and developed into a specification of the interfaces necessary for a CAPE-OPEN (CO) planning and scheduling component to be used by a CO executive by means of its interfaces. These specifications are provided in both COM and CORBA IDL.

The document additionally includes the prototype developed to validate the corresponding specifications.

ACKNOWLEDGEMENTS

CONTENTS

LIST OF FIGURES

1. Introduction

1.1 Overview

This document contains the interfaces conception, design and testing with a prototype.

For clarity reasons, the structure of this document is very similar to the other CO documents. First we show the main objectives set for this work, secondly the terminology and concepts associated with the scheduling problem are given. Later the analysis of the system requirements and information flows is outlined, then the interface design is presented. Finally, the scenario studied is described, and the correspondent prototype is explained.

1.2 Objectives

The CO main objective is to provide an Interface System as standard for the communication between different software components that carry up different tasks. For example, Planning & Scheduling Component to any compliant CO Executive, which performs the co-ordination and the connectivity between the different components available in one system. The goal of the Interface is to translate the requests for information or action, given by one of the components involved, into something any other component understand. In addition, the interfaces developed should be flexible enough to incorporate the latest developments in the scheduling area and to make possible the access to the component specific information. The minimum information that should be accessible to all components will be defined according to the standards ISA S95 and ISA S88.

The Interface can be thought of as a "socket" and a "plug", which are capable of exchanging information between them. The Executive and Planning & Scheduling Component do not need to know anything about the internal architecture of the other. This last point is very important in the planning and scheduling area since there is no standard method used in the industry but rather a lot of site-specific solutions exist. In this respect the Scheduling and Planning standard interface has to be as open as possible, defining the minimum information flow and functionality required to be accessed from other components and providing also a way of communicating with the information and calculation capabilities of each specific implementation.

It should be noted that unlike other CAPE-OPEN components the lack of standard methods and data requirements means that Planning and Scheduling components from different suppliers will not be interchanged as easily as, for example, a Physical Properties package.

1.3 About scheduling

1.3.1 Definitions

Process: (From S88) "A process is a sequence of chemical, physical or biological activities for the conversion, transport or storage of material or energy. Industrial manufacturing processes can generally be classified as continuous, discrete parts manufacturing, or batch"

> **Continuous process**: (From S88) "In a continuous process, materials are passed in a continuous flow through processing equipment. Once established in a steady operating state, the nature of the process is not dependent on the length of time operation"
>
> **Discrete parts manufacturing process**: (From S88)"In discrete parts manufacturing process, products are classified into production lots that are based on common raw materials, production requirements, and production histories. In a discrete parts manufacturing process,

a specified quantity of product moves as a unit (group parts) between workstations, and each part maintains its unique identity"

Batch process: (From S88) "The batch processes (…) lead to the production of finite quantities of material (batches) by subjecting quantities of input materials to a defined order of processing actions using one or more pieces of equipment. The product produced by a batch process is called a batch. Batch processes are discontinuous processes. Batch processes are neither discrete nor continuous; however, they have characteristics of both"

Resource: (From S95) "A resource is a collection of personnel, equipment, and/or material".

Unit: (From S88) "A collection of associated control modules and/or equipment modules and other process equipment in which one or more major processing activities can be conducted".

Recipe: (From S88) "The necessary set of information that uniquely defines the production requirements for a specific product. Note- There are four types of recipes defined in S88: general, site, master, and control".

Recipe Entity: (From S88) "The combination of a procedural element with associated recipe information (e. g., header, formula, equipment requirements, other information). General, site, master and control recipes are also recipe entities".

Process stages: (From S88)"The process consists of one or more process stages which are organised as an ordered set, which can be serial, parallel, or both. A process stage is a part of the process that usually operates independently from other process stages"

Process operations: (From S88) "Each process stage consists of an ordered set of one or more process operations. Process operations represent major processing activities. A process operation usually results in a chemical or physical change in the material being processed"

Commercial Transaction: in this document, commercial transaction will reference both demands for raw materials and orders for products.

Schedule: (From S88) "The batch schedule [...] contains information such as the products that are to be produced, how much of each product is to be produced, and when they are required for a specific process cell. It identifies which batches are to be made, their order, and the equipment to be used. This schedule also deals with issues such as personnel requirements, raw material options, and packaging requirements".

1.3.2 Introduction

In the chemical-processing context, production planning and scheduling collectively refers to the procedures and processes of allocating equipment over time to execute the chemical and physical-processing tasks required for manufacturing chemical products. Usually, the production planning component is directed at goal setting and aggregate allocation decision over long time scale (months, quarters or year), while scheduling focuses on shorter time scale allocation, timing and sequencing decisions required to executed the plan on the plant floor.

Besides, we recognise that scheduling can be performed over different time scales from planning a campaign of runs, to developing a master schedule for a given campaign, to modifying the master scheduling to respond to changes and unexpected perturbations which inevitably arise as the schedule is executed (Reklaitis, 1995). From this point of view, the scheduling problem is a rescheduling problem without initial perturbations.

Any system for producing chemical products contemplates three necessary components (Rippin 1993):

- A market for one or more products.

- A method to produce them (process).
- A set of available resources.

The market needs are expressed in the form of a production request (Silver et al., 1998). Basically there are two main strategies for introducing requests: Order Driven or Forecast Driven. In practice, hybrid combination of both is used. If production is forecast driven, the request could be interpreted like an "auto-request", that is, some production objectives are set by the same enterprise, as imposed by the market. The interface between the enterprise and the market is the business department.

The process: is a sequence of chemical, physical or biological activities for the conversion, transport, or storage of material or energy. Industrial manufacturing processes can be classified as continuous, discrete parts manufacturing, or batch. How a process is classified (S-88) depends on whether the output from the process appears in a continuous flow (continuous), in finite quantities of parts (discrete parts manufacturing), or in finite quantities of materials (batches). If the process is batchwise, the recipe represents it. Within the enterprise the responsible of the process is the production department.

The resources to be considered are the physical equipment available (units), the raw materials, the set of utilities (steam, electricity, etc.), money, manpower and so on. The management of resources has to do with the maintenance, production and inventory control tasks.

In the following section, the processes classification is given according to its operation and structure. Next, the scheduling problem and different solution approaches will be reviewed.

1.3.3 Types of processes

Industrial manufacturing processes can be classified as continuous, discrete parts manufacturing, or batch. How a process is classified depends on whether the output from the process appears in a continuous flow (continuous), in finite quantities of parts (discrete parts manufacturing), or in finite quantities of material (batches). Process definitions according to ISA- S88 can be found in the definition section of this document (Section 1.3.1).

This study is limited to batch and hybrid batch-continuous process. This kind of process can be classified according with the plant physical structure in: single path, multiple path and network.

A **simple path** structure is a group of units through which a batch passes sequentially (see Figure 1). A single-path structure could be a single unit, such as a reactor, o several units in sequence. Multiple input materials are typically used; multiple finished materials may be generated. Several batches may be in progress at the same time.

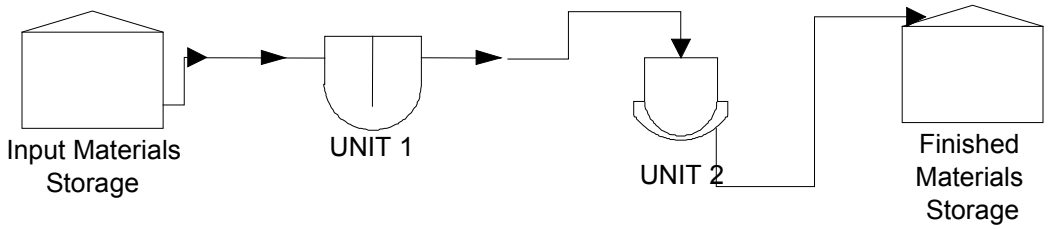


Figure 1: Single Path structure

A **multiple-path** structure is shown in Figure 2. It consists of multiple single-path structures in parallel with no product transfer between them. The units may share raw materials sources and product storage. Several batches may be in progress at the same time. Although units within a multi-path structure may be physically similar, it is possible to have paths and units within a multipath structure that are of radically different physical design.

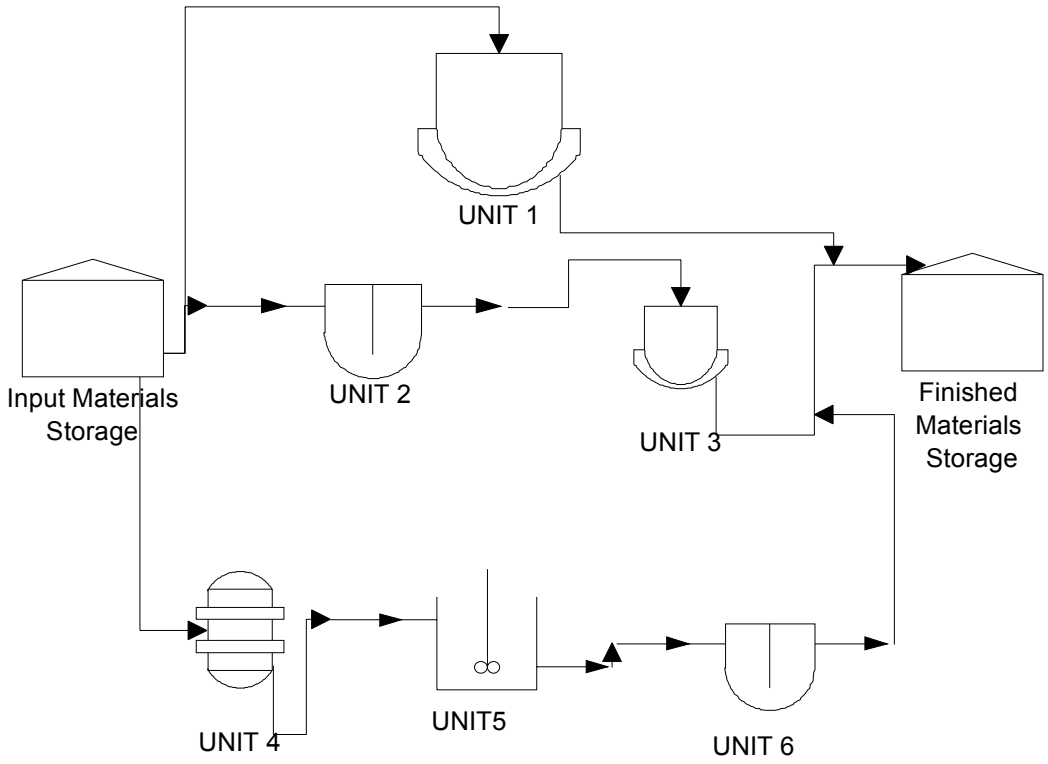


Figure 2: Multiple Path structure

A **network structure** is shown in Figure 3. The paths may be either fixed or variable. When the paths are fixed, the same units are used in the same sequence. When the path is variable, the sequence may be determined at the beginning of the batch or it may be determined as the batch is produced. The path could also be totally flexible. For example, a batch would not have to start at either Unit 1 or Unit 3; it could start within any feasible unit and take multiple paths through the process cell. The units themselves may be portable within the process cell. In this case, verification of the process connections may be an important part of the procedures. Note that several batches may be in production at the same time. The units may share raw materials sources and product storage.

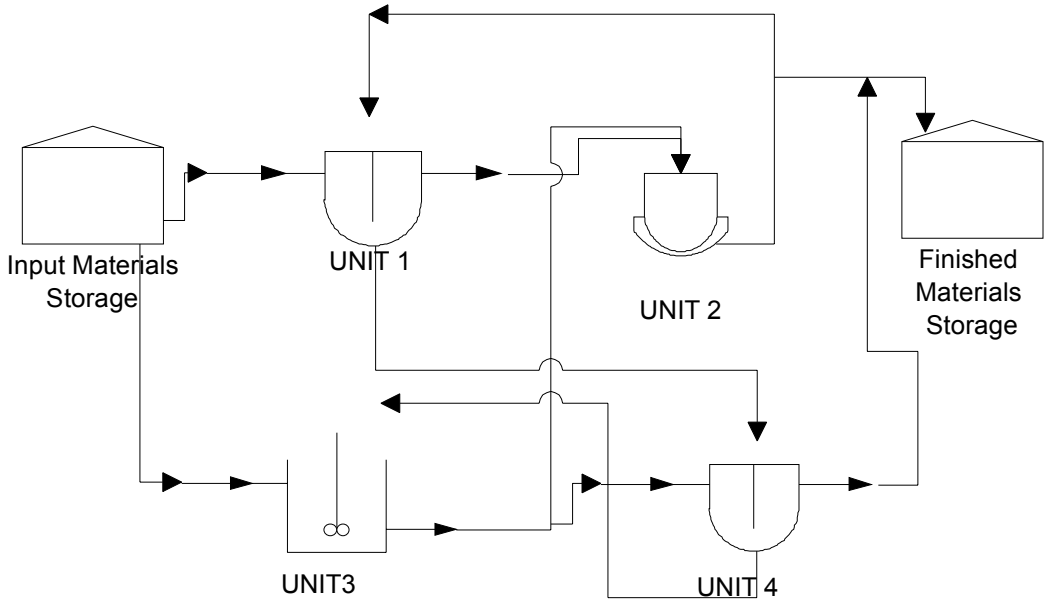


Figure 3: Network Structure

1.3.4 Planning and scheduling

The problem of short-term scheduling can be formulated as follows:

Given:

- The production recipe.
- The available units for each task, the units connectivity and their availability.
- The list of time and kind of utilities consumption required by each unit available for each task (could be variable with the amount of material being processed).
- The market requirements expressed as specific amounts of products at given time instants (product orders) other limitations on shared resources.

Determine:

- The optimal sequence of tasks performed in each unit (under specific performance criteria)
- The amount of material being processed at each time in each unit.
- The processing time of each task in each unit.
- The use of utilities as function of time.

Therefore, this problem involves:

- The assignment of units and resources to tasks.
- The sequencing of the tasks assigned to specific units.
- The determination of the start and end times for the execution of all tasks.
- The resources consumption at any time.

And also emerge additional questions, for instance:

- Considering incidences (assignment not allowed during certain time period for maintenance requirements).
- Moreover, discrepancies may arise between the proposed plan and that achievable. This may be caused by processing time variations, rush orders, failed batches, equipment breakdowns, etc. So, the schedule should be adjusted to incorporate new information. This is called reactive scheduling.

Once this information is processed, the solution of a scheduling problem is a proposed schedule. That is, the proposed sequence to carry out the recipe and unit assignment. The resource consumption along time is also given. All this information is usually represented using a Gantt Chart and Resource Diagram.

The Gantt chart shows the use of the different equipment units along time (see figure 4). With this chart it is possible to see the start and end a task, and which is the unit used for making that task. In this diagram, the colours correspond to different batches, and the numbers are task identifiers.

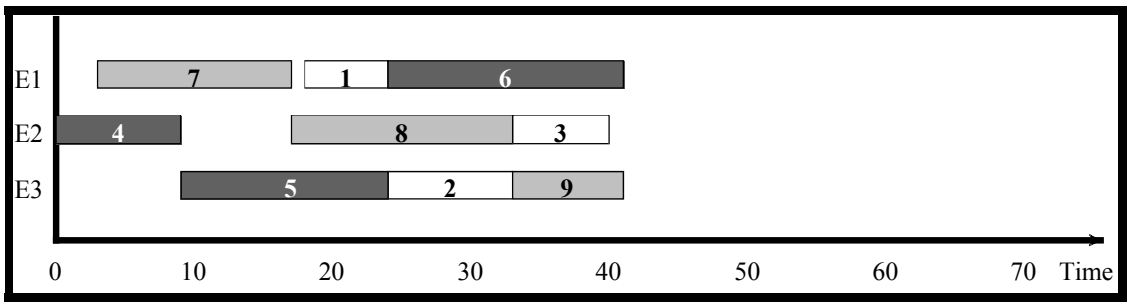


Figure 4: Gantt Chart

The Resource Diagram shows the consumption of resources like steam, electricity, manpower and others, also as a time function (see Figure 5).

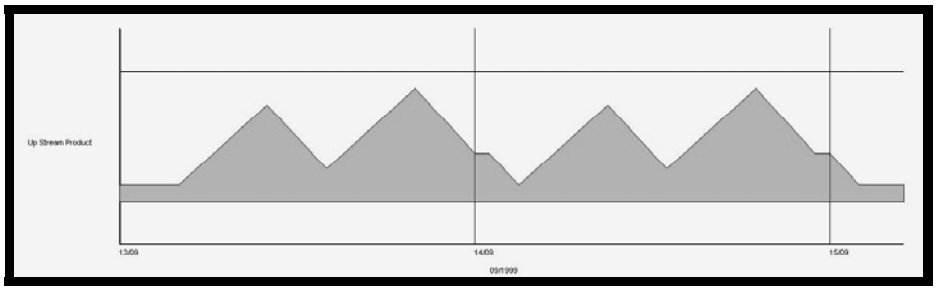


Figure 5: Resource Diagram

These diagrams are a major source of information for the production responsible, but they are also needed by the sales and supply departments for inventory control purposes and maintenance.

For the evaluation of a plan, there are several performance measures that can be used according with the specific needs of the enterprise. Some have to do with the time a job spends in the plant; others pertain to performance relative to due dates; and yet others concern utilisation of production resources (Silver et al., 1998).

Note that these three general goals may conflict with each other. For example, high resource utilisation increase the time spent, and may degrade the due date performance. Therefore in spite of the very short-term nature of the scheduling problem, much larger strategic concerns are at issue.

Managers should be very concerned with the trade-offs implied in the choice of which performance measure to use. For example, this choice may implicitly trade off work in process inventory cost (WIP) with customer satisfaction regarding meeting due dates, and with amortisation of capital investment. These three general goals can be specified in more precise performance measures.

One performance measure that is a primary focus of plant managers is the average WIP level. High WIP levels means that more money is invested in inventory, and therefore is not available for other purposes. Flowtime is the time that a batch spends from the moment it is ready for processing until its completion, and includes any waiting time prior to processing. The makespan is the total time for all jobs to finish processing.

The selection of the performance measure is a managerial decision related to the strategic objectives of each enterprise considered. Consequently, in spite of the acknowledged standard performance indexes, the need for customising specific **objective functions** for each particular case arises, thus the related interface requirements for such feature.

1.3.5 Different approaches

The scheduling problem at a single site is usually concerned with meeting fairly specific production requirements. Customer orders, stock imperatives, higher-level supply chain or long term planning would usually set these. It is concerned with the allocation over time of scarce resources between competing activities to meet these requirements in an efficient fashion (Shah, 1998).

The key components of the scheduling problem are resources, tasks and time. The resources need not be limited to processing equipment items, but may include material storage equipment, transportation equipment intra-and inter plant), operators, utilities (e.g. steam, electricity, cooling water), auxiliary services, raw materials and so on.

The problem has an inherent combinatorial nature, and then the calculation time grows at least potentially with the variable number. To solve this, then the typical strategies try to involve the formulation, to make a simpler problem and/ or use better calculation methods.

There are many different ways in which approaches to solve the scheduling problem may be classified. One of them is divide them in heuristic/stochastic search techniques and mathematical programming techniques and then further subdivide each according with their specific or generic application domains.

Most scheduling heuristics are concerned with formulating rules for determining sequences of activities. Stochastic search approaches are based on improvement of trial solutions by the application of an evolutionary algorithm, which modifies solutions and/or prioritises them from a list for further consideration.

The application of mathematical programming approaches implies the development of a mathematical model and an optimisation algorithm. Most approaches aim to develop models that are of a standard form (from linear programming (LP) models to mixed integer non-linear programming (MINLP) models. These may be solved by standard software or specialised algorithms that take account of problem structure.

The latter division is intended to reflect the scope of the technique in terms of plant structure and process recipes. Rippin (1983) classified different flexible plant structures.

- Multiproduct plants: Where each product has the same processing network.
- Multipurpose plants: where the products are manufactured via different processing networks, and there may be more than one way in which to manufacture the same product.

Therefore, we can find a lot of scheduling software packages able to solve the scheduling problem not only using different methods, but also different in data structure and capabilities. Moreover, from the industrial point of view, the scheduling packages tend to be very specific according to the plant needs, so considerable effort to set the standard interfaces will be needed.

In spite of the different solution methods, a common issue should be the model for the recipe description.

The structure of processes and related materials is characterised following the guidelines of the ISA S88 specification and three different levels of detail are introduced for the description of production recipes.

The first level or process level is defined using a network of processes and attached materials which describes the precedence for those substances considered. An explicit material balance is specified for each of the processes in terms of a stoichometric-like equation. Different processes may be specified for different material balances due to different recipe or equipment efficiency. Thus, this network provides a general description for production structures including common intermediates, synthesis and separation processes.

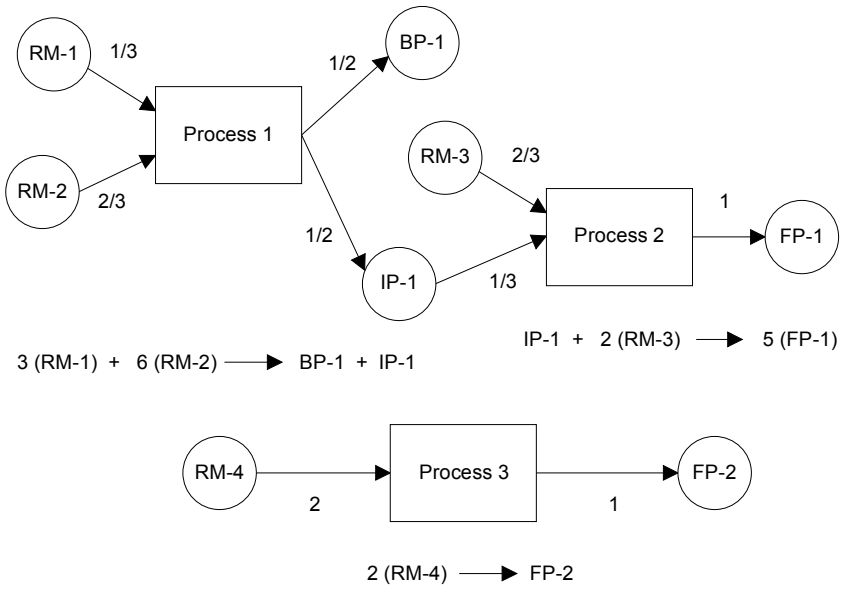


Figure 6: Processing Network for three processes leading to two final products

For each process, the second level provides the information about the possible unit assignment for those sequences of basic operations to be carried out in the same equipment unit, which are included into a stage.

