CHALMERS

System Modelling for Spacecraft On-Board Computers

Master of Science Thesis in the Programme Networks and Distributed Systems

Fatemeh Zarvani

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Göteborg, Sweden, March 2010

System Modeling for Spacecraft On-Board Computers

Fatemeh Zarvani

Examiner: Johan Karlsson

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden March 2010

Abstract

This study makes an evaluation of the usability of SysML, a system graphical modeling language, for capturing functional requirements for space electronic products. This is done by a case study in which a model of a Packet Telemetry Encoder system is developed.. The report includes a brief overview of various graphical modeling languages.. The case study illustrates how the SysML provides system engineers with a robust modeling language. SysML is an extension to UML (unified modeling language) which is widely used by software engineers. The extension to the UML profile includes requirements modeling, parametric modeling, allocation and extension to structural and activity modeling. These extensions of the UML language are known as the SysML Profile. SysML is a representation of domain-specific engineering analysis models and the great advantage of applying the language is providing a single model to capture requirements, system elements and the relationship between these model elements. This capability of SysML is satisfying the traceability of requirements which is required in every stage during the system development life cycle. SysML is a methodology and tool independent language.

Acknowledgements

There are many people that have helped me throughout my studies, particularly the thesis work, whom I wish to thank. I want to thank Björn Hansson, my supervisor at RUAG for all his help and advices throughout the work of this thesis. I wish to thank Johan Karlsson, the examiner of the thesis at Chalmers. I am grateful for the support and encourages from my parents and I also wish to thank my family and all my friends for always being there and supporting me.

Table of Contents

Abstract....3

Acknowledgements....4

1.Introduction....6

2.Examples of system modelling standards and languages....8
2.1.AADL....8
2.2.SysML....8
2.3.MARTE....9
2.4.UPDM....10

3.SysML language Description....11
3.1. SysML Diagrams....11
3.1.1 Requirement Modelling....12
3.1.2 Behavioural Modelling....13
3.1.3 Structural Modelling....14
3.2 SysML Views and View Points....16
3.3 SysML Crosscutting and Allocation....16
3.4 SysML Tools....17

4.Case study....19
4.1 Organizing the Model....19
4.1.1 Telemetry Data Flow....20
4.2 Operational domain (top-level functionalities)....23
4.3 System context....25
4.4 Modelling the requirements....26
4.5 Structural Modelling....27
4.5.1 PTME module (component specification and functionalities)....28
4.6 Modelling behaviour....35
4.6.1 Packet Telemetry Encoder initialisation time configuration....36
4.6.2 Individual Virtual Channel Operational configuration....41
4.6.3 Transfer Frame Generation....43
4.6.4 Channel coding, Synchronization and Pseudo-randomization....48
4.7 Requirements Analysis....51
4.8 Critical System Properties and Constraints....52

5.Related works....53
5.1 Simulation-based design....53
5.2 Model-based Requirements engineering....53
5.3 Modelling, validation and verification....54
5.4 SysML Profile for System on chip (Soc) Design....55

6.Summary of the PTME SysML model....56

7.Conclusions....58

8.Reference....60

1 Introduction

Graphical system modelling is a technique intended to assist developers in managing the complexity involved in the design and maintenance of large computers systems. It aims to improve the communication among different stakeholders, such as end-users, developers and project managers, and others who have an interest in a given system. Graphical models are abstractions of a real system that can be useful particularly in refining and managing system requirements. Multiple aspects of the system are addressed by functional and behavioural modelling, structural and component modelling, performance modelling and engineering analysis modelling.

A trade-off analysis must be performed by the system engineers to select a suitable architecture among different alternative solutions. The system architecture which is defined by system engineers is considered as requirement of the software and can be used by software engineers. In order to develop balanced architectural solutions for a system in response to the stakeholder's needs, system engineers must efficiently express and refine the system requirements from the early stages of the development process. In some cases the preliminary requirements specification are defined by the customer. In other cases, the starting point is a modified version of the requirements specification for a previous system. In both cases, it is the modeller's task to interpret and analyse the stakeholders' requirements developing a precise specification with a minimum amount of ambiguity. For traceability purposes, the identified system requirements must be linked to the associated preliminary requirements and further to the system components.

