APPENDIX B

BUSINESS PROJECT MANAGEMENT (BPROJM) DOMAIN MODEL UML SPECIFICATIONS

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

This is a documentation of the complete Business Project Management (BProjM) domain model which supplements the main thesis. The focus of the main thesis is to describe the domain modelling approach and thus, only extracts of the developed domain model are presented with the aim to illustrate the modelling steps. The best way to review the BProjM domain model would be to examine its original Objecteering 6.1 version, where the details of each of its components could be explored using the system functionalities provided by the Objecteering software. Given that the Objecteering software may not be accessible to all the readers, this document serves as the main source of reference where the developed domain model can still be studied without the support of the tool.

1.1 Purpose of this document

The primary purpose of this document is to provide full details of the developed BProjM domain model for reference by all interested parties. This includes using it as (1) the main reference for those who wish to understand the coverage of this domain model before adding on new research findings; as well as (2) a domain reference for those who wish to acquire a better understanding of business project management as a subject matter.

1.2 Organization of this document

This document is organized in to 4 chapters:

1) Introduction – This chapter provides background information about this documentation;

2) Underlying design principles and considerations – This chapter summarizes the key principles and considerations that guided the development of the model. This is so that the readers could comprehend the model from the developer's point of view and grasp the essence of the model very quickly;

3) Structural and behavioural specifications – This is the core chapter of this documentation where it presents the UML specifications of the domain model;

4) Usage scenarios – This chapter presents two scenarios that depict the applications of the UML specifications.

It is recommended that the readers go through Chapter 1 and 2 before reviewing the content of Chapter 3 and 4.

CHAPTER 2
DESIGN PRINCIPLES & CONSIDERATIONS

This chapter describes the design principles & considerations behind the development of each type of UML diagrams namely (1) *Package Diagram* and *Class Diagram* for the structural specifications; (2) *State Machine Diagram* and *Communication Diagram* for the behavioural specifications; and (3) *Sequence Diagram* for the *Use Case*. It is important to note that the focus of the research is to demonstrate how business project management knowledge can be represented in the form of a domain model using UML, based on the devised domain modelling approach. It is not intended for the research to produce a show-case UML specification and thus, the application of UML features may not be optimal in some instances.

2.1 Package Diagram

The first consideration of the *Package Diagram* is to design the *Package*s in such a way that level 1 *Packages* represent the key components of the domain model; while level 2 *Packages* denote logical grouping of related concepts within the level 1 *Packages*. Based on this principle, the level 1 /essential *Packages* of the domain model are:

1) BMM
2) BMM PlaceHolder (Assets & Liabilities)
3) BPDM
4) OSM
5) BProjM
6) Usage Scenario

The first 4 items are the OMG's standards for describing business enterprise at the conceptual level; item 5 "BProjM" is the main component of the domain model which specifies business project management structures and dynamics; and item 6 "Usage

Scenario" details how the domain model is used in (1) defining requirement specifications of the desired PMIS; and (2) assessing if an existing business project management practice has all the essential components in place.

2.2 Class Diagram

The *Classes* in the OMG's *Packages* contain only skeletal definitions since the OMG's standards are included in the domain model for the purpose of completeness, so that interfaces between business project management and its parent organization can be illustrated. The focal point of the modelling is the "BProjM" *Package* and the definition of its *Classes* is guided by the following principles and considerations:

- Each *Class* is defined with 2 mandatory *Attributes* namely "Name" and "Description" to ensure that their instantiations would be properly identified and described.

- All *Classes* defined with *State Machines* must have the *Attribute* "*Status*" to reflect the state of the *Object*; and an *Operation* "*UpdateXXXStatus()*" that effects the change in status (where "XXX" is the *Class* name).

- Validation rules for *Attribute*'s data type are defined as *Constraints* for the *Attribute.*

- Processing logics that must be incorporated by an *Operation* are defined as *Constraints* for the Operations.

- Non-trivial inter-*Class* relationships which could not be sufficiently represented by *Associations* are defined as *Constraints* for the *Class*.

- *Classes* of similar nature are defined using a same pattern. For example, each of the 4 *Classes* in the "BProjM Function" (which represent the 4 core business project management competencies) are defined with (1) *Operation* "*DefineYYY()"*to represent the processes in the "planning" process group; (2)

Operation "*UpdateYYY()"* to represent the processes of updating the deliverables created by the "planning" process group; and (3) *Operation* "*ManageYYY()"* to represent the processes in the "executing" and "controlling" process groups (where "YYY" refers to the respective PMBOK knowledge area).

- *Operations* of similar nature are named in a similar fashion to reflect their relevancies with each other. For example, the definition of TrackOutput() and TrackProgress() in the *Class* INTEGRATED MANAGEMENT; and TrackPerformance() in the *Class* INTEGRATED PROGRAMME MANAGEMENT. All these *Operations* are prefixed by the term "Track" to denote their roles in tracking the different components of project success.

- "Roles" of the *Classes* in an *Association* is defined only if the information adds value. This is because most of the *Classes* are playing the role as their names imply. An example of a special case where the roles need to be explicitly defined would be the relationship which denotes that tasks are organized into subtasks. In this case, the *Association "IsOrganizedInto"* must have the role "Task" defined for one end and "Subtask" defined for the other.

- *Communication Diagrams* (for the behavioural specifications) and *Sequence Diagrams (*of the *Use Cases)* are developed using generic *Objects* which are generic instances created for every *Class* in the model.

2.3 State Machine Diagram

Each transition of state is consistently defined with (1) "Received Event" i.e. the event that triggers the transition; (2) "Guard Condition" i.e. the condition that must be met before the *Class* can assume the current state; (3) "Expression of the Action" i.e. the *Operation* to be carried out after taking on the current state.

For the more sophisticated state, "sub-states" are introduced so that transition of states within states could also be described. Please refer to the *State Machine Diagram* of INITIATIVE for illustration.

2.4 Communication Diagram

The development of *Communication Diagram* is guided by the following:

- Unless restricted by the need to further condense the layout, *Objects* in the *Communication Diagram* are arranged from top to bottom, from left to right.

- *Communication Message* that defines the interaction between two *Objects* must be a predefined *Operation* in the receiving *Class*. This is to ensure integrity and consistency among different UML components in the domain model.

- To reflect interactions among *Objects* which take place one after the other, the *Communication Messages* are labelled serially to reflect the sequential relationship.

- To reflect relationships among *Communication Messages* at different levels of interactions, related messages at the lower level are assigned with a corresponding "sub-serial number". For example, if two messages at the highest level are labelled with 1 and 2 respectively; lower level interactions related to message 1 would have its messages labelled 1.1, 1.2, 1.3 etc., whilst those related to message 2 would be labelled 2.1, 2.2, 2.3 and so forth.

- To reflect exchange of *Communication Messages* which could take place only upon completion of some others, "*Dependency*" links are defined among them.

2.5 Sequence Diagram

The development of *Sequence Diagram* for the *Use Case* is guided by the following:

- Given the emphasis to demonstrate how *Communication Diagram* of the domain model can be used in the specification of domain-imposed requirements, only the domain-imposed requirements are defined as a sequence of actions and the

user-defined requirements are summarized in the form of a “Note”. This is to ensure that the developed *Sequence Diagram* remains legible.

- Most of the interactions are defined as “synchronize messages”, so that the *Operations* predefined in the receiving *Class* could be selected directly from the dialog box. By doing so, the internal consistency and integrity of the specifications could be maintained at all times.

CHAPTER 3
STRUCTURAL & BEHAVIOURAL SPECIFICATIONS

This chapter first presents a series of *Class Diagrams* to provide an overview of the domain model. The purpose is to set the scene and to facilitate better comprehension of the detailed UML specifications in the subsequent sections which are organized by *Packages*. Within each section dedicated for the *Package*, details (namely *Description*, *Operations, Associations, Attributes, State Machines)* of its *Classes* are provided.

In summary, the BProjM domain model contains the following level 1 components:

Packages :

- *BProjM*
- *BMM*
- *BMM PlaceHolder (Assets & Liabilities)*
- *BPDM*
- *OSM*

Classes :

- *Calendar*
- *Financial Mgmt*
- *Customer Services*
- *Employee*
- *Event*
- *HR Mgmt*
- *Key Performance Indicators*
- *Machine*
- *Marketing*
- *Procurement Mgmt*
- *Quality Mgmt*
- *Risk Mgmt*
- *Sales*
- *Skill*

The focal point of this research is the "BProjM" *Package* and thus its sub-*Package*s and *Classes* shall be presented first. This is followed by the skeletal definition of OMG *Packages* (i.e. BMM, BMM PlaceHolders, BPDM and OSM); and the list of other *Classes* defined at level 1. These level 1 *Classes* are not grouped into *Packages*

because they are recommended either for future work and/or for inclusion by the respective OMG standards. Please refer to Chapter 7.5, Findings and Discussions in the main thesis for details.

The final section of this chapter is the presentation of the *Communication Diagrams* which capture the behaviour of the model according to the 4 stages of the project life cycle, each of which corresponds to a state of the "ruling" *Class* INITIATIVE.

Note that a big portion of this chapter namely the *Package* and *Class* definitions are automatically generated by Objecteering's documentation tool, whilst the definition of *Constraints, Communication Diagrams* and *Sequence Diagrams* are extracted and compiled manually.

3.1 Class Diagrams

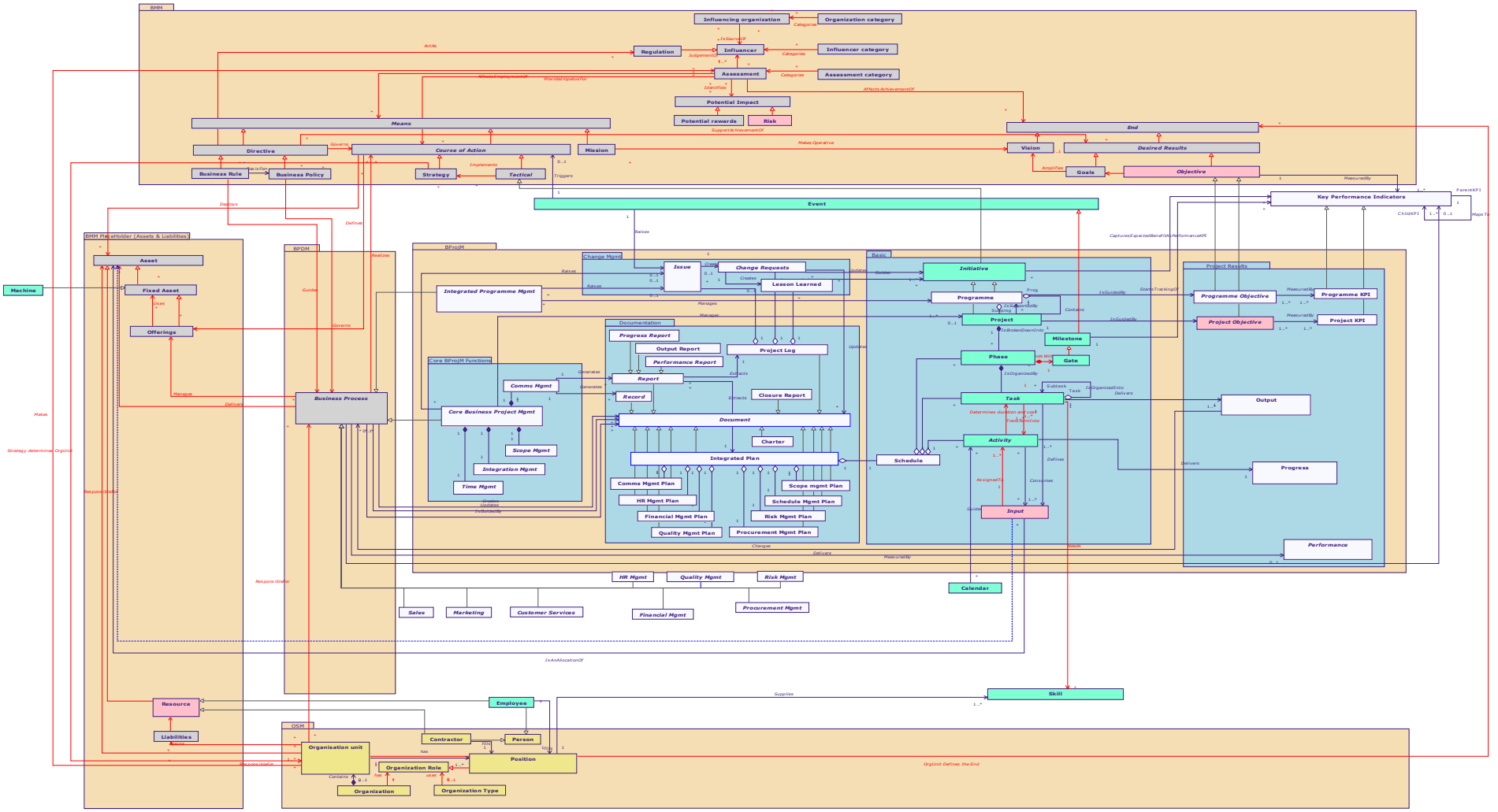
This section presents the following *Class Diagrams* to provide an overview as well as the different perspectives of looking at the BProjM domain model.

1) *Class Diagram (Complete)* – overview of the complete BProjM domain model reflecting all *Classes*, *Packages* and *Associations*.

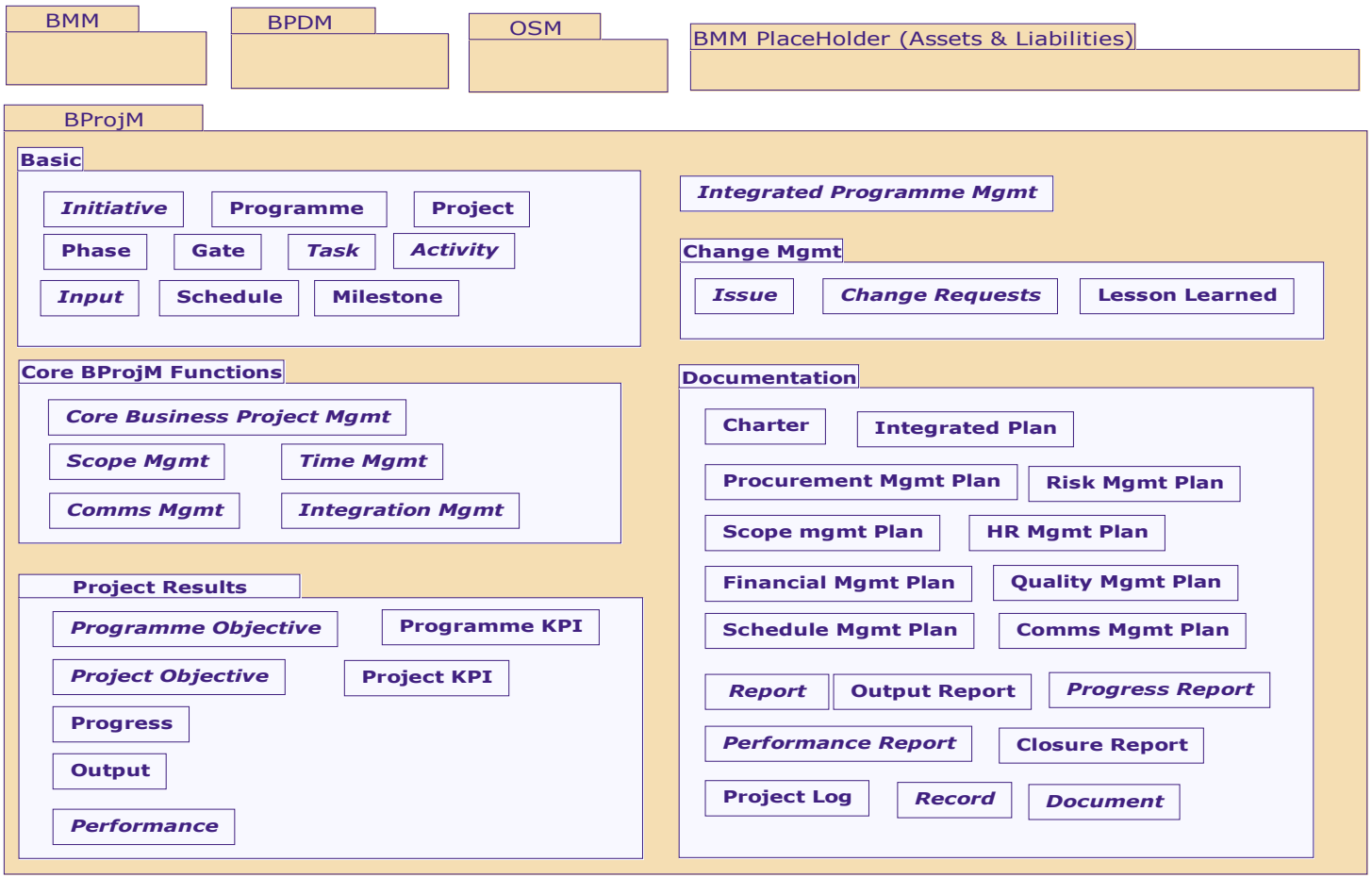
2) *Class Diagram (Packages)* – extract of the domain model that shows only the level 1 *Packages* and *Classes* in the "BProjM" *Package*.

3) *Class Diagram (BProjM Overview)* – extract of the domain model that shows only the details of the main scope of work i.e. "BProjM" *Package*.

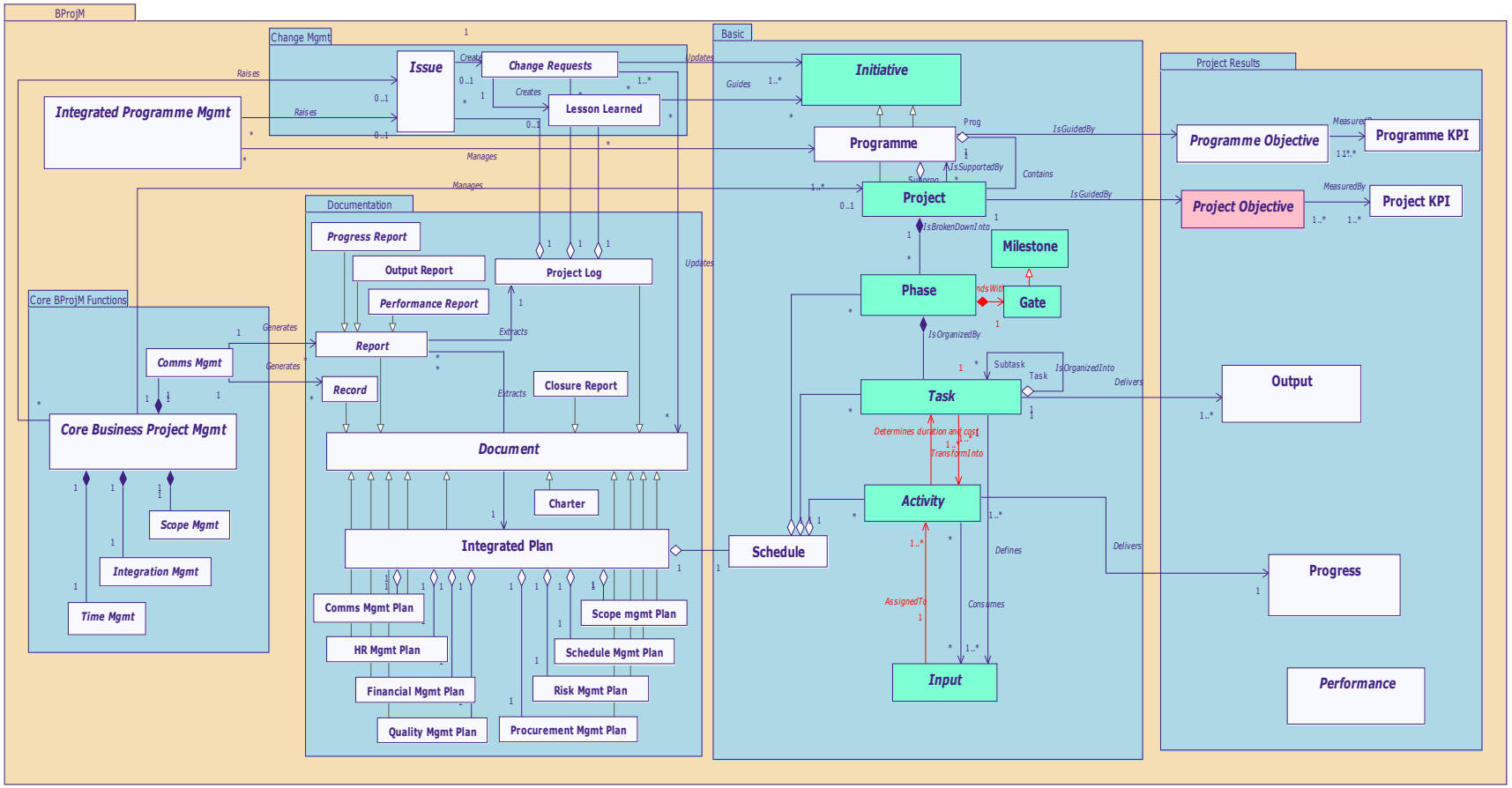
4) *Class Diagram (BProjM in Context)* – extract of the domain model that shows only the *Classes* in the level 1 *Packages* which are defined with cross-*Package* inter-relationships.



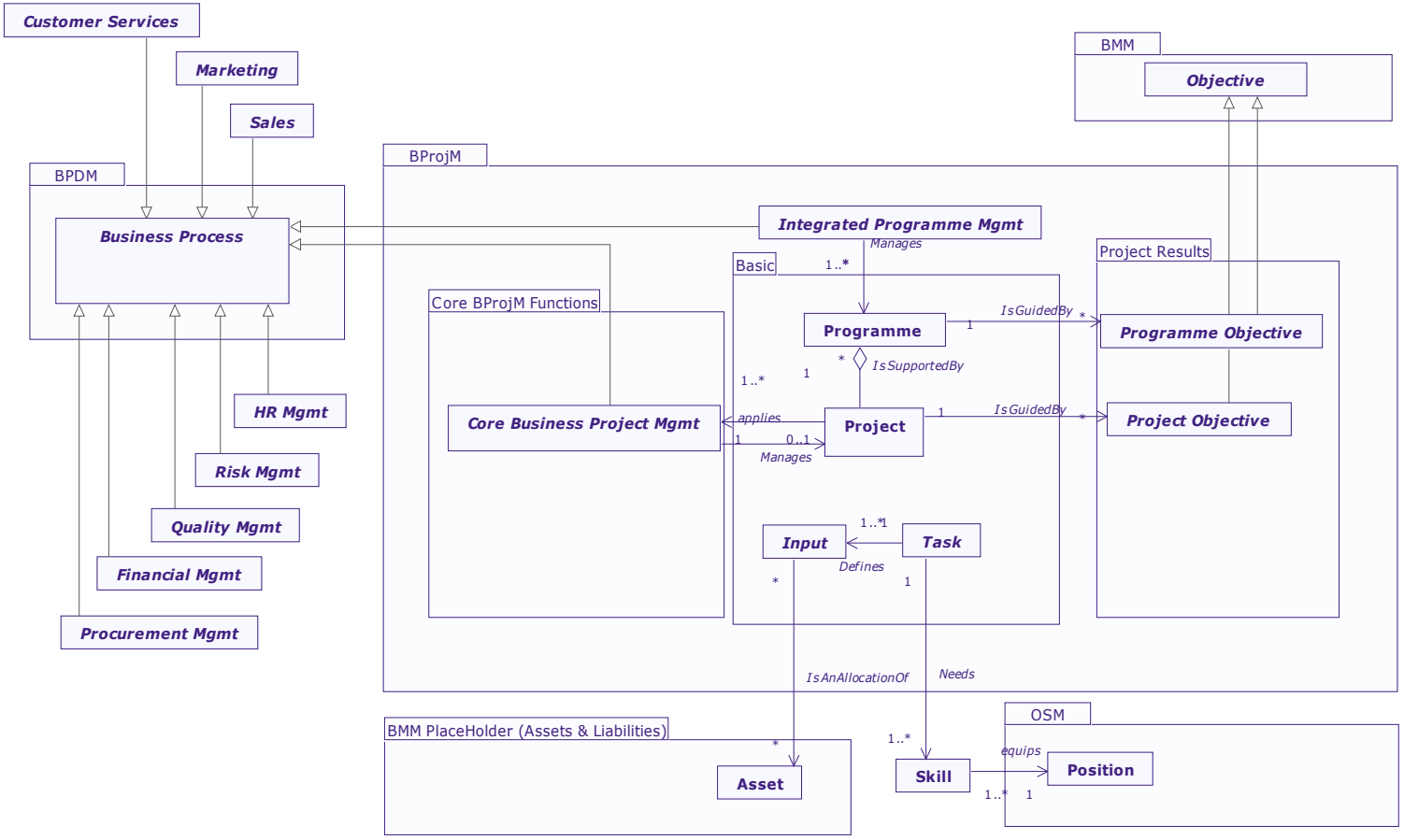
Class Diagram (Complete)



Class Diagram (Packages)



Class Diagram (BProjM Overview)



Class Diagram (BProjM in Context)

3.2 "BProjM" package

Description :

This Package contains both structural and behavioural specifications of the business project management.

Packages :

- *Core BProjM Functions*
- *Project Results*
- *Basic*
- *Change Mgmt*
- *Documentation*

Classes :

- *Integrated Programme Mgmt*

3.2.1 "Integrated Programme Mgmt" class

Description :

An organizational practice that includes processes to enable on-going initiation, execution and completion of top-down driven business change programme which may consist of many sub-programmes defined with specific organizational objectives.

Parent classes :

- *Business Process*

Operations :

- *public TrackPerformance ( )*
- *public ManageDeptSupport ( )*

Associations :

- *Manages : (1-*) as undefined to Programme (0-*) as undefined*
- *Raises : (0-1) as undefined to Issue (0-*) as undefined*

Attributes :

- *public char[120] Name*
- *public real[Date] ReviewDate*
- *public char[120] Reviewer*
- *public char[600] ReviewFindings*
- *public char[600] Description*

3.2.1.1 "TrackPerformance" operation

```
public TrackPerformance ()
```

This refers to processes involved in ensuring that project is delivering its expected contributions at the organizational level.

3.2.1.2 "ManageDeptSupport" operation

`public ManageDeptSupport ()`
This refers to the processes involved in securing and managing the support from functional departments on the 5 non-core business project management competency areas namely risk management, quality management, HR management, financial management, procurement management.

3.3 "Core BProjM Functions" package

Description :
This Package contains specifications for the core business project management functions.

Classes :

- *Core Business Project Mgmt*
- *Scope Mgmt*
- *Time Mgmt*
- *Comms Mgmt*
- *Integration Mgmt*

3.3.1 "Core Business Project Mgmt" class

Description :
This refers to core business project management competencies which are essential for achieving delivery success at the project level.

Parent classes :

- *Business Process*

Associations :

- *Manages : (0-1) as undefined to Project (1-1) as undefined*
- *Raises : (0-1) as undefined to Issue (0-*) as undefined*

Attributes :

- *public char[120] Name*
- *public real[Date] ReviewDate*
- *public char[600] ReviewFindings*
- *public char[120] Reviewer*
- *public char[600] Description*

3.3.2 "Scope Mgmt" class

Description :

A subset of project management that includes processes to ensure that the project includes all the work required and only the work required, to complete the project successfully.

Operations :

- *public ManageScope ( )*
- *public DefineScope ( )*
- *public UpdateScope ( )*

3.3.2.1 "ManageScope" operation

`public ManageScope ()`
This corresponds to the project scope management processes classified under the "Executing" and "Controlling" Process Groups in the PMBOK Guide.

3.3.2.2 "DefineScope" operation

`public DefineScope ()`
This corresponds to the project scope management processes classified under the "Initiating" Process Group in the PMBOK Guide.