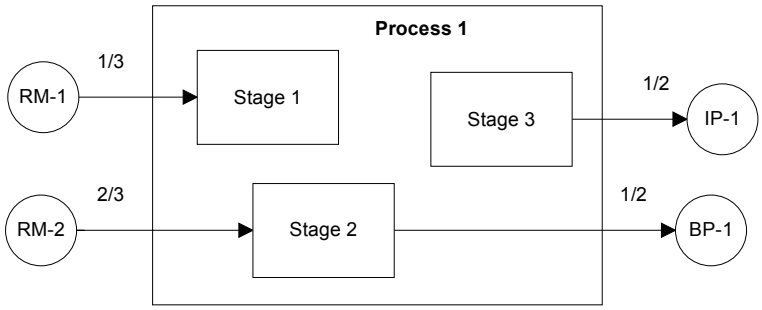


Figure 7: Stage network for one process

Finally, the deepest level of detail is the operation level. All the operations in each recipe are described including all the specific data which is needed: time required, resource requirements, inputs and outputs of flows and time relationships between operations, such precedence constraints, etc.

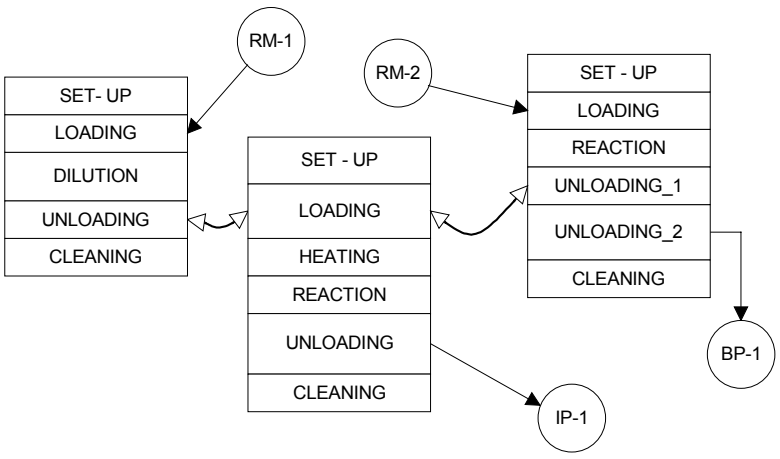


Figure 8: Structured set of operations for a process.

1.3.6 Planning and scheduling frameworks.

The complex problem of what to produce and where and how to produce it is best considered through an integrated, hierarchical approach which also acknowledges typical corporate structures and business processes (Rickard et al 1999).

The scheduling problem is essentially one of decision-making. This decision doesn't have to be taken in isolation. As well as providing a Gantt Chart and Resource use diagram, it is also desirable that a toolkit provides more information to support better decision making.

This encourages us to integrate a scheduling and planning tool into a simulation environment together with another software tools. It will improve our decisions because we can for instance, test a possible scheduling strategy by dynamic simulation and see more aspects of the problem. An integrated scheduling tool would allow to check automatically the flexibility of the proposed schedule, and for instance to see the quality of the products, operation condition, environmental and safety factors that previously were not taken in to account.

From a scheduling point of view, it is necessary to have a tool, which allows the correct simulation of the system under consideration.

1.3.7 Modelling of combined Discrete/Continuous Processes

The importance of batch and batch-semicontinuous chemical processes has been appreciated recently. Demand for their improvement, extension and control design has been increased as well (Czulek, 1988).

Analysis of continuous chemical processes can be reasonably performed by the aid of the well-proven flowsheeting packages. Given the characteristics that differentiate continuous and batch systems, these packages are unable to analyse batch and batch-continuous processes. Only a few contributions in general purpose simulation packages have been reported until now in this direction.

The dynamic behaviour of processing systems exhibits both continuous and significant discrete aspects (Barton, 1994). Process simulation is therefore a combined discrete/continuous simulation problem. In addition, there is a critical need for a declarative process-modelling environment to encompass the entire range of processing system operation, from purely continuous to batch. In fact, few processes can be considered to operate in an entirely continuous manner. Even the majority of "continuous" process experience significant discrete changes superimposed on their predominantly continuous behaviour. Such changes typically arise from the application of digital regulatory control, plant equipment failure, or as a consequence of planned operational changes, such as start-up or shutdown, feedstock and/or product campaign changes, process maintenance, and so on.

Coordination between simulators and schedulers in hybrid processing networks is not an easy task, and requires a precise definition of the information flows involved, which is precisely a major task in Work Package 5.3.

2. Requirements

2.1 Textual requirements

2.1.1 Architecture

The architecture of the CO Interface System is based on an object-oriented framework, which should allow software systems to be built from binary software components. These components are able to communicate to each other using CO Interfaces. The different components can come from different vendors and may reside in the same machine or be distributed along a network, with different components running on different Operating System platforms.

The CO Interface System gives a standard method for the communication between external software packages which carry up different tasks, as for example, the communication of Planning & Scheduling to any compliant CO Executive, which performs the co-ordination and the connectivity between the different packages available in one system. The Interface can be seen as a "socket" and "plug" system, which is capable of exchanging information between the two parts. The Executive and Scheduling and Planning Packages do not have to know anything about the internal architecture of each other. The goal of the Interface is to translate the requests for information or action, by each one of the packages involved, into something the other understands.

For example, the following figures show some of the ways in which external facilities can communicate with an Executive through the interfaces to be defined in the CO project. (S- Scheduling and Planning, N- Numeric (already defined in Cape-Open), C- Control (corresponds to part C of GCO workpackage 5.3))

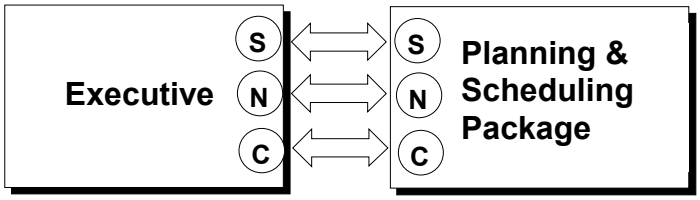


Figure 9 Architecture 1

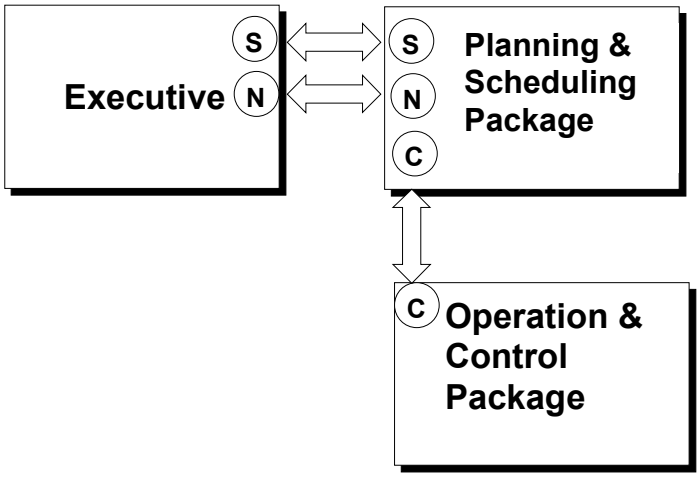


Figure 10 Architecture 2

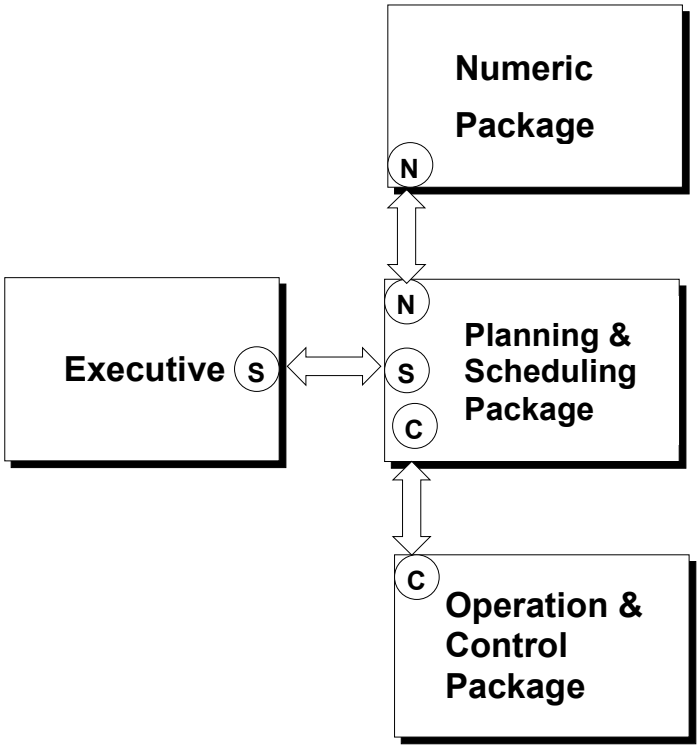


Figure 11 Architecture 3

2.1.2 Data Flows

The communications required for a Scheduling and Planning package can be summarised in the following figure (based in the specifications of ISA S95)

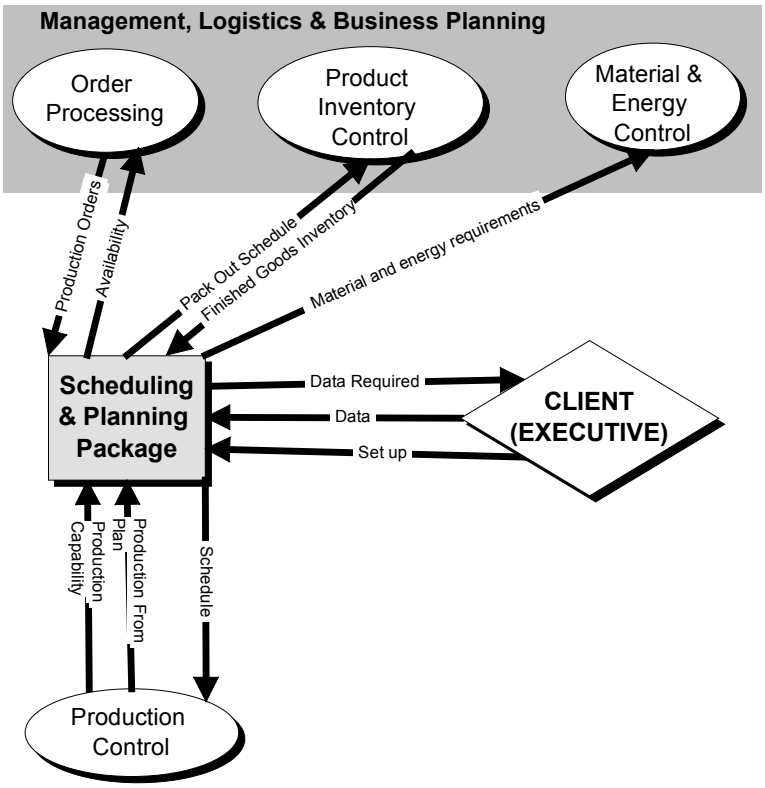


Figure 12: Information flow according to ISA S95

According to the figure 12, The Scheduling and planning package basically communicates with two levels of the information systems:

(i) Management, Logistics & Business Planning

(ii) Production Control

Communication should also be provided to allow the initial set-up and the data set-up from an external client. In the scope of CO project the external client is mainly the CO Executive.

COMMUNICATION BETWEEN THE PLANNING & SCHEDULING PACKAGE AND THE EXECUTIVE

Basically the communication between these two components has the following associated aspects:

Package Initialisation: The Executive should be able to recognise that a Planning and Scheduling Package (PSP) is available and should be able to initialise it.

Setting up: The Executive should be able to set up all the variables needed for the calculation of a schedule. These variables will be typically different from one package to another. This aspect can be performed in two ways.

Proprietary package GUI: The PSP could have a proprietary GUI which is able to provide the end user with tools for introducing all the data required.

GUI dependent of the Executive: The Executive should provide all the GUI necessary to introduce and to manage all the data of the package.

Schedule Calculation: The Executive asks the PSP for the list of calculation procedures available. The PSP gives access to the list of calculation functions available. Each one of the calculation capabilities has its own parameters. The parameters can be controlled using the PSP GUI specific capabilities or the Executive GUI tools. The Executive specifies one of the calculation capabilities, sends a request to the Package to begin the calculation and waits for completion.

Results reporting: The Executive asks the PSP to provide results. There are two main ways of doing it:

Proprietary package Report generator: The package is able to generate several reports. The Executive can select one of them and ask the Package to generate it.

Report generated by the Executive: Executive asks for the information available and uses it to generate its own report.

Table 1: Data flows between Package and Executive

Executive	Planning & Scheduling Package
Package Initialisation	
Ask Package for initialisation	Responds with Status
Setting up with proprietary GUI	
Ask the package to get its specific data	Obtains data using own input system
Setting up with Executive GUI	
Ask the package for the list of parameters needed	Responds with the list of parameters
Create an GUI to manage the list of parameters	
Sends data introduced to Package	Retrieves data from Executive
Schedule Calculation with proprietary GUI	
Ask the package to invoke its calculation procedures	Obtains additional info from own GUI
Wait for Calculation completion	Performs calculations
	Returns calculation complete status to Executive
Schedule Calculation with Executive GUI	
Ask the package for list of schedule calculation capabilities	Responds with the list of calculation capabilities

Executive	Planning & Scheduling Package
Use a GUI to select one of the calculations available	
Asks for parameters needed for an specific calculation	Return the list of parameters needed
Use a GUI to accepts inputs for parameters needed	
Sends the parameters values to package	Gets the parameter values
Invokes calculation of the calculation routine selected	Starts calculation with the given parameters
Wait for calculation completion	Performs calculations
	Returns calculation complete status to Executive
Results reporting	
Ask package what outputs it has	Send a list of available results
Ask package to send required results	Send requested results
Ask package to output a report	Outputs a report

COMMUNICATION BETWEEN THE PLANNING & SCHEDULING PACKAGE AND PRODUCTION CONTROL

The information required for the Production Control is basically the Schedule to be executed. This information is included in the Results Reporting of the previous part. Eventually a PSP will require from Production Control the real Schedule being realised and the production capacity of the plant.

Table 2: Information flow 2

Production Control	Planning & Scheduling Package
Schedule information retrieval	
Ask the PSP for the Schedule to be performed in the plant	Responds with the Schedule

COMMUNICATION BETWEEN THE PLANNING & SCHEDULING PACKAGE AND THE ORDER PROCESSING

Order processing requires from PSP the availability of a final product given the list of Production orders.

Table 3: Information flow 3

Order Processing	Planning & Scheduling Package
Schedule information retrieval	
Send new orders to be processed	Responds with the Schedule to perform the orders introduced

COMMUNICATION BETWEEN THE PLANNING & SCHEDULING PACKAGE AND PRODUCT INVENTORY CONTROL

The information required from Product Inventory Control is the amount of final products and intermediates to be produced along the time following the actual schedule. Inventory Control can provide information about the due dates for the required products.

Table 4: Information flow 4

Product Inventory Control	Planning & Scheduling Package
Inventory info retrieval	
Ask the package for variations in the inventory using the current schedule	Responds with the predicted variations in the inventory

COMMUNICATION BETWEEN THE PLANNING & SCHEDULING PACKAGE AND MATERIAL & ENERGY CONTROL

The information required from Material & Energy Control is the amount of raw materials end energy required to process the actual schedule.

Table 5: Information flow 5

Material & Energy Control	Planning & Scheduling Package
Inventory info retrieval	
Ask the package for material and Energy requirements with the predicted schedule	Responds with the forecasted requirements for the actual schedule

2.2 Use Cases

We present now the current versions of the use case diagrams (Booch et al. 1999, Muller 2000, Rational Rose 2000).

Problem statement:

Define a Scheduling problem, using a scheduling package, supplying the information required in order to perform calculations, retrieve results and the information generated.

2.2.1 Actors

Can be identified four main *Actor*s: The Resources Manager, The Process Manager, The Commercial Transactions Manager and The Production Manager. Different people inside an enterprise could do the functionality of every *Actor*, even a software system.

When no particular *Actor* is responsible for a specific *Use Case* the "Generic User" *Actor* will be referenced.

The Resources Manager is the responsible for managing the data associated to the resources use and availability. This is the information about materials, utilities and equipment. This *Actor* would also represent those responsible for maintenance and for material and energy control..

The Process Manager is the responsible for managing the information related to the Process Recipe. This is the sequence of task needed for the production of a specific product. Usually, the corresponding department inside an enterprise would be the Process-engineering department.

The Commercial Transactions Manager is the responsible for managing the information related to commercial transactions (production orders and raw material supplies).

The Production Manager is the responsible of making a production plan (using the appropriate software tool), supplying the necessary data to do this. In an enterprise, this would be done by the production planning and scheduling department.

Identification of key packages:

According with the problem structure, we identify five main packages: The General Management Package, The Resource Management Package, The Recipe Management Package, The Commercial Transaction Package and The Plan Management Package.

The Resource Management Package will include the *Use Cases* related with the functionality desired for the usage and definition of resources; these are materials, equipment and utilities.

The Recipe Management Package will include the *Use Cases* related with the functionality desired for the usage and definition of Recipes.

The Transactions Package will include the Uses Cases for the management and specification of commercial transactions.

The Plan Management Package will include the *Use Cases* related with the functionality desired for the usage and definition of production schedules.

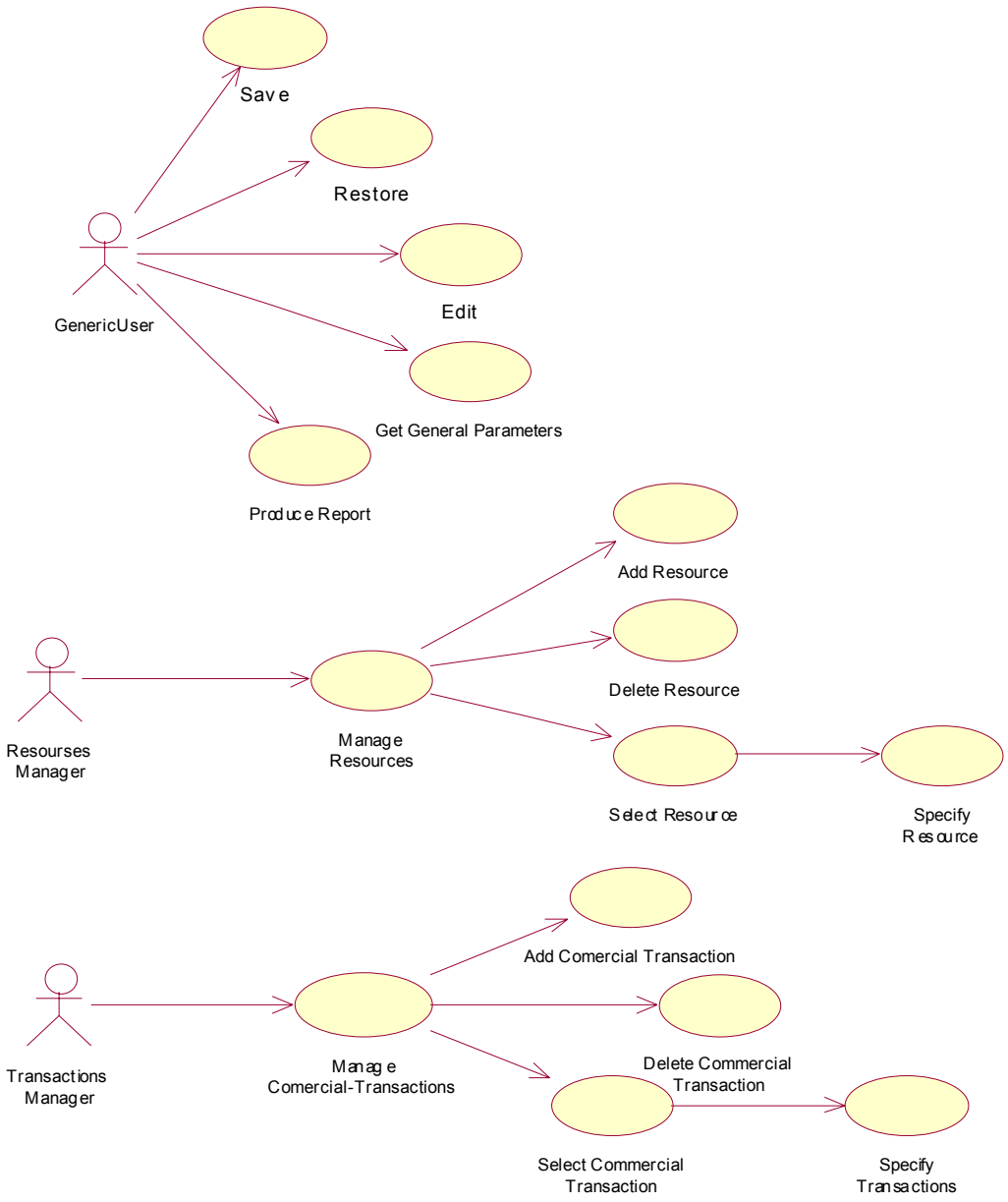
Finally, The General Management Package will include the uses cases related with the generic functionality associated to the initialisation, termination and some other functionality, which are not specific to any *Actor*.