The Term"Development environment" refers to the tools and repositories used to support a development process. Project Management tools, Engineering Analysis tools, Test Tools, System Modeling tools, Engineering Development tools, requirements management tools and Document Generation tools are considered as development tools. To support collaborative engineering there must be a logical connectivity between these tools. The role of the system model in this environment is to provide an integrated framework for other models. This is done for example by relating the text requirements to the design, providing the component specification for design models, supporting verification by providing the test case information and so on.

Comparison on system engineering vs. software engineering come up with this result that system engineering is broader discipline. System is the collection of different related components and software can be one of these elements. This thesis work aims to demonstrate the benefits that can be achieved by following a model-based approach to system development. Spacecraft onboard computers are examples of Software-intensive systems that consist of hardware and software.

Another purpose is to make an overview of existing standards and tools for model-based requirements analysis, and in particular assess the usefulness of the SysML language. As satellite systems are becoming increasingly reliant on software, the compatibility between the software models described in UML and the model of the system is an important issue, so there is a need for a unified modelling language and

a tool or at least compatible technique for modelling system, both hardware and software components to perform the transition from document-based approaches to model-based approach.

A major task is looking at the possibilities of modelling space electronic products at the system level as an alternative or complement to textual descriptions. The latter is made through a case study where the requirements for an existing Packet Telemetry Systems are formulated using SysML. The work started by studying how the elaboration of functional system requirements is done today at RUAG, system section, what is the procedure from requirements engineering point of view.

Today, engineers at RUAG use a document-based system engineering approach in which requirements are written in textual form using a requirement management tool called DOORS. DOORS support graphical modelling in terms of the block diagrams and functional diagrams. To express the system requirements and design specification, requirements are stored in requirements repository and captured in diagrams and tables, but graphics which are used to illustrate the design and textual requirements in tables are not fully associated. Therefore using DOORS, it is not possible to have different views of the system structure all describing the same concept but from different perspective. The idea is to have a better understanding of the system by having the text and diagrams in corporation which also provides the traceability of requirements. Here traceability refers to possible connections between the requirements and other part of the system, the structure, behaviour and so on, not only the requirements themselves.

The report is organized as follows. An overview of some system modelling languages is provided in the next section. Section 3 provides an introduction to the SysML modelling language and tools that implement the language. The description of the Packet Telemetry Encoder (PTME) system and its SysML model are covered in section 4. In section 5, an overview of related work is provided. Section 6 summarizes the SysML modelling capabilities that were explored in the modelling of the PTME system. Section 7 presents our conclusions and discusses directions for future work.

2 Examples of system modelling standards and languages

Modelling languages are used to visualize the System Engineering process (SE) by means of graphics. The system engineering process begins at an early stage of the system lifecycle with a requirements analysis to refine the system requirements specification that can be used in the following stages such as system level design, integration and validation. The main goal is to represent the allocation (selection of components which satisfy the requirements), binding (assignment of tasks to selected components) and scheduling (execution order for the task) by means of diagrams [1]. Some examples of modelling languages used in industry are described in this section.

2.1 AADL

Architecture Analysis & Design Language (AADL) is developed by the Society of Automotive Engineers (SAE) [2]. It is an industry standard that supports model-based embedded system engineering, allowing designers to model the software and hardware architecture of real-time, safety-critical, fault-tolerant and embedded systems with precise syntax and semantics. AADL supports many of the capabilities for system engineering purposes such as Requirement management in development process and provides a system design analysis to predict the performance. Using AADL to design a complex system satisfies both functional requirements and non-functional attributes, such as performance, reliability, security and so on [2]. There is an open source tool called OSATE (Open Source AADL Tool Environment) for editing models, model compilation and front-end analysis. Some commercial tools such as Illogic [33] and Artisan (UML tools) [33] also support AADL. A UML profile is included in the Language specification so AADL can be considered as a specialized modelling notation within the UML framework.