3.3.2.3 "UpdateScope" operation

`public UpdateScope ()`
This corresponds to the project scope management processes classified under the "Planning" Process Group in the PMBOK Guide.

3.3.3 "Time Mgmt" class

Description :

A subset of project management that includes processes to ensure timely completion of project.

Operations :

- *public ManageTimeline ( )*
- *public DefineTimeline ( )*
  public UpdateTimeline ( )

3.3.3.1 "ManageTimeline" operation

`public ManageTimeline ()`
This corresponds to the project time management processes classified under the "Controlling" Process Groups in the PMBOK Guide.

3.3.3.2 "DefineTimeline" operation

`public DefineTimeline ()`
This corresponds to the project time management processes classified under the "Planning" Process Group in the PMBOK Guide.

3.3.3.3 "UpdateTimeline" operation

`public UpdateTimeline ()`
This corresponds to the project time management processes classified under the "Planning" Process Group in the PMBOK Guide.

3.3.4 "Comms Mgmt" class

Description :

A subset of project management that includes processes to ensure timely and appropriate generation, collection, dissemination, storage and ultimate disposition of project information.

Operations :

- *public ManageComm ( )*
- *public DefineCommReq ( )*
- *public UpdateCommReq ( )*
- *public GenerateReport ( )*

Associations :

- *Generates : (0-*) as undefined to Report (1-1) as undefined*
- *Generates : (0-*) as undefined to Record (1-1) as undefined*

3.3.4.1 "ManageComm" operation

`public ManageComm ()`
This corresponds to the project communications management processes classified under the "Execution" and "Controlling" Process Groups in the PMBOK Guide.

3.3.4.2 "DefineCommReq" operation

`public DefineCommReq ()`
This corresponds to the project communications management processes classified under the "Planning" Process Group in the PMBOK Guide.

3.3.4.3 "UpdateCommReq" operation

`public UpdateCommReq ()`
This corresponds to the project communications management processes classified under the "Planning" Process Group in the PMBOK Guide.

3.3.4.4 "GenerateReport" operation

`public GenerateReport ()`
This refers to the specific processes involved in generating the management reports.

3.3.5 "Integration Mgmt" class

Description :

A subset of project management that includes processes to ensure that the various elements of the project are properly coordinated.

Operations :

- *public TrackOutput ( )*
- *public DefineIntegrationReq ( )*
- *public UpdateIntegrationReq ( )*
- *public TrackProgress ( )*
- *public ManageIntegration ( )*

3.3.5.1 "TrackOutput" operation

`public TrackOutput ()`
This refers to the processes involved in ensuring that the project is delivering the output according to specifications.

Constraints : {Validation}

Track Output can only be activated if the Integration Plan has been approved

3.3.5.2 "DefineIntegrationReq" operation

`public DefineIntegrationReq ()`
This corresponds to the project integration management processes classified under the "Planning" Process Group in the PMBOK Guide.

3.3.5.3 "UpdateIntegrationReq" operation

`public UpdateIntegrationReq ()`
This corresponds to the project integration management processes classified under the "Planning" Process Group in the PMBOK Guide.

3.3.5.4 "TrackProgress" operation

`public TrackProgress ()`
This refers to the processes involved in ensuring that the project is making progress in terms of completing the planned activities as specified in the project schedule.

Constraints : {Validation}
Track Progress can only be activated if the Integration Plan has been approved.

3.3.5.5 "ManageIntegration" operation

`public ManageIntegration ()`
This corresponds to the project integration management processes classified under the "Execution" and "Controlling" Process Groups in the PMBOK Guide.

3.4 "Project Results" package

Description :
This Package contains structural and behavioural specification for concepts related to project results.

Classes :

- *Programme Objective*
- *Project Objective*
- *Performance*
- *Output*
- *Programme KPI*
- *Project KPI*
- *Progress*

3.4.1 "Programme Objective" class

Description :
Purposes or goals of the programme, the aims to be achieved and the desired end results.

Parent classes :

- *Objective*

Associations :

- *MeasuredBy : (1-*) as undefined to Programme KPI (1-*) as undefined*

3.4.2 "Project Objective" class

Description :
Purposes or goals of the project, the aims to be achieved and the desired end results.

Parent classes :

- *Objective*

Associations :

- *MeasuredBy : (1-*) as undefined to Project KPI (1-*) as undefined*

3.4.3 "Performance" class

Description :

The contribution of project at the programme /organizational level.

Parent classes :

- *Programme Objective*

Operations :

- *public DefinePerformance ( )*
- *public UpdatePerformance ( )*
- *public UpdatePerformanceStatus ( )*
- *public GetPerformanceStatus ( )*
- *public ReviewPerformance ( )*

Associations :

- *BasisFor : (0-*) as undefined to Core Business Project Mgmt (0-*) as undefined*

Attributes :

- *public char[600] Description*
- *public char[120] Reviewer*
- *public char[60] Status*
- *public real[Date] ReviewDate*
- *public char[120] Name*
- *public char[600] ReviewFindings*
- *public real[Date] RealizationDateActual*
- *public real RealizationDatePlanned*

State machines :

- *PerformanceStateMachine*

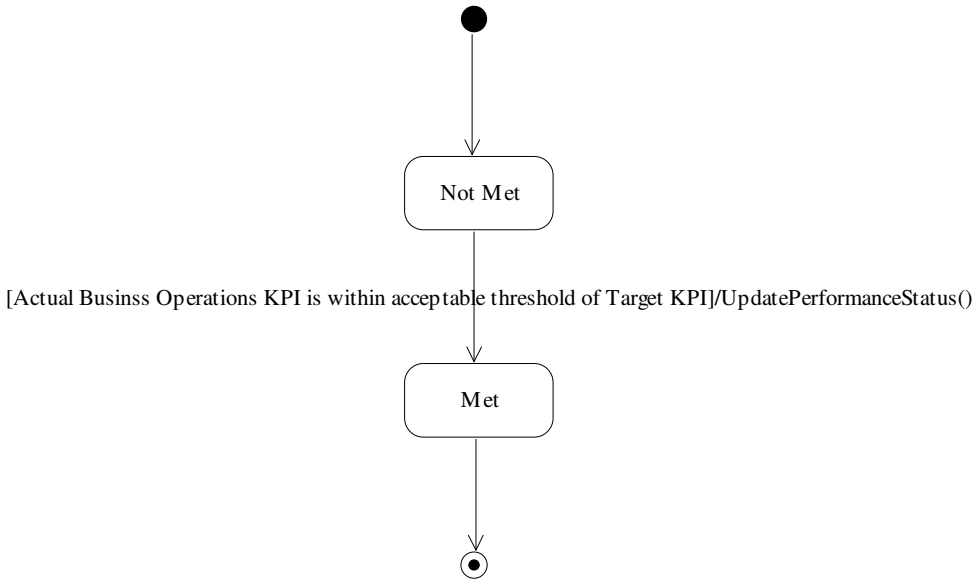
3.4.3.1 "PerformanceStateMachine" state machine

States :

- *Not Met*
- *Met*

Transitions :

- *Initial to Not Met*
- *Not Met to Met*
- *Met to Final*



Performance State Machine Diagram

3.4.3.2 "DefinePerformance" operation

`public DefinePerformance ()`
This refers to defining the expected project benefits and its contributions at the organizational level.

3.4.3.3 "UpdatePerformance" operation

`public UpdatePerformance ()`
This refers to updating the predefined project benefits or contributions at the organizational level especially the actual realization date.

3.4.3.4 "UpdatePerformanceStatus" operation

`public UpdatePerformanceStatus ()`
This refers to the updating performance status based on whether the corresponding performance KPIs have been met.

3.4.3.5 "GetPerformanceStatus" operation

`public GetPerformanceStatus ()`
This refers to retrieval of performance status typically to serve the generation of management reports.

3.4.3.6 "ReviewPerformance" operation

`public ReviewPerformance ()`
This refers to act of reviewing project performance and the capturing of the review findings at the end.

3.4.4 "Output" class

Description :
Actual outcome of an Initiative. Can be a physical product, a service or a document.

Operations :

- *public DefineOutput ( )*
- *public UpdateOutput ( )*
- *public ReviewOutput ( )*
- *public UpdateOutputStatus ( )*
- *public GetOutputStatus ( )*

Associations :

- *Changes : (0-1) as undefined to Business Process (1-1) as undefined*

Attributes :

- *public char[120] Name*
- *public char[120] Reviewer*
- *public char[600] Description*
- *public real[Date] ReviewDate*
- *public char[600] ReviewFindings*
- *public char[60] Status*

State machines :

- *OutputStateMachine*

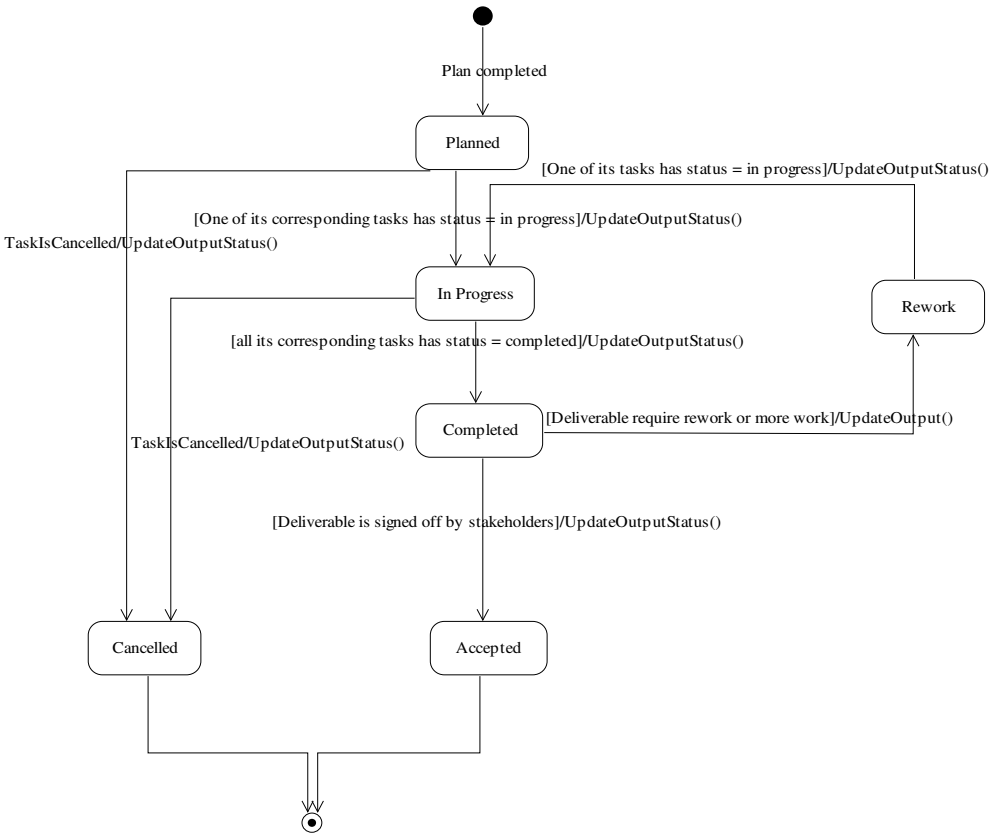
3.4.4.1 "OutputStateMachine" state machine

States :

- *Planned*
- *Completed*
- *In Progress*
- *Cancelled*
- *Accepted*
- *Rework*

Transitions :

- *Initial to Planned*
- *Planned to In Progress*
- *Planned to Branch*
- *Planned to Cancelled*
- *In Progress to Completed*
- *Completed to Rework*
- *Completed to Accepted*
- *Rework to In Progress*
- *In Progress to Branch*
- *In Progress to Cancelled*
- *Branch to Cancelled*
- *Cancelled to Final*
- *Accepted to Final*



Output State Machine Diagram

3.4.4.2 "DefineOutput" operation

`public DefineOutput ()`
This refers to defining the direct output or deliverables of the project.

3.4.4.3 "UpdateOutput" operation

`public UpdateOutput ()`
This refers to updating the details of a predefined output.

3.4.4.4 "ReviewOutput" operation

`public ReviewOutput ()`
This refers to the act of reviewing the status and quality of the output, as well as the capturing of review findings at the end.

3.4.4.5 "UpdateOutputStatus" operation

`public UpdateOutputStatus ()`
This refers to the update of output status depending on the conditions stated in the state machine diagram.

3.4.4.6 "GetOutputStatus" operation

`public GetOutputStatus ()`
This refers to the retrieval of output status typically to serve the need of generating management reports.

3.4.5 "Programme KPI" class

Description :
Key part of the measurable programme objectives.

Parent classes :

- *Key Performance Indicators*

3.4.6 "Project KPI" class

Description :
Key part of the measurable project objectives.

Parent classes :

- *Key Performance Indicators*

3.4.7 "Progress" class

Description :

The completion of planned activities.

Operations :

- *public UpdateProgressStatus ( )*
- *public GetProgressStatus ( )*
- *public ReviewProgress ( )*

Attributes :

- *public char[600] Name*
- *public char[60] Status*
- *public real[Date] ReviewDate*
- *public char[120] Reviewer*
- *public char[600] ReviewFindings*
- *public char[600] Description*

State machines :

- *ProgressStateMachine*

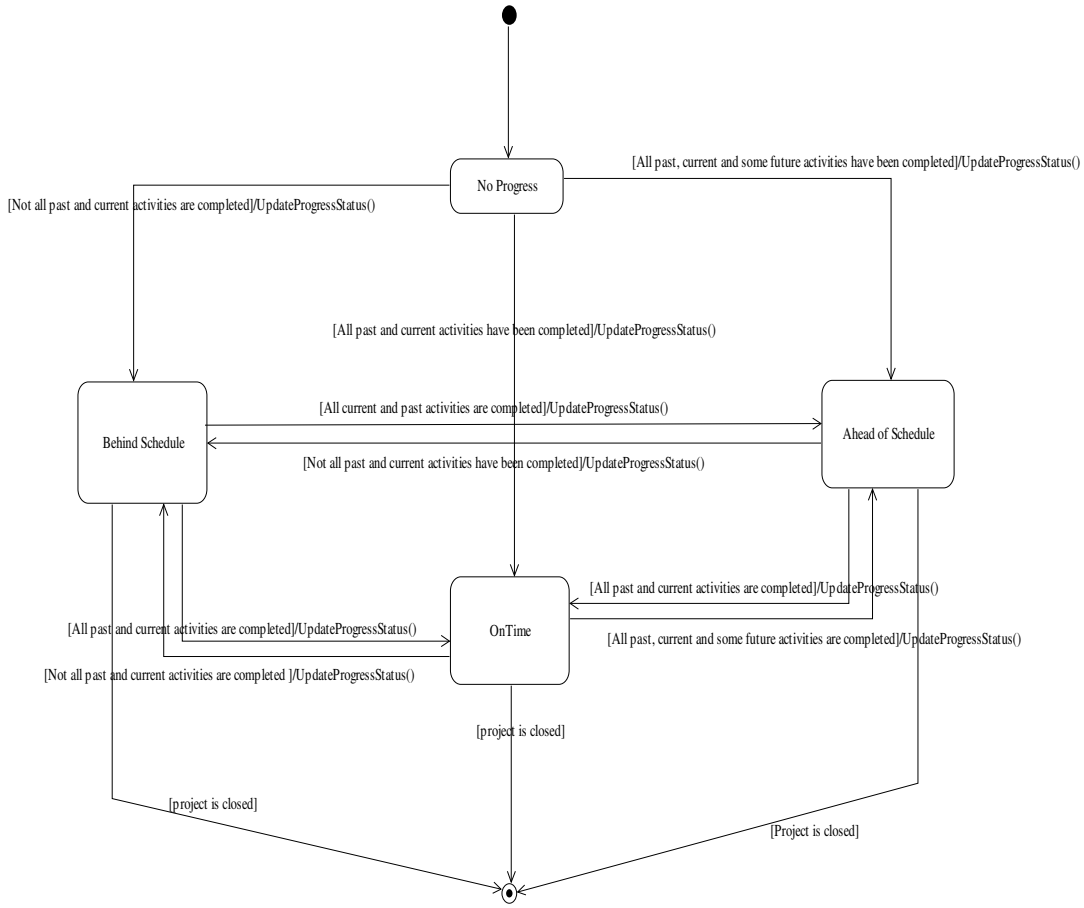
3.4.7.1 "ProgressStateMachine" state machine

States :

- *OnTime*
- *Behind Schedule*
- *Ahead of Schedule*
- *No Progress*

Transitions :

- *No Progress to OnTime*
- *Behind Schedule to OnTime*
- *Ahead of Schedule to OnTime*
- *OnTime to Behind Schedule*
- *OnTime to Ahead of Schedule*
- *OnTime to Final*
- *No Progress to Behind Schedule*
- *Ahead of Schedule to Behind Schedule*
- *Behind Schedule to Ahead of Schedule*
- *Behind Schedule to Final*
- *No Progress to Ahead of Schedule*
- *Ahead of Schedule to Final*
- *Initial to No Progress*



Progress State Machine Diagram

3.4.7.2 "UpdateProgressStatus" operation

`public UpdateProgressStatus ()`
This refers to the updating of progress status depending on the conditions defined in the state machine diagram.

3.4.7.3 "GetProgressStatus" operation

`public GetProgressStatus ()`
This refers to the retrieval of progress status typically to serve the need of generating management reports.

3.4.7.4 "ReviewProgress" operation

`public ReviewProgress ()`
This refers to the act of reviewing progress status and the capturing of review findings at the end.

3.5 "Basic" package

Description :

This Package contains structural and behavioural specifications for the fundamental concepts of business project management.

Classes :

- *Initiative*
- *Programme*
- *Project*
- *Milestone*
- *Phase*
- *Gate*
- *Task*
- *Activity*
- *Schedule*
- *Input*

3.5.1 "Initiative" class

Description :

Any intention or endeavour, super ordinate concept of project and programme.

Parent classes :

- *Tactical*

Operations :

- *public CreateInitiative ( )*
- *public UpdateInitiative ( )*
- *public UpdateInitiativeStatus ( )*
- *public KickOffInitiative ( )*
- *public CloseInitiative ( )*

Associations :

- *CapturesExpectedBenefitAsPerformanceKPI : (0-*) as undefined to Key Performance Indicators (0-*) as undefined*

Attributes :

- *public char[600] Description*
- *public real CostEstimated*
- *public real BenefitEstimated*
- *public real CostActual*
- *public real BenefitActual*
- *public char[120] Name*
- *public real[Date] StartDatePlanned*
- *public real[Date] EndDatePlanned*
- *public real[Date] StartDateActual*
- *public real[Date] EndDateActual*
- *public char[120] Owner*
- *public char[60] Status*

State machines :

- *InitiativeLifeCycleStateMachine*

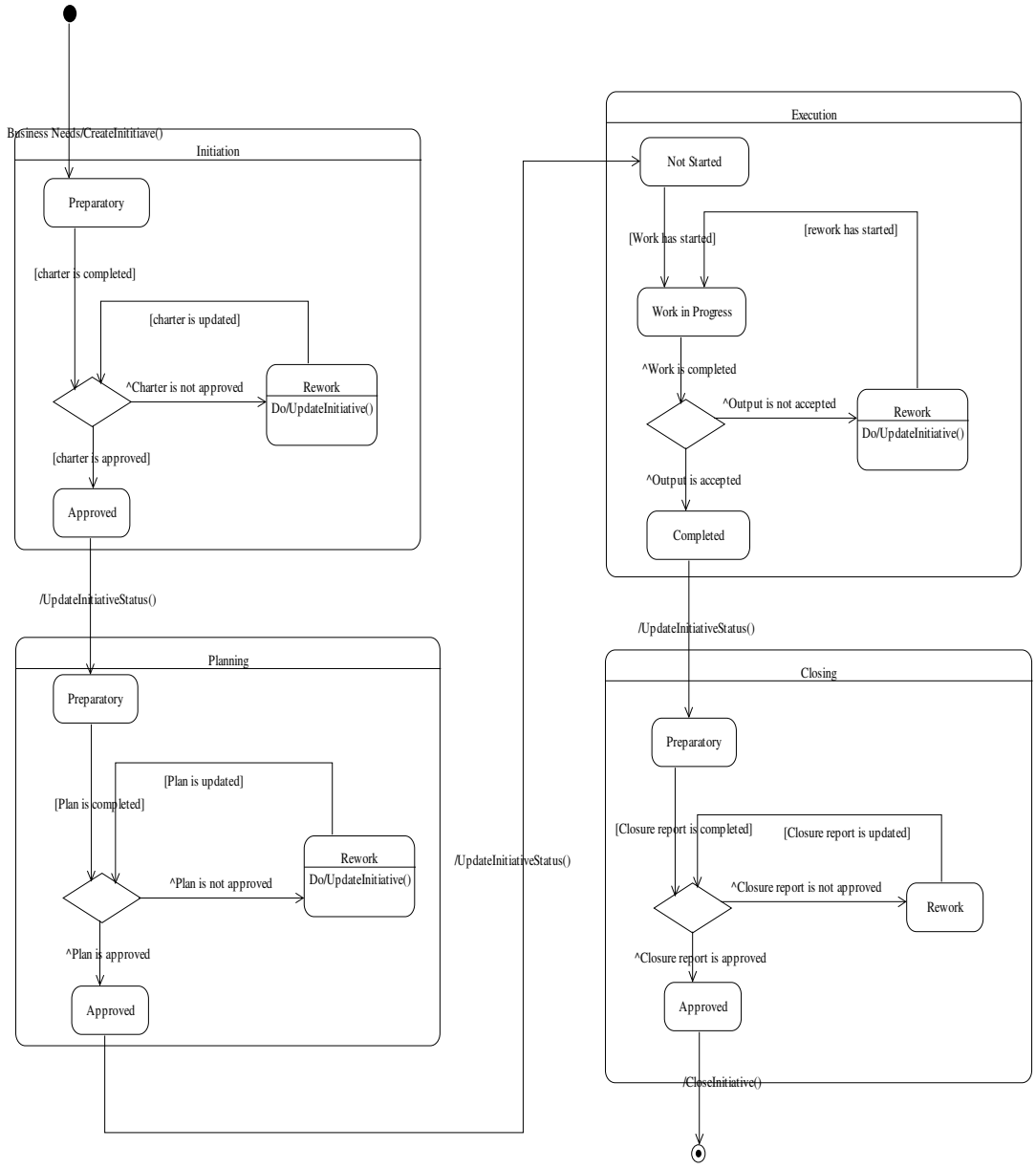
3.5.1.1 "InitiativeLifeCycleStateMachine" state machine

States :

- *Initiation*
- *Initiation::Preparatory*
- *Initiation::Rework*
- *Initiation::Approved*
- *Planning*
- *Planning::Preparatory*
- *Planning::Rework*
- *Planning::Approved*
- *Execution*
- *Execution::Not Started*
- *Execution::Work in Progress*
- *Execution::Rework*
- *Execution::Completed*
- *Closing*
- *Closing::Preparatory*
- *Closing::Rework*
- *Closing::Approved*

Transitions :

- *Initial to Preparatory*
- *Preparatory to Branch*
- *Branch to Rework*
- *Rework to Branch*
- *Branch to Approved*
- *Approved to Preparatory*
- *Preparatory to Branch*
- *Branch to Rework*
- *Rework to Branch*
- *Branch to Approved*
- *Approved to Not Started*
- *Not Started to Work in Progress*
- *Rework to Work in Progress*
- *Work in Progress to Branch*
- *Branch to Rework*
- *Branch to Completed*
- *Completed to Preparatory*
- *Preparatory to Branch*
- *Branch to Rework*
- *Rework to Branch*
- *Branch to Approved*
- *Approved to Final*



Initiative Life Cycle State Machine Diagram

3.5.1.2 "CreateInitiave" operation

`public CreateInitiave ()`
This refers to defining a new initiative when it is first initiated.

3.5.1.3 "UpdateInitiative" operation

`public UpdateInitiative ()`
This refers to updating the details of the created initiative.

3.5.1.4 "UpdateInitiativeStatus" operation

`public UpdateInitiativeStatus ()`
This refers to the update of initiative status depending on the conditions specified in the state machine diagram.

3.5.1.5 "KickOffInitiative" operation

`public KickOffInitiative ()`
This refers to processes related to officiating the commencement of the initiative.

3.5.1.6 "CloseInitiative" operation

`public CloseInitiative ()`
This refers to official closure of the initiative.

3.5.2 "Programme " class

Description :
A group of projects organized to meet a set of organizational objectives.

Parent classes :

- *Initiative*

Operations :

- *public GetProgrammes ( )*

Associations :

- *IsGuidedBy : (0-*) as undefined to Programme Objective (1-1) as undefined*
- *Contains : (0-*) as Prog to Programme (1-1) as Subprog*

3.5.2.1 "GetProgrammes" operation

`public GetProgrammes ()`
This refers to the retrieval of all sub-programmes classified under the given programme.

3.5.3 "Project" class

Description :

A group of people working together in order to achieve specific objective(s), given limited time and resources.

Parent classes :

- *Initiative*

Operations :

- *public GetProjects ( )*

Associations :

- *applies : (1-*) as undefined to Core Business Project Mgmt (1-1) as undefined*
- *IsGuidedBy : (0-*) as undefined to Project Objective (1-1) as undefined*

3.5.3.1 "GetProjects" operation

`public GetProjects ()`
This refers to retrieval of all projects classified under a given programme.

3.5.4 "Milestone" class

Description :

Event with significant meaning for project status.

Parent classes :

- *Event*

Operations :

- *public DefineMilestone ( )*
- *public UpdateMilestone ( )*
- *public GetMilestone ( )*

Associations :

- *StartsTrackingOf : (0-*) as undefined to Key Performance Indicators (1-1) as undefined*

3.5.4.1 "DefineMilestone" operation

`public DefineMilestone ()`
This refers to specifying a specific gate as milestone.

3.5.4.2 "UpdateMilestone" operation

`public UpdateMilestone ()`
This refers to updating the details of the defined Milestone.

3.5.4.3 "GetMilestone" operation

`public GetMilestone ()`
This refers to the retrieval of milestones in an INITIATIVE.

3.5.5 "Phase" class

Description :
Subdivision of project timing with specific objective. Often ends with a gate.