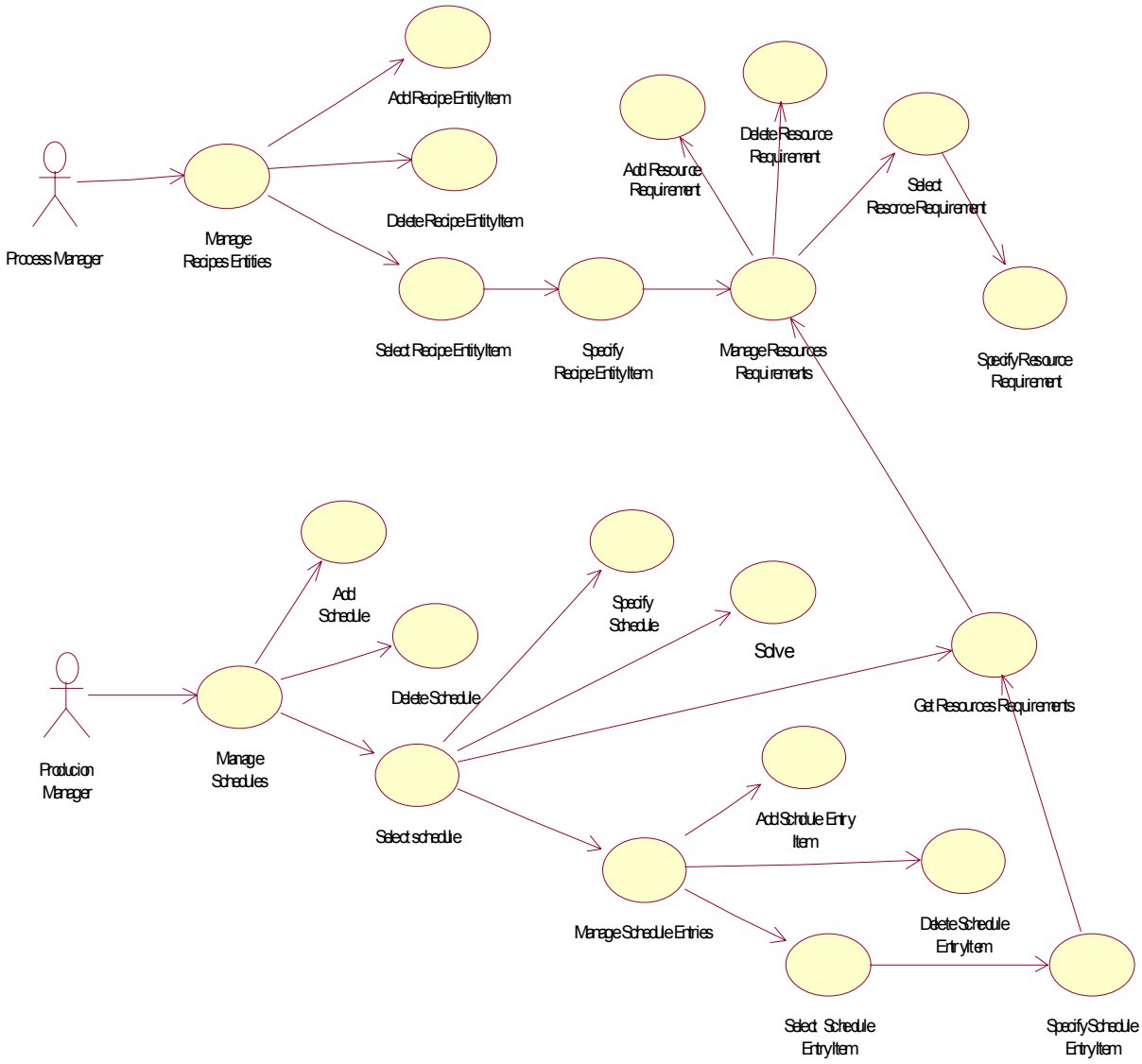
2.2.2 List of Use Cases

- UC-001: SAVE (SV.)
- UC-002 : RESTORE (RST.)
- UC-003 : EDIT (ED.)
- UC-004 : GET GENERAL PARAMETERS (GPRS.)
- UC-005 : PRODUCE REPORT RESTORE (PRRT.)
- UC-006 : MANAGE RESOURCES (MGRS.)
- UC-007 : ADD RESOURCE (ADRS.)
- UC-008 : DELETE RESOURCE (DLRS.)
- UC-009 : SELECT RESOURCE ITEM (SLRSIT.)
- UC-010 : SPECIFY RESOURCE (SPRS.)
- UC-011 : MANAGE RECIPES ENTITIES (MGRPENS.)
- UC-012 : ADD RECIPE ENTITY ITEM (ADRPENIT.)
- UC-013 : DELETE RECIPE ENTITY ITEM (DLRPENIT.)
- UC-014 : Select Recipe Entity Item (SlRpEnIt.)
- UC-015 : SPECIFY RECIPE ENTITY ITEM (SPRPENIT.)
- UC-016 : MANAGE RESOURCES REQUIREMENTS (MGRSRQ.)
- UC-017 : ADD RESOURCE REQUIREMENT (ADRSRQ.)
- UC-018 : DELETE RESOURCE REQUIREMENT (DLRSRQ.)
- UC-019 : SELECT RESOURCE REQUIREMENT ITEM (SLRSRQIT.)
- UC-020 : SPECIFY RESOURCE REQUIREMENT (SPRSRQ.)
- UC-021 : MANAGE COMMERCIAL TRANSACTIONS (MGCTS.)
- UC-022 : ADD COMMERCIAL TRANSACTION (ADCT.)
- UC-023 : DELETE COMMERCIAL TRANSACTION (DLCT.)
- UC-024 : SELECT COMMERCIAL TRANSACTION (SLCT.)
- UC-025 : SPECIFY COMMERCIAL TRANSACTION (SPCT.)
- UC-026 : MANAGE SCHEDULES (MGSCHS.)
- UC-027 : ADD SCHEDULE (ADSCH.)

- UC-028 : DELETE SCHEDULE (DLSCH.)
- UC-029 : SELECT SCHEDULE ITEM (SLSCHIT.)
- UC-030 : SPECIFY SCHEDULE (SPSCH.)
- UC-031 : SOLVE SCHEDULE (SVSCH.)
- UC-032: GET RESOURCES REQUIREMENTS (GTRSRQ.)
- UC-033 : MANAGE SCHEDULES ENTRIES (MGSCHETS.)
- UC-034 : ADD SCHEDULE ENTRY ITEM (ADSCHETIT.)
- UC-035 : DELETE SCHEDULE ENTRY ITEM (DLSCHETIT.)
- UC-036 : SELECT SCHEDULE ENTRY ITEM (SLSCHETIT.)
- UC-037 : SPECIFY SCHEDULE ENTRY ITEM (SPSCHENIT.)

2.2.3 Use Cases maps





2.2.4 Use Cases - General Package

UC-001: SAVE (SV.)

Actors: Generic User Actor Priority: Classification: Context: This Use Case allows saving of the data and results (if any) of the currently used PSP case. Pre-conditions: Flow of events: The PSP case information is saved under a specific name, format and location. Post-conditions:

Errors:

Uses:

Extends:

UC-002 : RESTORE (RST.)

Actors: Generic User Actor

Priority:

Classification:

Context: This Use Case allows restoring from disc the data and results (if any) of an available PSP case.

Pre-conditions:

Flow of events: The PSP case information is restored according to the specified name, format and location.

Post-conditions:

Errors:

Uses:

Extends:

UC-003 : EDIT (ED.)

Actors: Generic User Actor

Priority:

Classification:

Context: This Use Case allows the use of the PSP's editor.

Pre-conditions: The PSP was already initialised. The PSP software can be accessed from the Actors machine.

Flow of events: The Actor asks for the PSP editor The PSP, then offers its own graphical user interface, with all its functionality. If the request fails, the user is informed. If the request succeeds, the GUI is displayed on the screen.

Post-conditions:

Errors:

Uses:

Extends:

UC-004 : GET GENERAL PARAMETERS (GPRS.)

Actors: Generic User Actor

Priority:

Classification:

Context: By means of this Use Case, the generic user Actor asks the PSP for the list of all the general parameters for the current case.

Pre-conditions:

Flow of events: The generic user Actor asks for the set of general parameters offered by the PSP. The PSP will allow to the actor the specification of the available general parameters. The parameters values can be then supplied, consulted and/or modified.

Post-conditions:

Errors:

Uses:

Extends:

UC-005 : PRODUCE REPORT RESTORE (PRRT.)

Actors: Generic User Actor

Priority:

Classification:

Context: This Use Case is for reporting the data and results corresponding to the currently case.

Pre-conditions: The PSP offers some kind of report.

Flow of events: The generic user Actor asks for the available reports. Then, the PSP gives the list of available reports. The Actor selects one and configures its parameters. Then the Actor asks to the PSP for produce the report.

Post-conditions:

Errors:

Uses:

Extends:

2.2.5 Use Cases - Resources Management Package

UC-006 : MANAGE RESOURCES (MGRS.)

Actors: Resources Manager Actor

Priority:

Classification:

Context: The purpose of this Use Case is to allow to the Resources Manager Actor to manipulate the whole list of resources.

Pre-conditions:

Flow of events: The Actor selects the PSP manage resources option. The PSP gives to the Actor the whole list of resources. If there is no resource defined, the Actor can use the Add resource Use Case to create a new resource item. Then the Resources Manager Actor can select a specific resource item for further specification using the Select Item Use Case. Also, the Delete Resource Use Case can be used to delete an existing resource item.

Notes:

This Use Case is general to the any kind of resource (materials, utilities, equipments, etc).

Post-conditions:

Errors:

Uses: Add Resource, Delete Resource and Select Resource Item.

Extends:

UC-007 : ADD RESOURCE (ADRS.)

Actors: Resources Manager Actor

Priority:

Classification:

Context: This Use Case is for the addition of a new item of resource.

Pre-conditions:

Flow of events: Within the Manage Resources Use Case, the resource Manager asks the PSP to add a new resource item. The PSP creates the new item, which is added to the existing list. The new item can then be selected and later specified.

Post-conditions: The new resource item is successfully created.

Errors:

Uses:

Extends:

UC-008 : DELETE RESOURCE (DLRS.)

Actors: Resources Manager Actor

Priority:

Classification:

Context: This Use Case represents the deletion of an existing resource item.

Pre-conditions: The resource to be deleted exists.

Flow of events: Once that the Resources Manager has used the Manage resource Use Case, the Actor can delete an existing item from the resources list. The item to be deleted will be referenced by its name or position in the list.

Post-conditions: The item is successfully deleted.

Errors:

Uses:

Extends:

UC-009 : SELECT RESOURCE ITEM (SLRSIT.)

Actors: Resources Manager Actor

Priority:

Classification:

Context: This Use Case is for allowing to the Resources Manager to have access to a particular resource item information.

Pre-conditions:

Flow of events: Once the Resources Manager has used the manage resource Use Case, he or she can invoke this Use Case to specify a specific resource item. In order to get the item, the Resources Manager can use either the name or the position in the list. Then the PSP offers the particular item to the Resources Manager for consulting and / or modifying its current specification.

Post-conditions:

Errors:

Uses: Specify Resource Use Case

Extends:

UC-010 : SPECIFY RESOURCE (SPRS.)

Actors: Resources Manager Actor

Priority:

Classification:

Context: Using this Use Case the Resources Manager can consult, specify and/or change the information associated to the resource. This information will be different according to the kind of resource and the proprietary PSP.

Pre-conditions:

Flow of events: Once the Resources Manager has used the select resource item Use Case, he or she uses this Use Case to get the parameter list associated with the specific resource item. The parameters values can be then supplied, consulted and/or modified.

Post-conditions:

Errors:

Uses:

Extends:

2.2.6 Use Cases - Recipes Management Package

UC-011 : MANAGE RECIPES ENTITIES (MGRPENS.)

Actors: Recipes Manager Actor

Priority:

Classification:

Context: The purpose of this Use Case is to allow to the Recipes Manager Actor to manipulate the whole list of Recipes Entities.

Pre-conditions:

Flow of events: The Actor asks to the package for manage the Recipes Entities. The PSP gives to the Actor the whole list of Recipes Entities at the current recipe information level. If there is no Recipe Entity defined, the Actor can use the Add Recipe Entity Item Use Case for the addition of a new Recipe Entity Item. Then the Recipes Manager Actor can select one specific item of Recipe Entity for further specification using the Select Recipe Entity Item Use Case. Also, the Delete Recipe Entity Item Use Case can be used for deletion of an existing item of Recipe Entity.

Notes:

This Use Case is general to any recipe information level: recipe, stage and operation. The level covered will depend on the capabilities of the particular PSP.

Post-conditions:

Errors:

Uses: Add Recipe Entity, Delete Recipe Entity and Select Recipe Entity Item.

Extends:

UC-012 : ADD RECIPE ENTITY ITEM (ADRPENIT.)

Actors: Recipes Manager Actor

Priority:

Classification:

Context: This Use Case is for the addition of a new item of Recipe Entity

Pre-conditions:

Flow of events: Once that the Recipes Manager have used the Manage Recipe Entities Use Case, he asks to the PSP to add new Recipe Entity item. The PSP creates a new Recipe Entity Item instance and the new Recipe Entity item is added to the existing list. The new item is ready to be selected and later specified.

Post-conditions: The new Recipe Entity item is successfully created.

Errors:

Uses:

Extends:

UC-013 : DELETE RECIPE ENTITY ITEM (DLRPENIT.)

Actors: Recipes Manager Actor

Priority:

Classification:

Context: This Use Case represents the deletion of an existing Recipe Entity Item.

Pre-conditions: The Recipe Entity Item to delete exists.

Flow of events: Once that the Recipes Manager has used the Manage Recipe Entities Use Case, the Actor can delete one existing item of the Recipe Entities list. The item to delete will be referenced by its name or position in the list.

Post-conditions: The item is successfully deleted.

Errors:

Uses:

Extends:

UC-014 : SELECT RECIPE ENTITY ITEM (SLRPENIT.)

Actors: Recipes Manager Actor

Priority:

Classification:

Context: This Use Case is to allow to the Process Manager having access to a particular Recipe item information.

Pre-conditions:

Flow of events: Once the Process Manager has used the manage Recipe Use Case, can invoke this Use Case for the specification of a specific Recipe item. In order to get the item, the Process Manager can use either the name or the position in the list. Then the PSP offers the particular item to the Process Manager for consulting and / or modifying its current specification.

Post-conditions:

Errors:

Uses: Specify Recipe Use Case

Extends:

UC-015 : SPECIFY RECIPE ENTITY ITEM (SPRPENIT.)

Actors: Process Manager Actor

Priority:

Classification:

Context: Using this Use Case the Process Manager can consult, specify and/or change the information associated to the Recipe Entity Item. This information will be different according to the proprietary PSP.

Pre-conditions:

Flow of events: Once the Resources Manager has used the select resource item Use Case, uses this Use Case for get the parameters list associated with the specific Recipe Entity item. The parameters values can be then supplied, consulted and/or modified. This Use Case also allows to the Actor to have access to the resources requirements for the selected Recipe Entity Item.

Post-conditions:

Errors:

Uses: Manage Resources Requirements Use Case.

Extends:

UC-016 : MANAGE RESOURCES REQUIREMENTS (MGRSRQ.)

Actors: Recipes Manager Actor

Priority:

Classification:

Context: The purpose of this Use Case is to allow to the Recipes Manager Actor to manipulate the whole list of Resources Requirements for the selected Recipe Entity Item.

Pre-conditions:

Flow of events: The Actor asks to the currently selected Recipe Entity item for manage its Resources Requirements. The PSP gives to the Actor the whole list of Resources Requirements for this recipe entity item. If there is no Resources Requirements defined, the Actor can use the Add Resource Requirement Use Case for the addition of a new Resource Requirement item. Then the Recipes Manager Actor can select a specific item of Resource Requirement for further specification using the Select Resource Requirement Item Use Case. Also, the Delete Resource Requirement Use Case can be used for deletion of an existing item of Resource Requirement.

Note:

This Use Case will be used by the Process Manager, for the management of the estimation of resources requirements. On other hand, will be used by the Production Manager for also consulting the estimated resource requirements.

Post-conditions:

Errors:

Uses: Add Resource Requirement, Delete Resource Requirement and Select Resource Requirement.

Extends:

UC-017 : ADD RESOURCE REQUIREMENT (ADRSRQ.)

Actors: Process Manager Actor

Priority:

Classification:

Context: This Use Case is for the addition of a new item of resource requirement

Pre-conditions:

Flow of events: Once that the Process Manager has used the Manage Resources Requirements Use Case, he asks to the PSP for the addition of a new resource requirement item. The PSP creates a new resource requirement item instance and the new resource item is added to the existing list. The new item is ready to be selected and later specified.

Post-conditions: The new resource requirement item is successfully created.

Errors:

Uses:

Extends:

UC-018 : DELETE RESOURCE REQUIREMENT (DLRSRQ.)

Actors: Process Manager Actor

Priority:

Classification:

Context: This Use Case represents the deletion of an existing item of resource requirement

Pre-conditions: The resource requirement item to delete exists.

Flow of events: Once that the Process Manager have used the Manage Resources Requirements Use Case, the Actor can delete an existing item of the resources requirements list. The item to delete will be referenced by its name or position in the list.

Post-conditions: The item is successfully deleted.

Errors:

Uses:

Extends:

UC-019 : SELECT RESOURCE REQUIREMENT ITEM (SLRSRQIT.)

Actors: Process Manager Actor

Priority:

Classification:

Context: This Use Case is to allow to the Process Manager having access to a particular resource requirement item information.

Pre-conditions:

Flow of events: Once the Process Manager has used the manage resources requirements Use Case, can invoke this Use Case for the specification of a specific resource requirement item. In order to get the item, the Process Manager can use either the name or the position in the list. Then the PSP offers the particular item to the Process Manager for consulting and / or modifying its current specification).

Post-conditions:

Errors:

Uses: Specify Resource Requirement Use Case

Extends:

UC-020 : SPECIFY RESOURCE REQUIREMENT (SPRSRQ.)

Actors: Process Manager Actor

Priority:

Classification:

Context: Using this Use Case the Process Manager can consult, specify and/or change the information associated to the resource. This information will be different according to the proprietary PSP.

Pre-conditions:

Flow of events: Once the Process Manager has used the select resource requirement Use Case, uses this Use Case for get the parameters list associated with the specific resource requirement item. The parameters values can be then supplied, consulted and/or modified.

Post-conditions:

Errors:

Uses:

Extends:

2.2.7 Use Cases - Commercial Transactions Management Package

UC-021 : MANAGE COMMERCIAL TRANSACTIONS (MGCTS.)

Actors: Process Manager Actor

Priority:

Classification:

Context: The purpose of this Use Case is to allow to the Commercial Transaction Manager Actor to manipulate the whole list of Commercial Transactions.

Pre-conditions:

Flow of events: The Actor asks to the package for manage the commercial transactions. The PSP gives to the Actor the whole list of commercial transactions. If there is no commercial transaction defined, the Actor can use the Add Commercial Transaction Use Case for the addition of a new commercial transaction item. Then the Process Manager Actor can select a specific item of commercial transaction for further specification using the Select Item Use Case. Also, the Delete Commercial Transaction Use Case can be used for deletion of an existing item of commercial transaction.

Notes:

This Use Case is general to the any commercial transactions (production orders and buys).

Post-conditions:

Errors:

Uses: Add Commercial Transaction, Delete Commercial Transaction and Select Commercial Transaction Item.

Extends:

UC-022 : ADD COMMERCIAL TRANSACTION (ADCT.)

Actors: Commercial Transaction Manager Actor

Priority:

Classification:

Context: This *Use Case* is for the addition of a new item of commercial transaction.

Pre-conditions:

Flow of events: Once that the Commercial Transaction Manager has used the Manage Commercial Transaction Use Case, he asks to the PSP for the addition of a new Commercial Transaction item. The PSP creates a new commercial

transaction instance and the new commercial transaction item is added to the existing list. The new item is ready to be selected and later specified.

Post-conditions: The new commercial transaction item is successfully created and added to the existing list.

Errors:

Uses:

Extends:

UC-023 : DELETE COMMERCIAL TRANSACTION (DLCT.)

Actors: Commercial Transaction Manager Actor

Priority:

Classification:

Context: This Use Case represents the deletion of an existing item of resource.

Pre-conditions: The item to delete exists.

Flow of events: Once that the Commercial Transaction Manager has used the Manage Commercial Transaction Use Case, the Actor can delete an existing item from the Commercial Transactions list. The item to delete will be referenced by its name or position in the list.

Post-conditions: The item is successfully deleted.

Errors:

Uses:

Extends:

UC-024 : SELECT COMMERCIAL TRANSACTION (SLCT.)

Actors: Commercial Transaction Manager Actor

Priority:

Classification:

Context: This Use Case is to allow to the commercial transaction manager having access to a particular commercial transaction item information.

Pre-conditions: The selected item exists.

Flow of events: Once the commercial transaction manager has used the manage commercial transactions Use Case, can invoke this Use Case for the specification of a specific commercial transaction item. In order to get the item, the commercial transaction manager can use either the name or the position in the list. Then the PSP offers the particular item to the Commercial Transaction Manager for consulting and / or modifying its current specification.

Post-conditions: The item is successfully selected.

Errors:

Uses: Specify Commercial Transaction Use Case

Extends:

UC-025 : SPECIFY COMMERCIAL TRANSACTION (SPCT.)

Actors: Resources Manager Actor

Priority:

Classification:

Context: Using this Use Case the commercial transaction manager can consult, specify and/or change the information associated to one commercial transaction item. This information would be different according to the proprietary PSP.

Pre-conditions:

Flow of events: Once the commercial transaction manager has used the select commercial transaction item Use Case, uses this Use Case for getting the parameters list associated with the commercial transaction. The parameters values can be then supplied, consulted and/or modified.

Post-conditions:

Errors:

Uses:

Extends:

2.2.8 Use Cases - Schedules Management Package

UC-026 : MANAGE SCHEDULES (MGSCHS.)

Actors: Production Manager Actor

Priority:

Classification:

Context: The purpose of this Use Case is to allow to the Production Manager Actor to manipulate the whole list of Schedules.

Pre-conditions:

Flow of events: The Actor asks to the package for manage the schedules. The PSP gives to the Actor the whole list of schedules. If there is no schedules defined, the Actor can use the Add Schedule Use Case for the addition of a new Schedule item . Then the Production Manager Actor can select one specific item of Schedule for further specification and or calculation using the Select Item Use Case. Also, the Delete Schedule Use Case can be used for deletion of an existing item of resource.