2.2 SysML

Systems Engineering Modelling Language (SysML) is a graphical modelling language, derived from the Unified Modelling language (UML). SysML has been proposed by the Object Management Group (OMG) [30], together with the international Council on System Engineering (INCOSE) [31]. The official OMG SysML specification v1.0 became available in September 2007 and SysML v1.1 was adopted by OMG in September 2008. SysML is being used by systems engineers to specify requirements, system structure, functional behaviour and allocation during the specification and design phases. The system may include a variety of components, hardware, software, data, people and natural objects. As indicated in figure 2.1, which is adopted from Friedenthal et al, the subset of UML reused by SysML is called UML4SysML. In addition, SysML includes language constructs that are defined through the profile mechanism, a standard mechanism in UML to add a collection of domain specific notations. The SysML language consequently consists of two parts: UML4SysML and the SysML profile.

The modelling is done at multiple levels. SysML behavioural models represent function-based, message-based and state-based behaviours with corresponding interactions, sequential flows and states transitions. Components models depict structural composition with interconnections and classifications. Requirements models represent the requirements and their relationship to other requirements, components, activities, test cases and so on. Constraints on various system properties (functional and non-functional) are represented in parametric models. The SysML language is described in more detail in section 3.

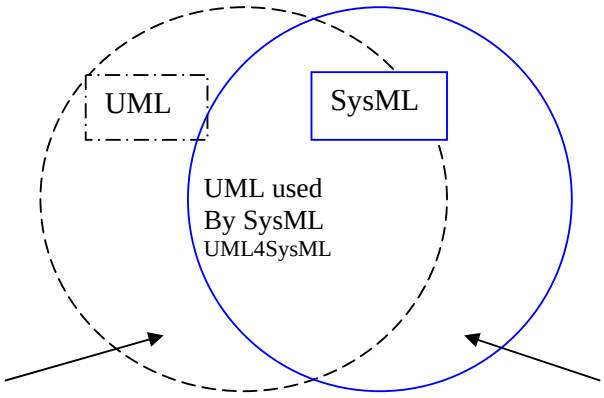

Figure 2.1 Relationships between SysML and UML

2.3 MARTE

Modelling and Analysis of Real Time and Embedded systems (MARTE) is a UML profile developed by the Object Management Group (OMG) supporting model-based development of real-time and embedded systems. MARTE provides support for specification, design and verification/validation of such systems, and defines a general framework for quantitative analysis, especially scheduling and performance analysis.

The Beta Specification of MARTE Profile was adopted by OMG in June 2007 and the Finalization Task Force (FTF) report was submitted in May 2008 to prepare the final specification [3].

When designing an embedded real-time system, scheduling analysis must be performed to ensure that all response time requirements are fulfilled. SysML and MARTE are two well known UML profiles that can be considered as complementary to identify design points that fulfill the timing constraints. They enable so called Design Space Exploration (DSE) analysis within the system design phase. DSE refers to discovering of optimal design solution among all possible ones [1]. While SysML is used to represent the system architecture and relating the functional and behavioral requirements to components, the MARTE profile

provides a general framework for quantative analysis including scheduling and performance analysis.

MARTE is composed of four main packages, MARTE foundations, MARTE Design Model, MARTE Analysis Models and MARTE annexes. The modeling of non-functional properties such as time modeling is contained in the MARTE foundations component. The activities of the left branch of “V-model” development life-cycle model are supported by the MARTE Design Model extension and the MARTE Analysis Model provides support for model-based analysis [29]. Papyrus (Eclipse) is the only tool until now that supports both the SysML and MARTE profiles. Artisan Software plans to implement an Artisan Studio Profile of the MARTE specification to support all the stereotypes and tag definitions of the MARTE specification [4].