Operations :

- *public DefinePhase ( )*
- *public UpdatePhase ( )*
- *public UpdatePhaseStatus ( )*

Associations :

- *EndsWith : (1-1) as undefined to Gate (1-1) as undefined*

Attributes :

- *public char[120] Name*
- *public char[600] Description*
- *public char[60] Status*
- *public real StartDatePlanned*
- *public real StartDateRevised*
- *public real StartDateActual*
- *public real EndDatePlanned*
- *public real EndDateRevised*
- *public real EndDateActual*

State machines :

- *PhaseStateMachine*

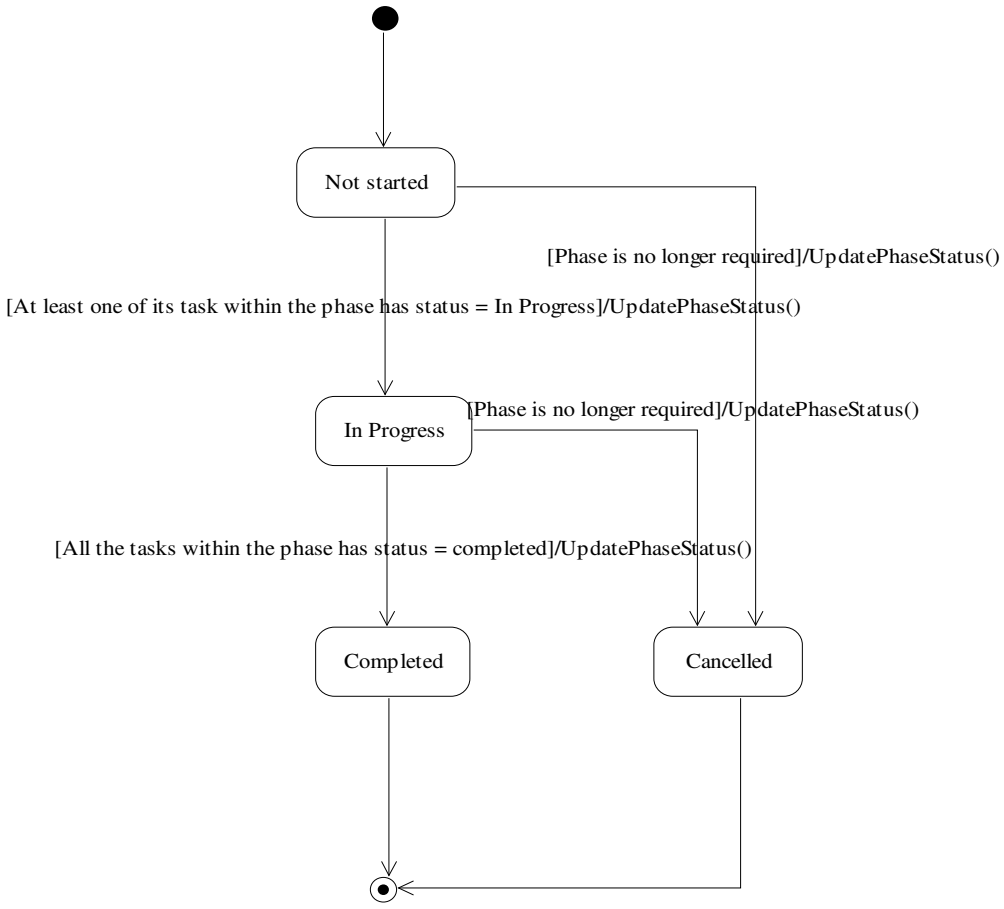
3.5.5.1 "PhaseStateMachine" state machine

States :

- *Not started*
- *In Progress*
- *Completed*
- *Cancelled*

Transitions :

- *Initial to Not started*
- *Not started to In Progress*
- *Not started to Cancelled*
- *In Progress to Completed*
- *In Progress to Cancelled*
- *Completed to Final*
- *Cancelled to Final*



Phase State Machine Diagram

3.5.5.2 "DefinePhase" operation

`public DefinePhase ()`
This refers to act of dividing a project into logical chucks, techinques such as Work Breakdown Structure (WBS) could be applied.

3.5.5.3 "UpdatePhase" operation

`public UpdatePhase ()`
This refers to updating the details of the defined Phase especially start and ene dates.

3.5.5.4 "UpdatePhaseStatus" operation

`public UpdatePhaseStatus ()`
This refers to the updating of Phase status depending on the conditions specified in the state machine diagram.

3.5.6 "Gate" class

Description :
Milestone ending a Phase. Usually associated with a formal review task.

Parent classes :

- *Milestone*

3.5.7 "Task" class

Description :
Project specific assignment. May be divided into sub-tasks. May be implemented by the application of a Process.

Operations :

- *public RollupConsumption ( )*
- *public UpdateTaskStatus ( )*
- *public DefineTask ( )*
- *public UpdateTask ( )*
- *public GetTaskStatus ( )*

Associations :

- *TransformInto : (1-*) as undefined to Activity (1-1) as undefined*
- *Needs : (1-*) as undefined to Skill (1-1) as undefined*
- *Defines : (1-*) as undefined to Input (1-1) as undefined*
- *Delivers : (1-*) as undefined to Output (1-1) as undefined*
- *IsOrganizedInto : (0-*) as Task to Task (1-1) as Subtask*

Attributes :

- *public char[120] Description*
- *public char[120] Name*
- *public char Owner*
- *public char[60] Status*
- *public real ConsumptionPlanned*
- *public real ConsumptionActual*
- *public real StartDatePlanned*
- *public real StartDateRevised*
- *public real StartDateActual*
- *public real EndDatePlanned*
- *public real EndDateRevised*
- *public real EndDateActual*

State machines :

- *Task State Machine*

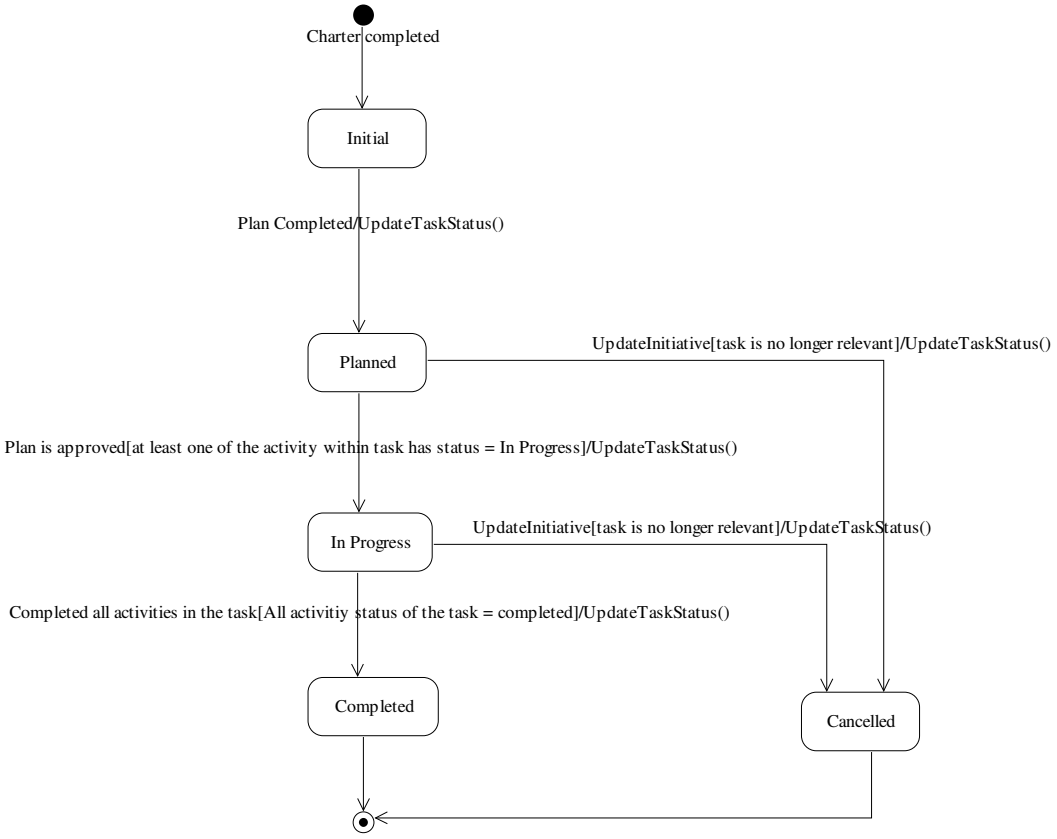
3.5.7.1 "Task State Machine" state machine

States :

- *Initial*
- *Planned*
- *In Progress*
- *Completed*
- *Cancelled*

Transitions :

- *Initial to Initial*
- *Initial to Planned*
- *Planned to In Progress*
- *Planned to Branch*
- *Planned to Cancelled*
- *In Progress to Completed*
- *In Progress to Branch*
- *In Progress to Cancelled*
- *Completed to Final*
- *Branch to Cancelled*
- *Cancelled to Final*



Task State Machine Diagram

3.5.7.2 "RollupConsumption" operation

`public RollupConsumption ()`
This refers to cumulating the actual consumptions of all the activities defined under the task.

3.5.7.3 "UpdateTaskStatus" operation

`public UpdateTaskStatus ()`
This refers to updating the task status depending on the conditions specified in the state machine diagram.

3.5.7.4 "DefineTask" operation

`public DefineTask ()`
This refers to defining the tasks required to support the completion of a project.

3.5.7.5 "UpdateTask" operation

`public UpdateTask ()`
This refers to the updating the details of a defined task especially the start and end date.

3.5.7.6 "GetTaskStatus" operation

`public GetTaskStatus ()`
This refers to the retrieval of the task status typically to serve the need of generating management reports.

3.5.8 "Activity" class

Description :

Task that has been assigned with specific resources. The assignment determines duration and costs of task execution.

Operations :

- *public UpdateActvity ( )*
- *public DefineActivity ( )*
- *public UpdateActivityStatus ( )*
- *public SequeceActivity ( )*
- *public EstimateActivityDuration ( )*
- *public GetActivityStatus ( )*
- *public GetActivityDetails ( )*

Associations :

- *Determines duration and cost : (1-1) as undefined to Task (1-*) as undefined*
- *Consumes : (0-*) as undefined to Input (0-*) as undefined*
- *Delivers : (1-1) as undefined to Progress (1-*) as undefined*

Attributes :

- *public char[600] Description*
- *public undefined Owner*
- *public char[120] Name*
- *public char[60] Status*
- *public real StartDatePlanned*
- *public real StartDateRevised*
- *public real StartDateActual*
- *public real EndDatePlanned*
- *public real EndDateRevised*
- *public real EndDateActual*

State machines :

- *Activity State Machine*

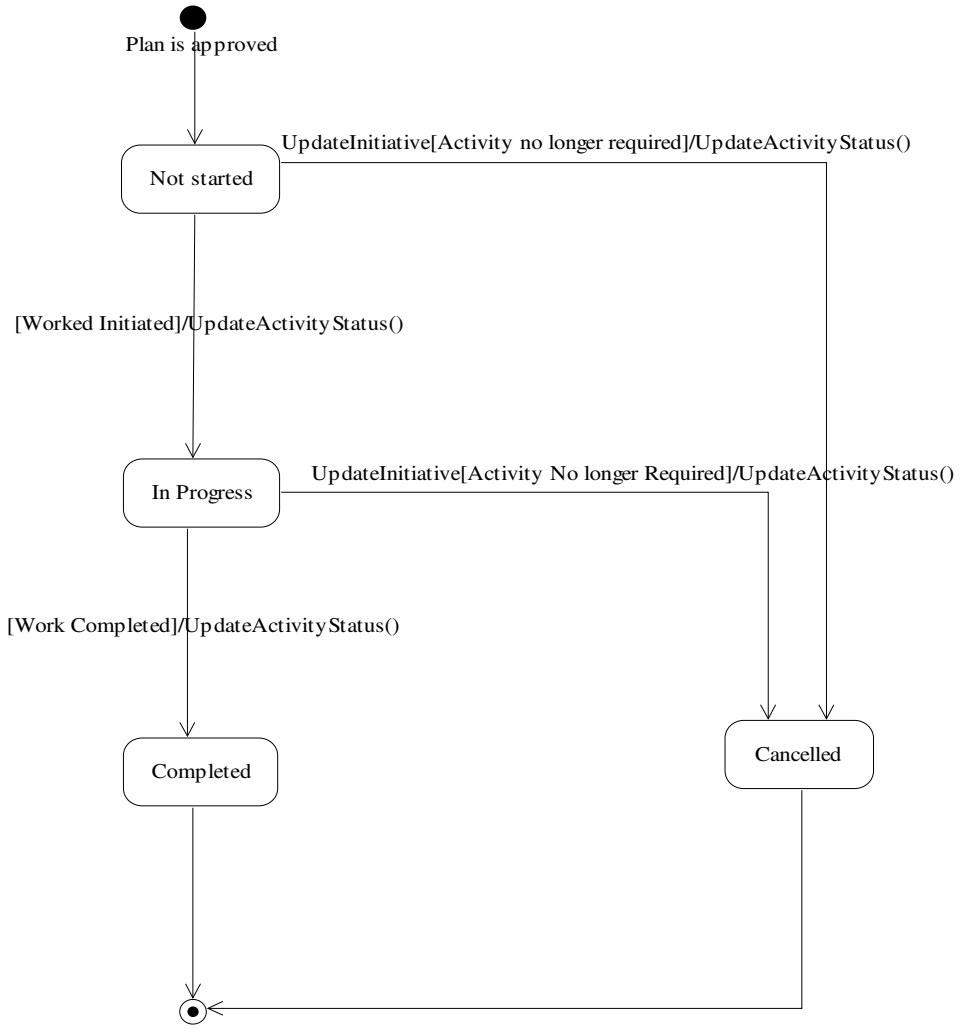
3.5.8.1 "Activity State Machine" state machine

States :

- *Not started*
- *In Progress*
- *Completed*
- *Cancelled*

Transitions :

- *Initial to Not started*
- *Not started to Branch*
- *Not started to Cancelled*
- *Not started to In Progress*
- *In Progress to Completed*
- *In Progress to Cancelled*
- *Completed to Final*
- *Cancelled to Final*



Activity State Machine Diagram

3.5.8.2 "UpdateActvity" operation

`public UpdateActvity ()`
This refers to the updating the details of defined activities especially the start and end dates.

3.5.8.3 "DefineActivity" operation

`public DefineActivity ()`
This refers to defining the activities within each task.

3.5.8.4 "UpdateActivityStatus" operation

`public UpdateActivityStatus ()`
This refers to updating of activity status depending on the conditions specified in the state machine diagram.

3.5.8.5 "SequeceActivity" operation

`public SequeceActivity ()`
This correponds to the "Activity Sequencing" related processes of project time management defined in PMBOK Guide.

3.5.8.6 "EstimateActivityDuration" operation

`public EstimateActivityDuration ()`
This correspond to the "Activity Duratin Estimating" related processes of project time management in the PMBOK Guide.

3.5.8.7 "GetActivityStatus" operation

`public GetActivityStatus ()`
This refers to the retrieval of activity status typically to serve the need of generating management reports.

3.5.8.8 "GetActivityDetails" operation

`public GetActivityDetails ()`
This refers to the retrieval of activity details typically bounded by a range of dates.

3.5.9 "Schedule" class

Description :
The planned dates for performing activities and the planned dates for meeting milestones.

Operations :

- *public CreateSchedule ( )*
- *public UpdateSchedule ( )*
- *public GetActivity ( )*
- *public GetTask ( )*
- *public GetPhase ( )*

Attributes :

- *public char[600] Description*
- *public char[120] Name*
- *public char[60] Version*
- *public undefined Phase/Task/Activity*
- *public real StartDateCurrent*
- *public real EndDateCurrent*
- *public real StartDateBaseline*
- *public real EndDateBaseline*

3.5.9.1 "CreateSchedule" operation

`public CreateSchedule ()`
This refers to the act of consolidating PHASE, TASKS and ACTIVITY to form a single reference.

3.5.9.2 "UpdateSchedule" operation

`public UpdateSchedule ()`
This refers to formation a new version of the schedule due to an update in PHASE, TASK or ACTIVITY.

3.5.9.3 "GetActivity" operation

`public GetActivity ()`
This refers to retrieval of activities defined in the schedule based on a given date.

3.5.9.4 "GetTask" operation

`public GetTask ()`
This refers to the retrieval of tasks based on a given date.

3.5.9.5 "GetPhase" operation

`public GetPhase ()`
This refers to the retrieval of Phases based on a given date.

3.5.10 "Input" class

Description :

Composite term which refers to all types of resources that is required to execute an INITIATIVE.
Operations :

- *public AllocateInput ( )*
- *public EstimateInput ( )*
- *public UpdateConsumption ( )*

Associations :

- *AssignedTo : (1-*) as undefined to Activity (1-1) as undefined*
- *IsAnAllocationOf : (0-*) as undefined to Asset (0-*) as undefined*

Attributes :

- *public char[120] Name*
- *public char[600] Description*
- *public real AmountEstimated*
- *public real AmountAllocated*
- *public real AmountConsumed*

3.5.10.1 "AllocateInput" operation

```
public  AllocateInput ()
```
This refers to the allocation of input made available by the parent organization for the project execution.

Constraints : {Validation}

Allocated resources must be predefined in RESOURCE or FIXED ASSET.

3.5.10.2 "EstimateInput" operation

```
public EstimateInput ()
```
This refers to the act of estimating the amount of INPUT required by the project execution.

3.5.10.3 "UpdateConsumption" operation

```
public UpdateConsumption ()
```
This refers to updating the actual consumption of the allocated input.

3.6 "Change Mgmt" package

Description :

This Package contains structural and behavioural specifications of change management related concepts in the context of project and/or programme.

Classes :

- *Issue*
- *Change Requests*
- *Lesson Learned*

3.6.1 "Issue" class

Description :

A point or matter of discussion, debate or dispute which may or may not lead to the creation of change request.

Operations :

- *public PerformImpactAnalysis ( )*
- *public RaiseIssue ( )*
- *public UpdateIssueStatus ( )*

Associations :

- *Creates : (0-1) as undefined to Change Requests (1-1) as undefined*

Attributes :

- *public char[600] Description*
- *public char[120] Impact*
- *public char[60] severity*
- *public char[60] Status*
- *public char[120] Name*
- *public char[600] Resolutions*

State machines :

- *IssueStateMachine*

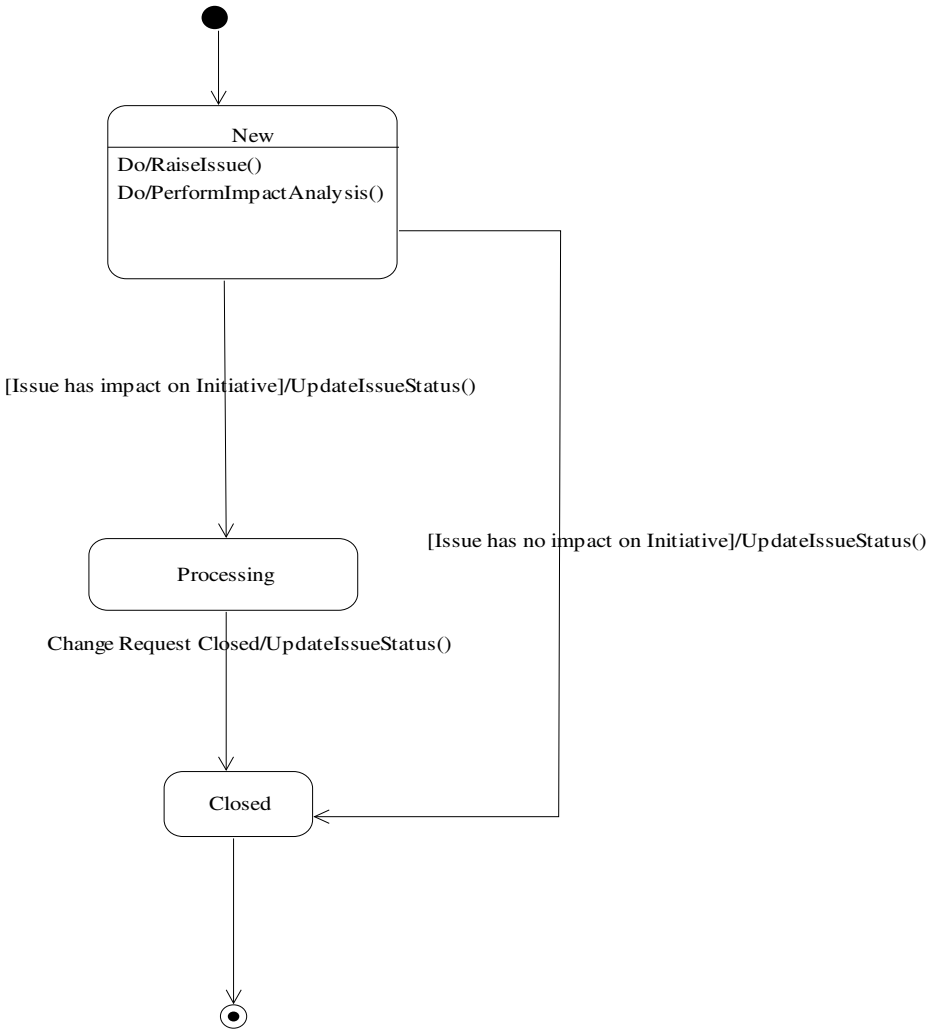
3.6.1.1 "IssueStateMachine" state machine

States :

- *New*
- *Processing*
- *Closed*

Transitions :

- *Initial to New*
- *New to Processing*
- *New to Closed*
- *Processing to Closed*
- *Closed to Final*



Issue State Machine Diagram

3.6.1.2 "PerformImpactAnalysis" operation

`public PerformImpactAnalysis ()`
This refers to the processes involved in determining the impact an issue has on an existing INITIATIVE.

3.6.1.3 "RaiseIssue" operation

`public RaiseIssue ()`
This refers to capturing the details of an issue which may pose an impact on existing INITIATIVE.

3.6.1.4 "UpdateIssueStatus" operation

`public UpdateIssueStatus ()`
This refers to updating the issue status based on the conditions specified in the state machine diagram.

3.6.2 "Change Requests" class

Description :
A document containing a call for an adjustment to one or more aspects of the project management.

Operations :

- *public ProcessChangeReqst ( )*
- *public RaiseChangeReqst ( )*
- *public UpdateChangeReqstStatus ( )*
- *public EstimateCostBenefits ( )*

Associations :

- *Updates : (1-*) as undefined to Initiative (1-*) as undefined*
- *Updates : (0-*) as undefined to Document (0-*) as undefined*
- *Creates : (0-1) as undefined to Lesson Learned (1-1) as undefined*

Attributes :

- *public char[600] Description*
- *public char[60] Status*
- *public char[120] Name*

State machines :

- *ChangeReqStateMachine*

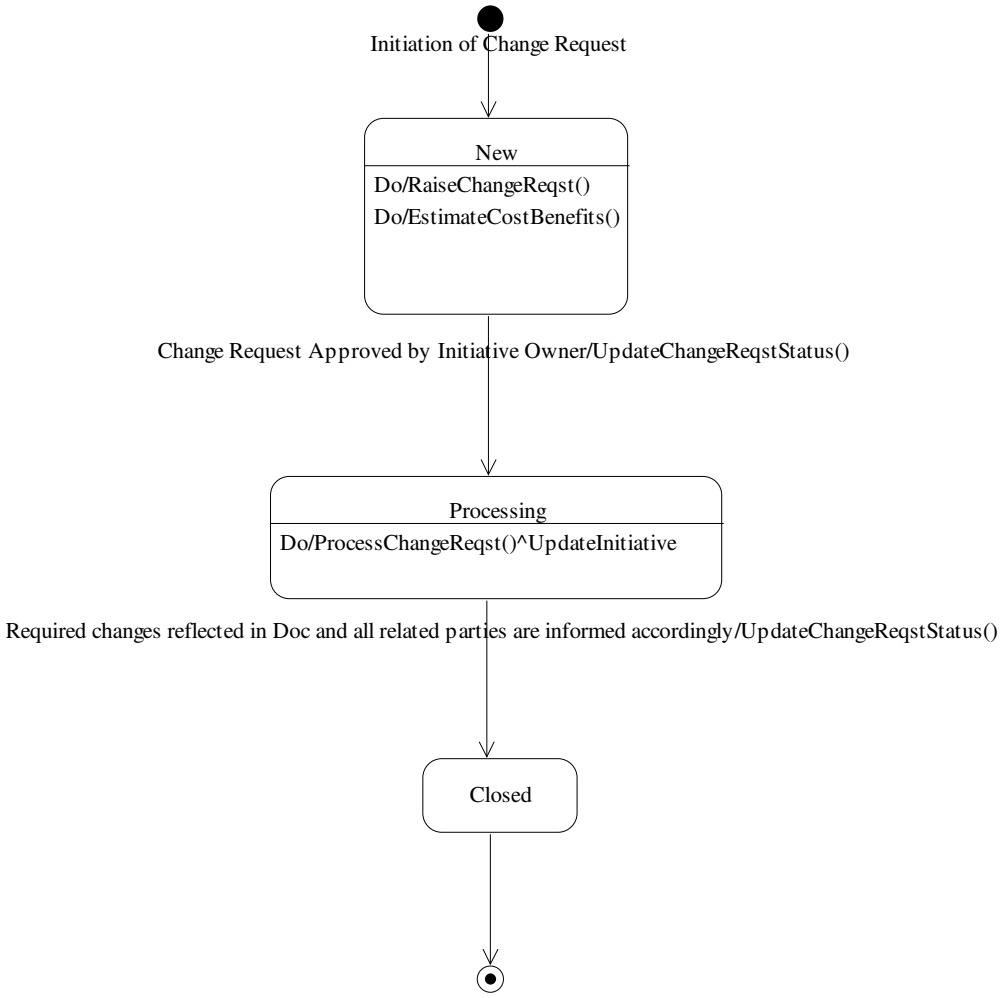
3.6.2.1 "ChangeReqStateMachine" state machine

States :

- *New*
- *Processing*
- *Closed*

Transitions :

- *Initial to New*
- *New to Processing*
- *Processing to Closed*
- *Closed to Closed*
- *Closed to Final*



Change Request State Machine Diagram

3.6.2.2 "ProcessChangeReqst" operation

`public ProcessChangeReqst ()`
This refers to the processes related to effecting the change request in the various aspect of project management work, as classified under the "Controlling" Process Group in the PMBOK Guide.

3.6.2.3 "RaiseChangeReqst" operation

`public RaiseChangeReqst ()`
This refers to defining the details of the change request.

3.6.2.4 "UpdateChangeReqstStatus" operation

`public UpdateChangeReqstStatus ()`
This refers to updating of change request status depending on the conditions specified in the state machine diagram.

3.6.2.5 "EstimateCostBenefits" operation

`public EstimateCostBenefits ()`
This refers to estimating and analyzing the cost and benefits of implementing the change reuqest.

3.6.3 "Lesson Learned" class

Description :
The cause of issue and/or change request, the reasoning behind the preventive and/or corrective action.

Operations :

- *public SearchLessonLearned ( )*
- *public ConsolidatesLessonLearned ( )*

Associations :

- *undefined : (0-1) as undefined to Closure Report (undefined-undefined) as closure Report*
- *Guides : (0-*) as undefined to Initiative (0-*) as undefined*

Attributes :

- *public char[120] Name*
- *public char[600] Description*

3.7 "Documentation" package

Description :
This Package contains structural and behavioural specificaitons of concepts related to project and/or programme documentation.

Classes :

- *Comms Mgmt Plan*
- *Charter*
- *Integrated Plan*
- *Project Log*
- *Document*
- *Record*
- *Progress Report*
- *Closure Report*
- *Performance Report*
- *HR Mgmt Plan*
- *Procurement Mgmt Plan*
- *Quality Mgmt Plan*
- *Financial Mgmt Plan*
- *Risk Mgmt Plan*
- *Scope mgmt Plan*
- *Schedule Mgmt Plan*
- *Output Report*
- *Report*

3.7.1 "Comms Mgmt Plan" class

Description :

A document that provides a collection and filing structure, a distribution structure, a description of the information to be distributed, production schedule for each type of communication, methods for accessing information between scheduled communications, a method for updating and refining the communication management plan as the project progresses and develops.

Parent classes :

- *Document*

3.7.2 "Charter" class

Description :

A document issued by senior management that formally authorizes the existence of an INITIATIVE.

Parent classes :

- *Document*

3.7.3 "Integrated Plan" class

Description :

A formal, approved document used to guide both execution and control of an INITIATIVE. The primary uses of he plan is to document planning assumptions and decisions, facilitate communication among stakeholders and document approved scope, cost and schedule baselines.

Parent classes :

- *Document*

3.7.4 "Project Log" class

Description :

A document which keeps chronological records of issues and its related change requests and lesson learned.

Parent classes :

- *Document*

Operations :

- *public SearchLog ( )*

3.7.4.1 "SearchLog" operation

`public SearchLog ()`

This refers to the search for information in the PROJECT LOG based on the given search criteria.

3.7.5 "Document" class

Description :

A consolidated form of record that presents related information in an organized fashion.