Notes:

This Use Case is general to the both functionalities, definition (or redefinition, in case of rescheduling) of a problem, and for retrieving the results (time and utilities consumption).

Post-conditions:

Errors:

Uses: Add Schedule, Delete Schedule and Select Schedule Item

Extends:

UC-027 : ADD SCHEDULE (ADSCH.)

Actors: Production Manager Actor

Priority:

Classification:

Context: This Use Case is for the addition of a new item of Schedule

Pre-conditions:

Flow of events: Once that the Production Manager have used the Manage Schedules Use Case, he asks to the PSP to add new schedule item. The PSP creates a new schedule instance and the new schedule item is added to the existing list. The new item is ready to be selected, specified and solved.

Post-conditions: The new schedule item is successfully created.

Errors:

Uses:

Extends:

UC-028 : DELETE SCHEDULE (DLSCH.)

Actors: Production Manager Actor

Priority:

Classification:

Context: This Use Case represents the deletion of an existing item of schedule.

Pre-conditions: The item to delete exists.

Flow of events: Once that the Production Manager has used the Manage Schedules Use Case, the Actor can delete an existing item from the schedules list. The item to delete will be referenced by its name or position in the list.

Post-conditions: The item is successfully deleted.

Errors:

Uses:

Extends:

UC-029 : SELECT SCHEDULE ITEM (SLSCHIT.)

Actors: Production Manager Actor

Priority:

Classification:

Context: This Use Case is to allow to the Production Manager having access to a particular schedule item information.

Pre-conditions:

Flow of events: Once the Production Manager has used the manage schedules Use Case, can invoke this Use Case for the specification of a specific schedule item. In order to get the item, the Production Manager can use either the name or

the position in the list. Then the PSP offers the particular item to the Production Manager for consulting and / or modifying its current specification.

Post-conditions:

Errors:

Uses: Specify Schedule Use Case

Extends:

UC-030 : SPECIFY SCHEDULE (SPSCH.)

Actors: Production Manager Actor

Priority:

Classification:

Context: Using this Use Case the Production Manager can consult, specify and/or change the information associated to the schedule. This information will be different according to the proprietary PSP.

Pre-conditions:

Flow of events: Once the Production Manager has used the select schedule item Use Case, uses this Use Case for get the parameters list associated with the specific schedule item. The parameters values can be then supplied, consulted and/or modified.

Post-conditions:

Errors:

Uses:

Extends:

UC-031 : SOLVE SCHEDULE (SVSCH.)

Actors: Production Manager Actor

Priority:

Classification:

Context: Using this Use Case the Production Manager asks to the PSP for the solution of the scheduling problem, according to the information and parameters supplied.

Pre-conditions: The schedule was correctly specified.

Flow of events: Once the Production Manager has used the select schedule item Use Case, and specified the schedule, uses this Use Case for solve it. The PSP will solve the problem, according with the data supplied and the current configuration. The information that result of this calculation will be contained by the same schedule.

Note:

In order to get the calculation results, the Actor should use the manage schedule entries and the get resources requirements Use Cases.

Post-conditions: The scheduling problem is successfully solved.

Errors:

Uses:

Extends:

UC-032: GET RESOURCES REQUIREMENTS (GTRSRQ.)

Actors: Production Manager Actor

Priority:

Classification:

Context: Using this Use Case the Production Manager asks to the PSP for have access to the resources requirements of the currently selected schedule.

Pre-conditions:

Flow of events: Once the Production Manager has used the select schedule item Use Case, can use this Use Case for consult or specify the estimated resources requirements. Then, the PSP will give him access to the list of resources requirement, in an analogous way as did with manage resources requirements Use Case.

Post-conditions:

Errors:

Uses: Manage Resources Requirements Use Case.

Extends:

UC-033 : MANAGE SCHEDULES ENTRIES (MGSCHETS.)

Actors: Production Manager Actor

Priority:

Classification:

Context: The purpose of this Use Case is to allow to the Production Manager Actor to manipulate the whole list of Schedules Entries.

Pre-conditions:

Flow of events: The Actor asks to the package for manage the Schedules Entries. The PSP gives to the Actor the whole list of Schedules Entries at the current Schedule information level. If there is no Schedule Entry defined, the Actor can use the Add Schedule Entry Item Use Case for the addition of a new Schedule Entry Item. Then the Schedules Manager Actor can select a specific item of Schedule Entry for further specification and / or consulting, using the Select Schedule Entry Item Use Case. Also, the Delete Schedule Entry Item Use Case can be used for deleting an existing item of Schedule Entry.

Notes:

This Use Case is general to any Schedule information level (Campaign, Batch, Unit procedure, Operation and Phase). The level covered will depend on the capabilities of the particular PSP.

Post-conditions:

Errors:

Uses: Add Schedule Entry, Delete Schedule Entry and Select Schedule Entry Item.

Extends:

UC-034 : ADD SCHEDULE ENTRY ITEM (ADSCHETIT.)

Actors: Production Manager Actor

Priority:

Classification:

Context: This Use Case is for the addition of a new item of Schedule Entry.

Pre-conditions:

Flow of events: Once that the Schedules Manager have used the Manage Schedule Entries Use Case, he asks to the PSP to add new Schedule Entry item. The PSP creates a new Schedule Entry Item instance and the new Schedule Entry item is added to the existing list. The new item is ready to be selected, specified, and consulted.

Post-conditions: The new Schedule Entry item is successfully created.

Errors:

Uses:

Extends:

UC-035 : DELETE SCHEDULE ENTRY ITEM (DLSCHETIT.)

Actors: Schedules Manager Actor

Priority:

Classification:

Context: This Use Case represents the deletion of an existing Schedule Entry Item.

Pre-conditions: The Schedule Entry Item to delete exists.

Flow of events: Once that the Schedules Manager have used the Manage Schedule Entries Use Case, the Actor can delete an existing item of the Schedule Entries list. The item to delete will be referenced by its name or position in the list.

Post-conditions: The item is successfully deleted.

Errors:

Uses:

Extends:

UC-036 : SELECT SCHEDULE ENTRY ITEM (SLSCHETIT.)

Actors: Schedules Manager Actor

Priority:

Classification:

Context: This Use Case is to allow to the Schedule manager having access to a particular Schedule Entry Item information.

Pre-conditions:

Flow of events: Once the Schedule Manager has used the manage Schedule Use Case, can invoke this Use Case for the specification of a specific Schedule item. In order to get the item, the Schedule Manager can use either the name or the position in the list. Then the PSP offers the particular item to the Schedule manager for consulting and / or modifying its current specification.

Post-conditions:

Errors:

Uses: Specify Schedule Entry Use Case

Extends:

UC-037 : SPECIFY SCHEDULE ENTRY ITEM (SPSCHENIT.)

Actors: Schedules Manager Actor

Priority:

Classification:

Context: Using this Use Case the Schedule manager can consult, specify and/or change the information associated to the Schedule Entry Item. This information will be different according to the proprietary PSP.

Pre-conditions:

Flow of events: Once the Production Manager has used the Select Scheduling Entry Item Use Case, uses this Use Case for get the parameters list associated with the specific Schedule Entry Item. The parameters values can be then supplied, consulted and/or modified. This Use Case also allows to the Actor to have access to the resources requirements for the selected Schedule Entry Item.

Post-conditions:

Errors:

Uses: Manage Resources Requirements Use Case

Extends:

2.3 Sequence diagrams

SQ- 1 ANALYSIS SEQUENCE DIAGRAM

This Diagram illustrates the dynamic interaction between actors and logic packages.