2.4 UPDM

Unified Modeling Language (UML) Profile for DoDAF and MoDAF (UPDM)
The US Department of Defense Architecture Framework (DoDAF) [34] is a reference model for how to organize enterprise architectures and system architectures for complex systems. It defines standardized views of systems information and helps the system engineers to know what kinds of models to create so sets of views are used to express the architecture. DoDAF does not require a specific tool or notation, but modelling languages such as UML can be used for its implementation. The UK’s Ministry of Defense Architecture Framework (MoDAF) uses aspects of the existing DoDAF together with additional Viewpoints that are required to support UK MOD processes, procedures, and organizational structures [5]. Both architectures can be used to define a system and its users in an effective way along with the activities that take place to fulfill the requirements. UPDM is an industry standard UML profile that supports both DoDAF and MoDAF. It was developed by UPDM group, within Object Management Group. UPDM includes UML compliance levels and fully leverage SysML features which facilitates the integration of system of systems (SoS) modelling with system modelling [6].

3 SysML language Description

According to Sanford Friendenthal et al [7], SysML language concepts can be described in three parts: Abstract syntax, Semantics and Concert syntax. Abstract syntax describes the characteristics of the concepts and their interrelationships in the language using meta-models. Meta-models consist of meta-classes to describe individual concepts. For example “Data Type” is a meta-class used to describe values of attributes. Other examples of meta-classes are packages, blocks, and activities and so on. Model elements are instances of the meta-classes defined in the model libraries to support reusability. For example different components of a system are represented in user models as a model element that can be captured in a SysML block definition diagram. Semantics describe the meaning of the language concepts in the system engineering domain. In SysML semantics are described in English text. Concert syntax or notations are representation of the meaning, in other words; notations visualize SysML concepts as graphical or textual elements. The relationships between the concepts are also described by abstract syntax such as association or generalization. Stereotypes together with constraints are concepts which are defined as domain specific notations in Profile to extend the language. The stereotypes are similar to meta-classes that can be derived from meta-classes to create new or modified concepts for further language customization. For example “Value Type” is a stereotype derived from “Data Type”.

3.1 SysML Diagrams

SysML notations visualize SysML concepts as graphical symbols on diagrams or textual elements in tables and matrixes. SysML includes nine diagrams to represent a particular aspect of the system model graphically. A subset of the UML language is reused by SysML and the extension to UML includes:

- Requirements modelling.
- Extensions to system structure modeling by defining blocks and value properties.
- Parametric modelling.
- Allocations between model elements to establish relations between structure and behavior in general.
- Extensions to activity modeling by introducing “Item flow” and “continues flows” and so on.

Figure 3.1 shows the SysML diagram taxonomy (adopted from Fridenthal et al).
Several UML diagrams such as deployment diagram and communication diagram is omitted and the ones that are shown in figure 3.1 are either new diagrams (Requirement diagram and Parametric Diagram), modified versions of UML diagrams (Activity Diagram, Block Definition Diagram and Internal Block Diagram) diagram or the same as those used in UML (Sequence Diagram, State

Machine diagram, Use Case Diagram and Package Diagram). Common concepts such as Frame, Header and Description apply to all SysML diagrams.