Operations :

- *public CreateDoc ( )*
- *public UpdateDoc ( )*
- *public ReviewDoc ( )*
- *public DistributeDoc ( )*
- *public ApproveDoc ( )*
- *public UpdateDocStatus ( )*
- *public GetDocStatus ( )*

Attributes :

- *public char[120] Name*
- *public char[4] Version*
- *public char[600] Description*
- *public real[Date] CreationDate*
- *public real[Date] RevisedDate*
- *public real[Date] ApprovedDate*
- *public real[Date] DistributionDate*
- *public char[60] Status*
- *public undefined Distribution list*

State machines :

- *DocumentStateMachine*

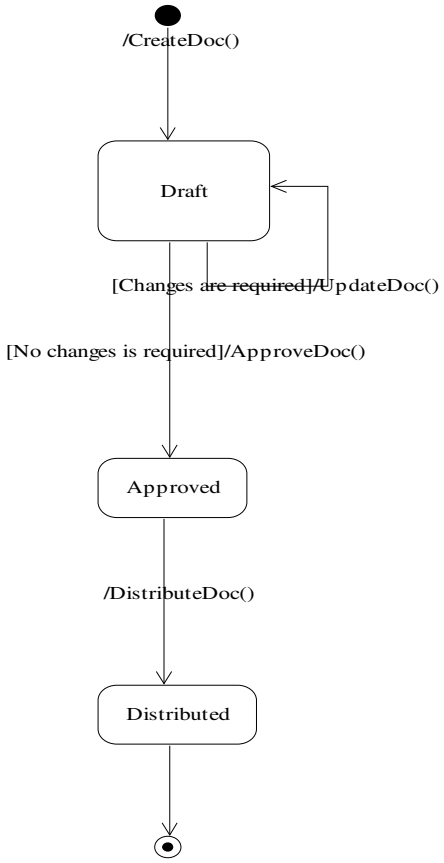
3.7.5.1 "DocumentStateMachine" state machine

States :

- *Draft*
- *Distributed*
- *Approved*

Transitions :

- *Initial to Draft*
- *Draft to Draft*
- *Draft to Approved*
- *Approved to Distributed*
- *Distributed to Final*



Document State Machine Diagram

3.7.5.2 "CreateDoc" operation

`public CreateDoc ()`
This refers to the creation of the document.

3.7.5.3 "UpdateDoc" operation

`public UpdateDoc ()`
This refers to the updating the content of document.

3.7.5.4 "ReviewDoc" operation

`public ReviewDoc ()`
This refers to the act of reviewing the document for the purpose of approval.

3.7.5.5 "DistributeDoc" operation

`public DistributeDoc ()`
This refers to the distribution of approved documents according to the distribution list.

3.7.5.6 "ApproveDoc" operation

`public ApproveDoc ()`
This refers to the act of approving the document.

3.7.5.7 "UpdateDocStatus" operation

`public UpdateDocStatus ()`
This refers to the updating of document status depending on the conditions specified in the state machine diagram.

3.7.5.8 "GetDocStatus" operation

`public GetDocStatus ()`
This refers to the retrieval of document status of a given document.

3.7.6 "Record" class

Description :
Correspondences, memos and documents describing the INITIATIVE.

Parent classes :

- *Document*

Attributes :

- *public char[60] Format*

3.7.7 "Progress Report" class

Description :

Formal report on the extent of which planned project activities are completed.

Parent classes :

- *Report*

3.7.8 "Closure Report" class

Description :

Formal report on the closure of the INITIATIVE.

Parent classes :

- *Document*

3.7.9 "Performance Report" class

Description :

Formal report on the contribution of the project at the programme /organizational level.

Parent classes :

- *Report*

3.7.10 "HR Mgmt Plan" class

Description :

A document that describes when and how human resources will be brought into and taken off the project team. Other supporting details include organization chart, job descriptions, training needs etc.

Parent classes :

- *Document*

3.7.11 "Procurement Mgmt Plan" class

Description :

A document that describes how the procurement processes (from solicitation planning through contract closeout) will be managed.

Parent classes :

- *Document*

3.7.12 "Quality Mgmt Plan" class

Description :

A document that describes how the project management team will implement its quality policy and quality system.

Parent classes :

- *Document*

3.7.13 "Financial Mgmt Plan" class

Description :

A document that describes how project financials or cost variances will be managed. Supporting details include basis for estimates, assumptions made, cost baseline and budget etc

Parent classes :

- *Document*

3.7.14 "Risk Mgmt Plan" class

Description :

A document that describes how risk identification, qualitative and quantitative analysis, response planning, monitoring, and control will be structured and performed during the project life cycle.

Parent classes :

- *Document*

3.7.15 "Scope mgmt Plan" class

Description :

A document that describes how project scope will be managed and how scope changes will be integrated into the project.

Parent classes :

- *Document*

3.7.16 "Schedule Mgmt Plan" class

Description :

A document that defines how changes to the schedule will be managed.

Parent classes :

- *Document*

3.7.17 "Output Report" class

Description :

Formal report on the status of delivering the direct output of the project.

Parent classes :

- *Report*

3.7.18 "Report" class

Description :

Formal report on the status and contributions of the INITIATIVE.

Parent classes :

- *Document*

Associations :

- *Extracts : (1-1) as undefined to Project Log (0-*) as undefined*
- *Extracts : (1-1) as undefined to Integrated Plan (0-*) as undefined*

Attributes :

- *public undefined Format*

3.8 "BMM" package

Description :

This Package contains structural specification for the BMM concepts.

Classes :

- *Assessment*
- *Assessment category*
- *Business Policy*

- *Business Rule*
- *Course of Action*
- *Desired Results*
- *Directive*
- *End*
- *Influencer*
- *Influencer category*
- *Influencing organization*
- *Means*
- *Mission*
- *Objective*
- *Organization category*
- *Potential rewards*
- *Regulation*
- *Risk*
- *Strategy*
- *Tactical*
- *Vision*
- *Potential Impact*
- *Goals*

3.8.1 "Assessment" class

Associations :

- *Identifies : (0-*) as undefined to Potential Impact (0-*) as undefined*
- *ProvideImpetusFor : (0-*) as undefined to Course of Action (0-*) as undefined*
- *JudgementOf : (1-*) as undefined to Influencer (0-*) as undefined*
- *AffectsAchievementOf : (0-*) as undefined to End (0-*) as undefined*
- *AffectsEmploymentOf : (0-*) as undefined to Means (0-*) as undefined*

3.8.2 "Assessment category" class

Associations :

- *Categories : (0-*) as undefined to Assessment (0-*) as undefined*

3.8.3 "Business Policy" class

Parent classes :

- *Directive*

Associations :

- *Governs : (0-*) as undefined to Business Process (0-*) as undefined*

3.8.4 "Business Rule" class

Parent classes :

- *Directive*

Associations :

- *BasisFor : (0-*) as undefined to Business Policy (0-*) as undefined*
- *Guides : (0-*) as undefined to Business Process (0-*) as undefined*

3.8.5 "Course of Action" class

Parent classes :

- *Means*

Associations :

- *Defines : (0-*) as undefined to Offerings (0-*) as undefined*
- *Deploys : (0-*) as undefined to Asset (0-*) as undefined*

3.8.6 "Desired Results" class

Parent classes :

- *End*

Associations :

- *undefined : (0-1) as undefined to Vision (undefined-undefined) as vision*

3.8.7 "Directive" class

Parent classes :

- *Means*

Associations :

- *Governs : (0-*) as undefined to Course of Action (0-*) as undefined*
- *ActAs : (0-*) as undefined to Regulation (1-*) as undefined*
- *SupportAchievementOf : (0-*) as undefined to Desired Results (0-*) as undefined*

3.8.8 "Influencer category" class

Associations :

- *Categories : (0-*) as undefined to Influencer (0-*) as undefined*

3.8.9 "Influencing organization" class

Associations :

- *IsSourceOf : (0-*) as undefined to Influencer (0-*) as undefined*

3.8.10 "Mission" class

Parent classes :

- *Means*

Associations :

- *MakesOperative : (0-1) as undefined to Vision (0-*) as undefined*

3.8.11 "Objective" class

Parent classes :

- *Desired Results*
- *Project Objective*

Operations :

- *public DefineObjective ( )*
- *public UpdateObjective ( )*

Associations :

- *Quantifies : (0-*) as undefined to Goals (0-*) as undefined*
- *MeasuredBy : (1-*) as undefined to Key Performance Indicators (1-1) as undefined*

3.8.12 "Organization category" class

Associations :

- *Categories : (0-*) as undefined to Influencing organization (0-*) as undefined*

3.8.13 "Potential rewards" class

Parent classes :

- *Potential Impact*

3.8.14 "Regulation" class

Parent classes :

- *Influencer*

3.8.15 "Risk" class

Parent classes :

- *Potential Impact*

3.8.16 "Strategy" class

Parent classes :

- *Course of Action*

Associations :

- *Strategy determines OrgUnit : (0-*) as undefined to Organisation unit (1-*) as undefined*

3.8.17 "Tactical" class

Parent classes :

- *Course of Action*

Associations :

- *Implements : (0-*) as undefined to Strategy (0-*) as undefined*

3.8.18 "Vision" class

Parent classes :

- *End*

3.8.19 "Goals" class

Parent classes :

- *Desired Results*

Associations :

- *Amplifies : (0-1) as undefined to Vision (0-*) as undefined*

3.9 BMM PlaceHolder (Assets & Liabilities)" package

Description :

This Package contains structural specification of the related concepts parked under the BMM Place Holders, in particular, those related to "Assets" and "Liabilities".

Classes :

- *Asset*
- *Fixed Asset*
- *Offerings*
- *Resource*
- *Liabilities*

3.9.1 "Fixed Asset" class

Parent classes :

- *Asset*

3.9.2 "Offerings" class

Parent classes :

- *Fixed Asset*

Associations :

- *Uses : (0-*) as undefined to Fixed Asset (0-*) as undefined*

3.9.3 "Resource" class

Parent classes :

- *Asset*

3.9.4 "Liabilities" class

Associations :

- *claims : (0-*) as undefined to Resource (0-*) as undefined*

3.10 "BPDM " package

Description :

This Package contains structural specification for the BPDM concepts.

Classes :

- *Business Process*

3.10.1 "Business Process" class

Description :

This refers to the core project management competencies which the business project manager must posess in order to assure delivery success at the project level

Operations :

- *public CreateBusinessProcess ( )*
- *public ImplementChangedBusinessProcess ( )*
- *public RemoveBusinessProcess ( )*

Associations :

- *Realizes : (0-*) as undefined to Course of Action (0-*) as undefined*
- *Delivers : (0-*) as undefined to Offerings (0-*) as undefined*
- *Manages : (0-*) as undefined to Asset (0-*) as undefined*
- *Creates : (0-*) as undefined to Document (0-*) as undefined*
- *Delivers : (0-1) as undefined to Performance (undefined-undefined) as undefined*
- *MeasuredBy : (0-1) as undefined to Key Performance Indicators (undefined-undefined) as undefined*
- *Updates : (0-*) as undefined to Document (0-*) as undefined*
- *IsGuidedBy : (0-*) as undefined to Document (0-*) as undefined*

3.11 "OSM" package

Description :

This Package contains structural specification of the OSM concepts.

Classes :

- *Person*
- *Contractor*
- *Position*
- *Organisation unit*
- *Organization*
- *Organization Role*
- *Organization Type*

3.11.1 "Contractor" class

Parent classes :

- *Person*
- *Resource*

Associations :

- *fills : (1-1) as undefined to Position (1-1) as undefined*

3.11.2 "Position" class

Parent classes :

- *Organization Role*

Associations :

- *Supplies : (1-*) as undefined to Skill (1-1) as undefined*

3.11.3 "Organisation unit" class

Parent classes :

- *Asset*

Associations :

- *has : (1-*) as undefined to Position (1-1) as undefined*
- *ResponsibleFor : (0-*) as undefined to Business Process (0-*) as undefined*
- *Makes : (0-*) as undefined to Assessment (1-*) as undefined*
- *ResponsibleFor : (0-*) as undefined to Liabilities (0-*) as undefined*
- *ResponsibleFor : (0-*) as undefined to Asset (0-*) as undefined*
- *OrgUnit Defines the End : (0-*) as undefined to End (1-*) as undefined*

3.11.4 "Organization" class

Associations :

- *has : (0-*) as undefined to Organization Role (1-1) as undefined*
- *Contains : (0-*) as undefined to Organisation unit (0-1) as undefined*

3.11.5 "Organization Role" class

Associations :

- *undefined : (0-1) as undefined to Organization (undefined-undefined) as Organization*

3.11.6 "Organization Type" class

Associations :

- *uses : (0-1) as undefined to Organization Role (1-1) as undefined*

3.12 "Calendar" class

Description

A timetable indicating the date, day, week and month in a calendar year.

Associations :

- *Guides : (0-*) as undefined to Activity (0-*) as undefined*

3.13 "Financial Mgmt" class

Description

Financial management concepts and functions.

Parent classes :

- *Business Process*

Operations :

- *public ExecuteCostMgmtPlan ( )*
- *public UpdateCostMgmtPlan ( )*
- *public EstimateCost ( )*
- *public DefineCostMgmtPlan ( )*
- *public PerformCBA ( )*
- *public ReviseCost ( )*

3.13.1 "ExecuteCostMgmtPlan" operation

`public ExecuteCostMgmtPlan ()`
This refers to execution of the cost management plan devised for the INITIATIVE.

3.13.2 "UpdateCostMgmtPlan" operation

`public UpdateCostMgmtPlan ()`
This refers to the updating of details in the cost management plan.

3.13.3 "EstimateCost" operation

`public EstimateCost ()`
This refers to estimating of cost involved in implementing the INITIATIVE.

3.13.4 "DefineCostMgmtPlan" operation

`public DefineCostMgmtPlan ()`
This refers to defining the financial /cost management plan for an INITIATIVE.

3.13.5 "PerformCBA" operation

`public PerformCBA ()`
This refers to the conduct of cost and benefits analysis for the INITIATIVE.

3.13.6 "ReviseCost" operation

`public ReviseCost ()`
This refers to revising the implementation cost for the INITIATIVE.

3.14 "Customer Services" class

Description
Customer Services concepts and functions.

Parent classes :

- *Business Process*

3.15 "Employee" class

Description
Person working for the ORGANIZATION UNIT. subset of RESOURCE.

Parent classes :

- *Person*
- *Resource*

Associations :

- *fills : (1-1) as undefined to Position (1-1) as undefined*

3.16 "Event" class

Description
Occurrence of an action at a specific point in time.

Operations :

- *public CaptureEvent ( )*
- *public BroadcastEvent ( )*

Associations :

- *Raises : (0-1) as undefined to Issue (1-1) as undefined*
- *Triggers : (0-1) as undefined to Course of Action (1-1) as undefined*

Attributes :

- *public real[date] Date*
- *public char[600] Description*
- *public char[120] Name*

3.16.1 "CaptureEvent" operation

`public CaptureEvent ()`
This refers to capturing the details of an event.

3.16.2 "BroadcastEvent" operation

`public BroadcastEvent ()`
This refers to broadcasting information about an event to a targeted group of audiences.

3.17 "HR Mgmt" class

Description
Human resources management concepts and functions.

Parent classes :

- *Business Process*

Operations :

- *public DefineHRmgmtReq ( )*
- *public UpdateHRmgmtReq ( )*
- *public ExecuteHRmgmtPlan ( )*

3.17.1 "DefineHRmgmtReq" operation

`public DefineHRmgmtReq ()`
This refers to defining the HR management requirements and preparation ofthe HR management plan for an INITIATIVE.

3.17.2 "UpdateHRmgmtReq" operation

`public UpdateHRmgmtReq ()`
This refers to the updating of details in the HR management plan.

3.17.3 "ExecuteHRmgmtPlan" operation

`public ExecuteHRmgmtPlan ()`
This refers to the execution of the HR management plan devised to support the INITIATIVE.

3.18 "Key Performance Indicators" class

Description
Key part of the measurable organizational objectives.

Operations :

- *public DefineKPI ( )*
- *public UpdateKPI ( )*
- *public MapKPI ( )*
- *public GetKPIDetails ( )*
- *public CaptureKPIActual ( )*

Associations :

- *MapsTo : (1-*) as ParentKPI to Key Performance Indicators (1-1) as ChildKPI*

Attributes :

- *public char[1] Type*
- *public char[600] Description*
- *public real EffectiveDatePlanned*
- *public real Data Source*
- *public undefined Formula*
- *public real Value*
- *public char[120] Name*
- *public char[60] Nature*
- *public real Threshold*
- *public real EffectiveDateActual*

3.18.1 "DefineKPI" operation

`public DefineKPI ()`
This refers to defining the KPIs for the measurabe objectives.

3.18.2 "UpdateKPI" operation

`public UpdateKPI ()`
This refers to updating details of a previously defined KPI.

3.18.3 "MapKPI" operation

`public MapKPI ()`
This refers to mapping of KPIs to reflect their hierarchical relationships.

3.18.4 "GetKPIDetails" operation

`public GetKPIDetails ()`
This refers to the retrieval of KPI details.

3.18.5 "CaptureKPIActual" operation

`public CaptureKPIActual ()`
This refers to capturing the actual KPI by (1) getting the raw data from the data source (2) apply the formula to calculate the KPI

3.19 "Machine" class

Description :

A non-consumable, non-human RESOURCE.

Parent classes :

- *Fixed Asset*

3.20 "Marketing" class

Description :

Marketing concepts and functions.

Parent classes :

- *Business Process*

3.21 "Procurement Mgmt" class

Description :

Procurement management concepts and functions.

Parent classes :

- *Business Process*

Operations :

- *public DefineProcurementMgmtReq ( )*
- *public UpdateProcurementMgmtReq ( )*
- *public ExecuteProcurementMgmtPlan ( )*

3.21.1 "DefineProcurementMgmtReq" operation

`public DefineProcurementMgmtReq ()`
This refers to defining the procurement management plan for an INITIATIVE.

3.21.2 "UpdateProcurementMgmtReq" operation

`public UpdateProcurementMgmtReq ()`
This refers to updating details in the procurement management plan.

3.21.3 "ExecuteProcurementMgmtPlan" operation

`public ExecuteProcurementMgmtPlan ()`
This refers to the execution of the procurement management plan devised to support an INITIATIVE.

3.22 "Quality Mgmt" class

Description :
Quality management concepts and functions.

Parent classes :

- *Business Process*

Operations :

- *public DefineQltyMgmtReq ( )*
- *public ExecuteQltyMgmtPlan ( )*
- *public UpdateQltyMgmtReq ( )*

3.22.1 "DefineQltyMgmtReq" operation

`public DefineQltyMgmtReq ()`
This refers to defining quality management plan for an INITIATIVE.

3.22.2 "ExecuteQltyMgmtPlan" operation

`public ExecuteQltyMgmtPlan ()`
This refers to execution of the quality management plan devised to support an INITIATIVE.

3.22.3 "UpdateQltyMgmtReq" operation

`public UpdateQltyMgmtReq ()`
This refers to updating the details in a quality management plan.

3.23 "Risk Mgmt" class

Description :
Risk management related concepts and functions

Parent classes :

- *Business Process*

Operations :

- *public DefineRiskMgmtReq ( )*
- *public UpdateRiskMgmtReq ( )*
- *public ExecuteRiskMgmtPlan ( )*

3.23.1 "DefineRiskMgmtReq" operation

`public DefineRiskMgmtReq ()`
This refers to defining the risk management plan for an INITIATIVE.

3.23.2 "UpdateRiskMgmtReq" operation

`public UpdateRiskMgmtReq ()`
This refers to updating details in the Risk Management Plan.

3.23.3 "ExecuteRiskMgmtPlan" operation

`public ExecuteRiskMgmtPlan ()`
This refers to execution of the risk management plan devised to support an INITIATIVE.

3.24 "Sales" class

Description :
Sales related concepts and functions.

Parent classes :

- *Business Process*

3.25 "Skill" class

Description :
Property and potential of a RESOURCE to satisfy requirement of a TASK.

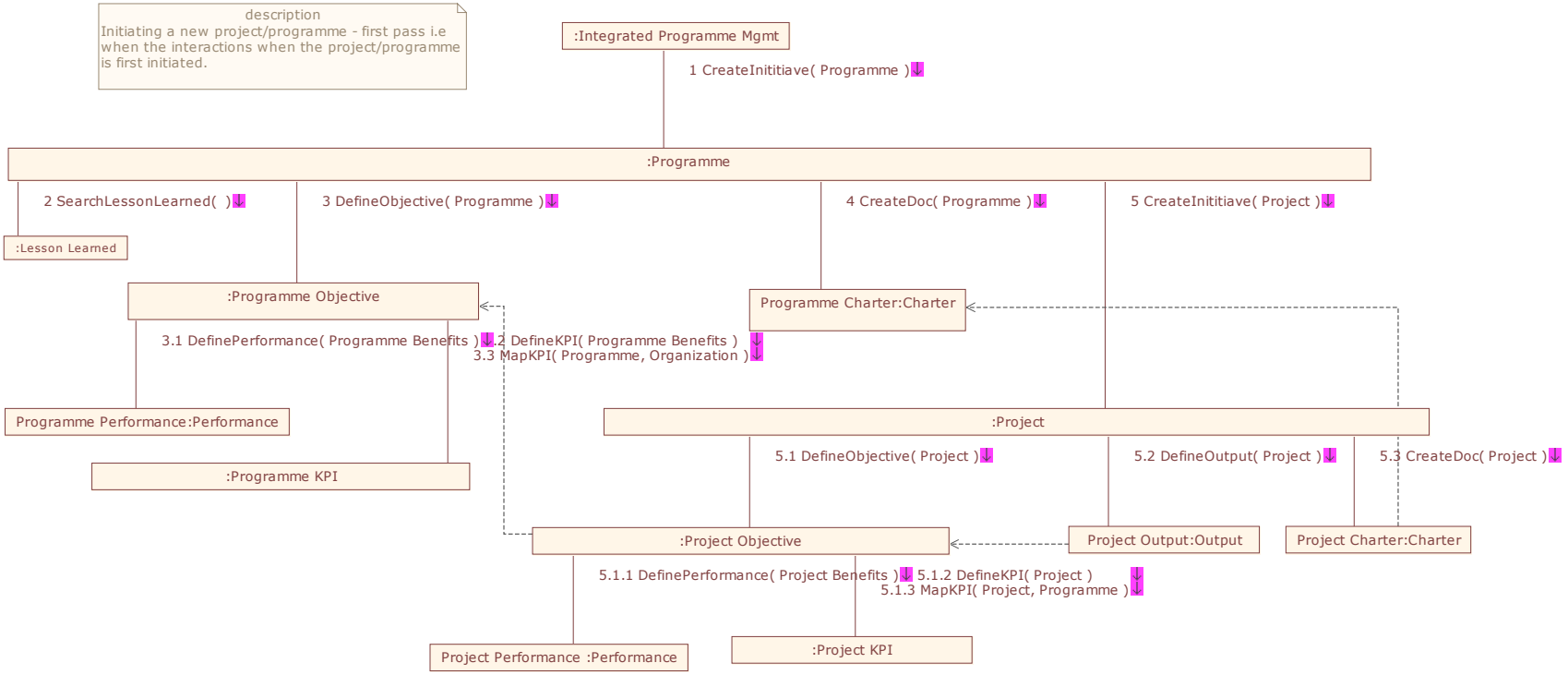
Associations :

- *equips : (1-1) as undefined to Position (1-*) as undefined*

3.26 Communications Diagrams

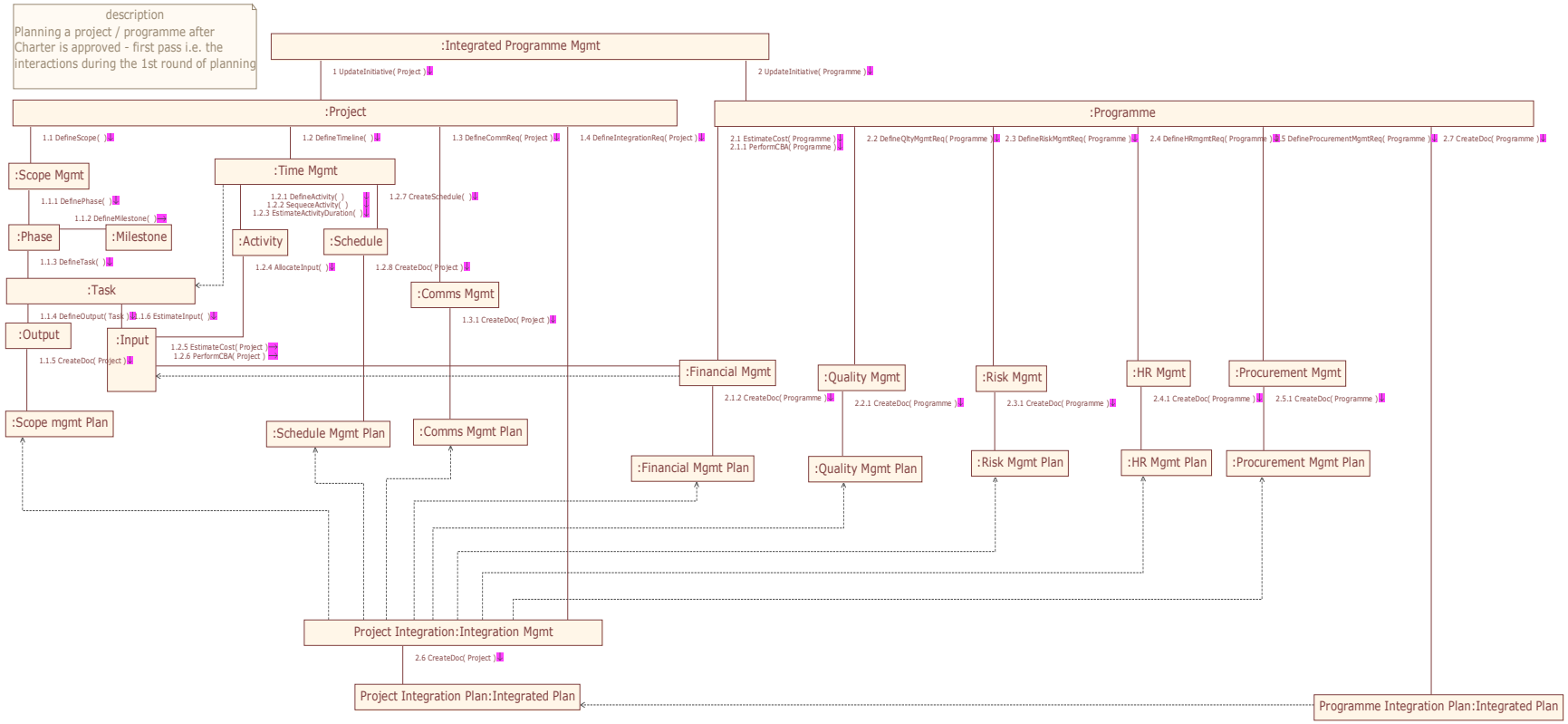
This section details the interactions among *Classes* during each state of the INITIATIVE, i.e. each stage of the project life cycle.