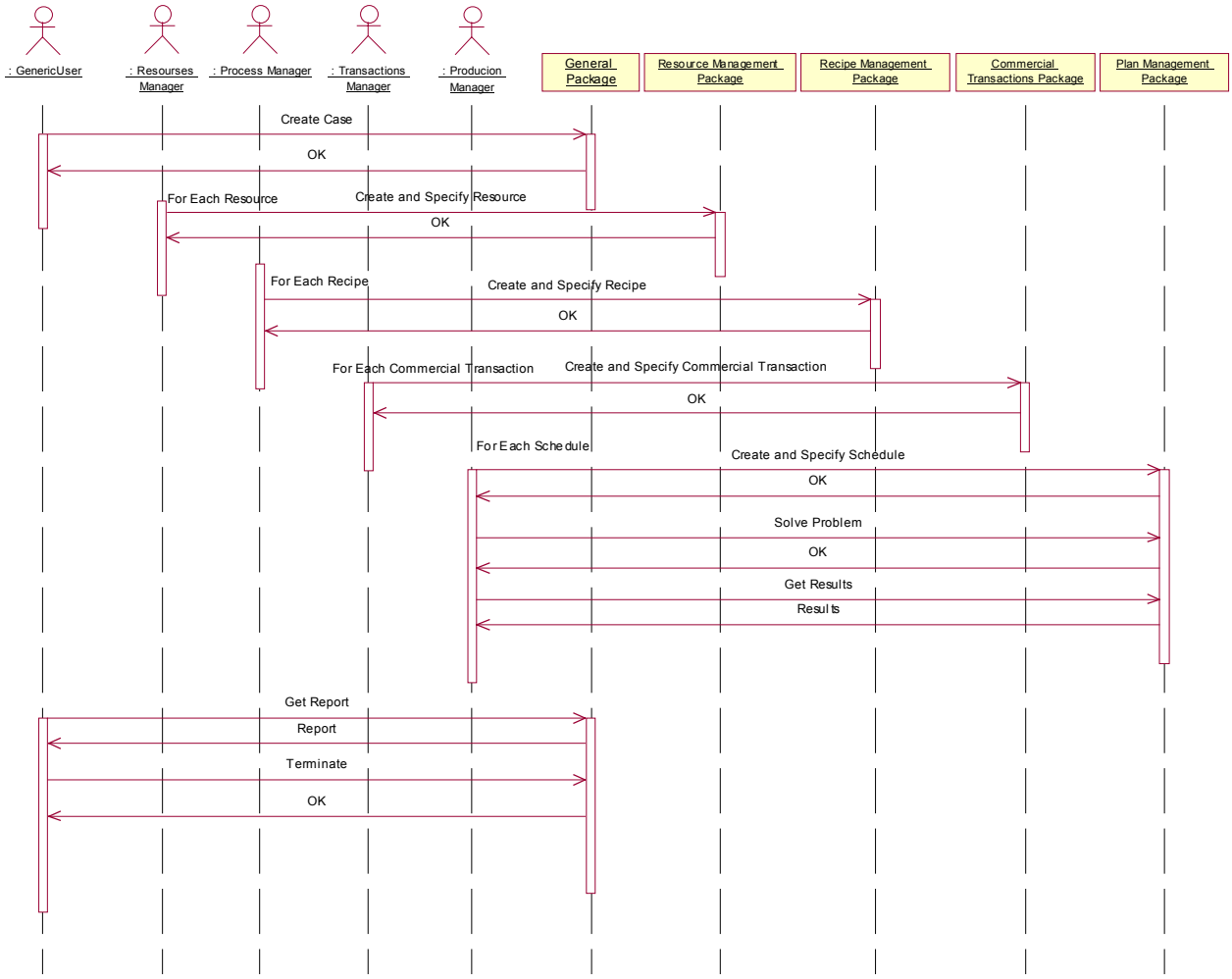


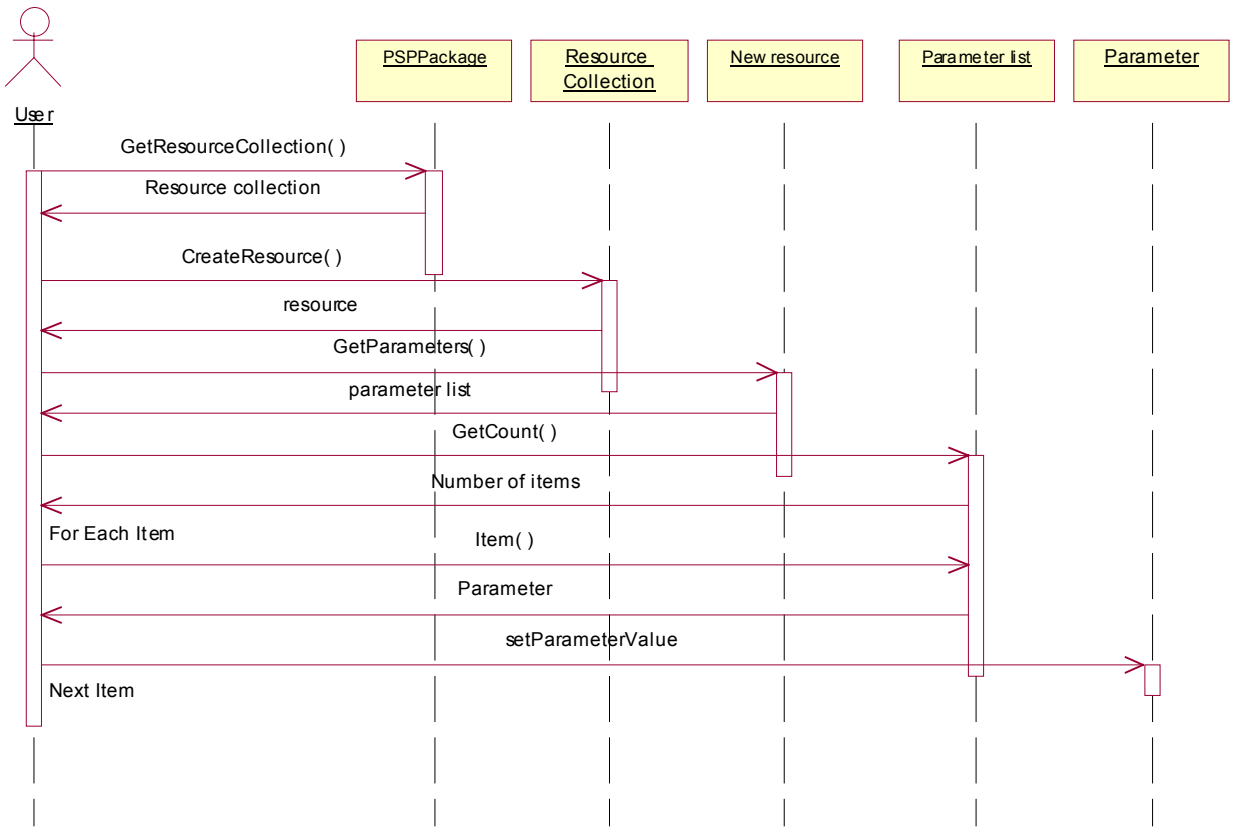
Figure 13 Sequence diagram

3. Analysis and design

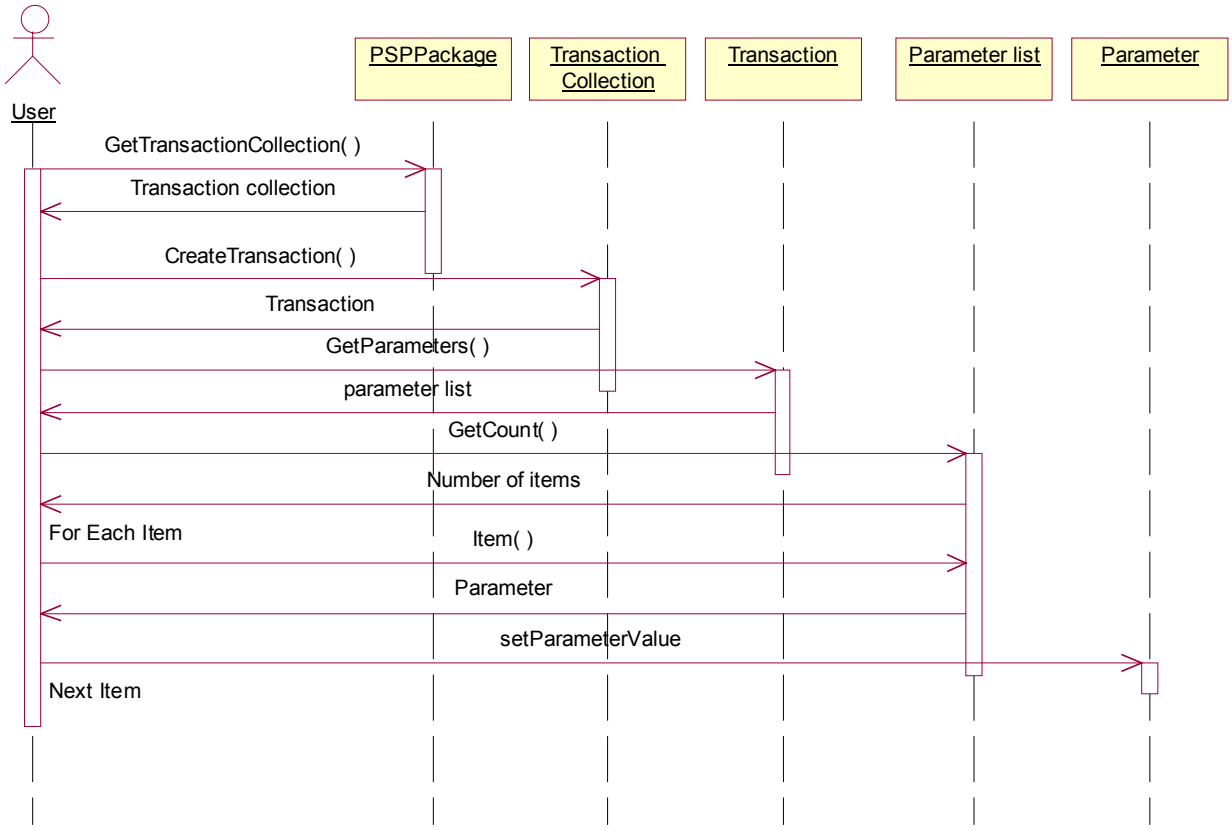
3.1 Overview

3.2 Sequence diagrams

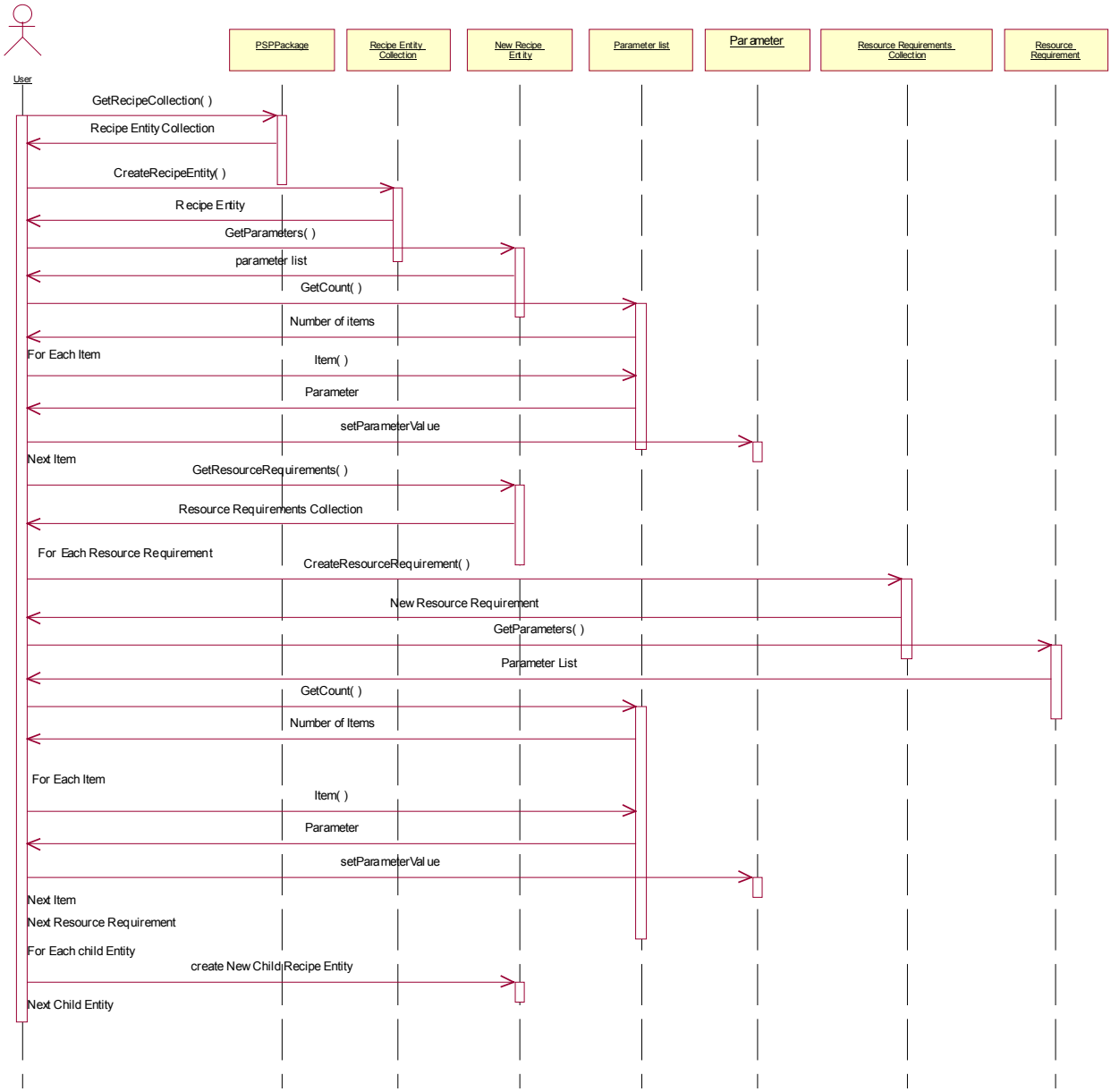
SQ- 1 CREATE A RESOURCE



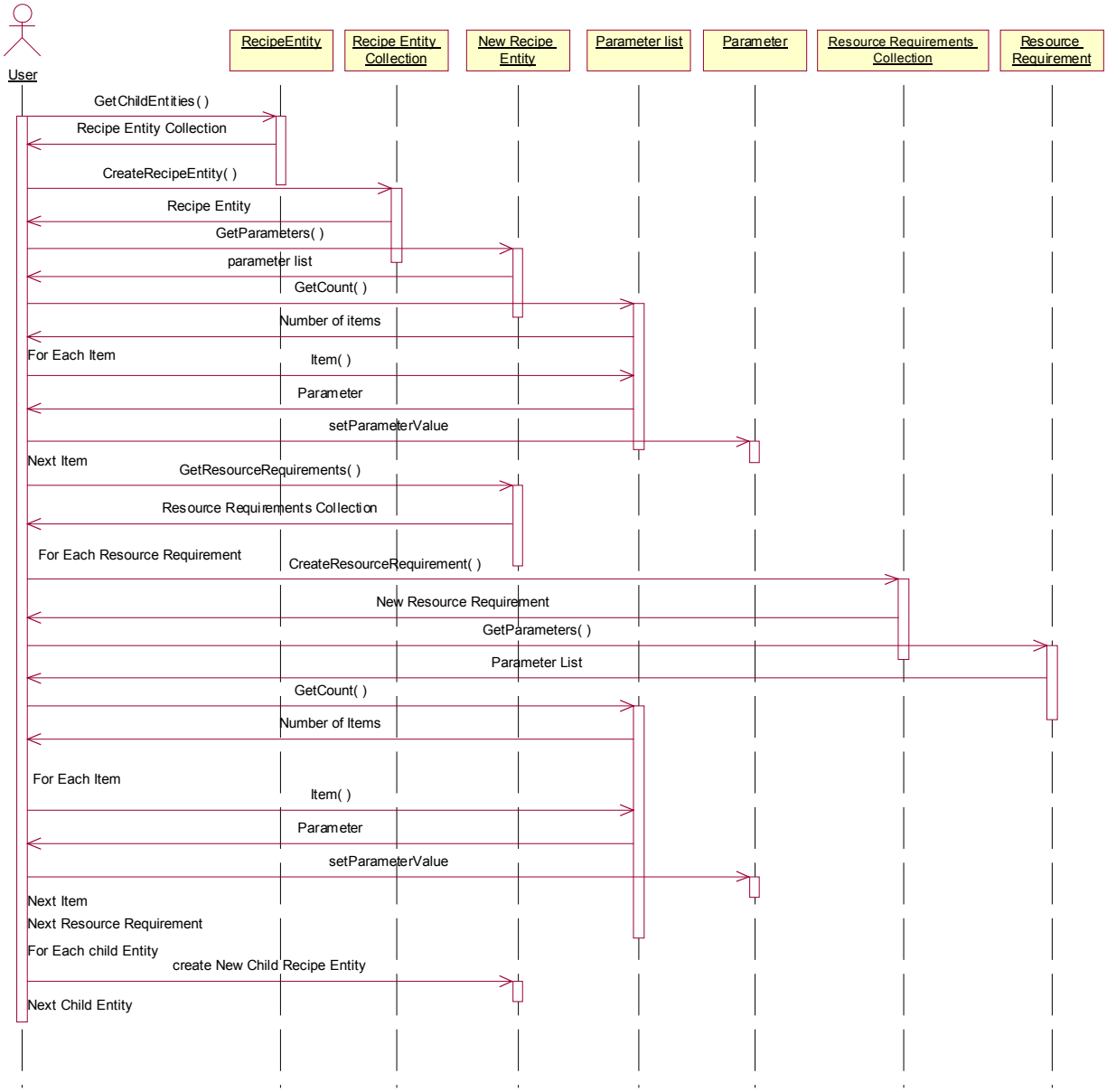
SQ- 2 CREATE A COMMERCIAL TRANSACTION



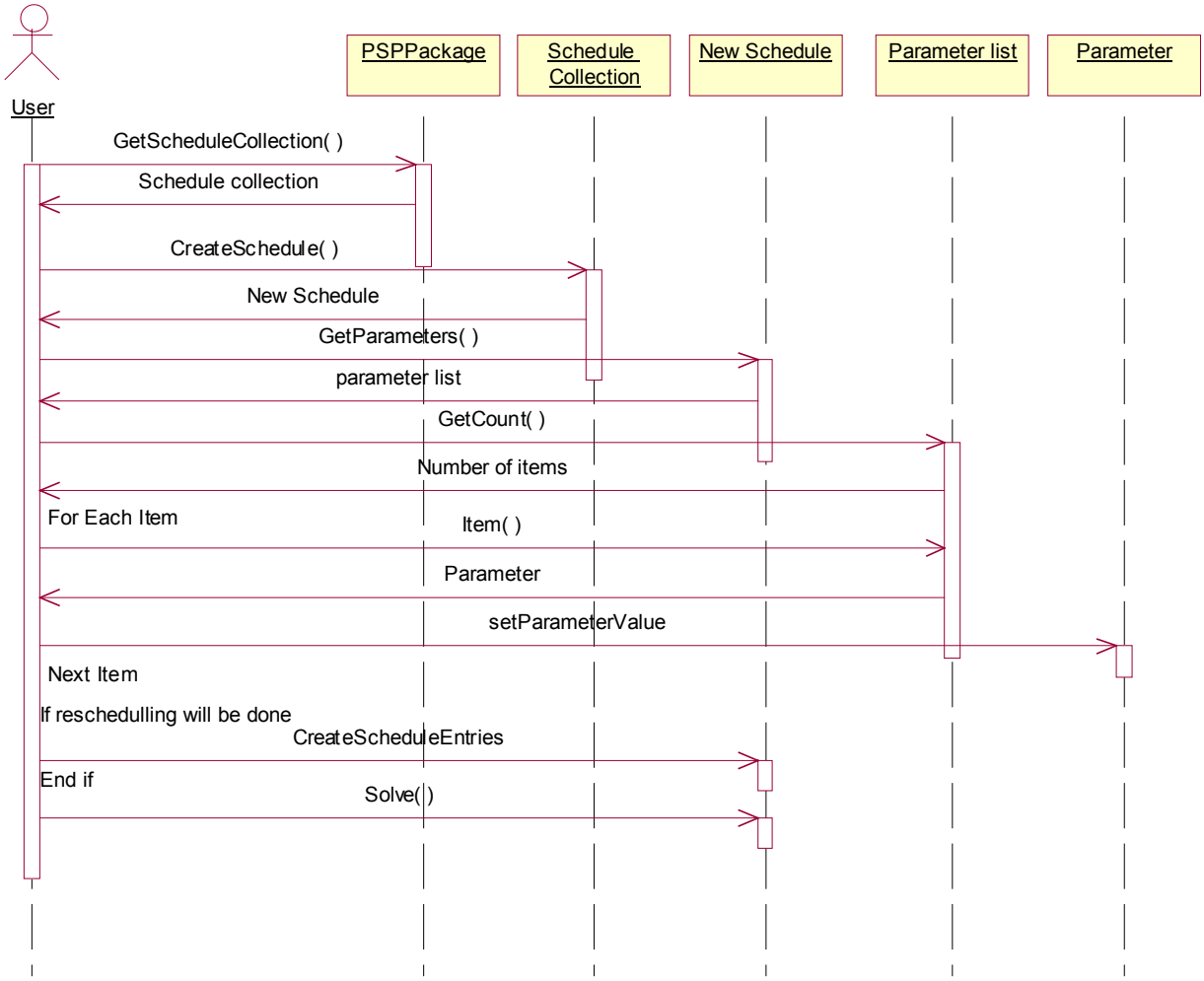
SQ- 3 CREATE A RECIPE ENTITY


User
PSPPackage
Recipe Entity Collection
New Recipe Entity
Parameter list
Parameter
Resource Requirements Collection
Resource Requirement
GetRecipeCollection( )
Recipe Entity Collection
CreateRecipeEntity( )
Recipe Entity
GetParameters( )
parameter list
GetCount( )
Number of items
For Each Item
Item( )
Parameter
setParameterValue
Next Item
GetResourceRequirements( )
Resource Requirements Collection
For Each Resource Requirement
CreateResourceRequirement( )
New Resource Requirement
GetParameters( )
Parameter List
GetCount( )
Number of Items
For Each Item
Item( )
Parameter
setParameterValue
Next Item
Next Resource Requirement
For Each child Entity
create New Child Recipe Entity
Next Child Entity

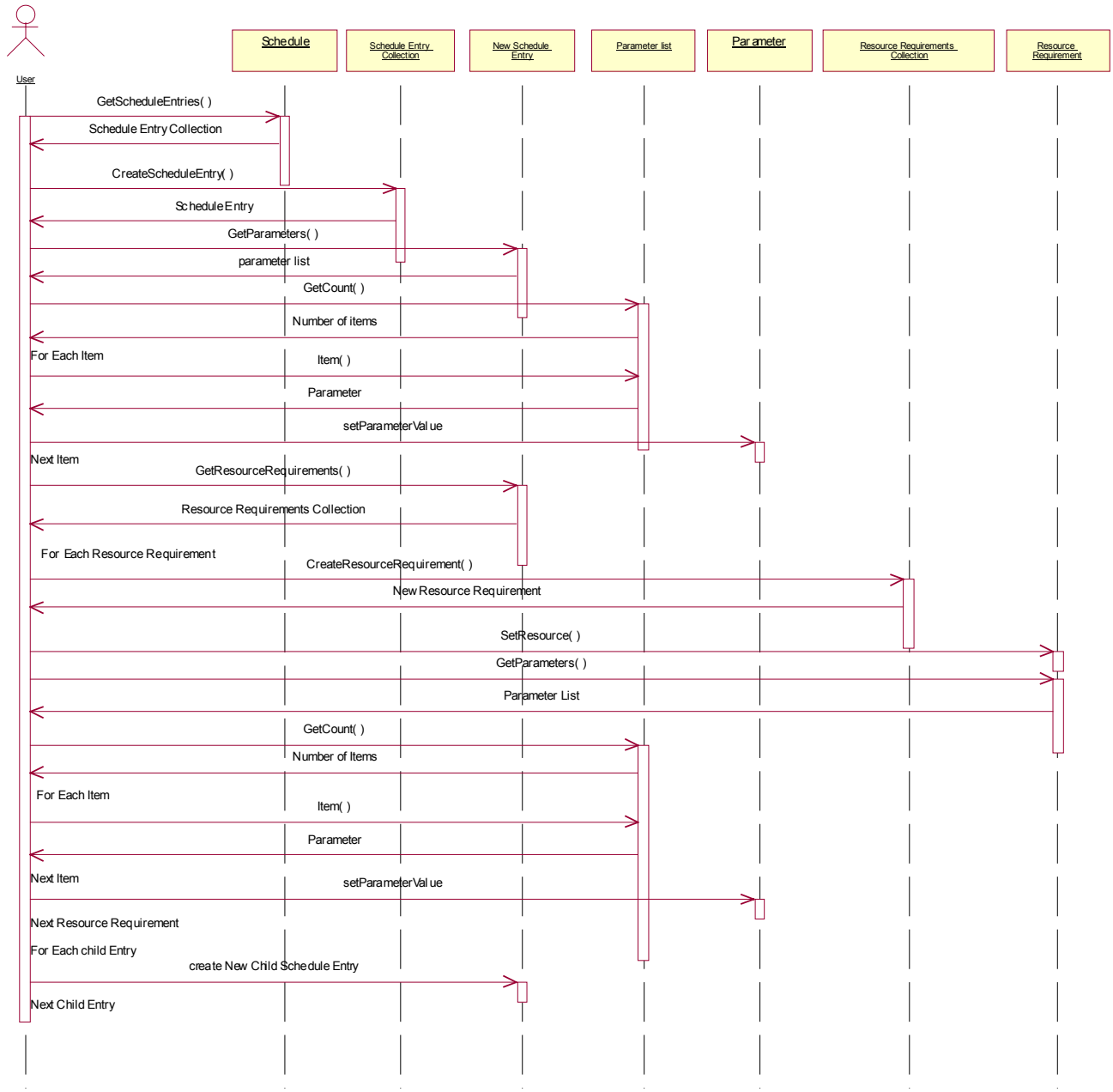
SQ- 4 CREATE A CHILD RECIPE ENTITY



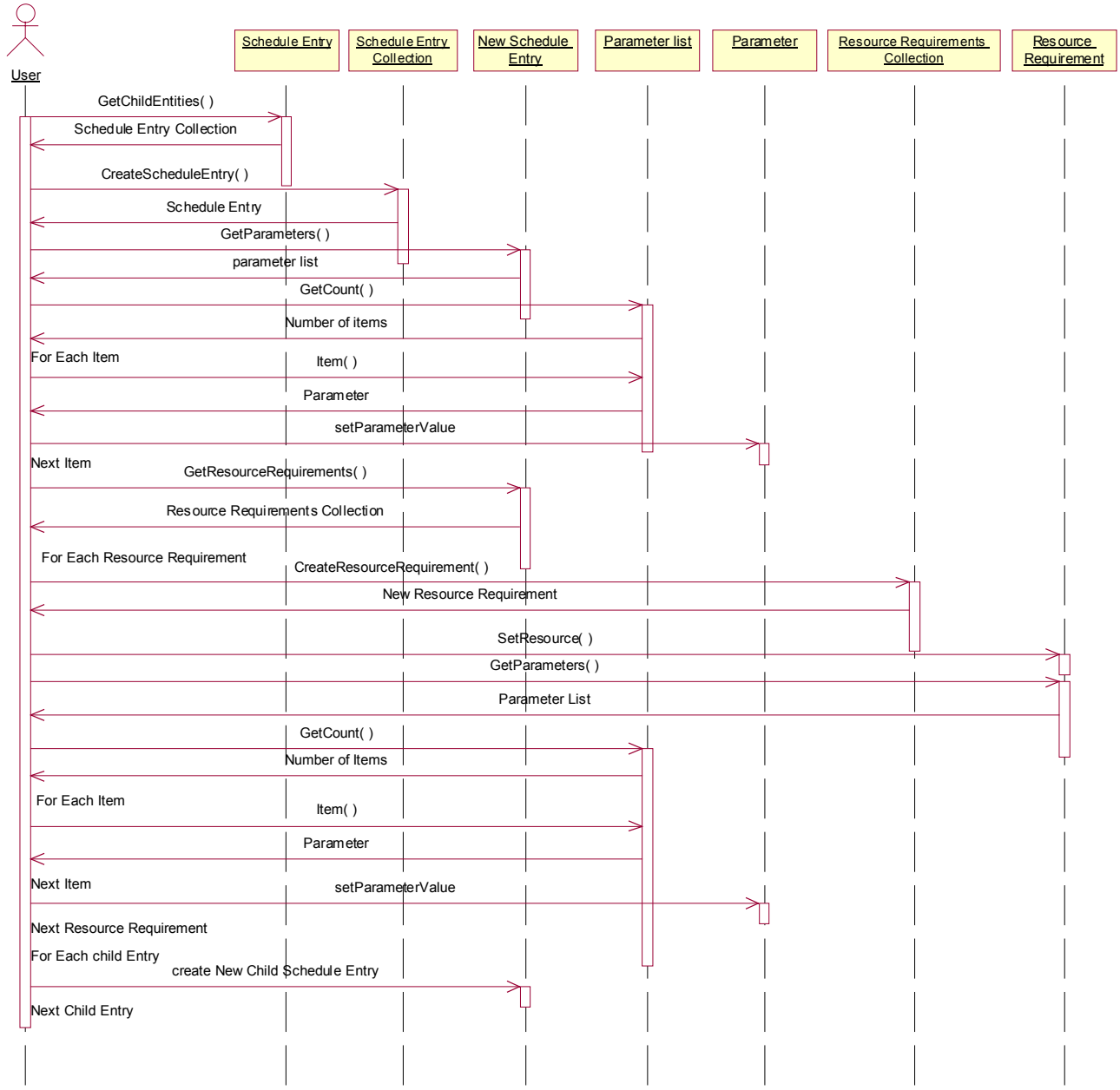
SQ- 5 CREATE AND SOLVE A NEW SCHEDULE



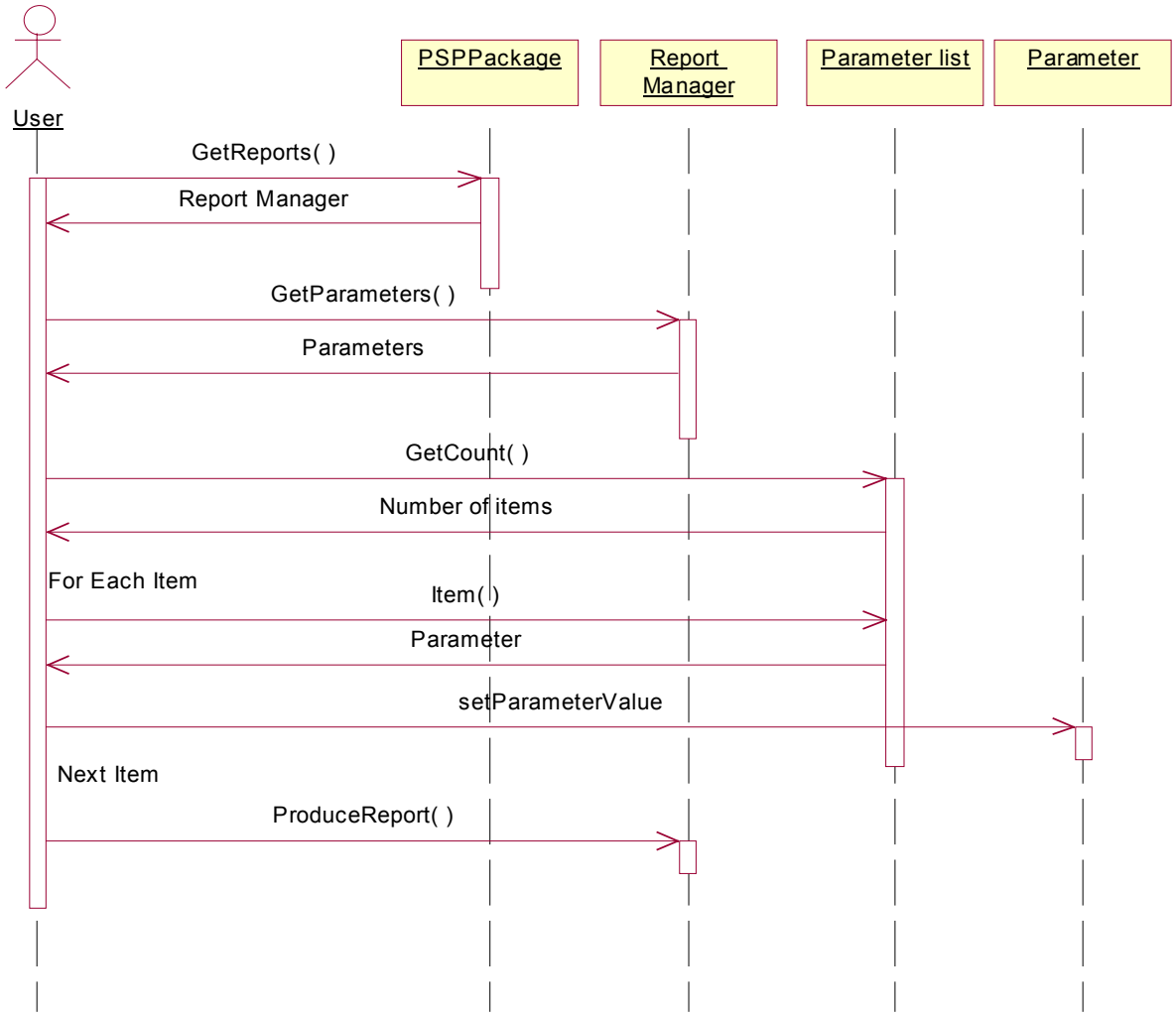
SQ- 6 CREATE A NEW SCHEDULING ENTRY



SQ- 7 CREATE A CHILD SCHEDULING ENTRY



SQ- 8 GENERATE A REPORT



3.3 Interface diagrams

3.3.1 Comments

The following diagram shows the relationships between the different component of a PSP system and their corresponding interfaces for external communication.

In order to clarify the diagram, the inheritance relationships between ICapeIdentification and the corresponding child interfaces (ICapePSP, ICapePSPResource, ICapePSPRecipeEntity, ICapePSPSchedule, ICapeScheduleEntry, ICapePSPTransaction, ICapePSPReport and ICapePSPResourceRequirement) are not shown.

The inheritance relationship among ICapePSPCollection and all the ICapePSPsomethingCollection interfaces is also excluded. ICapePSPCollection inherits from ICapeCollection that comes from the Common Collection Interface.

ICapePSP is a PMC primary object and so inherits from ICapeUtilities and provides persistence capability.