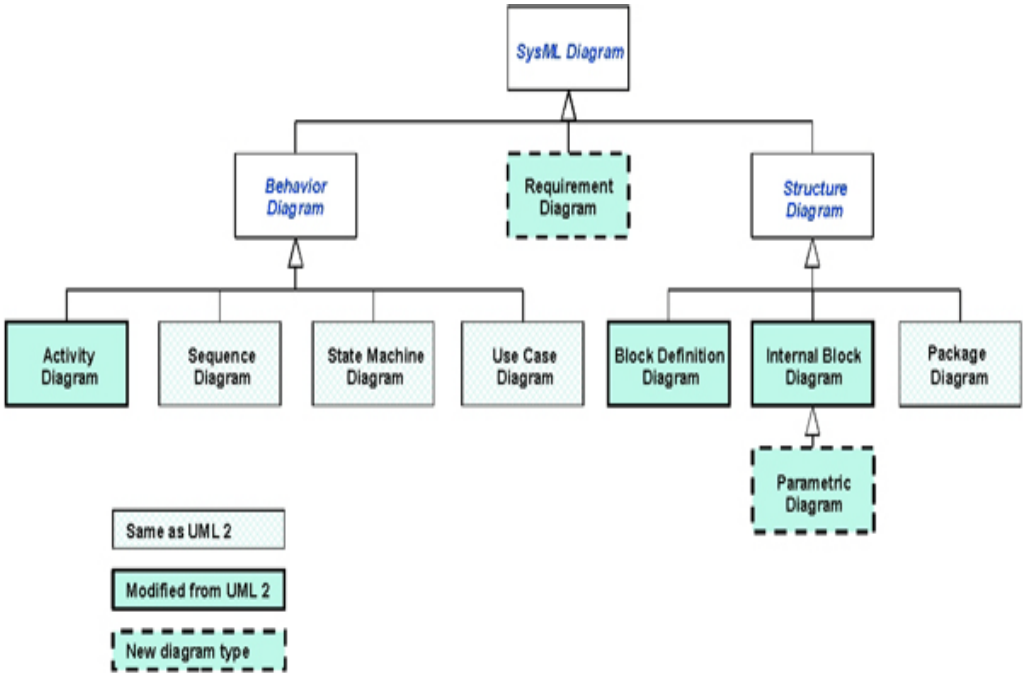

Figure 3.1 SysML Diagram Types

3.1.1 Requirement Modelling

Requirement Diagram: SysML has introduced a requirement diagram to capture text-based requirements in the model, using both graphical and tabular notations. The «requirement» stereotype extends the language. Some predefined properties are included in each captured requirement such as identifier and text. Depending on the application it is possible to extend the «requirement» stereotype to represent certain types of requirements, Examples of such extension include specify the «performanceRequirement» stereotype and the «functionalRequirement» stereotype .The containment relationship is used to decompose requirements hierarchically into several levels of requirements. . It is also possible to relate the requirements to other requirements as well as to other model elements. The latter is called cross-cutting relationships.

A set of stereotypes are defined in the SysML profile to implement relations. The «derive» represents the derive relationship which is defined between derived and source requirements. Containment is used to represent the requirements hierarchy. Another relationship which is defined between requirements is «copy»; this is used to show reuse of requirements in a different hierarchy. Cross-cutting relationships are defied as follows. «Satisfy» indicates the satisfy relationship which is defined between design elements and requirements. For example use cases may satisfy one

or several requirements. The verify relationship is defined between requirements and test cases and indicated by the «verify» stereotype. Model elements can be used to clarify the requirements, this dependency between the model elements and requirements is indicated by the «refine» stereotype. The «trace» relationship is used to relate external documents (source of the requirements) to the captured requirements in the model. To represent the relationship between the requirements and model elements the requirement construct can be shown on other diagrams such as Block Definition Diagram and Use Case Diagram.

The requirements relationships can be depicted using direct notations, Compartment notations and Callout Notations. A dashed arrow together with the name of relationship (e.g., «satisfy») is used in direct notation. Relationships can be displayed using callout notations. There will be a link between the requirement and the callout which specifies the model element at the other end of the relationship and corresponding relation name. Another alternative method is to use compartment notations which display the relation explicitly. This is required that corresponding model element supports compartment. Rationale is a concept which is provided by SysML Requirement Diagram to write comments on requirements. Unlike UML notes, rationale is considered as a model element. The rational information can be helpful when analyzing the requirement and further changes.

3.1.2 Behavioural Modelling

Use Case Diagram: like UML, use cases describe the behaviors of the system in terms of functionalities and uses of a system. In use case diagrams, the requirements are grouped into several use cases therefore Use Case Diagram provides a means of expressing functional requirements. It also offers a more precise way of expressing functional requirements than a text specification, and thus provides a means for reducing ambiguity in SysML models. Requirement blocks which capture the text-based functional requirements can be related to use cases using the refine relationship in both Requirement Diagram and Use Case Diagram.
An actor represents any entity that participates in the use of system in a use case model and can be a human, organization or any external system. Three kinds of relationships are defined to relate different use cases, specialization, inclusion and extension. Inclusion is used to denote that one use case contains another use case functionality. Extension is used to denote that a given use case extends the behaviour of another use case. Specialization refers to “is a” relationship to classify different actors that share some common features.