3.26.1 “Initiating” state



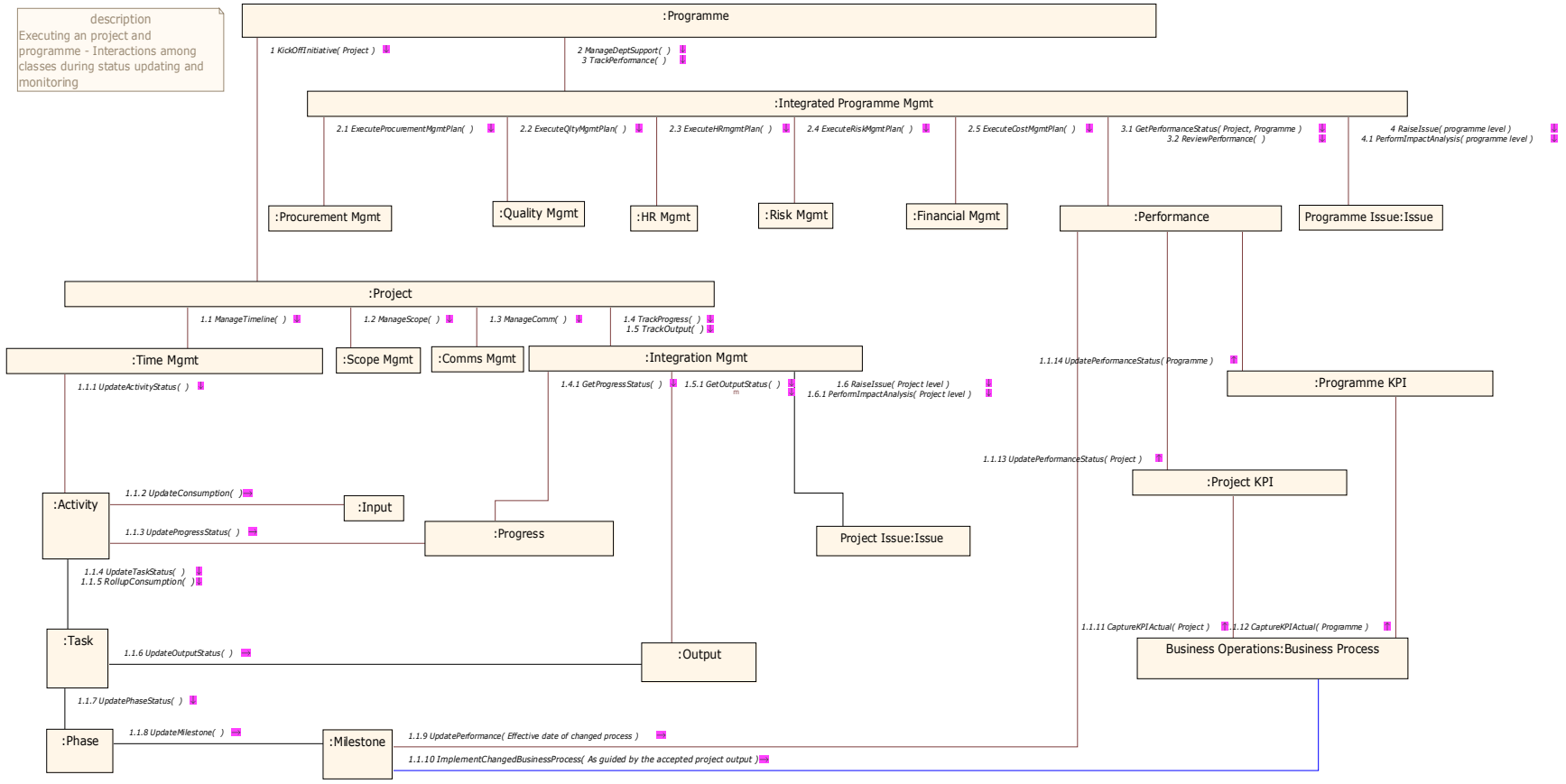
Communication Diagram (Initiating)

3.26.2 “Planning” state

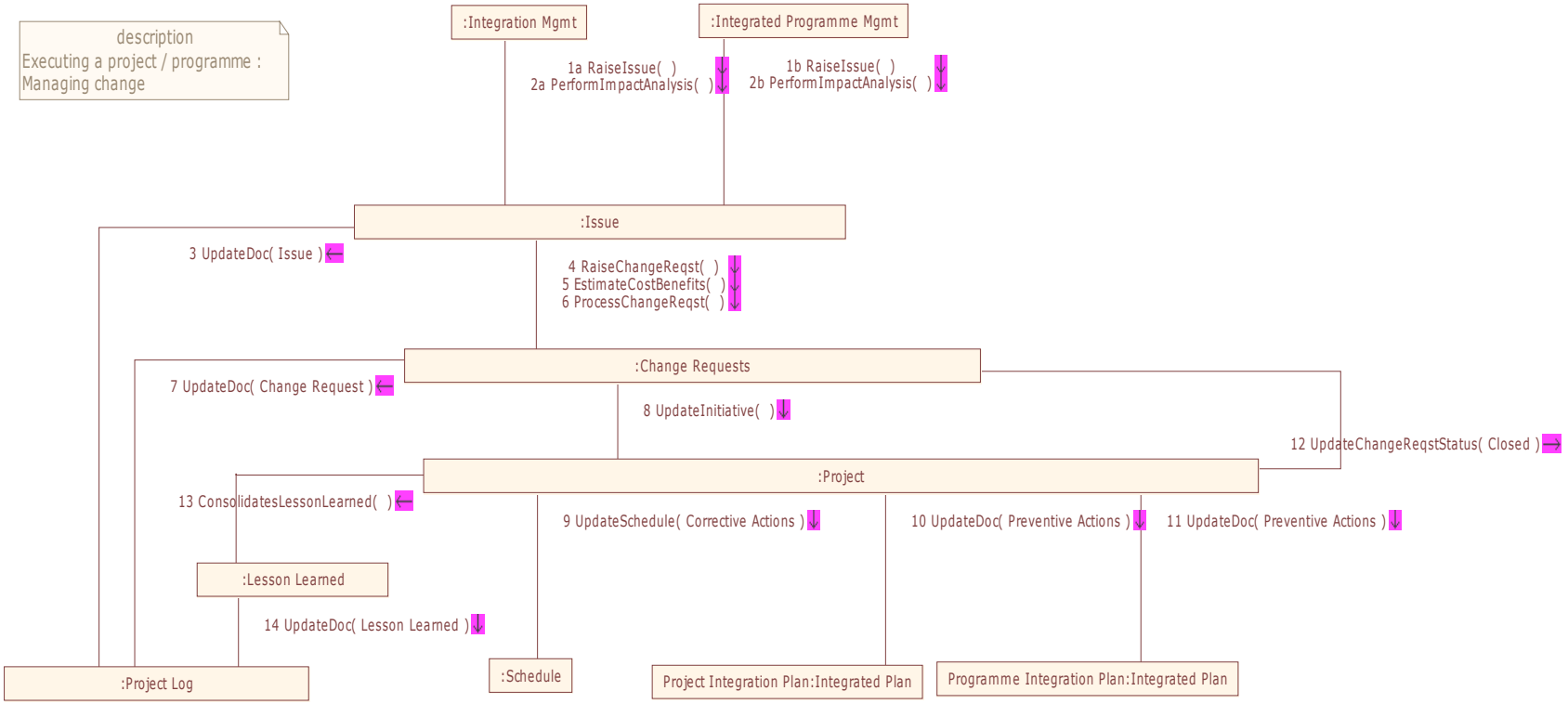


Communication Diagram (Planning)

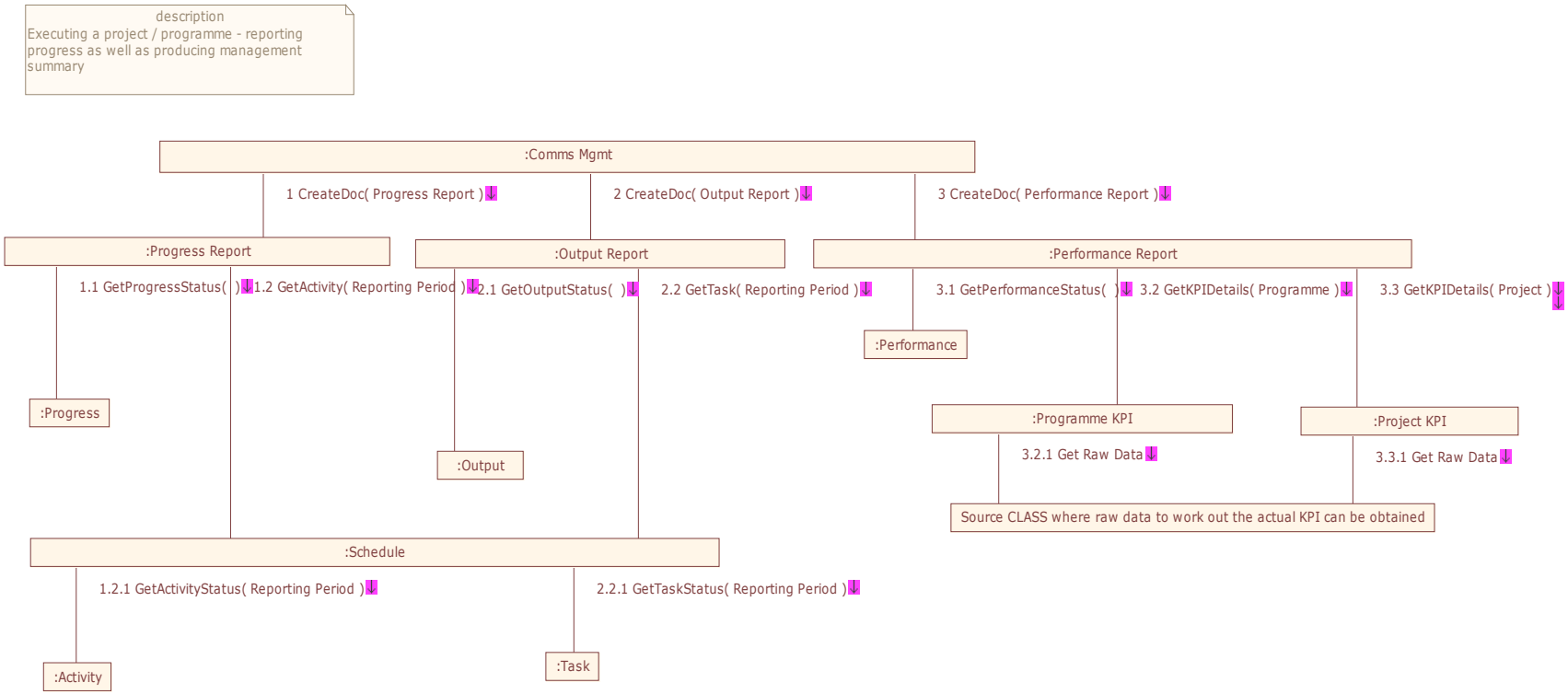
3.26.3 “Execution” state



Communication Diagram (Executing) – Status Update & Monitoring

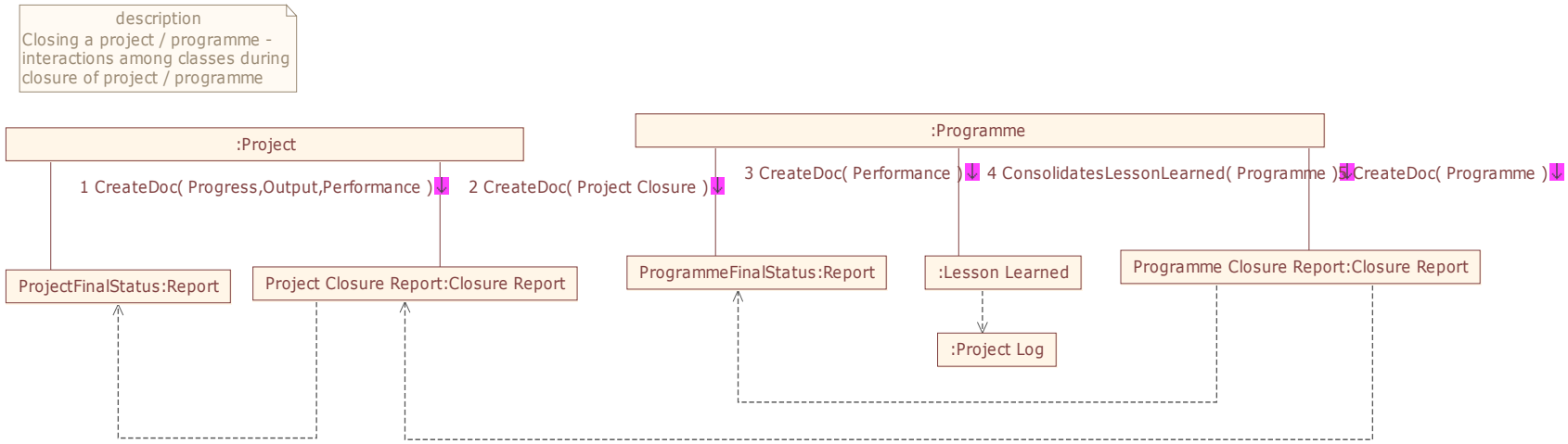


Communication Diagram (Execution) – Change Management



Communication Diagram (Execution) – Reporting

3.26.4 “Closing” state



Communication Diagram (Closing)

CHAPTER 4
USAGE SCENARIO

Similar to Chapter 3, the content of this chapter is organized by *Package* which is defined for each usage scenario, within which the details of each of its *Use Cases* will be described. Two usage scenarios are defined in the domain model and they are:

- *PMIS for Business Projects*
- *BProjM Practice Assessment*

4.1 "PMIS for Business Projects" package

Description :

This Package describes the usage scenario where the UML specifications of the domain model (especially the Communication Diagrams) can be used to define requirements of a PMIS.

Packages :

- *Actors*

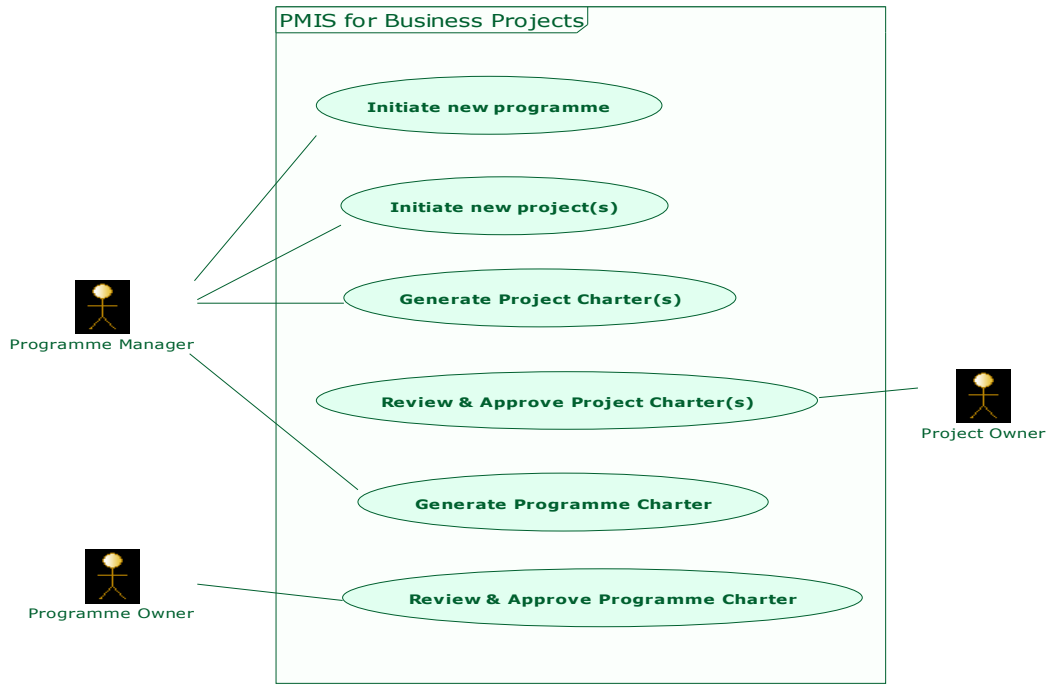
Classes :

- *System Module*

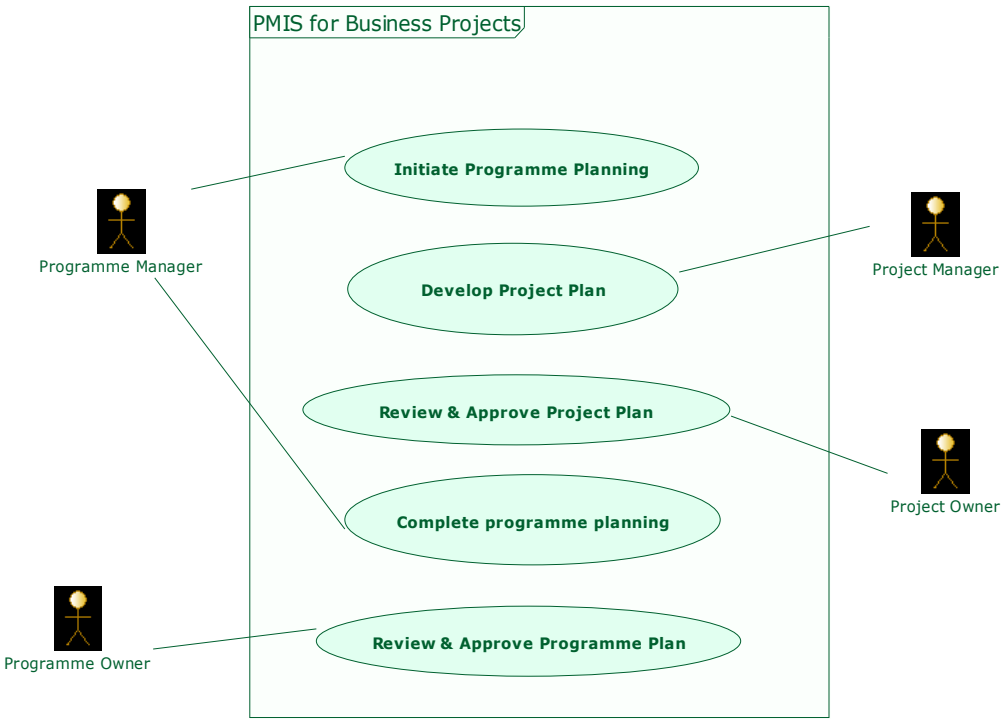
Use cases :

- *Review Project Status*
- *Review Project Performance*
- *Retrieve Work Assignment*
- *Update Work Status*
- *Review Programme Dashboard*
- *Initiate new project(s)*
- *Initiate new programme*
- *Generate Programme Charter*
- *Generate Project Charter(s)*
- *Develop Project Plan*
- *Complete programme planning*
- *Initiate Programme Planning*
- *Review & Approve Project Charter(s)*
- *Review & Approve Programme Charter*
- *Review & Approve Project Plan*
- *Review & Approve Programme Plan*
- *Prepare Project Closure Report*
- *Review & Close Programme*
- *Review & Close Project*
- *Prepare Programme Closure Report*

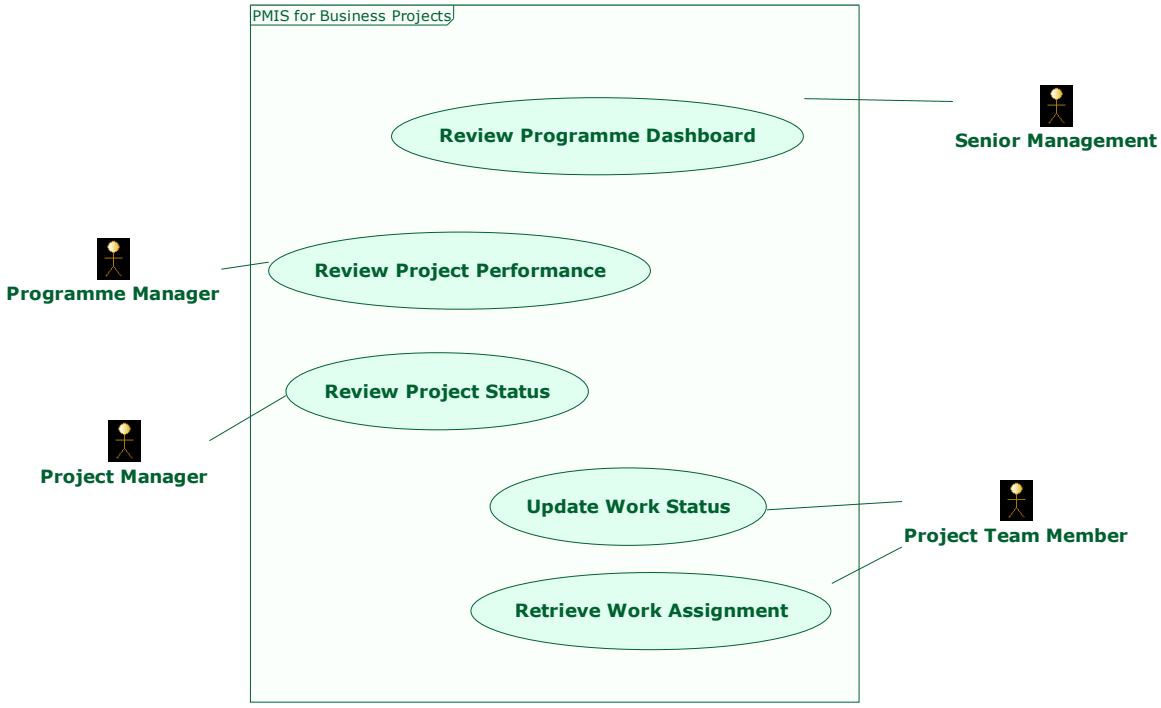
4.1.1 Use cases



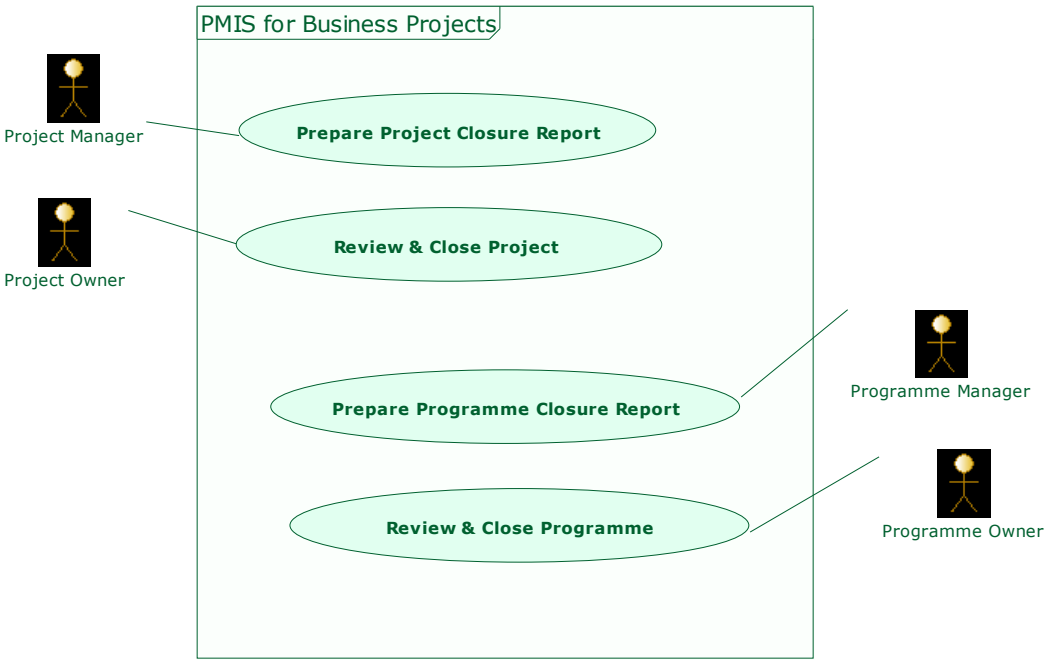
Use case diagram : Initiating state



Use case diagram : Planning state



Use case diagram : Executing state



Use case diagram : Closing state

4.1.1.1 "Review Project Status" use case

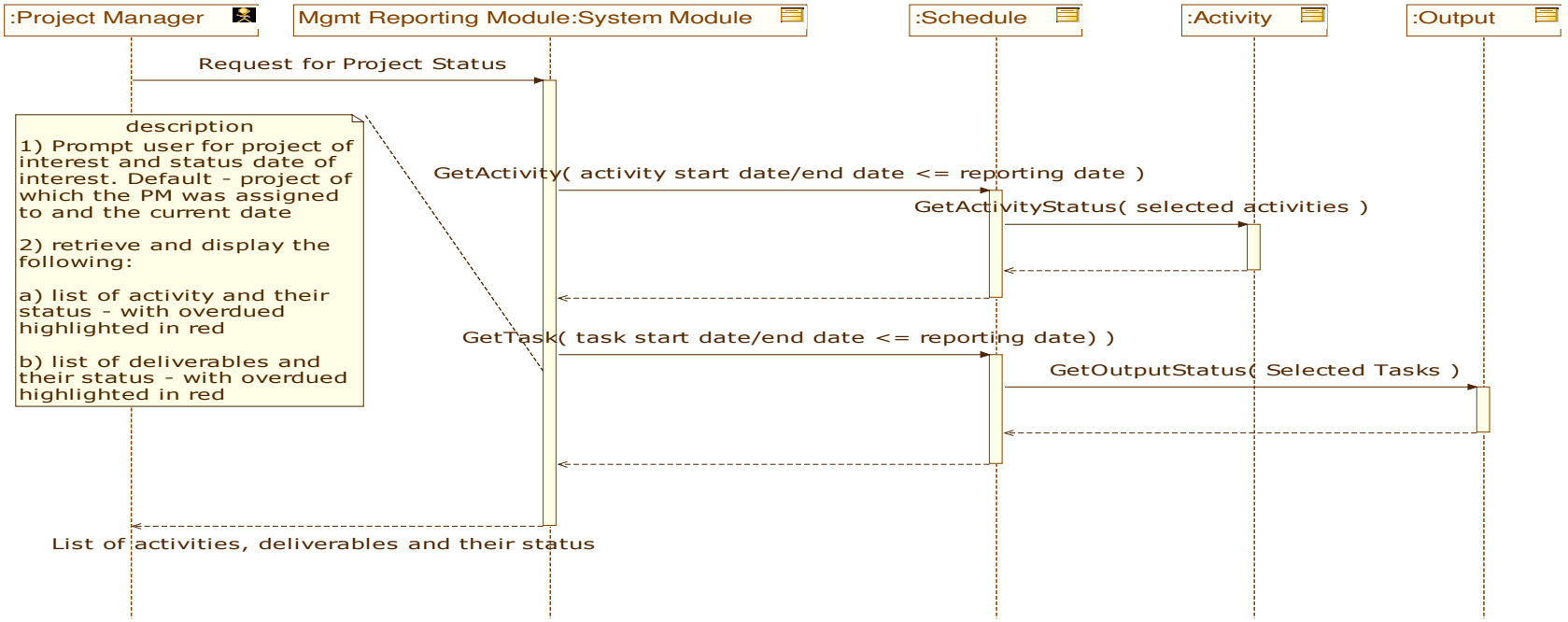
Description

The objective is to review the status of the project, in particular, if there is any slip in timeline for completing activity and in terms of delivering the output

Intervening actors :

- *Project Manager*

Sequence Diagram :