3.3.2 Diagram

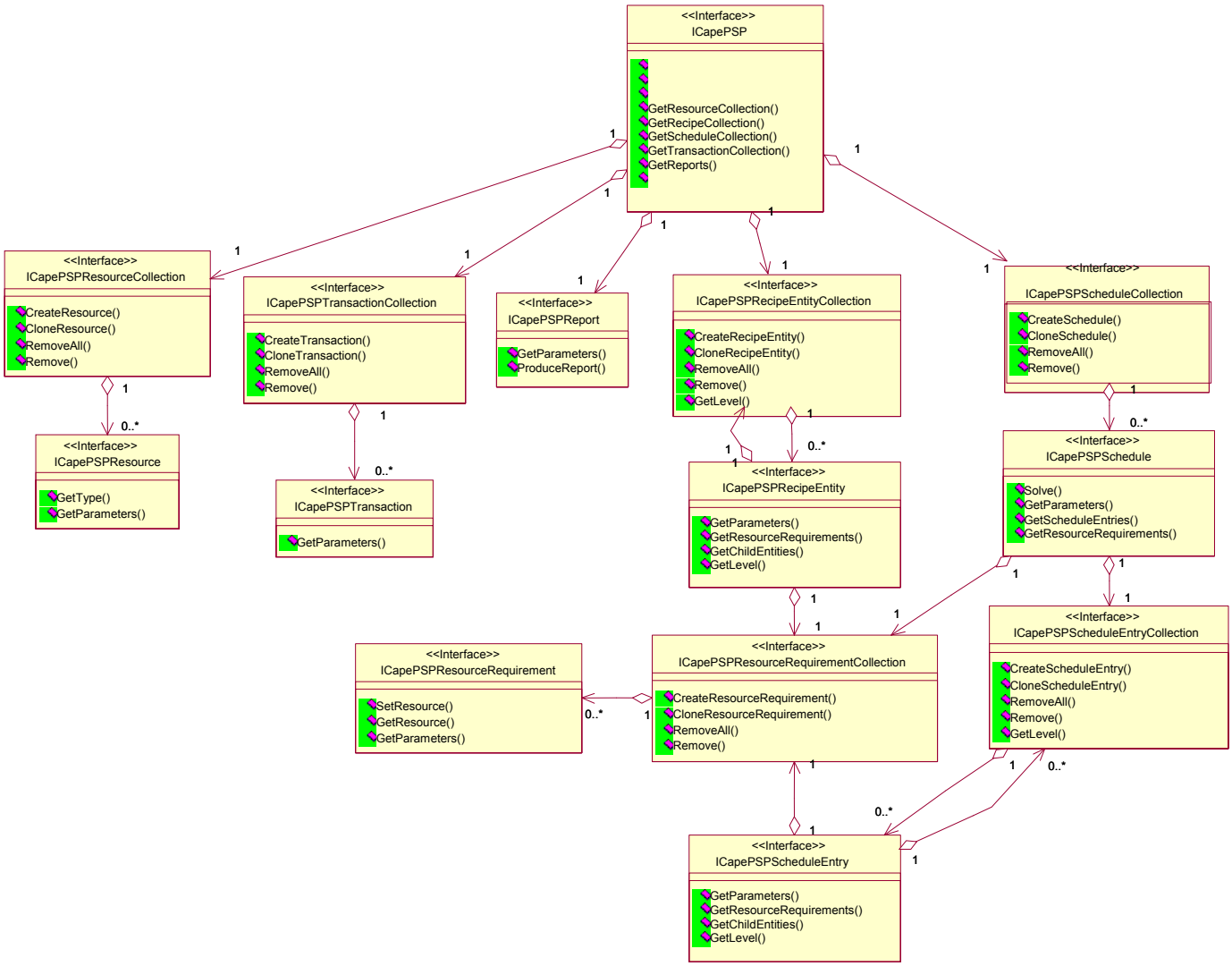


Figure 14 Interface diagram

3.4 State diagrams

3.5 Other diagrams

3.6 Interfaces descriptions

3.6.1 ICapePSP

Interface Name	ICapePSP
Method Name	GetResourceCollection
Returns	ICapePSPResourceCollection

Description

Returns the collection of resources (ICapePSPResourceCollection, ICapePSPCollection).

From here the client can use the item method to get and specific resource. The client can also use the remaining methods of the ICapePSPResourceCollection and ICapePSPCollection interfaces.

Arguments

None.

Errors

ECapeUnknown, EcapeFailedInitialisation, ECapeBadInvOrder

Interface Name	ICapePSP
Method Name	GetRecipeCollection
Returns	ICapePSPRecipeEntityCollection

Description

Returns the collection of higher level Recipe Items. Normally the higher level items will be Master Recipes, but can also be Unit procedures in some scheduling and planning packages. (i. e. ICapePSPRecipeEntityCollection).

From here the client can use the item method to get and specify recipe items. The client can also use the remaining methods of the ICapePSPRecipeCollection interface.

According to the ISA S88-2 Standard, the recipe Item detail levels should be:

- Master Recipe
- Unit Procedure
- Operation
- Phase

The detail level contemplated is up to the PSP package implementation.

Arguments

None.

Errors

ECapeUnknown, EcapeFailedInitialisation, ECapeBadInvOrder

Interface Name	ICapePSP
Method Name	GetScheduleCollection
Returns	ICapePSPScheduleCollection

Description

Returns an ICapePSPScheduleCollection interface.

From here the client gets the access to the different Schedules configuration calculation and data and creates new schedules. The client can perform calls to the ICapePSPCollection and ICapePSPCollection methods.

Arguments

None.

Errors

ECapeUnknown, ECapeFailedInitialisation, ECapeBadInvOrder

Interface Name	ICapePSP
Method Name	GetTransactionCollection
Returns	ICapePSPTransactionCollection

Description

Returns the collection of commercial transactions, this is the demands for raw materials and orders for products. (ICapePSPTransactionCollection).

From here the client can use the item method to get and specify transactions. The client can also use the remaining methods of the ICapePSPTransactionCollection and ICapePSPCollection interfaces.

Arguments

None.

Errors

ECapeUnknown, ECapeFailedInitialisation, ECapeBadInvOrder

Interface Name	ICapePSP
Method Name	GetReports
Returns	ICapePSPReport

Description

Returns an ICapePSPReport interface.

From here the client gets the access to the different Reports configuration and generation.

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation, ECapeBadInvOrder

3.6.2 ICapePSPReport

Interface Name	ICapePSPReport
Method Name	ProduceReport
Returns	--

Description

Produce a report according to the parameters information that has been already supplied trough the corresponding parameters interface.

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation, ECapeNoImpl, ECapeBadInvOrder

Interface Name	ICapePSPReport
Method Name	GetParameters
Returns	CapeInterface (ICapePSPCollection)

Description

Returns an ICapePSPCollection containing the parameters of the report selected. If no report is selected an error is raised.

Arguments

None

Errors

ECapeUnknown, , ECapeFailedInitialisation, ECapeNoImpl, ECapeBadInvOrder

3.6.3 ICapePSPCollection

The generic interface for PSP Collections of items, it contains all the methods for accessing the different items, Specific Collection interfaces are inherited from this interface when creation and destroying specific methods are required.

Interface Name	ICapePSPCollection
Method Name	Index
Returns	CapeLong

Description

Returns the index of the element identified with name otherwise raises an error.

Arguments

Name	Type	Description
[in] name	CapeString	The name of the element whose index is requested.

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeFailedInitialisation

3.6.4 ICapePSPResource

Interface Name	ICapePSPResource
Method Name	GetType
Returns	CapeLong

Description

It gets the type of resource: Material, Unit, or whatever the proprietary PSP allows. According with its value, the resource will have different parameters.

The suggested minimum types of resource are:

- 0- Not defined
- 1- Material
- 2- Unit
- 3-99 Reserved
- 100- ... Specific package defined

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPResource
Method Name	GetParameters
Returns	ICapePSPCollection

Description

It returns the collection of parameters (ICapePSPCollection).

These are delivered as a collection of elements exposing the interface ICapeParameter. From there, the client could extract the ICapeParameterSpec interface or any of the typed interfaces such as ICapeRealParameterSpec, once the client establishes that the Parameter is of type double.

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

3.6.5 ICapePSPResourceCollection

Inherits from ICapePSPCollection

Interface Name	ICapePSPResourceCollection
Method Name	CreateResource
Returns	ICapePSPResource

Description

Allows the creation of a new Resource of a specific kind, which is added to the Resource collection. If the name or kind is invalid an error is raised.

The suggested minimum kinds of resource are:

0- Not defined

1- Material

2- Unit

3-99 Reserved

100- ... Specific package defined

Arguments

Name	Type	Description
[in] name	CapeString	Name of the resource to be added
[in] kind	CapeLong	Kind of resource to be added

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeBadArgument, ECapeFailedInitialisation CloneResource

Interface Name	ICapePSPResourceCollection
Method Name	CloneResource
Returns	ICapePSPResource

Description

Allows the creation of a new Resource with the same property values and kind as the original resource, and a new name. This is used, for example, to specify a second storage tank similar to the first. An error is raised if the new name already exists for a resource of the same kind.

Arguments

Name	Type	Description
[in] origin	ICapePSPResource	Resource to be copied
[in] name	CapeString	Name of the new resource created.

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeBadArgument, ECapeFailedInitialisation

Interface Name	ICapePSPResourceCollection
Method Name	RemoveAll
Returns	--

Description

Deletes, if possible, all the resources in the resource collection. If a resource is used somewhere (i.e. a Resource requirement of a RecipeEntity), the corresponding resource will not be deleted forcing an error.

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPResourceCollection
Method Name	Remove
Returns	--

Description

Deletes, if possible the resource with an specific index. If the resource is used somewhere (eg. a Resource requirement of a RecipeEntity), the resource can not be deleted forcing an error.

Arguments

Name	Type	Description
[in] index	CapeLong	Index of the resource to be deleted.

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeOutOfBounds, ECapeFailedInitialisation

3.6.6 ICapePSPScheduleCollection

Inherits form ICapePSPCollection

Interface Name	ICapePSPScheduleCollection
Method Name	CreateSchedule
Returns	ICapePSPSchedule

Description

Allows the creation of a new Schedule, which is added to the Schedule collection. If the name is not valid (repeated) it returns an error.

Arguments

Name	Type	Description
[in] name	CapeString	Name of the schedule to be added

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeFailedInitialisation

Interface Name	ICapePSPScheduleCollection
Method Name	CloneSchedule
Returns	ICapePSPSchedule

Description

Allows the creation of a new Schedule with the same property values and contents of the schedule resource provided. The new schedule is created with the name provided. An error is raised if the new name already exists for an item of the same kind.

Arguments

Name	Type	Description
[in] origin	ICapePSPSchedule	schedule to be copied
[in] name	CapeString	New name for the copied schedule.

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeBadArgument, ECapeFailedInitialisation

Interface Name	ICapePSPScheduleCollection
Method Name	RemoveAll
Returns	--

Description

Deletes, if possible, all the schedules in the schedule collection. If a schedule can not be deleted an error is raised.

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPScheduleCollection
Method Name	Remove
Returns	--

Description

Deletes, if possible the schedule with a specific index. If a schedule can not be deleted an error is raised.

Arguments

Name	Type	Description
[in] index	CapeLong	Index of the schedule to be deleted.

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeOutOfBounds, ECapeFailedInitialisation

3.6.7 ICapePSPTransactionCollection

Inherits from ICapePSPCollection

Interface Name	ICapePSPTransactionCollection
Method Name	CreateTransaction
Returns	ICapePSPTransaction

Description

Allows the creation of a new Commercial transaction of the specified type, which is added to the Transaction collection. If the name or type is not valid it returns an error.

Possible values of transactions types are:

- 0 Not Defined
- 1 Order
- 2 Demand
- 3-99 Reserved
- 100-... Specific PSP package defined.

Arguments

Name	Type	Description
[in] name	CapeString	Name of the transaction to be added
[in] type	CapeLong	Type of the transaction to be added

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeBadArgument, ECapeFailedInitialisation

Interface Name	ICapePSPTransactionCollection
Method Name	CloneTransaction
Returns	ICapePSPTransaction

Description

Allows the creation of a new transaction with the same parameter and type values of the transaction provided and a different name. An error is raised if the new name already exists for an item of the same kind.

Arguments

Name	Type	Description
[in] origin	ICapePSPTransaction	Transaction to be copied
[in] name	CapeString	Name for the new transaction.

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeBadArgument, ECapeFailedInitialisation

Interface Name	ICapePSPTransactionCollection
Method Name	RemoveAll
Returns	--

Description

Deletes, if possible, all the Transactions in the Transaction collection. If a transaction can not be deleted (i.e. it is used somewhere) an error is raised.

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPTransactionCollection
Method Name	Remove
Returns	--

Description

Deletes, if possible the transaction with a specific index. If a transaction can not be deleted an error is raised.

Arguments

Name	Type	Description
[in] index	CapeLong	Index of the transaction to be deleted.

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeFailedInitialisation

3.6.8 ICapePSPRecipeEntityCollection

Inherits from ICapePSPCollection

Interface Name	ICapePSPRecipeEntityCollection
Method Name	CreateRecipeEntity
Returns	ICapePSPRecipeEntity

Description

Allows the creation of a new Recipe Entity, which is added to the Recipe Entity collection. If the name is not valid it returns an error. The detailed level of the recipe entity added can be obtained using the method GetLevel.

According to the ISA S88-2 Standard, the recipe Item detail levels should be:

- 0- Invalid
- 1- Master Recipe
- 2- (Reserved)
- 3-Unit Procedure
- 4-Operation
- 5-Phase
- 6..99- (Reserved)
- 100+ Specific PSP defined

The detail level contemplated is up to the PSP package implementation.

Arguments

Name	Type	Description
[in] name	CapeString	Name of the Recipe Entity to be added

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeFailedInitialisation

Interface Name	ICapePSPRecipeEntityCollection
Method Name	CloneRecipeEntity
Returns	ICapePSPRecipeEntity

Description

Allows the creation of a new Recipe Entity with the same parameter values of the Recipe Entity provided and a different name. The Provided Recipe entity level should be the same as the entity level provided with the method GetLevel. An error is raised if the new name already exists for an item of the same kind.

Arguments

Name	Type	Description
[in] origin	ICapePSPRecipeEntity	Recipe Entity to be copied
[in] name	CapeString	New name for the new recipe entity.

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeBadArgument, ECapeFailedInitialisation

Interface Name	ICapePSPRecipeEntityCollection
Method Name	RemoveAll
Returns	--

Description

Deletes, if possible, all the Recipe Entities in the Recipe entities collection. If a recipe entity can not be deleted (i.e. is used somewhere) an error is raised.

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPRecipeEntityCollection
Method Name	Remove
Returns	--

Description

Deletes, if possible the Recipe Entity with an specific index. If a recipe entity can not be deleted an error is raised.

Arguments

Name	Type	Description
[in] index	CapeLong	Index of the recipe entity to be deleted.

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeFailedInitialisation, ECapeOutOfBounds

Interface Name	ICapePSPRecipeEntityCollection
Method Name	GetLevel
Returns	CapeLong

Description

Returns the detail level of the recipe entities stored.

According to the ISA S88-2 Standard, the recipe Item detail levels should be:

- 0- Invalid
- 1- Master Recipe
- 2- (Reserved)
- 3-Unit Procedure
- 4-Operation
- 5-Phase
- 6..99- (Reserved)
- 100+ Specific PSP defined

Arguments

Name	Type	Description
[out,return] level	CapeLong	Level or the Recipe entities stored

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeFailedInitialisation

3.6.9 ICapePSPScheduleEntryCollection

Interface Name	ICapePSPScheduleEntryCollection
Method Name	CreateScheduleEntry
Returns	ICapePSPScheduleEntry

Description

Allows the creation of a new Schedule Entry, which is added to the Schedule Entry collection. If the name is not valid it returns an error. The detailed level of schedule entry added can be obtained using the method GetLevel.

According to ISA S88-2 Standard the possible levels are:

- 0 Invalid
- 1 Campaign
- 2 Batch
- 3 Unit procedure
- 4 Operation
- 5 Phase
- 6-99 (Reserved)
- 100+ Specific for each PSP

Arguments

Name	Type	Description
[in] name	CapeString	Name of the Schedule Entry to be added

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeFailedInitialisation

Interface Name	ICapePSPScheduleEntryCollection
Method Name	CloneScheduleEntry
Returns	ICapePSPScheduleEntry

Description

Allows the creation of a new Schedule Entry with the same parameter values of the Schedule Entry provided. The Provided Schedule Entry entity level should be the same as the entity level provided with the method GetLevel. An error is raised if the new name already exists for an item of the same kind.

Arguments

Name	Type	Description
[in] origin	ICapePSPScheduleEntry	Schedule entry to be copied
[in] name	CapeString	Name of the new schedule entry.

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeBadArgument, ECapeFailedInitialisation

Interface Name	ICapePSPScheduleEntryCollection
Method Name	RemoveAll
Returns	--

Description

Deletes, if possible, all the Schedule Entries in the Schedule Entries collection. If a schedule entry can not be deleted (i.e. is used somewhere) an error is raised.

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPScheduleEntryCollection
Method Name	Remove
Returns	--

Description

Deletes, if possible the Schedule Entry with an specific index. If a schedule entry can not be deleted an error is raised.

Arguments

Name	Type	Description
[in] index	CapeLong	Index of the Schedule entry to be deleted.

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeOutOfBounds, ECapeFailedInitialisation

Interface Name	ICapePSPScheduleEntryCollection
Method Name	GetLevel
Returns	CapeLong

Description

Returns the detail level of the schedule entries stored.

According to ISA S88-2 Standard the possible levels are:

- 0 Invalid
- 1 Campaign
- 2 Batch
- 3 Unit procedure
- 4 Operation
- 5 Phase
- 6-99 (Reserved)
- 100+ Specific for each PSP

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

3.6.10 ICapePSPResourceRequirementCollection

Inherits from ICapePSPCollection

Interface Name	ICapePSPResourceRequirementCollection
Method Name	CreateResourceRequirement
Returns	ICapePSPResourceRequirement

Description

Allows the creation of a new Resource Requirement, which is added to the Resource Requirement collection. If the name is not valid it returns an error.

Arguments

Name	Type	Description
[in] name	CapeString	Name of the Resource Requirement to be added

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeFailedInitialisation

Interface Name	ICapePSPResourceRequirementCollection
Method Name	CloneResourceRequirement
Returns	ICapePSPResourceRequirement

Description

Allows the creation of a new Resource Requirement with the same parameter values of the resource requirement provided and with a different name. An error is raised if the new name already exists for an item of the same kind.

Arguments

Name	Type	Description
[in] origin	ICapePSPResourceRequirement	Resource Requirement to be copied
[in] name	CapeString	New name for the new resource requirement.

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeBadArgument, ECapeFailedInitialisation

Interface Name	ICapePSPResourceRequirementCollection
Method Name	RemoveAll
Returns	--

Description

Deletes, if possible, all the Resource Requirements in the Resource requirement collection. If a Resource Requirement can not be deleted (i.e. is used somewhere) an error is raised.

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPResourceRequirementCollection
Method Name	Remove
Returns	--

Description

Deletes, if possible the Resource Requirement with an specific index. If a Resource Requirement can not be deleted an error is raised.

Arguments

Name	Type	Description
[in] index	CapeLong	Index of the resource requirement to be deleted.

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeOutOfBounds, ECapeFailedInitialisation

3.6.11 ICapePSPRecipeEntity

Interface Name	ICapePSPRecipeEntity
Method Name	GetParameters
Returns	ICapePSPCollection

Description

It returns the collection of parameters (ICapePSPCollection).

These are delivered as a collection of elements exposing the interface ICapeParameter. From there, the client could extract the ICapeParameterSpec interface or any of the typed interfaces such as ICapeRealParameterSpec, once the client establishes that the Parameter is of type double.

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPRecipeEntity
Method Name	GetResourceRequirements
Returns	ICapePSPResourceRequirementCollection

Description

Returns the collection of the resource requirements for the current recipe entity (ICapePSPResourceRequirementCollection).

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPRecipeEntity
Method Name	GetChildEntities
Returns	ICapePSPRecipeEntityCollection

Description

Returns the collection of child recipe entities for the current recipe entity (ICapePSPRecipeEntityCollection). The Recipe Entity Collection level should be lower than the parent recipe entity obtained with the method GetLevel

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPRecipeEntity
Method Name	GetLevel
Returns	CapeLong

Description

Returns the level of the recipe entity.

According to the ISA S88-2 Standard, the recipe Item detail levels should be:

- 0- Invalid
- 1- Master Recipe
- 2- (Reserved)
- 3-Unit Procedure
- 4-Operation
- 5-Phase
- 6..99- (Reserved)
- 100+ Specific PSP defined

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

3.6.12 ICapePSPScheduleEntry

Interface Name	ICapePSPScheduleEntry
Method Name	GetParameters
Returns	ICapePSPCollection

Description

It returns the collection of parameters (ICapePSPCollection).

These are delivered as a collection of elements exposing the interface ICapeParameter. From there, the client could extract the ICapeParameterSpec interface or any of the typed interfaces such as ICapeRealParameterSpec, once the client establishes that the Parameter is of type double.

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPScheduleEntry
Method Name	GetResourceRequirements
Returns	ICapePSPResourceRequirementCollection

Description

Returns the collection of the resource requirements for the current Schedule Entry (ICapePSPResourceRequirementCollection).

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPScheduleEntry
Method Name	GetChildEntities
Returns	ICapePSPScheduleEntryCollection

Description

Returns the collection of child schedule entries for the current schedule entry (ICapePSPScheduleEntryCollection). The Schedule Entry Collection level should be lower than the parent schedule entry obtained with the method GetLevel

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPScheduleEntry
Method Name	GetLevel
Returns	CapeLong

Description

Returns the level of the schedule entry.

According to ISA S88-2 Standard the possible levels are:

- 0 Invalid
- 1 Campaign
- 2 Batch
- 3 Unit procedure
- 4 Operation
- 5 Phase
- 6-99 (Reserved)
- 100+ Specific for each PSP

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

3.6.13 ICapePSPSchedule

Interface Name	ICapePSPSchedule
Method Name	GetParameters
Returns	ICapePSPCollection

Description

It returns the collection of parameters (ICapePSPCollection).

These are delivered as a collection of elements exposing the interface ICapeParameter. From there, the client could extract the ICapeParameterSpec interface or any of the typed interfaces such as ICapeRealParameterSpec, once the client establishes that the Parameter is of type double.

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPSchedule
Method Name	GetResourceRequirements
Returns	ICapePSPResourceRequirementCollection

Description

Returns the collection of the resource requirements for the current Schedule (ICapePSPResourceRequirementCollection).

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPSchedule
Method Name	GetScheduleEntries
Returns	ICapePSPScheduleEntryCollection

Description

Returns the collection of schedule entries for the current schedule (ICapePSPScheduleEntryCollection).

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPSchedule
Method Name	Solve
Returns	--

Description

Perform a calculation that fulfils the specifications provided with the parameters. It can return an Error.

Arguments

None

Errors

ECapeUnknown, ECapeSolvingError, ECapeFailedInitialisation

3.6.14 ICapePSPResourceRequirement

Interface Name	ICapePSPResourceRequirement
Method Name	GetParameters
Returns	ICapePSPCollection

Description

It returns the collection of parameters (ICapePSPCollection).