Activity diagram: SysML Activity diagram is a modified version of the UML activity diagram, which is used to model behaviour in terms of the flow of input, output and control. One activity comprises several actions. Functional requirements are grouped in actions. “CallBehaviourAction” is a special type of action which can be associated to an activity. This association captures the composition of activities and makes it possible to call an activity from another. In SysML control is treated as data for starting and disabling actions. The actions are triggered by inputs and outputs through a controlled sequence. One action stops, it initiates the other action and when all the actions are complete the activity is complete. Conditional activities and interchanges between activities can also be captured in the diagram. Events can

also trigger an action and it means that something has happened and needs to be handled by the system. In SysML, flows can be typed by value types, data types, and blocks.

New concepts are introduced in SysML activity diagram by means of stereotypes. « Continuous »is one of the important stereotypes. SysML supports continuous flow modeling in a way that continues output and input can be produced and consumed by an action while it is executing like flow of water. « Control Operator » stereotype is applied to action and operation in order to enable or disable the action. When it applies to an action, control values are needed as an input or the action provides control values as an output. These new concepts are explained by Friedentalet al.

Sequence Diagram: Behaviour of the system can be represented by the sequence diagram in term of message passing between multiple lifelines. There is no difference between UML sequence diagram and SysML sequence diagram. Each lifeline represents an entity which is participating in an interaction. Messages can be synchronous or asynchronous. A sender of a synchronous messages waits for a reply from the receiver, while a sender of an asynchronous message continues its execution immediately after the message is sent. Synchronous messages are denoted a by a closed arrowhead, whereas asynchronous messages are denoted by an open arrowhead. SysML sequence diagram is effective for capturing the communication between discrete types of behaviour. It provides a useful mechanism for verifying system interaction against the use case description. There are techniques to enhance the scalability of SysML sequence diagrams compare to UML sequence diagrams. One added feature in SysML is the possibility to use control logics such as conditional statements, parallel statements and loop statements or Reference Sequences when illustrating the behaviour of the system. Reference Sequences is used when trying to illustrate one sequence which contains more sequences. So the modeller refers to other Sequence diagram by using this notation which is linked to it. In a sequence diagram, messages are represented on horizontal axis and time is represented on vertical axis.

State machines: SysML state machine is as same as UML 2.0 state machine and represents event-based behaviour modelling. Considering the behavioural concepts of the system in term of different states during the life-cycle of the system, state machines are used to represent the state changing. Three types of events are supported, change event, time event and signal event. The goal is to track state transition with respect to corresponding events.

3.1.3 Structural Modelling

Package Diagram: The SysML package diagram is the same as the UML package diagram and is used to organize the model. The organization can be done regarding diagram types such as use cases packages, requirements packages, structure packages and so on, or regarding different levels such as enterprise, system physical design, logical design and so on. View points are alternative ways to organize the model. This concept is described in a separate subsection hereafter.

Block definition diagram: Block Definition Diagrams aim to describe the structure of a system. It is based on the UML class diagram and composite structure diagram. The enhancement is done by providing two capabilities, reusability of model elements and interconnections. A block in SysML can be used to represent any type of component in a system, functional, physical and even human, etc. It is possible to decompose the system into sub-systems and show the association, composition and specialization relationships between the blocks such as a block hierarchy in the block definition diagram. To specify the block characteristics in details the modeller can add some features to define its properties, operations, its interfaces and constraints. It is also possible to allocate other model elements from or to the block. For example allocation of activities to the block, that can be used to define its role in the system. Different types of allocation are described later in a separate subsection. If the block satisfies a particular requirement, this can be captured in the block definition diagram. Blocks are reusable model elements in multiple diagrams.

Parts can be described as the block properties which are composite blocks that almost represent the physical components of the blocks. References are parts that are not owned by the enclosing block. The value properties of the Block represent quantifiable properties with units, dimensions and probability distribution.

Internal block diagram: The internal block diagram represents the internal structure of a higher level block. Part properties of a block can be shown on internal block diagram to visualize the blocks interconnections and collaboration to realize the behaviour of the block. The parts can be connected using connectors and ports. Ports provide access to the internal structure of a block. In SysML there are two types of ports, **flow ports** represent the interfaces, through which the block communicates with other blocks, and **standard ports** represent the interfaces for handling requests and invocations of the services with other blocks [8].