4.1.1.2 "Review Project Performance" use case

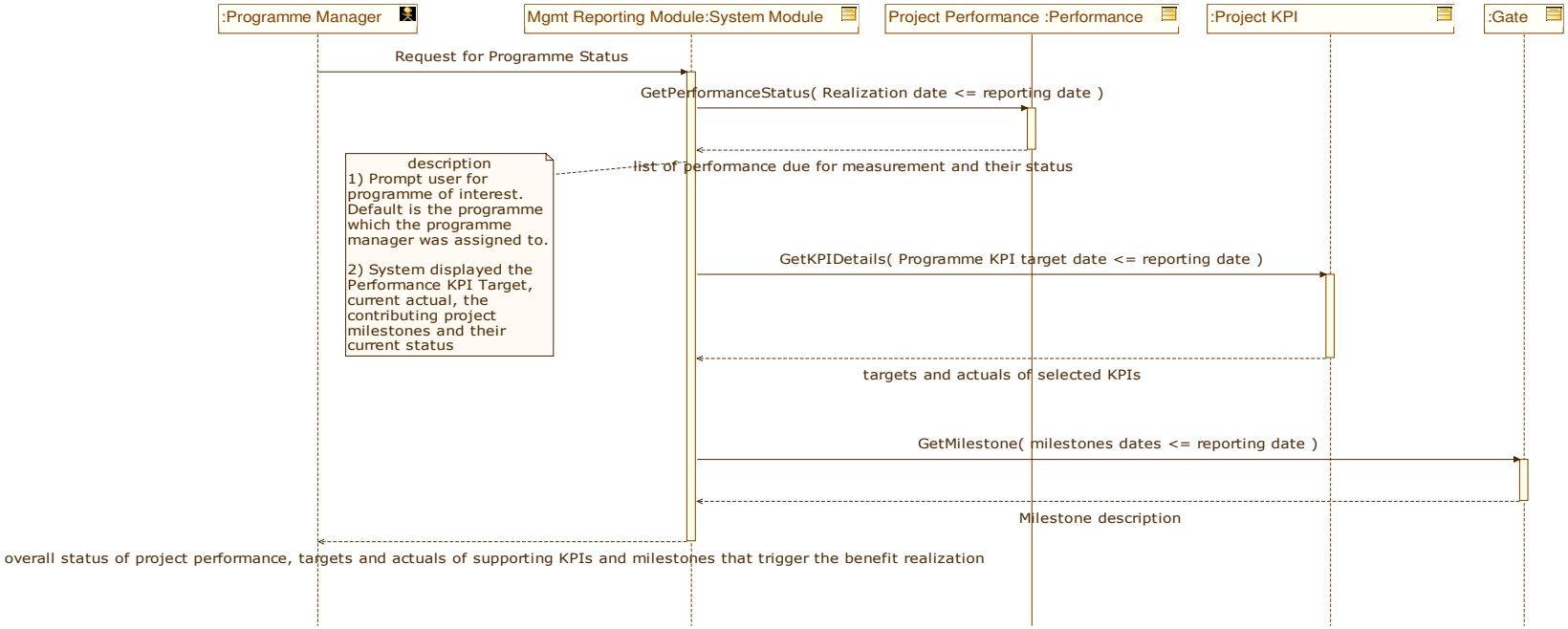
Description

The objective is to be able to review the status of the projects under the programme, in particular the performance KPIs

Intervening actors :

- *Programme Manager*

Sequence Diagram :



4.1.1.3 "Retrieve Work Assignment" use case

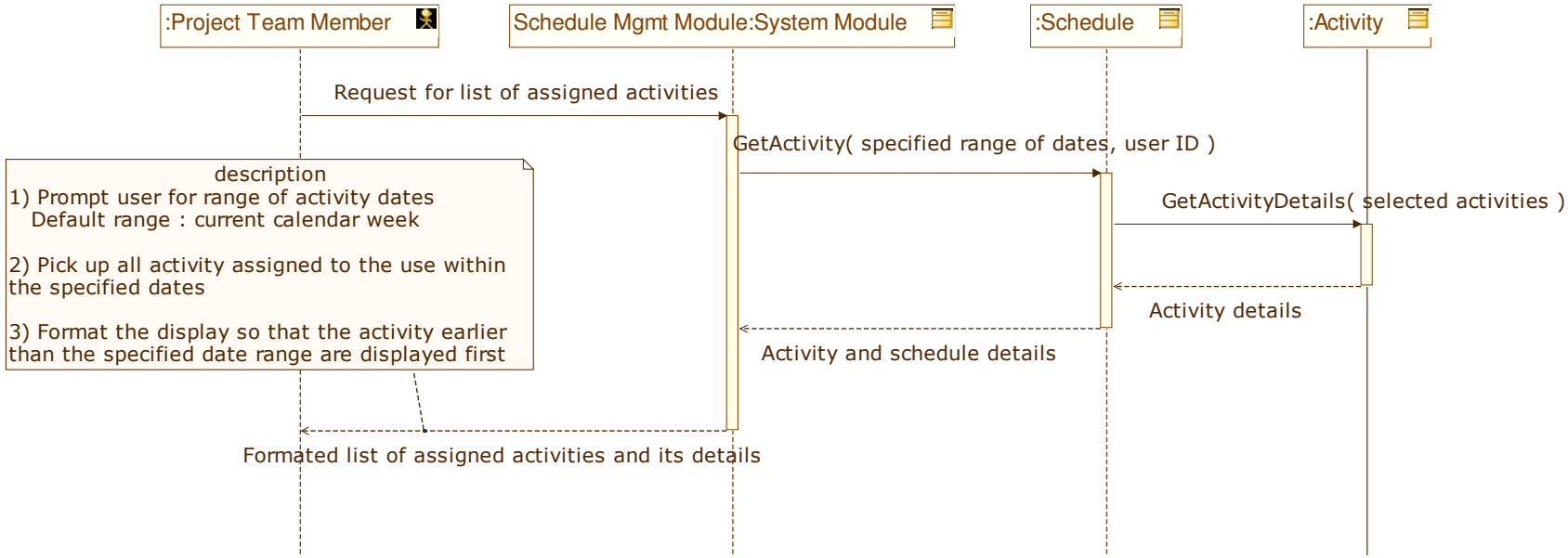
Description

The objective is to obtain a list of activity that must be completed within a specified period of time to help plan the work for the week

Intervening actors :

- *Project Team Member*

Sequence Diagram :



4.1.1.4 "Update Work Status" use case

Description

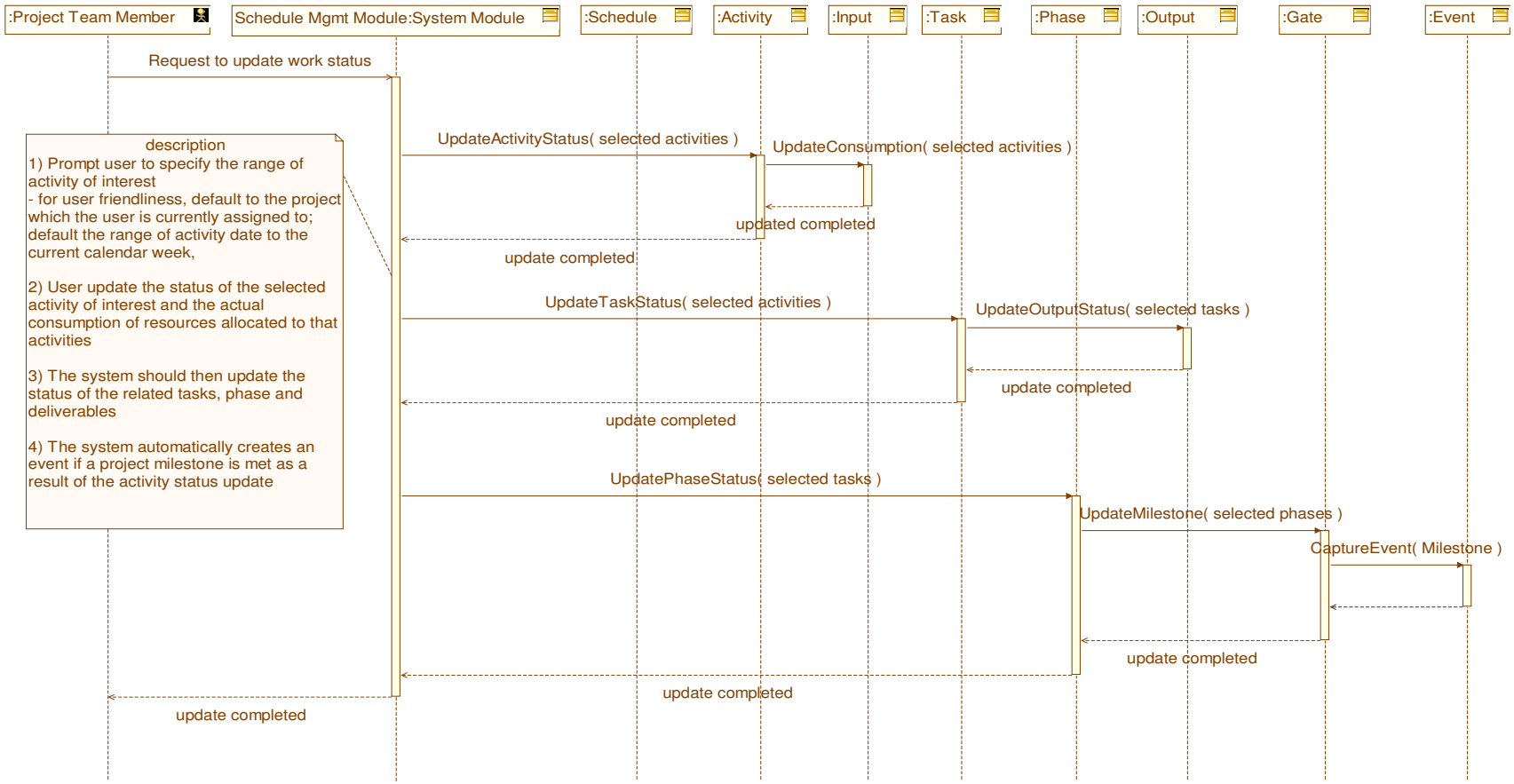
The objective is to capture the status of the activity assigned to individual in order to facilitate project reporting

Intervening actors :

- *Project Team Member*

Please refer to the next page for the Sequence Diagram.

Sequence Diagram :



4.1.1.5 "Review Programme Dashboard" use case

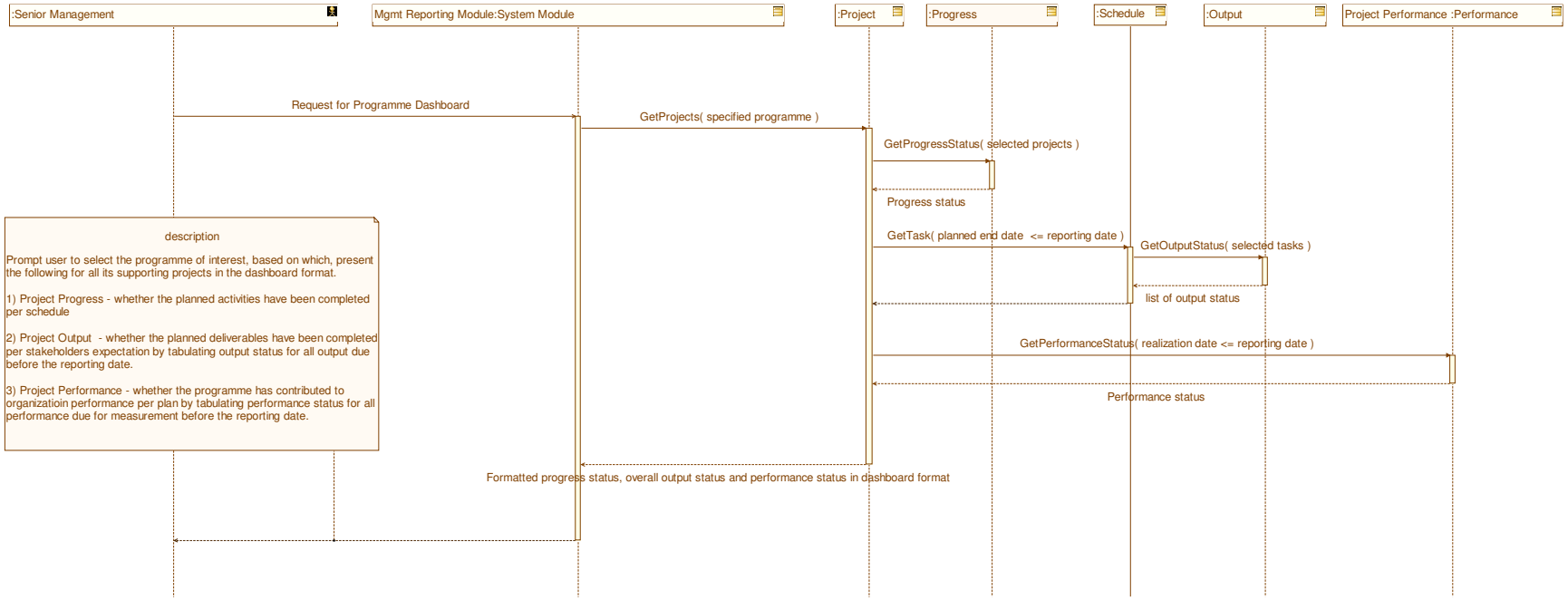
Description

The objective is to obtain a status summary of all the on-going initiatives

Intervening actors :

- *Senior Management*

Sequence Diagram :



4.1.1.6 "Initiate new project(s)" use case

Description

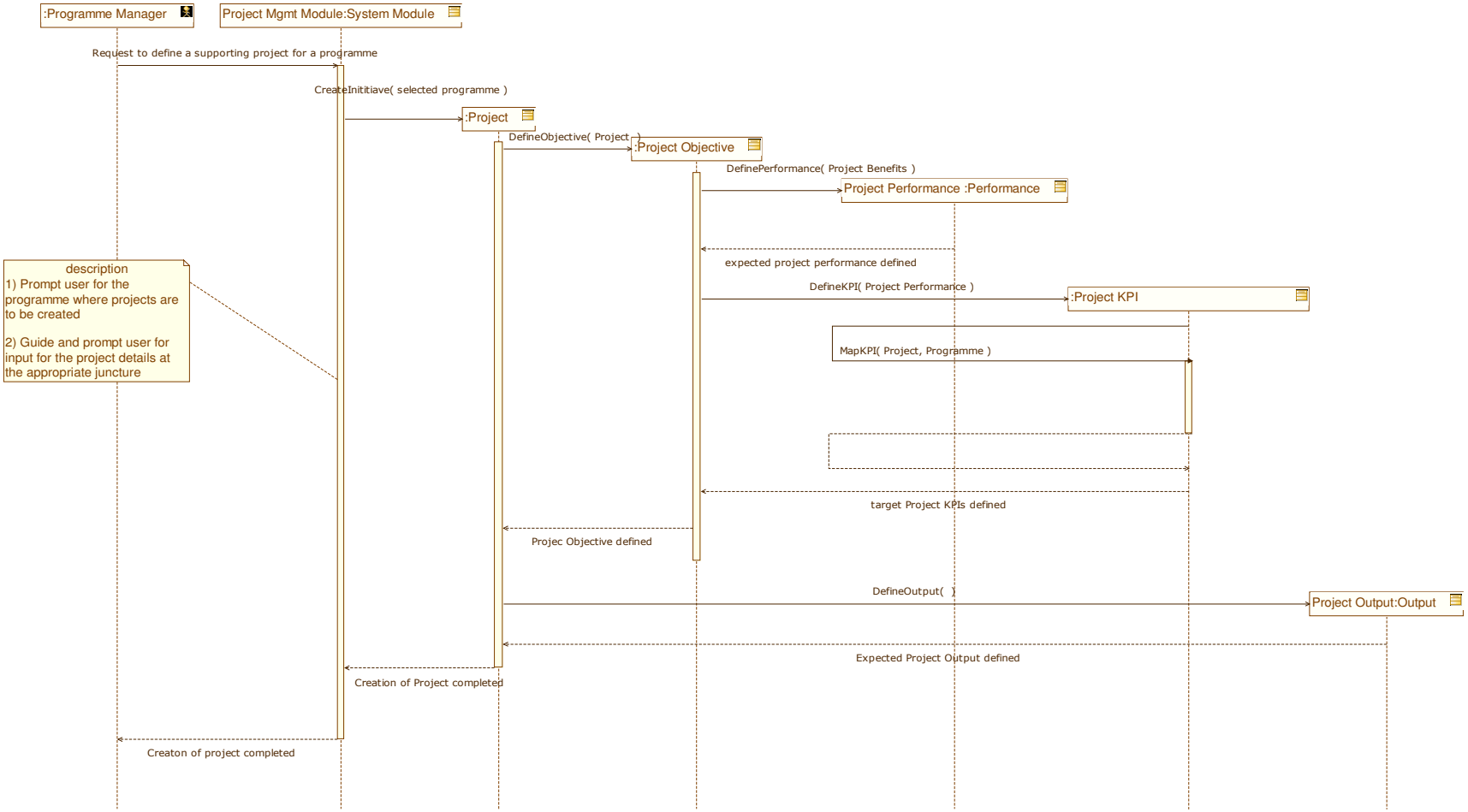
The objective is to capture the objectives and performance KPIs of the underlying projects which contribute towards the newly initiated programme

Intervening actors :

- *Project Manager*
- *Programme Manager*

Please refer to the next page for the Sequence Diagram.

Sequence Diagram :



4.1.1.7 "Initiate new programme" use case

Description

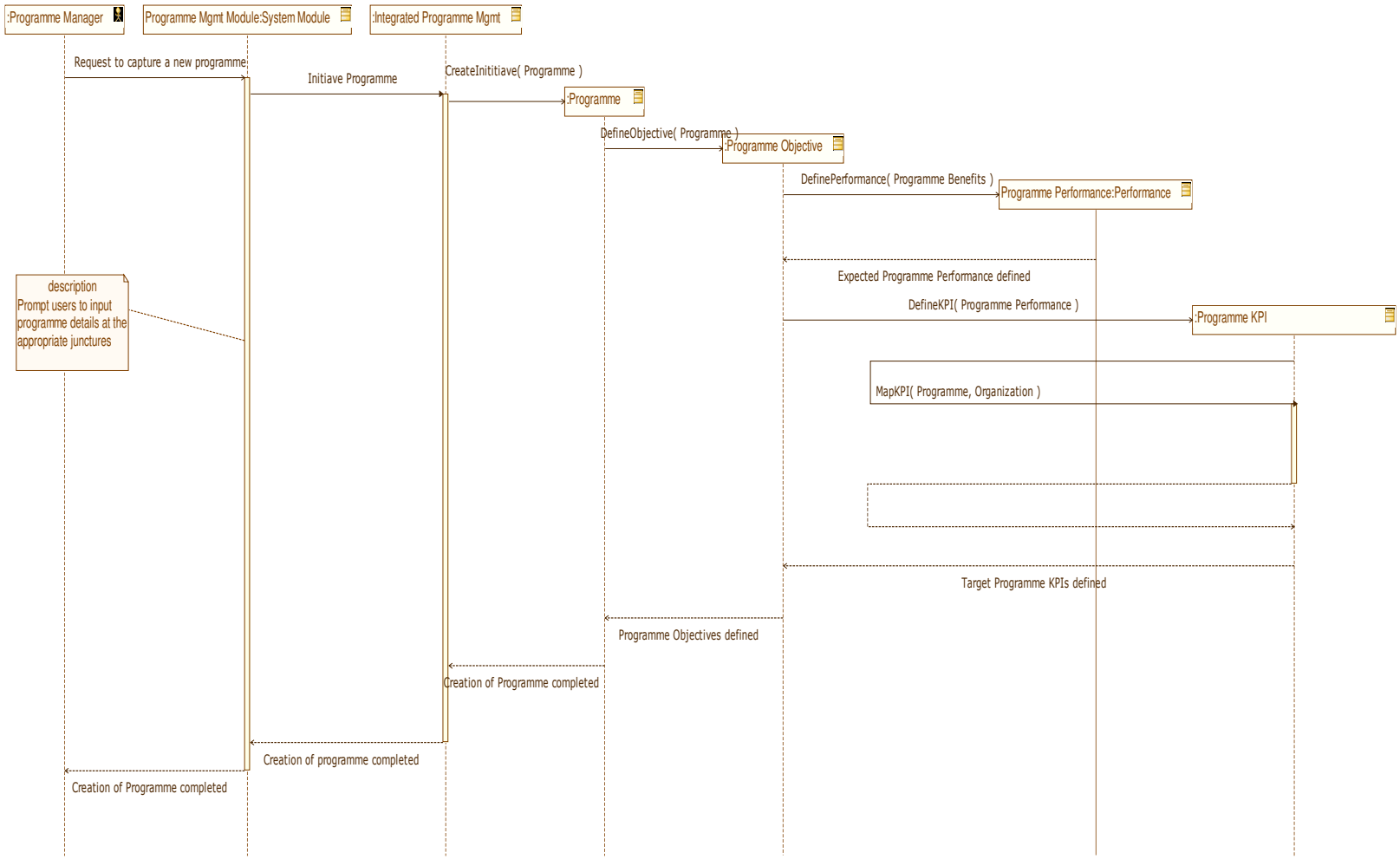
The objective is to capture the objectives and performance KPIs of the new programme, based on which contributing projects will be defined

Intervening actors :

- *Project Manager*
- *Programme Manager*

Please refer to the next page for the Sequence Diagram.

Sequence Diagram :



4.1.1.8 "Generate Programme Charter" use case

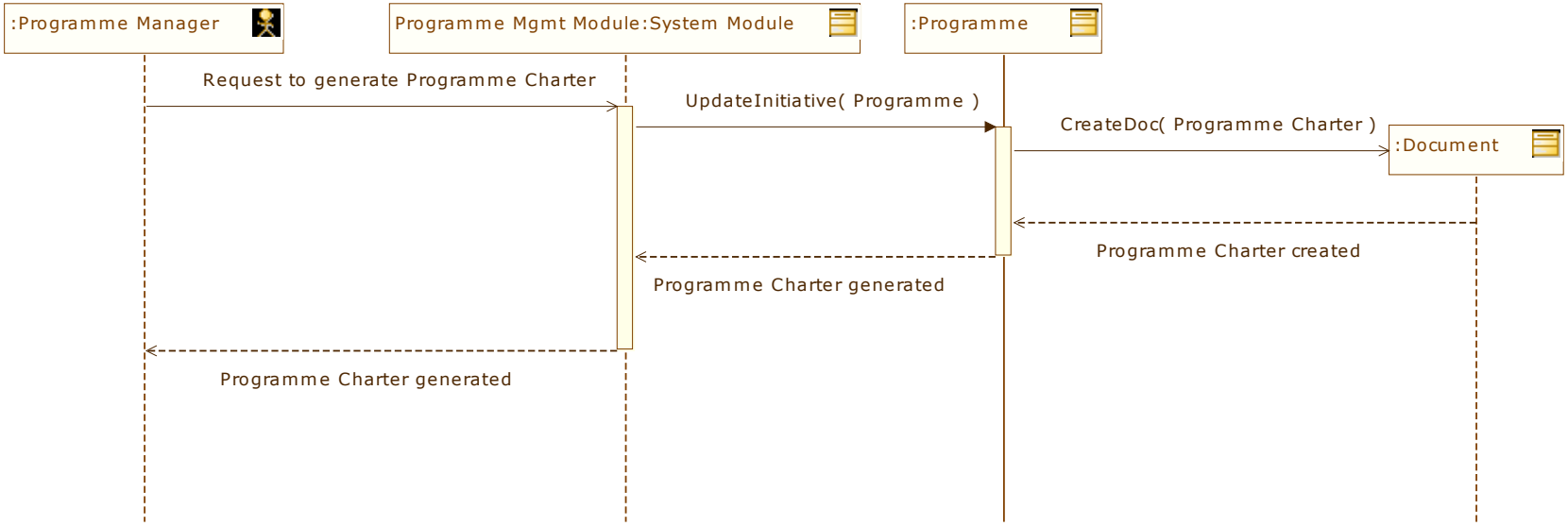
Description

The objective is to generate the consolidated programme charter comprising terms of references of both the new programme and its contributing new projects, for approval by the programme owner

Intervening actors :

- *Programme Manager*

Sequence Diagram :



4.1.1.9 "Generate Project Charter(s)" use case

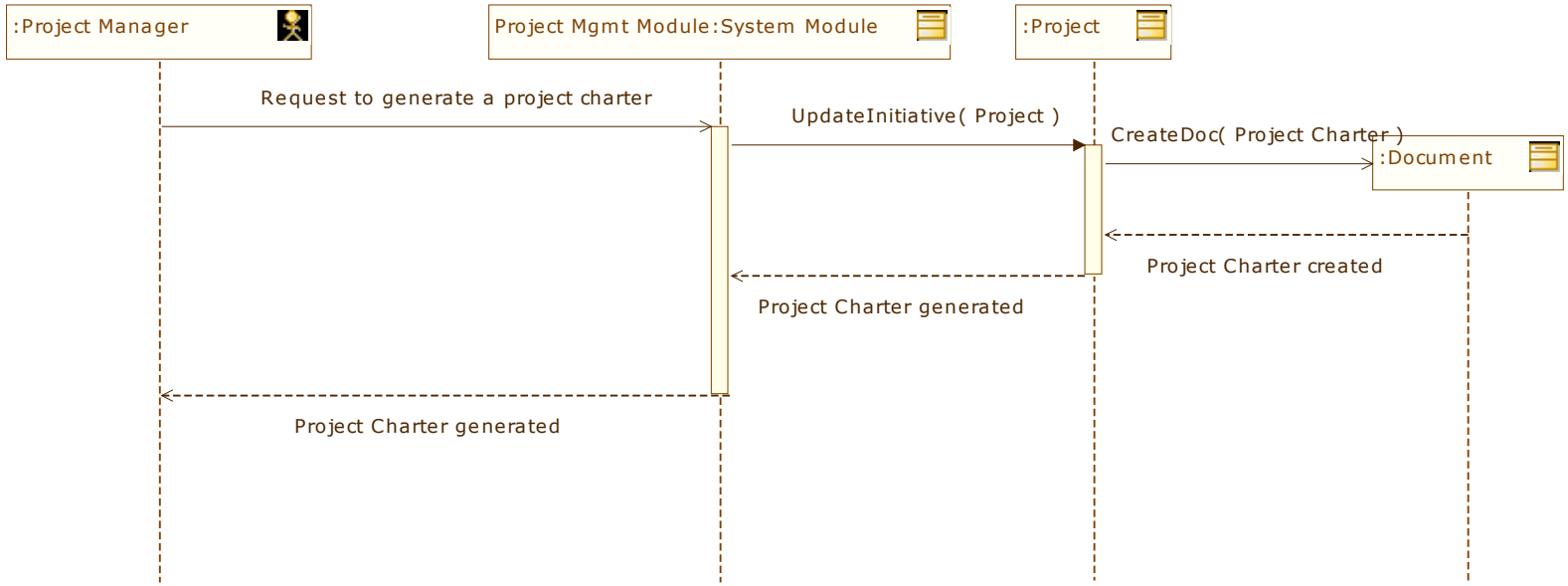
Description

The objective is to auto-generate the terms of references of each project which have already been captured by the PMIS for approval by the project owner

Intervening actors :

- *Project Manager*
- *Programme Manager*

Sequence Diagram :



4.1.1.10 "Develop Project Plan" use case

Description

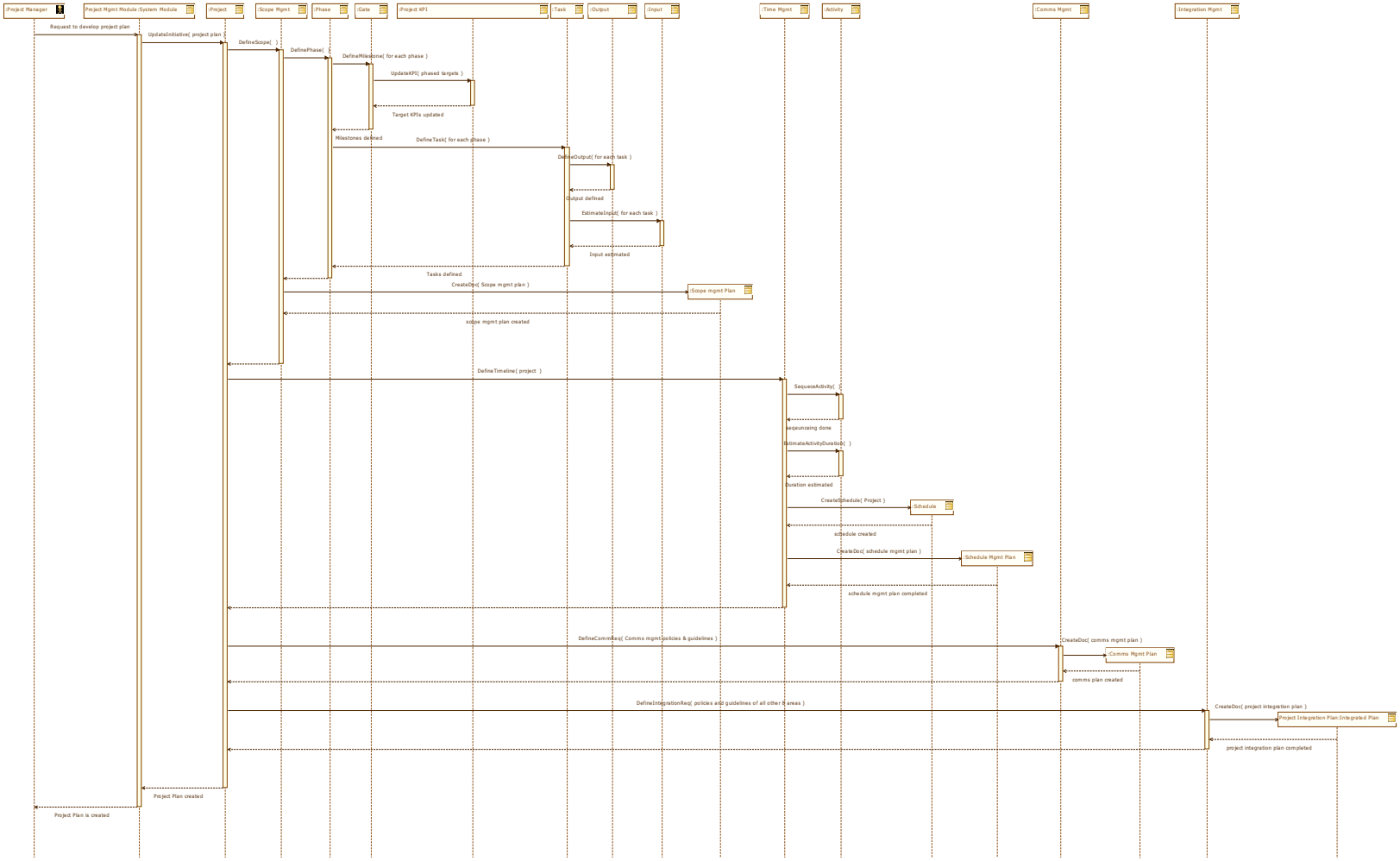
The objective is to support detailed project planning within the parameters as defined by the programme

Intervening actors :

- *Project Manager*

Please refer to the next page for the Sequence Diagram.