These are delivered as a collection of elements exposing the interface ICapeParameter. From there, the client could extract the ICapeParameterSpec interface or any of the typed interfaces such as ICapeRealParameterSpec, once the client establishes that the Parameter is of type double.

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

Interface Name	ICapePSPResourceRequirement
Method Name	Get/SetResource
Returns	ICapePSPResource/--

Description

Gets or set the resource for a Resource Requirement

Arguments

Name	Type	Description
[--][in] resource	ICapePSPResource	The interface of the resource.

Errors

ECapeUnknown, ECapeInvalidArgument, ECapeFailedInitialisation

3.6.15 ICapePSPTransaction

Interface Name	ICapePSPTransaction
Method Name	GetParameters
Returns	ICapePSPCollection

Description

It returns the collection of parameters (ICapePSPCollection).

These are delivered as a collection of elements exposing the interface ICapeParameter. From there, the client could extract the ICapeParameterSpec interface or any of the typed interfaces such as ICapeRealParameterSpec, once the client establishes that the Parameter is of type double.

Arguments

None

Errors

ECapeUnknown, ECapeFailedInitialisation

3.7 Scenarios

4. Interface specifications

4.1 COM IDL

```
// You can get these intructions in Psp.idl file from CAPE-OPENv1-0-0.zip
```

4.2 CORBA IDL

```
// You can get these intructions in CAPE-OPENv1-0-0.idl within the
CAPEOPEN100::Business::Other::Psp module
```

5. Notes on analysis and interface specifications

The purpose of these notes is to record the rationale for decisions made in designing the planning and Scheduling interfaces, methods and attributes and to indicate those issues that require further investigation.

5.1 Issues to be resolved

The following issues have been identified, but not yet fully resolved. Experience gained in analysing the implementation of the Planning and Scheduling Package prototypes and interaction with the other subpackages of workpackage 5.3 will be a major factor in their resolution.

- Parameters: Further work is needed to define new kinds of generic parameters that will be useful in this context, especially to allow parameter containing (a parameter, which is a collection of other parameters).

- Error handling has not been implemented in the prototype and further changes in the errors raised by each methods can be appear according to the feedback provided by future versions of the prototype.

- The sequence diagrams for communication between scheduling and planning and control are not defined yet but a new document showing the interaction between the different parts of workpackage 5.3 has been released to RFC.

5.2 Decision rationale

5.2.1 Flexibility is the key element

A recursive structure for recipe and schedule definition has been adopted following the ISA SP-88-2 Standard. The basic idea can be summarised in the following figure:

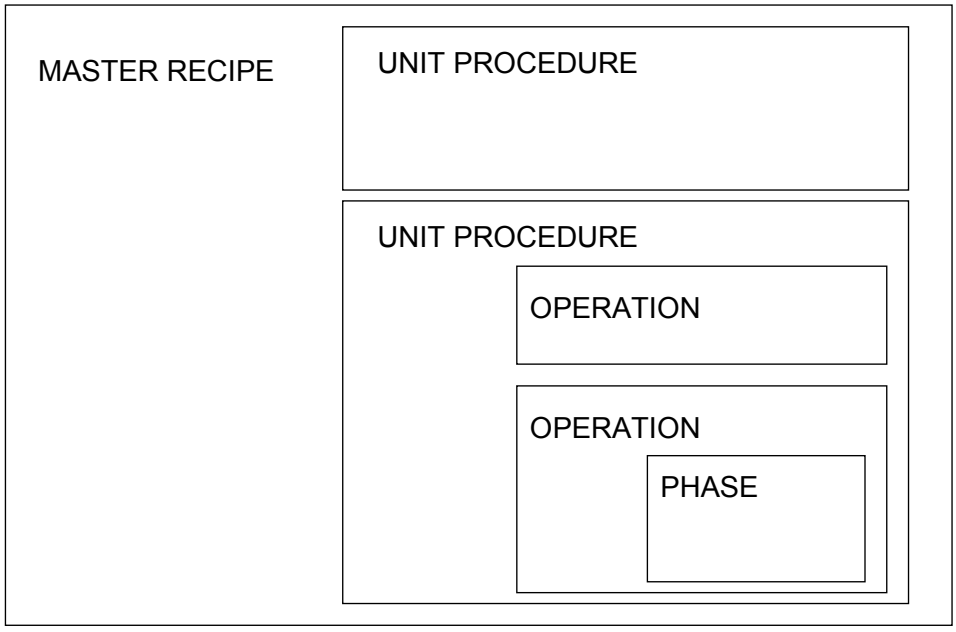


Figure 15 Recursive structure

The master recipe can contain Unit procedures, the unit procedures can contain operations and the operation can contain phases.

This recursive approach allows describing the recipe data in a standardised way at any level. This concept is very valuable in the Planning and Scheduling package as the data required for each planning and scheduling software available usually differs and it do not have a common structure.

The recursive approach allows describing with the same interface structured recipes in a hierarchical fashion as well as structures like STN or RTN.

The same basic idea is used in the description of the schedule data, an hierarchical recursive structure of Schedule entries is used. With the following hierarchy:

- Campaign
- Batches
- Unit procedure
- Operation
- Phase

The detail levels of the description will depend on the planning and scheduling package used.

5.2.2 Parameter use is intensive

The proposed use of parameters is intensive, as there are many different possible information structures for scheduling and planning. Parameters definition should be improved to include nesting of parameters (a parameter which is really a group of parameters), tables and arrays.

5.2.3 Rescheduling

We assume that re-scheduling is the same process as scheduling, but with different parameters, if the Solve method is called with the definition of the actual state of the plant, in fact, a rescheduling is performed

5.2.4 Information Format

The proposed interfaces provide ways for saving and restoring information. Even the formats can be implementation dependent, it is desirable that the PSP use some standard way to do it. Regarding this aspect, the authors strongly recommend that the file format were xml (E**X**tendend **M**ark-up **L**anguage), using structures already proposed (i. e. World Batch Forum).

6. Proposed scenario

The scenario proposed for scheduling and planning is based on a flexible configurable pilot plant located at the Chemical Engineering Department at UPC. The plant is described later on.

This new scenario is introduced since previous CO scenarios do not suit the needs for the scheduling and planning activities to be studied in workpackage 5.3.

This plant is also proposed as the scenario for the workpackage 6.2. So this situation will allow testing the communication between the two packages in a whole-integrated environment. Additionally, some points of the workpackage 2 could be also tested.

The different operation modes and recipes described in this scenario have been designed to address the testing of the most relevant issues in planning and scheduling systems such as:

(i) Use and management of different recipes

(ii) Unit assignment

(iii) Interaction of batch processes with continuous processes

(iv) On line rescheduling

6.1 Plant Description

The plant (ProCel), proposed also as a basic environment for the Open simulation and Optimisation in a Real-Time Environment package scenario, is located on the lab installations at the UPC Chemical Engineering Department. It is constituted by three tank reactors, three heat exchangers and the necessary pumps and valves to allow changes in the configurations. Equipment of the plant is fully connected and the instrumentation allows the changes of configuration by software. The flowsheet of this plant is represented in the following figure.

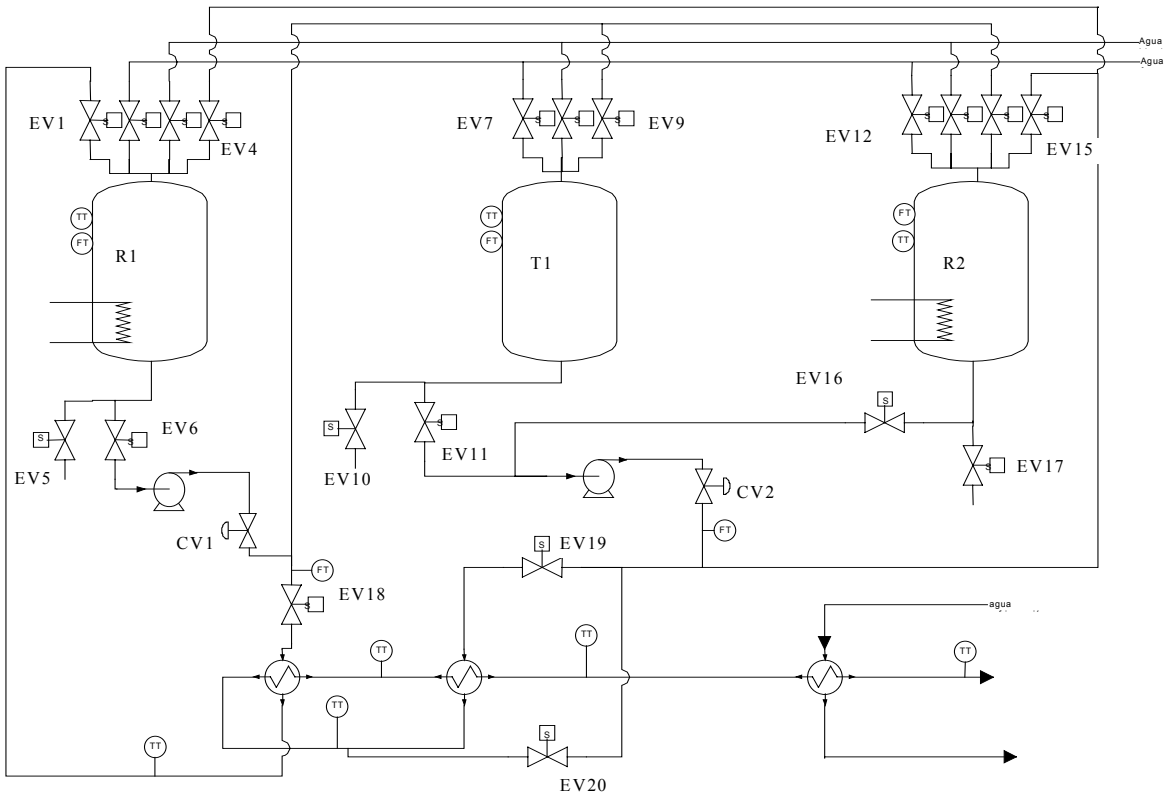


Figure 16: Pilot plant

This pilot plant is fully controlled with a distributed control commercial system and is designed to work in different modes. This flexibility allows making experiments in batch, continuous and batch-continuous mode simply configuring the control software. It also allows the TCP/IP communication with the general computer network. This communication allows on-line rescheduling experimentation.

6.2 Planning And Scheduling Package

The objective of the scheduling and planning system is determining the detailed set of operating decisions (unit to task assignment, sequencing and timing) satisfying plant (resources) and process (product recipes) constraints that maximise (minimise) a given objective function (makespan, due dates, etc.). This schedule can be eventually used directly by a control system and the scheduling system can eventually use information of the control system to perform on line rescheduling.There are three proposed modes of plant operation for the testing of this package:

(i) Multiproduct case with three stage recipes

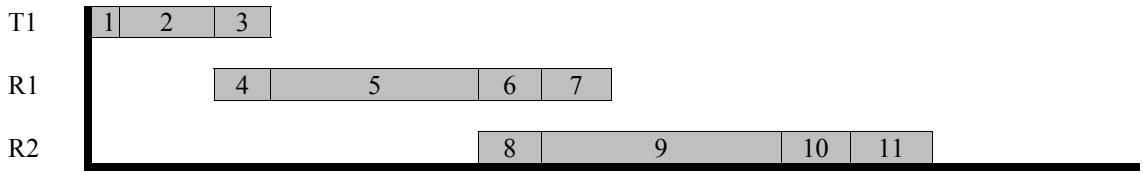
(ii) Multiproduct case with two-stage recipes and out-of-phase operation

(iii) Monoproduct case contemplating the use of continuous and discontinuous operations with a controled buffer tank.

6.2.1 Mode 1: Multiproduct Case Using All Three Batch Units

There will be 5 different recipes each with three stages and with different operation times in operations number 2, 5, 7,9 and 11 due to different operation conditions.

Representation of recipes is as follows



Operation Code	Stage Code	Operation description	Unit
1	1	Load tank 1	T1
2	1	Stirring / homogenising	T1
3	1	Discharge to reactor 1	T1
4	2	Load reactor 1	R1
5	2	Processing of Reactor 1 (Heating to a temperature set)	R1
6	2	Discharge to reactor 2	R1
7	2	Cleaning of Reactor 1	R1
8	3	Load of reactor 2	R2
9	3	Operation of reactor 2 (Heating)	R2
10	3	Discharge of reactor 2	R2
11	3	Cleaning of reactor 2	R2

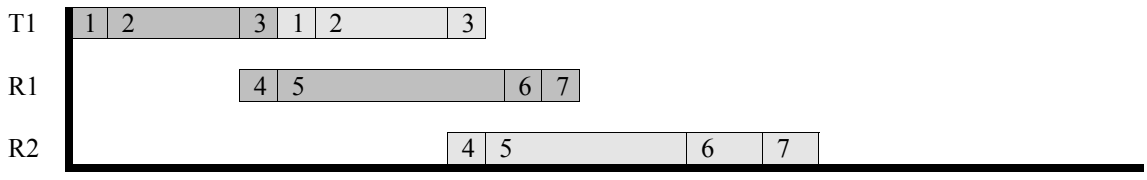
The provisional table of times for the 5 recipes are described in the following table (Final times may be changed according to the experiments realised in the real plant).

Processing times for all the operations is in minutes

Operation Code	Recipe 1	Recipe 2	Recipe 3	Recipe 4	Recipe 5
1	5	5	5	5	5
2	5	10	15	5	10
3	5	5	5	5	5
4	5	5	5	5	5
5	15	20	15	25	25
6	5	5	5	5	5
7	10	5	10	5	10
8	5	5	5	5	5
9	25	20	20	15	20
10	5	5	5	5	5
11	5	10	5	10	5

6.2.2 Mode 2: Two Stage Multiproduct Case With Out-Of-Phase Operation

In this case there will be two recipes with two stages each. In both recipes the operation time of the operation with code 5 will be different depending on the reactor chosen to perform the second stage. Representation of recipes is as follows (Representation of a Gantt chart performing two batches)



Operation code	Stage code	Operation description	Unit
1	1	Load tank 1	T1
2	1	Stirring / homogenising	T1
3	1	Discharge to reactor 1 or 2	T1
4	2	Load Reactor	R1/R2
5	2	Process of Reactor (Heating to a temperature set)	R1/R2
6	2	Discharge of final product	R1/R2
7	2	Cleaning of Reactor	R1/R2

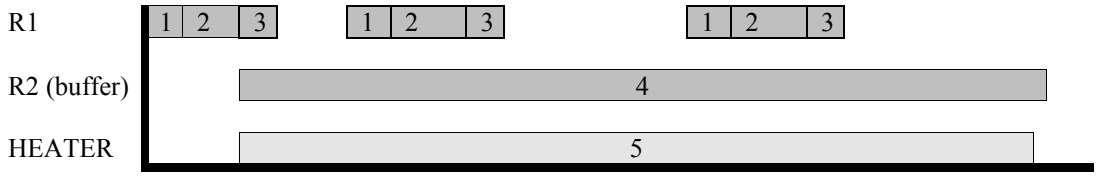
The table of operation times is the following one:

Operation	Recipe 1	Recipe 2
1	5	5
2	5	10
3	5	5
4 (R1,R2)	5	5
5(R1)	15	20
5 (R2)	20	25
6(R1,R2)	5	5
7(R1,R2)	10	5

6.2.3 Mode 3: Monoproduct Case Contemplating Continuous And Batch Operations Using A Buffer Tank

The main objective of this operation mode is to test the capability of the scheduling system for taking into account the presence of continuous and discontinuous processes linked by a buffer tank. The scheduling system should be able to manage the information flow in the buffer tank and make the appropriate timing of the operations thus predicting the evolution of the level in the buffer tank.

Representation of recipes is as follows (Representation of a Gantt chart performing three batches and the operation of the continuos part and the occupation of the buffer tank)



Operation code	Stage Code	Operation description	Unit
1	1	Load Reactor	R1
2	1	Stirring / homogenising/heating	R1
3	1	Discharge to buffer	R1
4	continuous	Buffer tank control	R2
5	continuous	Operation of heat exchangers	HEATER

The times corresponding to the batch recipe are the following

Operation	Recipe 1
1	5
2	15
3	5

7. Prototypes implementation

The main objective of this section is to produce a Planning and Scheduling Package, which implements the CO interface specifications. Therefore, it is useful for testing and validation of the proposed specification. Besides, this work is also expected to be a useful guideline for programmers interested in adapting a PSP into a CO complaint one.

For such proposes, this section is organised as follows: first, an overall structure overview is presented. Then the original problem is described and the way in which the information is sent from the client to the server is summarised. Later, the wrapping work carried out is briefly explained. Once at this point, the reader can test the annexed programs. And finally, the corresponding conclusions and comments, which were used in order to provide feedback to the draft interface specifications, are presented. Besides, the appendix includes the instructions for running the prototype. The prototype, the source code and the corresponding documentation in html format are also available at the BSCW.

7.1 Methodology

Basically, the PSP Prototype contains the following functional items:

- A Client: This may be any piece of software able to set the problem and/or get the solution (i.e. a Coordination Control Package, a Supply Chain Package, etc). In this prototype, the client performs both functions, it sets the problem and obtains the results.

- A PSP: This is an application able to solve a Planning and Scheduling problem but is not Cape Open Compliant.

- A PSP Server: Which wraps the original PSP into a Cape Open Compliant in order to implement the specified interfaces.

The following figure illustrates their relationships:

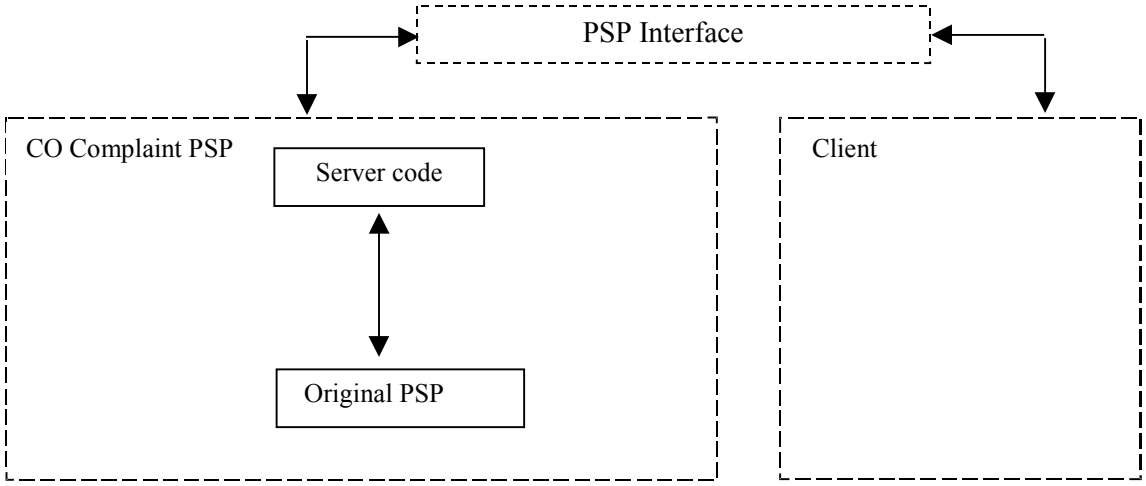


Figure 17 PSP prototype 1

The prototype developed is based on the Interface Specification Draft provided by UPC (Canton et al. 2000a). The specifications were written in two neutral Interface Definition Languages (CIDL and MIDL) corresponding to the Common Object Request Broker Architecture (CORBA IDL) and Distributed Component Object Model (DCOM) respectively. These IDL's characterise all the methods and input/output arguments that determine the entire PSP interface functionality. However, IDL's are purely declarative

languages, which have to be precompiled in a programming language in order to provide the implementation details.

In this prototype, Java Development Kit (JDK) version 1.3 has been adopted as the language to generate both the server and client codes. All communications between objects are managed by the CORBA middleware from the Object Management Group's (OMG), following the CAPE-OPEN guidelines (Orfali and Harkley 1998)

7.2 The case study

7.2.1 Context

The example used corresponds to the two stage multiproduct case with out-of-phase operation (mode 2) described the prior section.

7.2.2 Input data

Resources:

Resource Name	Associated Information
T1	Description: Tank 1
R1	Description: Reactor #1
R2	Description: Reactor #2

Recipes

Process	Stage	Operation		Unit	Time
P1	P1-S1	P1-S1-O1	Load	T1	5
		P1-S1-O2	Stirring	T1	5
		P1-S1-O3	Discharge	T1	5
	P1-S2	P1-S2-O4	Load Reactor	R1 or R2	15
		P1-S2-O5	Process	R1 or R2	20
		P1-S2-O6	Discharge	R1 or R2	5
		P1-S2-O7	Cleaning	R1 or R2	10
P2	P2-S1	P2-S1-O1	Load	T1	5
		P2-S1-O2	Stirring	T1	10
		P2-S1-O3	Discharge	T1	5
	P2-S2	P2-S2-O4	Load Reactor	R1 or R2	15
		P2-S2-O5	Process	R1 or R2	20
		P2-S2-O6	Discharge	R1 or R2	5
		P2-S2-O7	Cleaning	R1 or R2	10

Products Demand:

Demand
2 Batches of product A
2 Batches of product B

The objective function is to minimize the Makespan.

7.2.3 Results and output

The solution of this planning and scheduling problem involves the determination of:

- The sequence of tasks to be perfomed in each equipment unit
- The timing of these tasks
- The value of the resulting objective function.