Standard ports are typed either as a required interface or provided interface. As the names indicate required interface specifies the set of operations which are requested by the owning block and provided interface specified the set of operations that the block provides. Flow ports are typed either as an atomic or non-atomic. Atomic flow port shows the interaction of a single item between two blocks (in/out or in-out). If there are more one item that flows in different direction between two blocks, the specification of flow is described by «flow specification». Regardless of type of flow ports, the item that flow can be in any type, a block, a value type, a data type or any other user defined types.

Parametric diagram: SysML parametrics diagrams are new modelling capabilities which are not part of UML. Parametric diagrams integrate engineering analysis models with system requirements and design models. They are used to define equations that express constraints for block properties [17]. Performance, reliability, and physical characteristics are examples of critical system parameters that can be expressed by parametric diagrams. Constraints are linked to the quantifiable characteristics of the system and its components which are called “value property”. They are captured in a block which is tagged by the «constraint» stereotype. The particular usage of constraint blocks are described by the “Constraints properties“. The equations represented by constraint properties in the model, specify the relation among the value properties which are the system properties. For example “F=m*a”

is a constraint and “F”, “m” and “a”, are parameters each describing value properties of the system.

Parametric diagram captured blocks, which are composed of constraint properties and constraints blocks which in turn can be composed of other constraint blocks. The binding connectors are used to associate parameters of constraint blocks with the owning block or constraint block properties.

3.2 SysML Views and View Points

According to “Recommended Practice for Architectural Description of Software-Intensive Systems” [9], a view is a “representation of the whole system from the perspective of a related set of concerns” and a Viewpoint is “a specification of the conventions for constructing and using a view - a pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis”. This standard is a recommended practice, one kind of IEEE standard [9]. It is notation-independent, which means that it does not specify the format or the media for the architectural description. The main objective of the standard is to establish a conceptual framework and vocabulary for talking about architectural issues of systems. Detailed presentation through several views makes a system model easier for the analysts and stakeholders to comprehend. Views do not contain elements but instead import model elements from different packages in order to collect them into a common namespace. The Reference Model for Open Distributed Processing (RM-ODP) is a standardized model implementing the viewpoint approach [10]. It recognizes five viewpoints: Enterprise, Information, Computation, Engineering and Technology [11].

SysML models and model elements visualize different aspects of a system. SysML introduces two concepts, views and viewpoints which are consistent with the IEEE 1471 standard [9]. A view of the system supports a particular stakeholder’s interest and a viewpoint describes this particular perspective. A SysML block which is indicated by the «view» stereotype represents a description of a specific interest. The viewpoint is a block indicated by «viewpoint» stereotype, and includes a set of properties. The purpose of specific perspective is identified as a property in viewpoints. Other properties of viewpoints are the language which is used to present the view, the particular stakeholders and the method which is used to establish the view. The view conforms to a particular viewpoint.

Several model elements such as activities and requirements can cooperate to capture a specific view of the system. A package Diagram can be used to depict both viewpoint and the view that conforms to it together with other related model elements. Several views of each model can be described by SysML, each view presenting specific aspect of the model.

3.3 SysML Crosscutting and Allocation

Separation of concerns is the process of breaking a system into distinct features that overlap in functionality as little as possible (Dijkstra, 1974)[12]. According to this

process, complex systems can be decomposed into several components with a single functionality. Sommerville views concerns as non-functional requirements or constraints that needs to be identified early on in the requirements definition phase of a project to assess impact, and provide traceability [13]. Non-functional requirements such as safety and traceability are needed to be satisfied by the system components. “Cross-cutting” is a concept that refers to mapping these non-functional requirements to functional requirements. The flexibility of the means to address the concerns is important because they may change during the evolution of system. One example of such concerns is security [14].