Sequence Diagram :



4.1.1.11 "Complete programme planning" use case

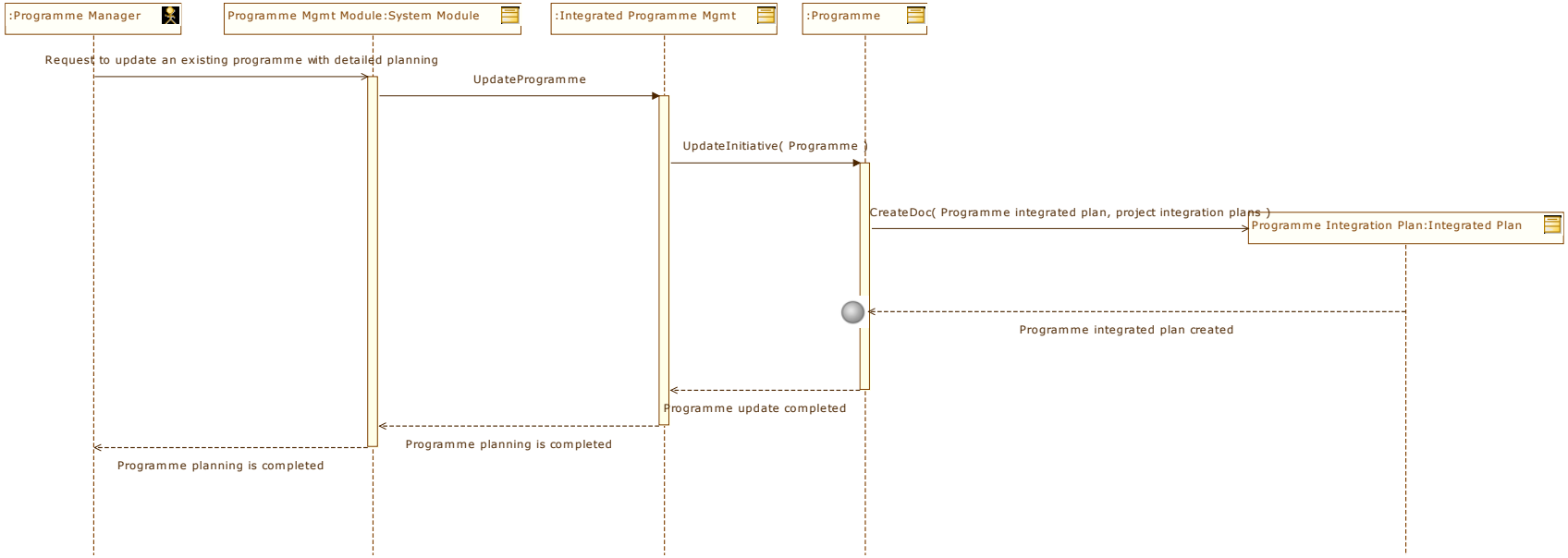
Description

The objective is to initiate the programme planning where the underlying projects are defined in relations with each other in terms of timing, deliverables, integration points etc.

Intervening actors :

- *Programme Manager*

Sequence Diagram :



4.1.1.12 "Initiate Programme Planning" use case

Description

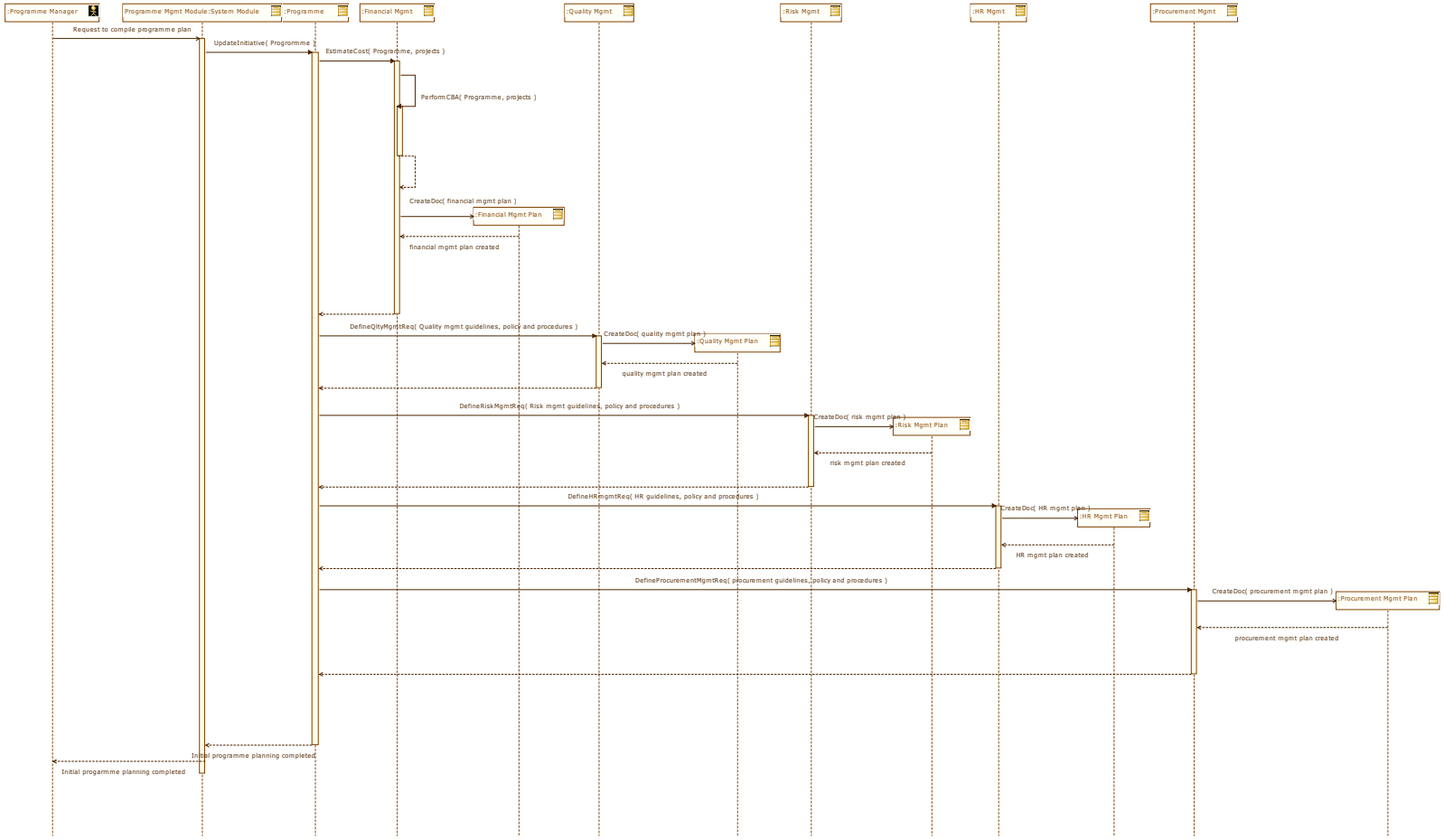
The objective is for the programme manager to consolidate the integrated project plan of the contributing projects with the common underlying programme management components namely HR management, Financial management, Procurement management, Quality Management, Risk Management

Intervening actors :

- *Programme Manager*

Please refer to the next page for the Sequence Diagram.

Sequence Diagram :



4.1.1.13 "Review & Approve Project Charter(s)" use case

Description

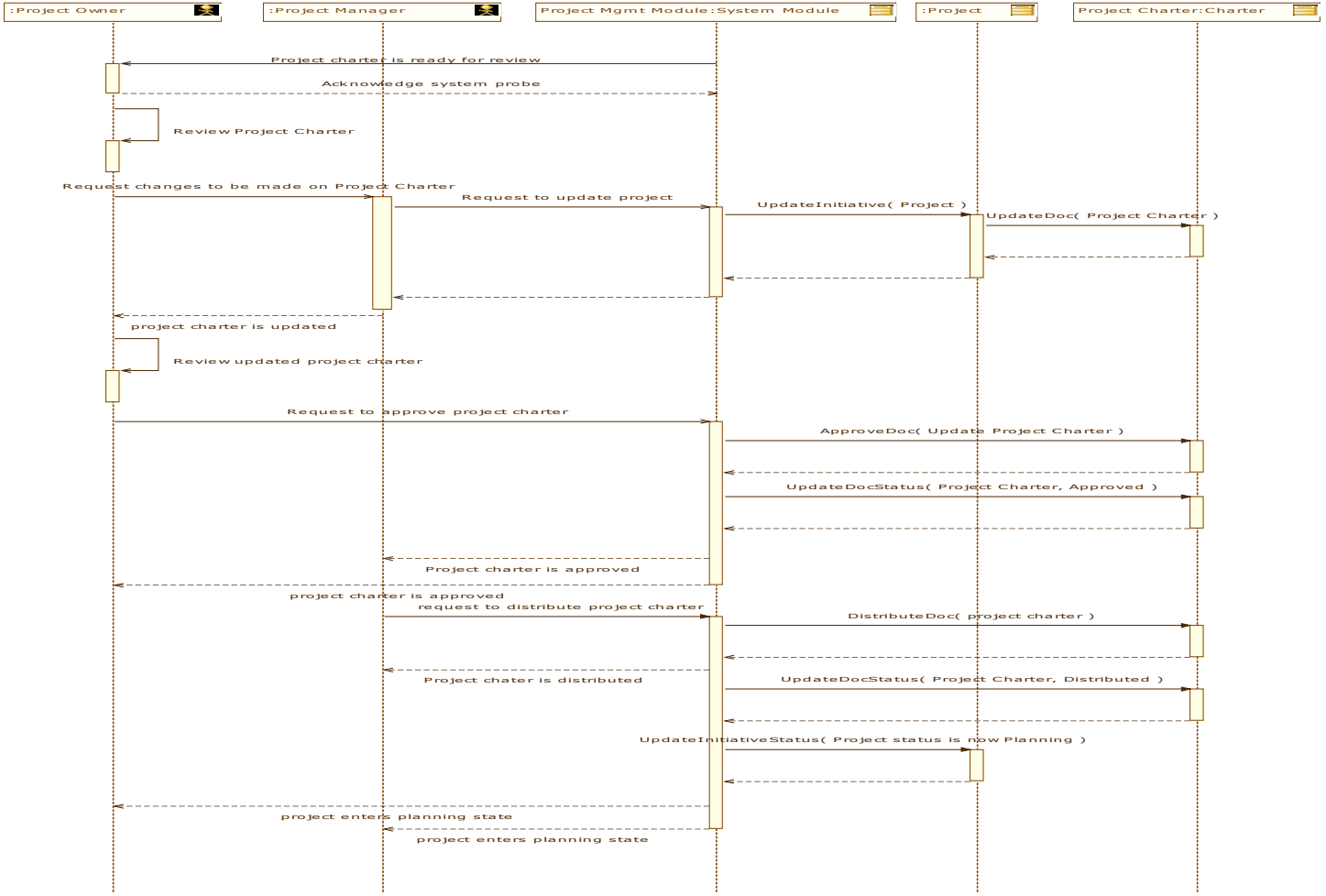
The objective is to allow the project owner to review and approve the project charter

Intervening actors :

- *Project Owner*

Please refer to the next page for the Sequence Diagram.

Sequence Diagram :



4.1.1.14 "Review & Approve Programme Charter" use case

Description

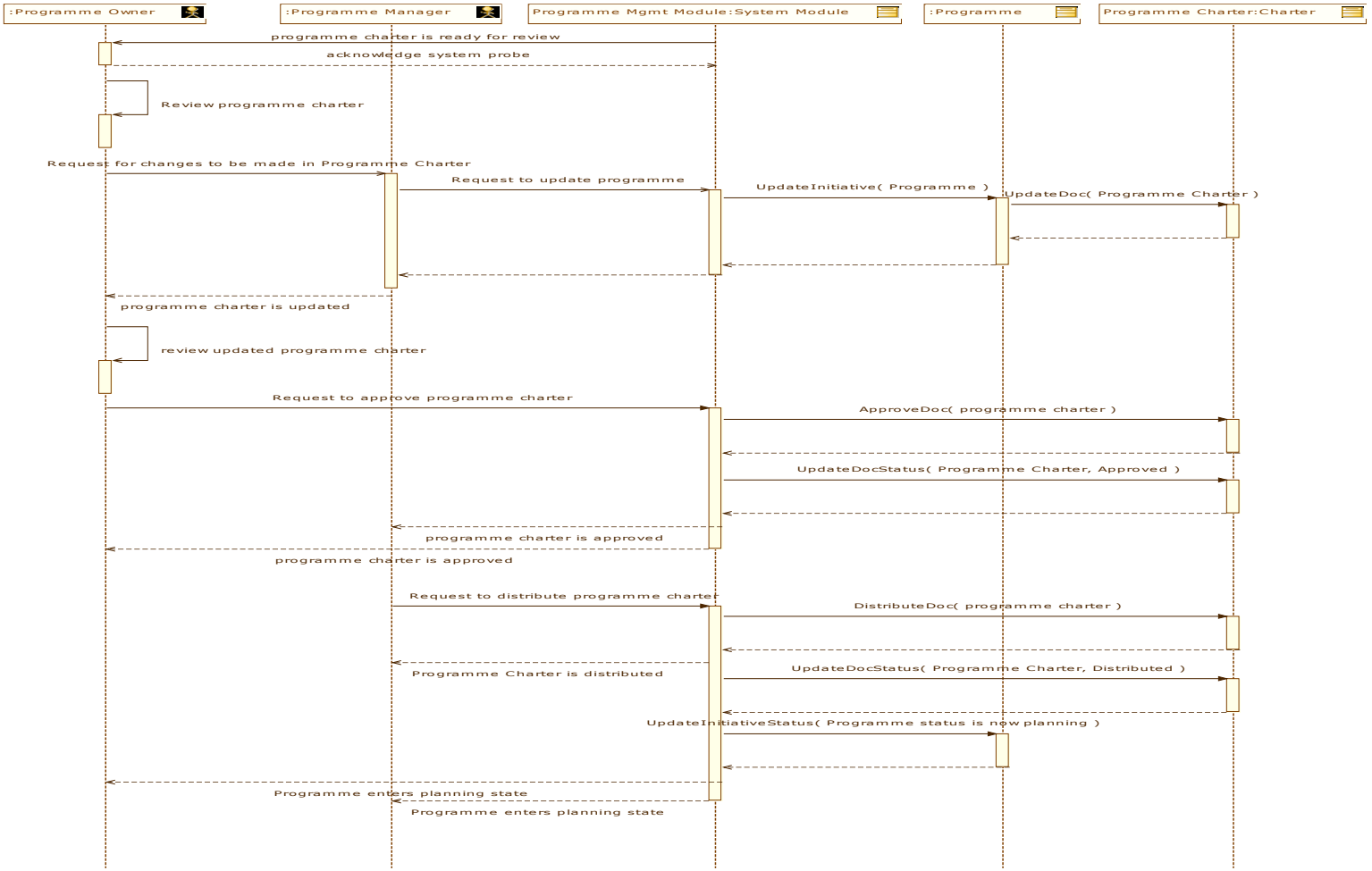
The objective is allow the programme owner to review and approve the programme charter

Intervening actors :

- *Programme Owner*

Please refer to the next page for the Sequence Diagram.

Sequence Diagram :



4.1.1.15 "Review & Approve Project Plan" use case

Description

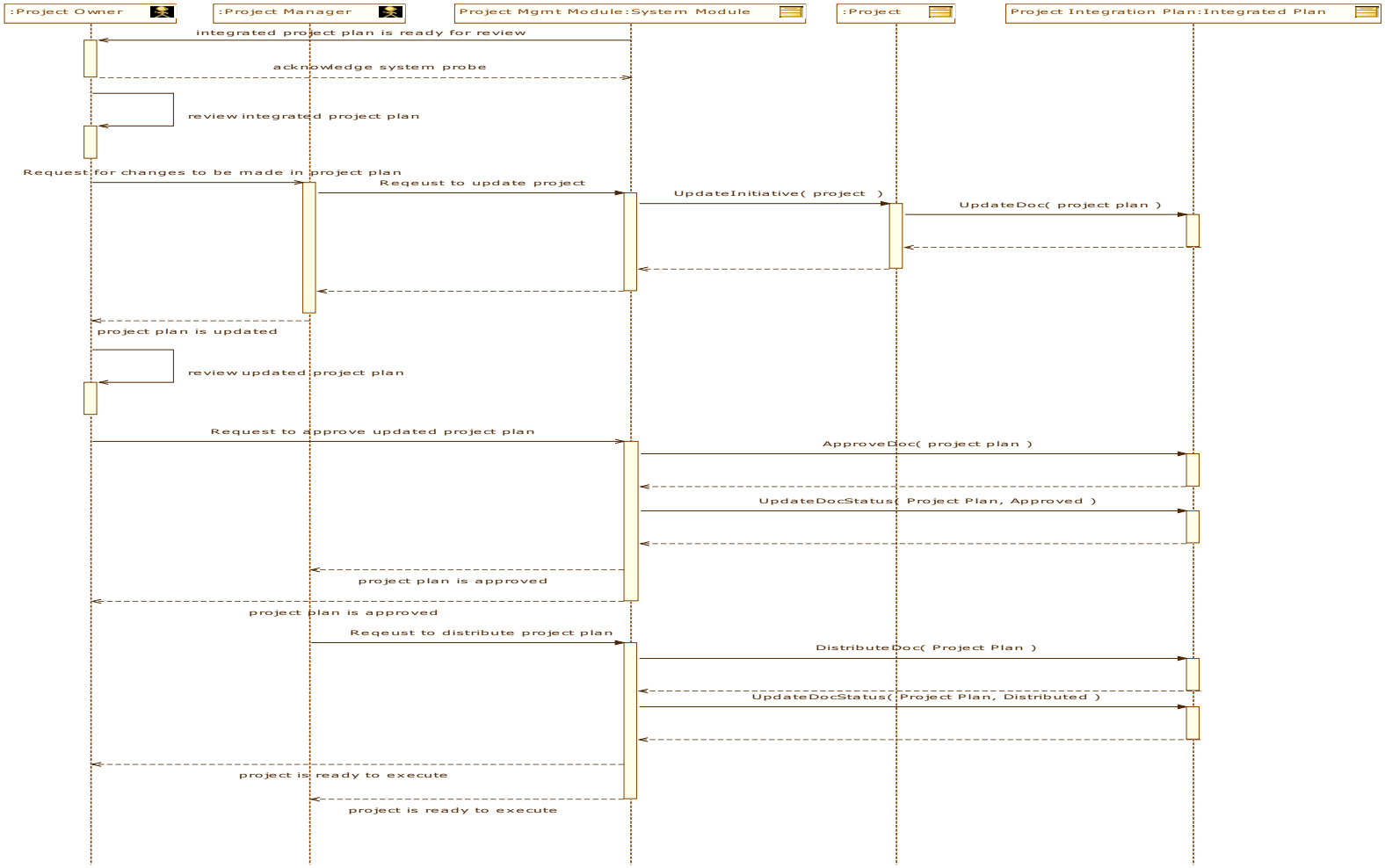
The objective is for the project owner to review and approve the defined project plan

Intervening actors :

- *Project Owner*

Please refer to the next page for the Sequence Diagram.

Sequence Diagram :



4.1.1.16 "Review & Approve Programme Plan" use case

Description

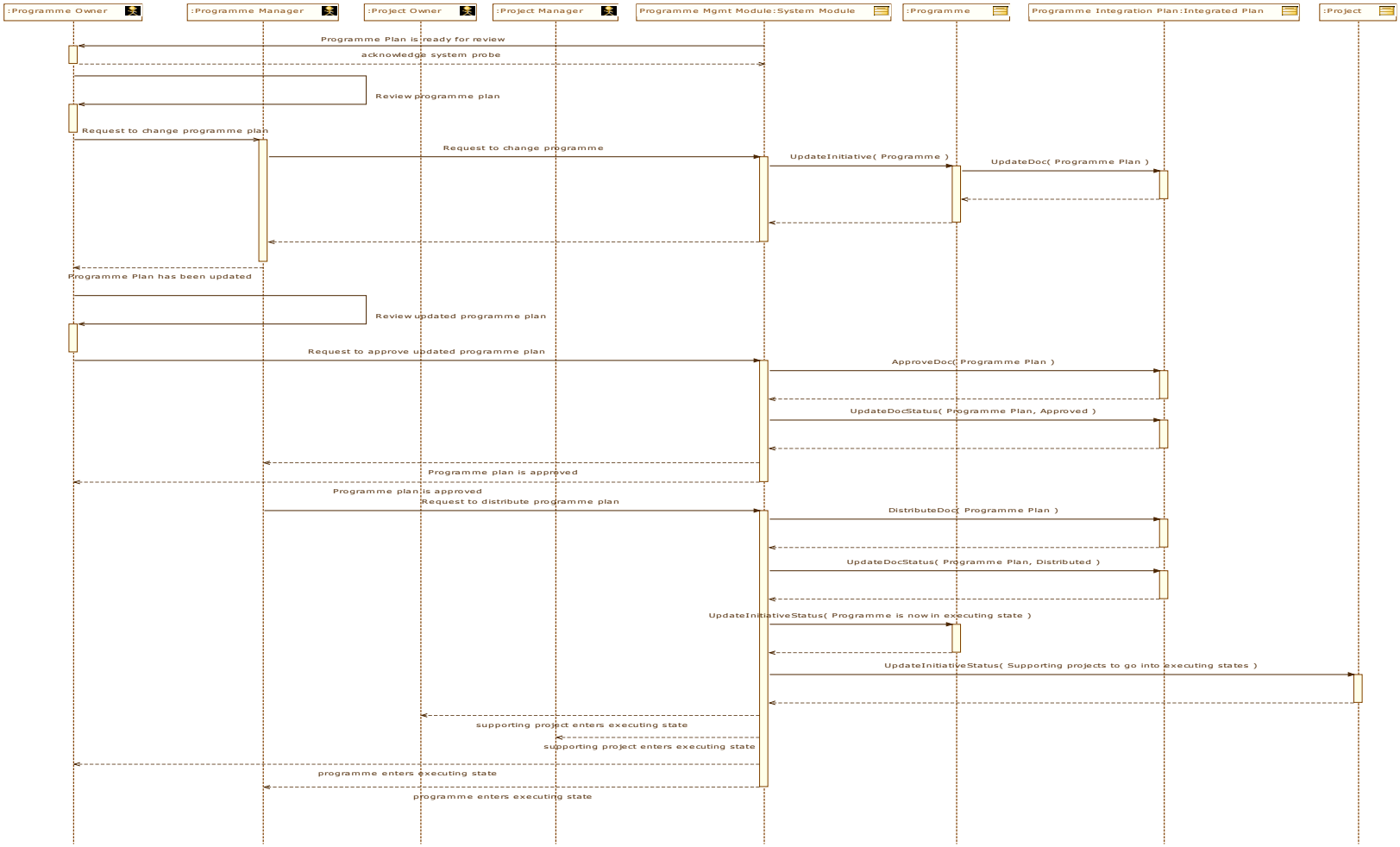
The objective is to for the programme owner to review and approve the defined programme plan.

Intervening actors :

- *Programme Owner*

Please refer to the next page for the Sequence Diagram.

Sequence Diagram :



4.1.1.17 "Prepare Project Closure Report" use case

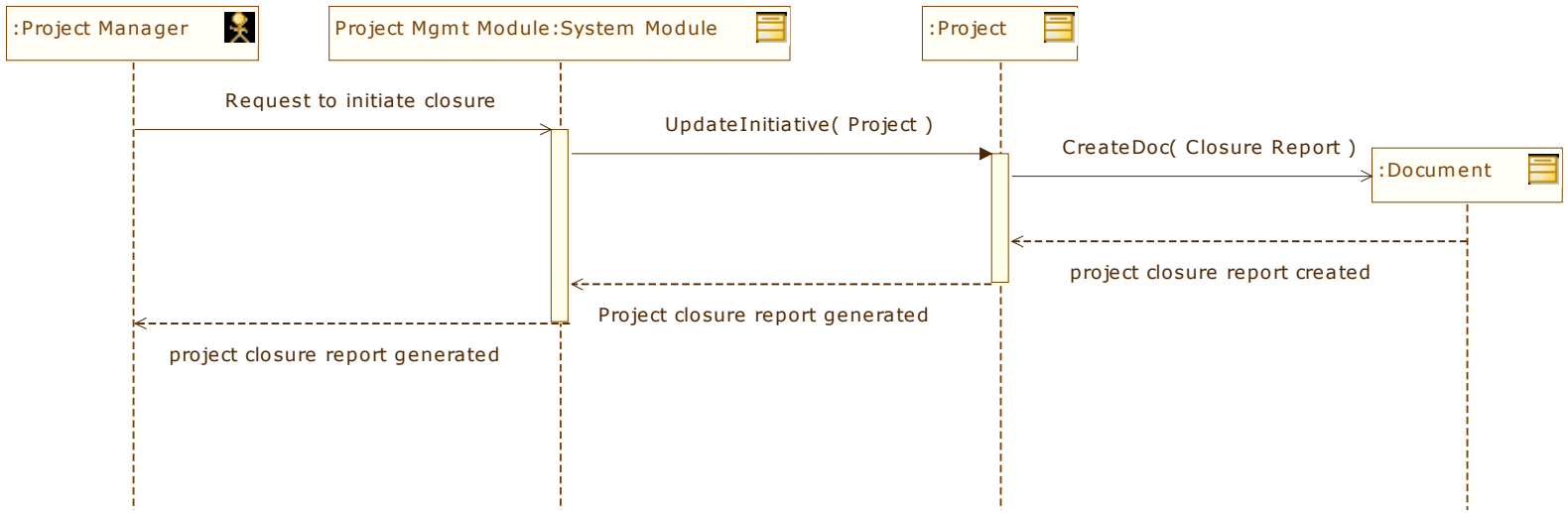
Description

The objective is to obtain the latest status and performance of the project and prepare the closure report.