7.2.4 Mathematical formulation

In order to solve the above problem we have used the following formulation:

A-Timing

Minimize the schedule makespan (MS) and a weighted contribution of event times and waiting times.

$$Z = C^1 \cdot MS + \sum_{m=1}^{M} C_m^2 \cdot TW_m + \sum_{n=1}^{N} C_n^3 \cdot T_n \qquad (1)$$

Subject to:

$$MS \geq T_n \qquad \text{(events)} \qquad (2)$$

$$T_n \geq T_n^{\min} \qquad \text{(events)} \qquad (3)$$

$$T_{NF_m} - T_{NI_m} - TOP_m = TW_m \qquad \text{(operations)} \qquad (4)$$

$$0 \leq TW_m \leq TW_m^{\max} \qquad \text{(operations)} \qquad (5)$$

$$T_{ND_k} \geq T_{NO_k} + \Delta T_k \qquad \text{(links)} \qquad (6)$$

B-Assignment and Sequences

Allocation binary variables are introduced. They equal to 1 to identify allocation between production stages and equipment units. Each stage should be assigned at least to one unit.

$$\sum_{u \in U_{bs}} Y_{bsu} = 1 \quad \forall b \in B, s \in S_b \qquad (7)$$

Sequence constraints complementary to equation 6 should also be introduced. Equation (6) was used in the original formulation not only for expressing the time constraints within a recipe, but also to represent the constraints between tasks in a given sequence of batches. Each stage has associated a start event (SN_{bs}) and an end event (EN_{bs}). The predecessor concept is introduced using a binary variable, ($P_{bb'}$) which relates a pair of batches (b and b'). If batch b is processed before batch b' then the predecessor variable equals to 1.

Once defined the predecessor variables, the sequence constraints are reformulated using a big M so that:

$$T_{SN_{b's}} \geq T_{EN_{bs}} - M.(1 - P_{bb'}) - M.(2 - Y_{bsu} - Y_{b'su}) \qquad \forall b, b' \in B, b' > b, s \in S_{bb'} \qquad (8)$$

$$T_{SN_{bs}} \geq T_{EN_{b's}} - M.P_{bb'} - M.(2 - Y_{bsu} - Y_{b'su}) \qquad \forall b, b' \in B, b' > b, s \in S_{bb'} \qquad (9)$$

Were:

DN_k	End event of link ***k***
DT_k	Minimum time between events of link ***k***
EN_{bs}	End event of task ***s*** of batch ***b***
FN_m	End event of operation ***m***
IN_m	Start event of operation ***m***
M	A big number >> Makespan
MS	Makespan
ON_k	Start event of link ***k***
$P_{bb'}$	Binary variable equal to 1 if the batch ***b*** is performed before batch ***b'***
S_b	Set of stages belonging to batch ***b***
$S_{bb'}$	Set of stages of batches ***b*** and ***b'*** that can have common units
SN_{bs}	Start event of task ***s*** of batch ***b***
T_n	Time associated to event ***n***
T_n^{min}	Minimum time associated to event ***n***
TOP_m	Operation time of operation ***m***
TW_m	Waiting time of operation ***m***
TW_m^{max}	Maximum waiting time of operation ***m***
U_{bs}	Set of units that can perform the stage ***s*** of batch ***b***
Y_{bsu}	Binary variable equal to 1 if the stage ***s*** of batch ***b*** is assigned to unit ***u***

7.3 The Server

7.3.1 The PSP

The PSP itself in this case, is a set of classes that basically covers the following functionality:

(i) To keep the input data.

(ii) To formulate the problem (in this case a mathematical programming problem).

(iii) To use a solver (in this case an LP) in order to solve the above-formulated problem.

(iv) To hold the solution data.

The simplified diagram in Figure 16 shows the corresponding classes and their relationships:

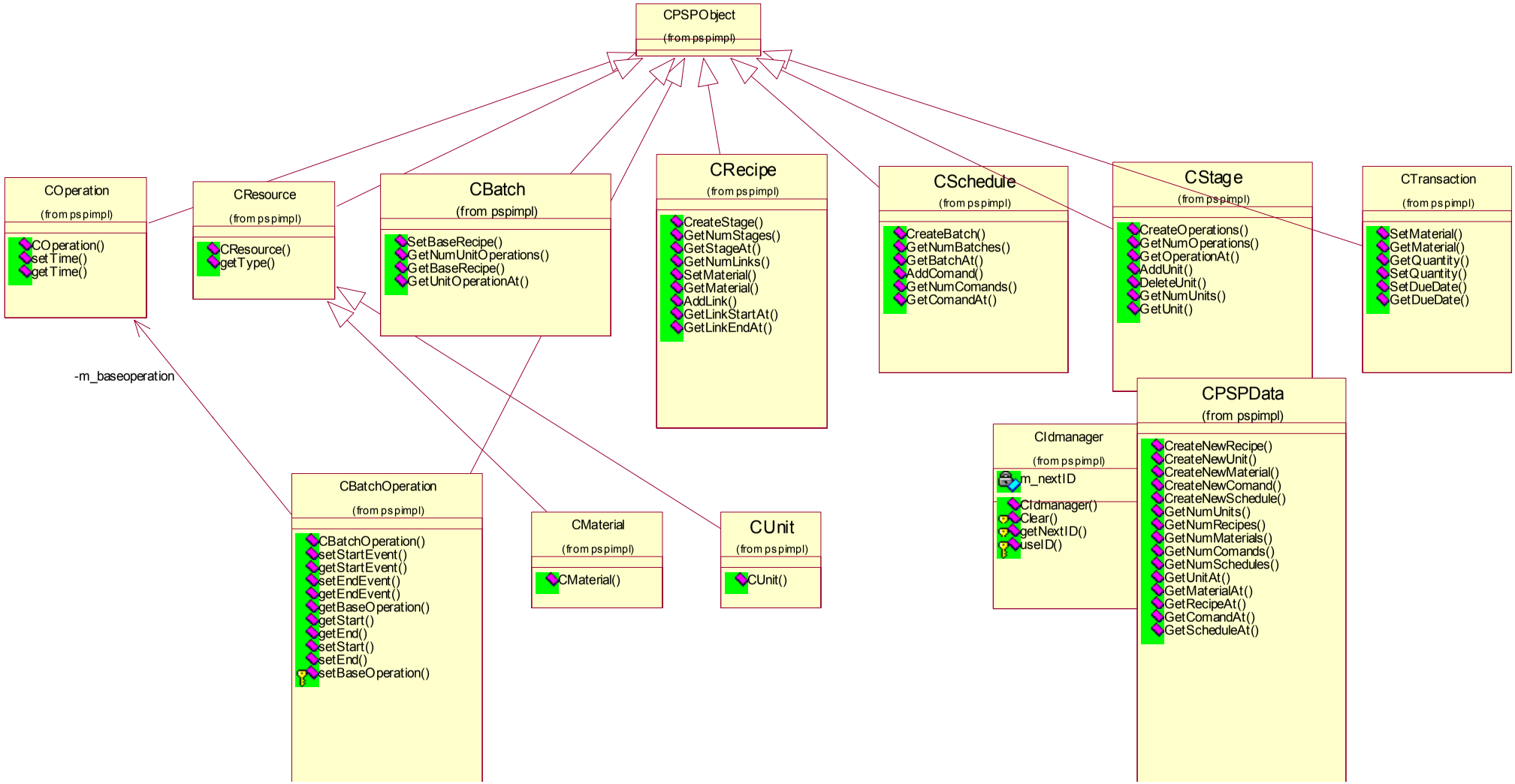


Figure 18 PSP prototype 2

For the sake of simplicity, the classes corresponding to the solver are not included here (for more information, see the html documentation).

7.3.2 The PSP Server

The PSP server is a piece of software that implements the specified interface. Therefore it is the responsible of the PSP wrapping and allows to the client a standard way of interaction with the PSP. Its classes clearly reflect the interface structure.

The next figure shows the implemented interface diagram.

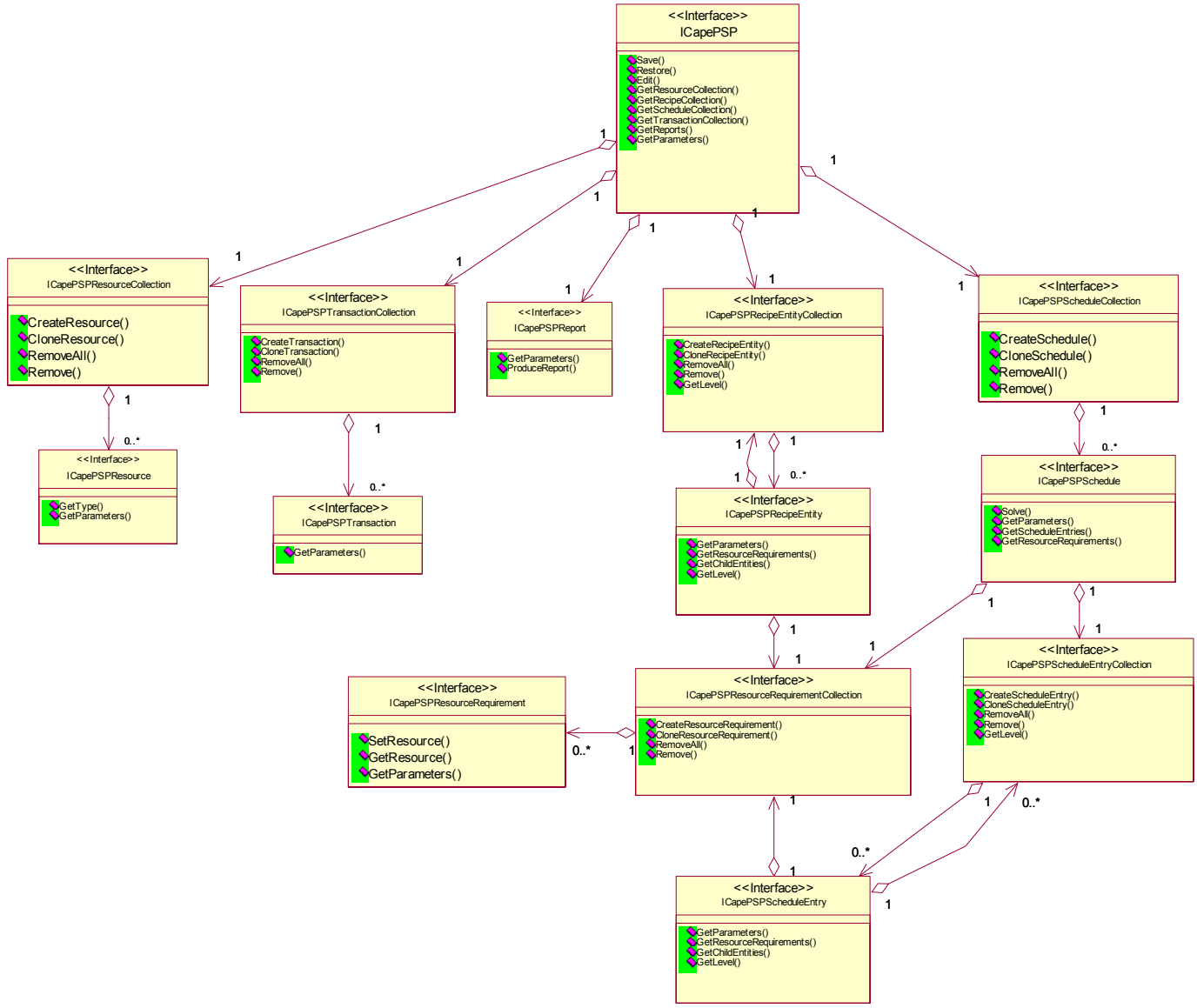


Figure 19 PSP prototype 3

For clarity reasons two classes are not included:

ICapeIdentification, from which inherit the following classes:

- ICapePSP
- ICapePSPResource

- ICapePSPRecipeEntity
- ICapePSPTransaction
- ICapePSPSchedule
- ICapePSPScheduleEntry

And, ICapePSPCollection, which have only tree methods (GetCount, Item and Index) and from wich inherite the following classes:

- ICapePSPResourceCollection
- ICapePSPRecipeEntityCollection
- ICapePSPTransactionCollection
- ICapePSPScheduleCollection
- ICapePSPScheduleEntryCollection.

7.4 The Client

As can be deduced, the client has no strong responsibilities. It just sends the problem input data to the PSP Server and, when the problem is solved, retrieves the results.

When more complex cases are studied (for example a reactive scheduling problem), the interaction between the client (or clients) and the server is significantly increased, and therefore a more complex client side structure may be required.

A simplified sequence diagram is presented in the following figure, which illustrates the interaction between the client, server and original PSP (besides the MILP solver):

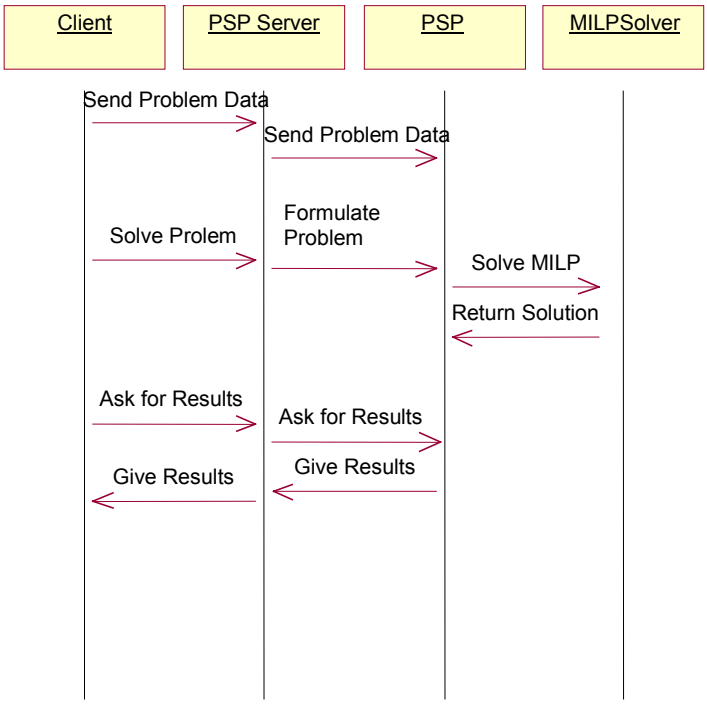


Figure 20 PSP prototype 4

7.5 Evaluation and Conclusion

The prototype development allows clarifying the relationships among modules and hence some changes were done to the original interface specifications. Even though these changes were not significant, they are next summarized:

1- The two following methods of the ICapePSPReport interface were removed: GetReportList and SelectReport, because their functionality can be delegated to the ICapeParameter interface.

2- Spelling errors in the original document, the interface descriptions and also the IDL files, were corrected.

3-The implementation of the ICapeParameter Interface is really messy and time consuming when working with arrays. Specific suggestions will be send to the M&T group in order to simplify its implementation.

4-The error handling was not implemented in the current prototype, but it will be incorporated in future versions. As the general framework is running, the additional coding needed is not difficult at all, although requires a significant time for coding due to the size of the prototype.

Hence, the development of this prototype has provided the feedback required to the Draft Interface Specification for producing the Final Interface Specifications.

8. Specific glossary terms

9. Bibliography

Avraam, M. P.; Shah N. and Pantelides, C.C. "Modelling and Optimisation of General Hybrid Systems in the Continuous Time Domain". *Comput. Chem. Engng*. Vol **22**. Suppl., pp. S221-S228, **1998**.

Barton, P.I. and Pantelides, C.C. ."Modelling of Combined Discrete/Continuous Processes". *AIChE Journal*, Vol. **40**., Nº 6 , **1994**.

Booch, G.; Rumbaugh, J.; and Jacobson, I. "The Unified Modeling Language User Guide", Addison-Wesley Object Technology Series, **1999.** ISBN 0 -201-57168-4

Brandl, D. "Make Batch Plants Bloom". *Chem. Eng.* (June **1998**).

Cantón, J.; Sequeira, S. E., Graells, M., Espuna, A., Puigjaner, L. WP5.3 Proposed Scenario (GCO-5.3-UPC-03), GCO Repository, **2000b**.

Cantón, J.; Sequeira, S. E.; Graells, M., Espuna, A., Puigjaner, L. PSP Interface Specification Draft (GCO-5.3-UPC-03), GCO Repository **2001a**.

CAPE-OPEN Methods and Tools Guidelines V 1.0

Chin, Kristine "Make Data Connection Across the Enterprise". *Chem. Eng.* (May **1999**).

Czulek, A. J. "An Experimental Simulator For Batch Chemical Processes", *Comput. Chem. Engng.* Vol. **12** No. 2/3. pp 253-259, **1988**.

Dimitriadis, V.D.; Saha, N. and Pantelides, C.C. "Modelling and Safety Verification of Discrete/Continuous Processing Systems" . *AIChE Journal,* Vol. **43** pp. 1041-1059, Nº4, **1997.**

Djavdan, P. "Design of an On-line Scheduling Strategy for a Combined Batch/Continuous Plant using Simulation". *Comput. Chem. Engng,* Vol. **17** Nº 5/6, pp. 561-567, **1993**.

International Soc. for Measurement and Control. *ISAdS88.01*-**1995** "Batch Control, Part 1: Models and Terminology".

International Soc. for Measurement and Control. *ISAdS95.01*-**1999**. Draft 14 "Enterprise-Control System Integration, Part 1: Models and Terminology".

Muller, Pierre-Alain . "Modelado de Objetos con UML". Ediciones Gestion 2000 S.A. Barcelona, (**1997**).

Nowwicki, P., Brandl, D. "Integrate Bussines Systems with Batch Manufatcuring". *Chem. Eng*. (March **2000**).

Orfali, Robert and Dan Harkley "Client/Server programming with Java and CORBA" John Wiley & sons, Inc., **1998**.

Rational Rose 2000. Using Rose Revision 6.5 (September **1999**).

Reklaitis, G. V. "Computer-Aided Design and Operation of Batch Processes" *Chem. Eng. Educ.* Vol. **29**, pp. 76-85. **1995**.

Reklaitis, G.V. "Overview of Scheduling and Planning of Batch Process Operations". *NATO ASI Series F*, Vol **143**, pp. 661-705, **1996**.

Rickard, J.G.; Machieto, S. and Shah, N. "Integrated Decision Support in Flexible Multipurpose Plants", *Comput. Chem. Engng. Suppl.*, pp. S547-S550, **1999**.

Rippin, D. W. T. “Batch Process Systems Engineering: A Retrospective And Prospective Review”. *Comput. Chem. Engng.* Vol. **17.** Suppl. pp S1-S536, **1993**.

Shah, N. “Single-and multisite planning and scheduling: Current Status and Future Challenges”, *FOCAPO ‘ 98*, **1998**.

Silver, E.; Pyke, D.; Peterson, R. “Inventory Management and Production Planning and Scheduling”, John Wiley & Sons **1998**.

World Batch Forum web site: http://www.wbf.org

10. Appendices

10.1 Annex A: Building and running the Client and Server sides

The source files of the prototypes are available for the testing of the PSP Prototype. Two main folders are supplied: PSPServer for the server sources, PSPClient for the client sources. The remaining folders are the skeleton and stubs generated with the IDLTOJAVA tool, and other auxiliary classes. The Java Development Kit (JDK) Version 1.3 should be installed in order to compile and run the prototype. An appropriate JDK can be downloaded from http://java.sun.com according to the software and hardware available for running the prototype. The source files provided has been tested under Windows NT, Solaris and Linux.

The detailed description of the different directories contends is the following:

Folder	Description	Used in server	Used in client
Parameter	Idltojava- generated Parameter interfaces stubs and skeletons	Yes	Yes
Common	Idltojava- generated Common interfaces stubs and skeletons	Yes	Yes
Base	Idltojava- generated Base interfaces stubs and skeletons	Yes	Yes
Identification	Idltojava- generated Identification interfaces stubs and skeletons	Yes	Yes
Psp	Idltojava- generated PSP interfaces stubs and skeletons	Yes	Yes
Lp	Lp-solve classes	Yes	No
Pspserver	Wraper classes for the PSP interfaces	Yes	No
Pspprototype/pspsolver	Implementation of a lp based PSP solver.	Yes	No
Pspprototype/pspimpl	Implementation of the PSP classes used by the psp solver	Yes	No
Gantt	Small java based gantt-chart implementation	No	Yes
Pspclient	PSP client which uses the PSP server	No	Yes

Building the Java files

All the Java files must be compiled before running the prototype. This can be done by using the following commands from the prototype root directory:

```
javac -classpath . lp\*.java
javac -classpath . Parameter\*.java
javac -classpath . Common\*.java
javac -classpath . Base\*.java
javac -classpath . Identification\*.java
javac -classpath . Psp\*.java
javac -classpath . pspserver\*.java
javac -classpath . pspprototype\pspsolver\*.java
javac -classpath . pspprototype\pspimpl\*.java
javac -classpath . gantt\*.java
javac -classpath . pspclient\*.java
```

Those commands are in a windows format. In some Unix-based operating systems the backslash "\" should be replaced for the normal slash "/".

These tasks can be automatically done using the files: **compileall.bat** (for Windows) and **compileall.sh** (for Unix/Linux).

Running the prototype

There are three steps required to run the prototype:

(i) Initialising a Name Server

(ii) Start the PSP Server

(iii) Start the PSP Client

The Name Server and the PSP Server should be running in the same machine. The Client can be run either in the same machine or in a different one.

Running the client and the server in the same computer

The orders required are the following:

1- Start the name server

```
tnameserv –ORBInitialPort 1050
```

This order starts the name server provided with the JDK1.3 listening in the port number 1050. The port number can be changed to any number greater than 1024.

2- Start the PSPServer

```
java -classpath . PSPServer.CapePSPServer -ORBInitialPort 1050
```

This order starts the PSPServer to be run in port 1050. The port number should be the same as the one chosen for the name server. Notice that both, the PSP server and nameserver will run until being interrupted or killed.

3- Start the PSPClient

```
java -classpath . PSPClient.PSPClient -ORBInitialPort 1050
```

This order starts the PSPClient using the PSP server located in the same machine. Notice that the same port number should be used to initialise the Name Server, the PSPServer and the PSPClient.

In order to perform automatically these tasks, use the files **runnameserv.* , runserver.*** and **runclient.*** (**"*"** means "bat" for Windows and it means "sh" for Unix/ Linux).

Running the client and the server in different computers

In this case the appropriate JDK should be installed in both computers. The operating system of both computers may be different.

To run the prototype on two computers, first start the Name Server and the PSPServer in the same computer with the same procedure described before.

Then start the PSPClient in the second computer:

```
java -classpath . PSPClient.PSPClient –ORBInitialHost host_IP -ORBInitialPort 1050
```

The host_IP should be the IP name or number where the server is running (i.e. myhost.mydomain.com or 111.222.111.222).