Intervening actors :

- *Project Manager*

Sequence Diagram :



4.1.1.18 "Review & Close Programme" use case

Description

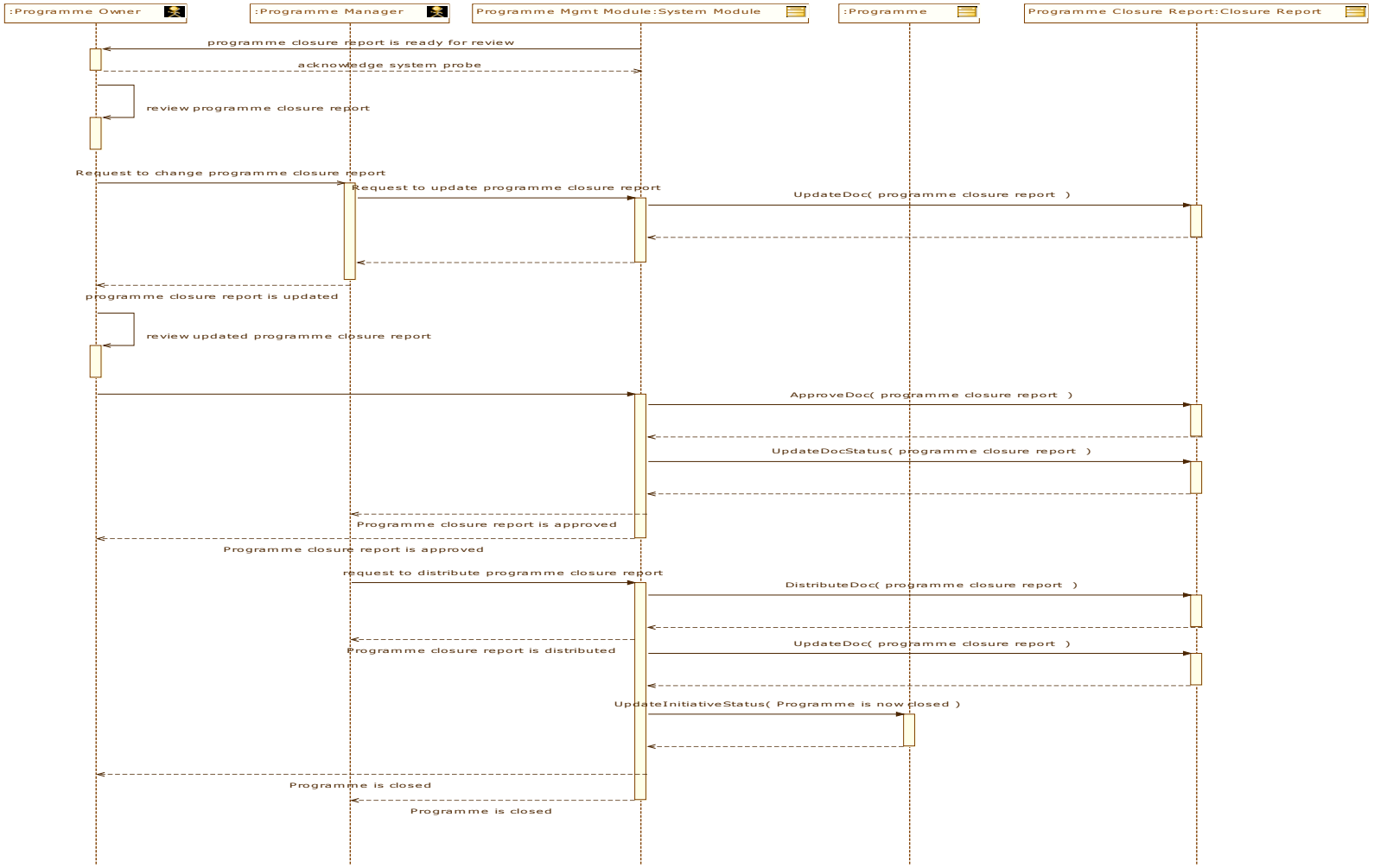
The objective is to review the programme closure report and close the programme.

Intervening actors :

- *Programme Owner*

Please refer to the next page for the Sequence Diagram.

Sequence Diagram :



4.1.1.19 "Review & Close Project " use case

Description

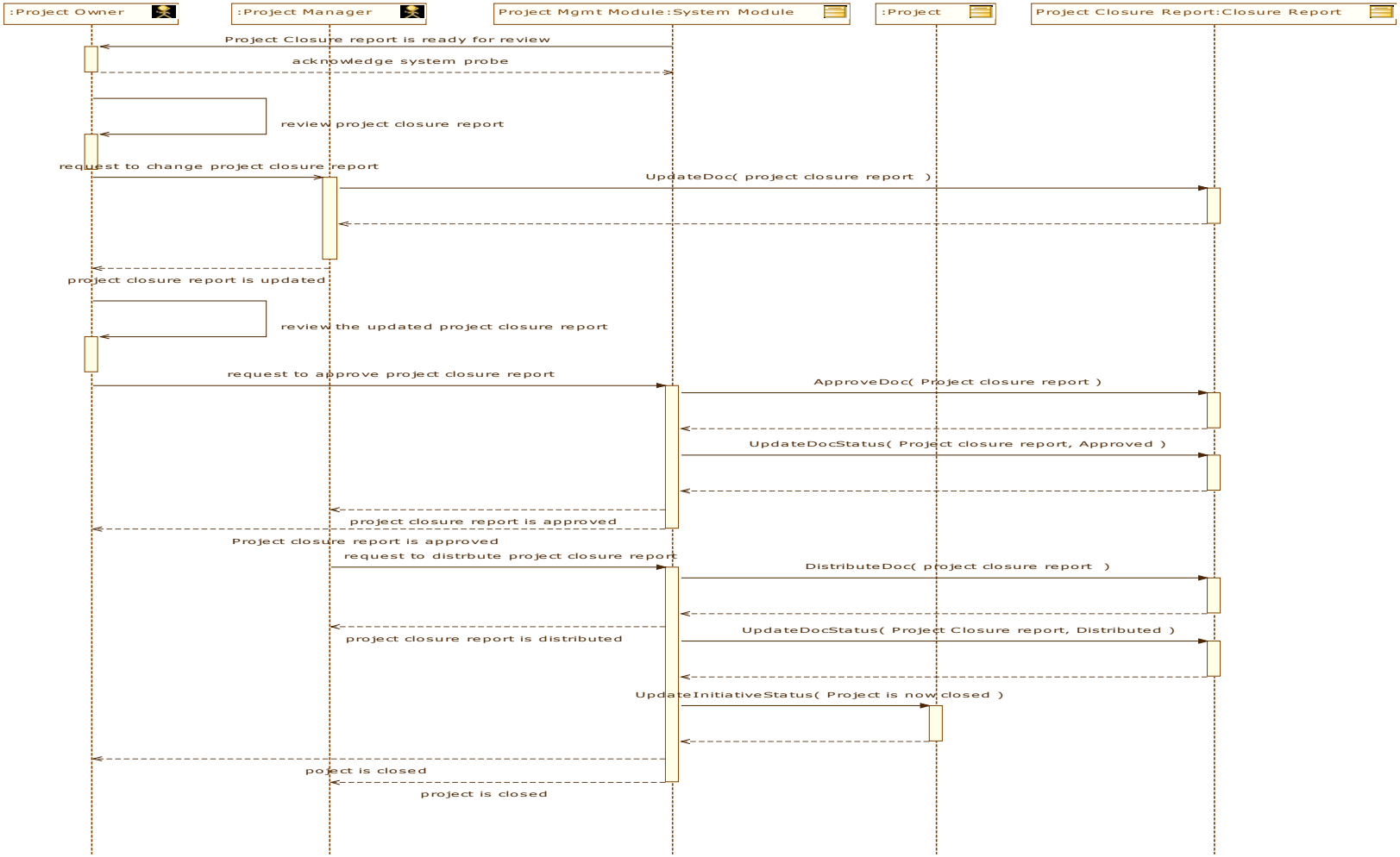
The objective is to review the project closure report and close the project.

Intervening actors :

- *Project Owner*

Please refer to the next page for the Sequence Diagram.

Sequence Diagram :



4.1.1.20 "Prepare Programme Closure Report" use case

Description

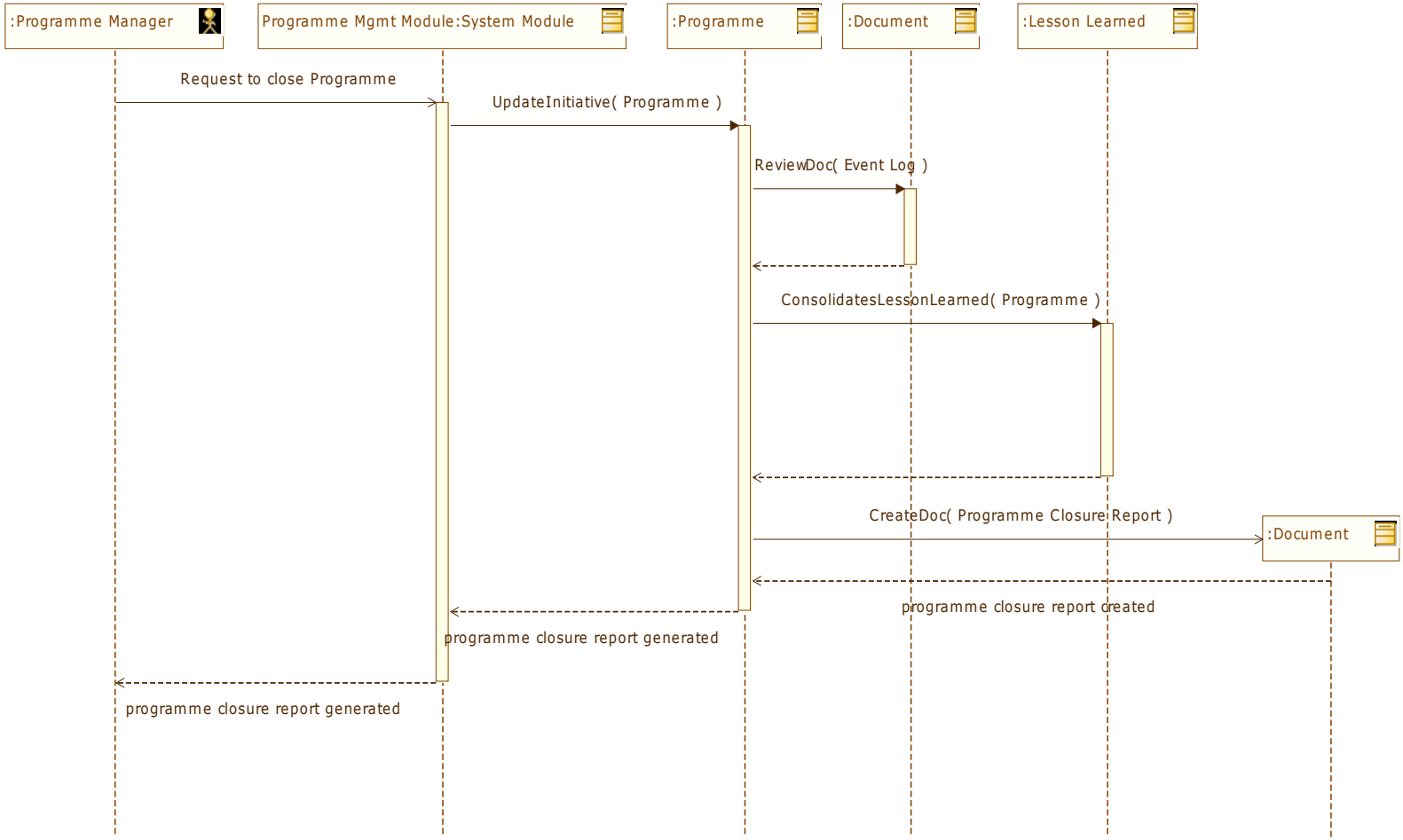
The objective is to obtain latest status and performance of the programme and prepare the closure report.

Intervening actors :

- *Programme Manager*

Please refer to the next page for the Sequence Diagram.

Sequence Diagram :



4.1.2 "Actors" package

Description :

This Package defines the end users of the PMIS.

Actors :

- *Programme Manager*
- *Project Manager*
- *Project Team Member*
- *Senior Management*
- *Project Owner*
- *Programme Owner*

4.1.2.1 "Programme Manager" actor

Cooperating use cases :

- *Review Project Performance*
- *Initiate new programme*
- *Initiate new project(s)*
- *Generate Programme Charter*
- *Complete programme planning*
- *Initiate Programme Planning*
- *Prepare Programme Closure Report*
- *Generate Project Charter(s)*

4.1.2.2 "Project Manager" actor

Cooperating use cases :

- *Review Project Status*
- *Initiate new programme*
- *Initiate new project(s)*
- *Generate Project Charter(s)*
- *Develop Project Plan*
- *Prepare Project Closure Report*

4.1.2.3 "Project Team Member" actor

Cooperating use cases :

- *Retrieve Work Assignment*
- *Update Work Status*

4.1.2.4 "Senior Management" actor

Cooperating use cases :

- *Review Programme Dashboard*

4.1.2.5 "Project Owner" actor

Cooperating use cases :

- *Review & Approve Project Charter(s)*
- *Review & Approve Project Plan*
- *Review & Close Project*

4.1.2.6 "Programme Owner" actor

Cooperating use cases :

- *Review & Approve Programme Charter*
- *Review & Approve Programme Plan*
- *Review & Close Programme*

4.2 "BProjM Practice Assessment" package

Description :

This Package describes the usage scenario where the UML specifications of the domain model could be used to identify missing components in a business project management practice.

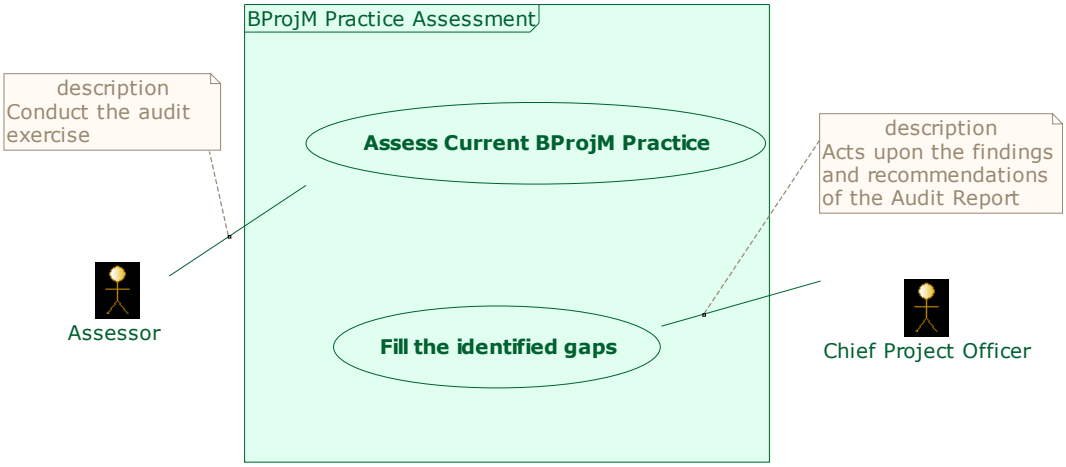
Actors :

- *Assessor*
- *Chief Project Officer*

Use cases :

- *Assess Current BProjM Practice*
- *Fill the identified gaps*

4.2.1 Use cases



Use case diagram : BProjM Practice Assessment

4.2.2 Actors

4.2.2.1 "Assessor " actor

Cooperating use cases :

- *Assess Current BProjM Practice*

4.2.2.2 "Chief Project Officer" actor

Cooperating use cases :

- *Fill the identified gaps*

4.2.2.3 "Assess Current BProjM Practice" use case

Description

The objective is to review the current business project management practice and identify if any essential components are missing. This can be achieved by performing the following:

1. Based on the BProjM Class Diagram, collect information about the current projects and programmes.

2. Instantiate the Class Diagram using the collected information to create an Object Diagram. Establish the links between Objects (as guided by tool such as Objecteering if available).

3. Identify the gaps i.e. the Classes which could not be instantiated and links which could not be established because this information does not exist in the current practice.

4. Prepare an audit report detailing the identified gaps with recommendations on how they can be filled.

Intervening actors :

- *Assessor*

4.2.2.4 "Fill the identified gaps" use case

Description

The objective is to implement the necessary measures to fill the gaps which have been uncovered by the assessment exercise, so that the business projects in the future will stand a higher chance of success.

Intervening actors :

- *Chief Project Officer*

4.2.3 Examples

Description :

Two examples of how the domain model can be used as a practice assessment tool are presented below:

1. The object diagram created based on data collected during Co1 case study (EBP programme)

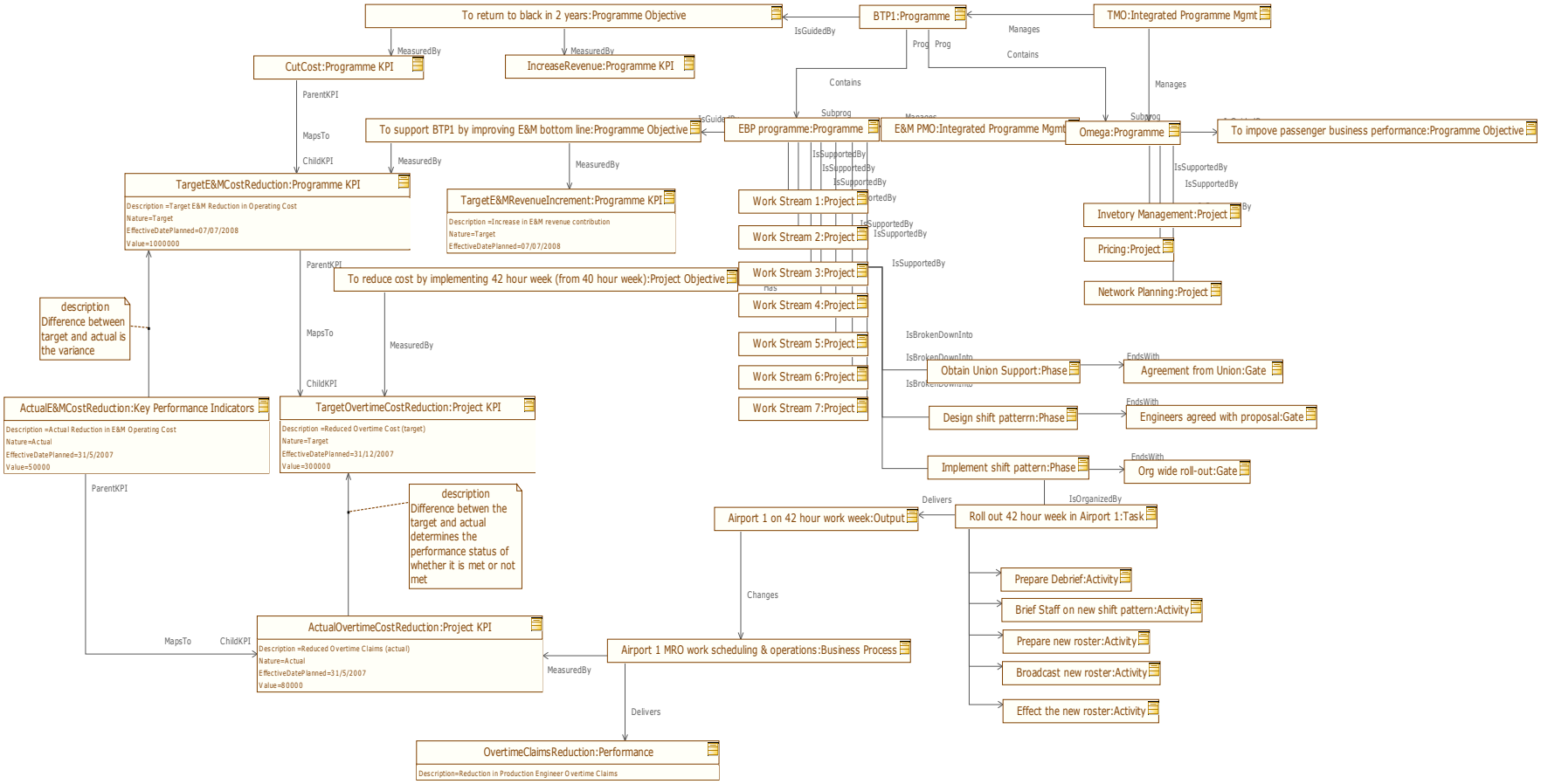
2. The object diagram created based on data collected during Co2 case study

In the case of Co1, no apparent gap was found. In the case of Co2, the identified gaps are the shaded objects and the links which are labelled “non-existing relationship”.

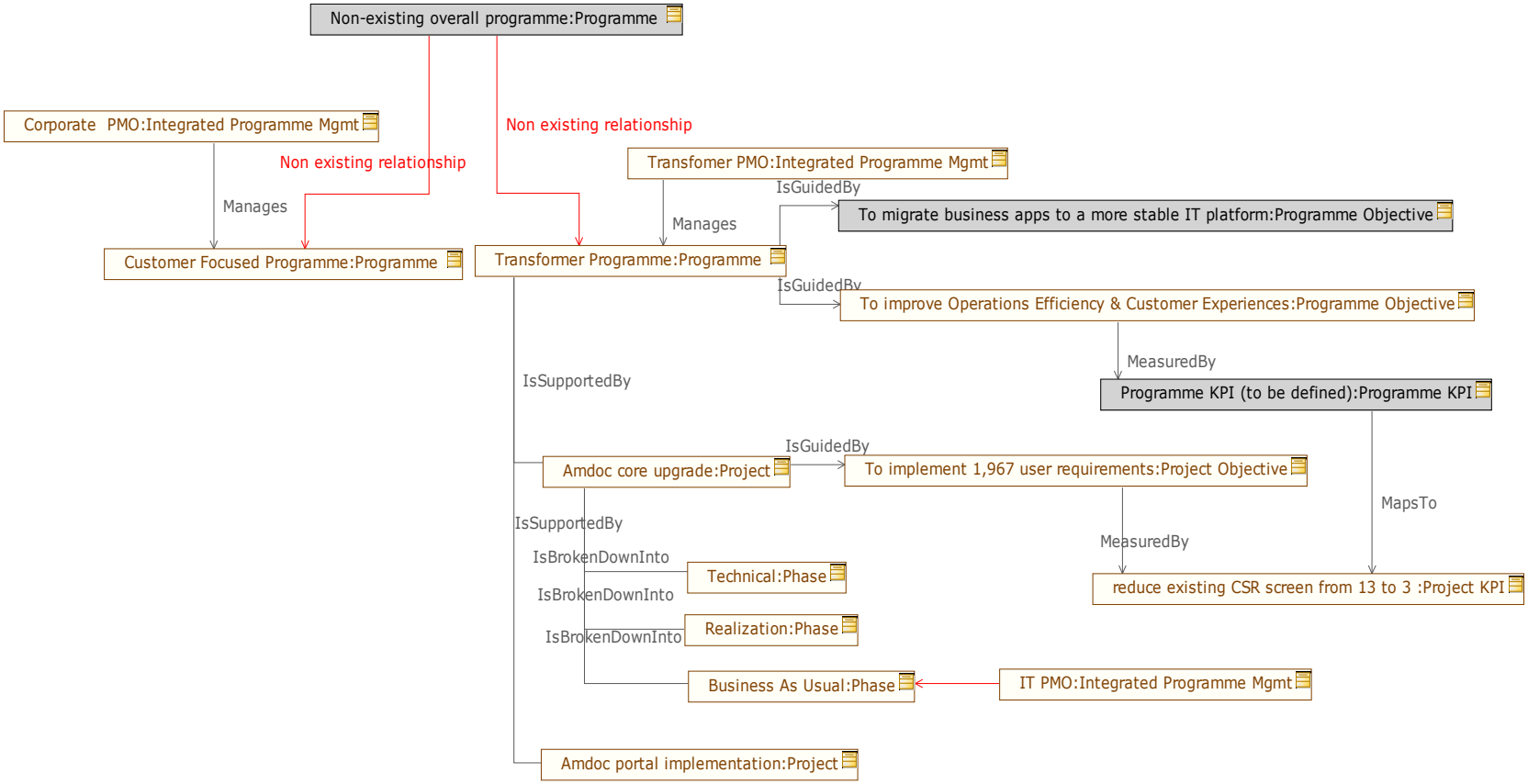
Declared instances :

- *EBP programme:Programme*
- *Work Stream 1:Project*
- *Work Stream 2:Project*

- *Work Stream 3:Project*
- *Work Stream 4:Project*
- *Work Stream 5:Project*
- *Work Stream 6:Project*
- *Work Stream 7:Project*
- *Omega:Programme*
- *Invetory Management:Project*
- *Pricing:Project*
- *Network Planning:Project*
- *To return to black in 2 years:Programme Objective*
- *To support BTP1 by improving E&M bottom line:Programme Objective*
- *To impove passenger business performance:Programme Objective*
- *TargetE&MRevenueIncrement:Programme KPI*
- *TargetE&MCostReduction:Programme KPI*
- *To reduce cost by implementing 42 hour week (from 40 hour week):Project Objective*
- *TargetOvertimeCostReduction:Project KPI*
- *CutCost:Programme KPI*
- *IncreaseRevenue:Programme KPI*
- *Obtain Union Support:Phase*
- *Agreement from Union:Gate*
- *Design shift patterrn:Phase*
- *Engineers agreed with proposal:Gate*
- *Implement shift pattern:Phase*
- *Org wide roll-out:Gate*
- *Brief Staff on new shift pattern:Activity*
- *Prepare new roster:Activity*
- *Broadcast new roster:Activity*
- *Effect the new roster:Activity*
- *Prepare Debrief:Activity*
- *Roll out 42 hour week in Airport 1:Task*
- *Airport 1 on 42 hour work week:Output*
- *ActualOvertimeCostReduction:Project KPI*
- *OvertimeClaimsReduction:Performance*
- *Airport 1 MRO work scheduling & operations:Business Process*
- *ActualE&MCostReduction:Key Performance Indicators*
- *Transformer Programme:Programme*
- *Technical:Phase*
- *Realization:Phase*
- *Business As Usual:Phase*
- *Customer Focused Programme:Programme*
- *Transfomer PMO:Integrated Programme Mgmt*
- *Corporate PMO:Integrated Programme Mgmt*
- *TMO:Integrated Programme Mgmt*
- *E&M PMO:Integrated Programme Mgmt*
- *Amdoc core upgrade:Project*
- *Amdoc portal implementation:Project*
- *IT PMO:Integrated Programme Mgmt*
- *To migrate business apps to a more stable IT platform:Programme Objective*
- *To implement 1,967 user requirements:Project Objective*
- *To improve Operations Efficiency & Customer Experiences:Programme Objective*
- *reduce existing CSR screen from 13 to 3 :Project KPI*
- *BTP1:Programme*
- *Non-existing overall programme:Programme*
- *Programme KPI (to be defined):Programme KPI*



Object Diagram (Co1)



Object Diagram (Co2)