Allocation is the most important cross-cutting construct which is a general relationship that maps one model element to another. This mapping is mentioned in five categories. Behavioral allocation maps functions to components in SysML activity diagrams. Structural allocation maps logical components to physical components. Allocation of software to hardware units is an example of structural allocations which is captured in SysML block definition diagrams and internal block diagrams. Allocation of requirements refers to various ways that requirements can be related to each other and to other model element, as mentioned in section 3.1.1. Allocation of flows refers to the mapping of flows that are captured in internal block diagrams to the corresponding flows which are represented in activity diagrams. Allocation of properties is captured in Parametrics diagrams by mapping the constraints to the system properties.

As mentioned earlier the use of SysML requirement diagrams make it is possible to associate the requirements with elements of the system model which facilities analysis, management, testing and refinement of requirements during the development of a SysML model. To satisfy system-level requirements it is needed to specify requirements for the components. Huase in [14] called this mapping mechanism a backward mechanism as the mapping is from the design model to the design requirements; Forward mapping refers to the mapping from system analyses to system design.

3.4 SysML Tools

Drawing tools are an important aspect of the introduction of model-based system development. There are several tools that implement UML profiles (SysML, MARTE and UPDM) by different tool vendors, such as “Papyrus” [15] and “Telelogic Tau” [16]. As we are following a unified approach for system and software modelling, Artisan Real-Time Studio has been chosen because it supports both UML and SysML. Artisan tool supports the OMG SysML standard for requirements engineering purposes by implementing the SysML profile.

The profile includes the following stereotypes and corresponding definitions to extend the UML language:

- Activities stereotypes and tag definitions
- Allocations stereotypes and tag definitions
- Blocks stereotypes and tag definitions
- Constraints blocks stereotypes and tag definitions
- Model elements stereotypes and tag definitions
- Ports and Flows stereotypes and tag definitions
- Requirements stereotypes and tag definitions

A requirements profile allows requirements to be fully integrated into the modelling environment. Management and traceability between requirements and other model elements are big issues and critical to development effort. Artisan tool supports these features. Artisan Studio DOORS synchronization provides synchronization of the contents of Artisan Studio and DOORS. This capability allows the modeller to create, modify or delete requirements in either of these tools. The exchange of requirements and their relationships is included in the logical interface between Artisan studio and DOORS. All the relations between the requirements which are maintained in the Requirements Management tool are captured in requirements models if the tools are synchronized any proposed updates to the requirements in the modelling tool will be applied to the corresponding requirements in the Requirements Management tool. Reqtify [32] is an alternative tool that can be used to support traceability analysis and requirements coverage by providing interface between the modelling tool and requirement management tool such as DOORS.

4 Case study

This section contains an example that illustrates the application of SysML to the design of a spacecraft packet telemetry system (PTM). The case study is oriented around the functional structure of the PTM system, rather than the detailed implementation. A packet telemetry system (PTM) permits a spacecraft on-board data system to transmit telemetry data units to the users located on the ground across a space link. The system consists of three main parts, an onboard control system, a ground control system and external entities that produce the transferred data. In this case study, we present SysML models for a subset of the physical and logical components of the on-board part of the PTM system.

The packet telemetry encoder (PTME) is a hardware module which is part of the onboard control system module and comprises several sub-modules such as encoders, modulators, interfaces and buffers. The core of the PTME is a hardware component called the telemetry encoder (TME). A TME is included in the CROME ASIC, a proprietary hardware component developed by RUAG Space. The CROME ASIC is an integrated device providing on-board telemetry and telecommand services via standardized interfaces. A commercial off-the-shelf version of the TME, AT7909, is available from Atmel.

The development of the PTME SysML model involved the following steps:

- Organization of the model and identification of a reusable library of components.
- Elicitation of operational domain and toptop-level functions by via use cases.
- Defining the system context.
- Modelling the requirements.
- Modelling of the system structure and behaviour
- Analysis of the system requirements.
- Capturing and evaluation of parametric and constraints.

4.1 Organizing the Model

The initial organization of the model includes definitions of packages containing model elements and libraries of components. The package diagram in figure 4.1 describes the high level organization for the PTME model. The model contains packages for use cases, requirements, structural components, value types, item types, activities and parametrics for analysis. The activities package contains behavioural diagrams including activity diagrams, sequence diagrams and state diagrams. The analysis package contains constraints blocks which can be used for performance analysis purposes.

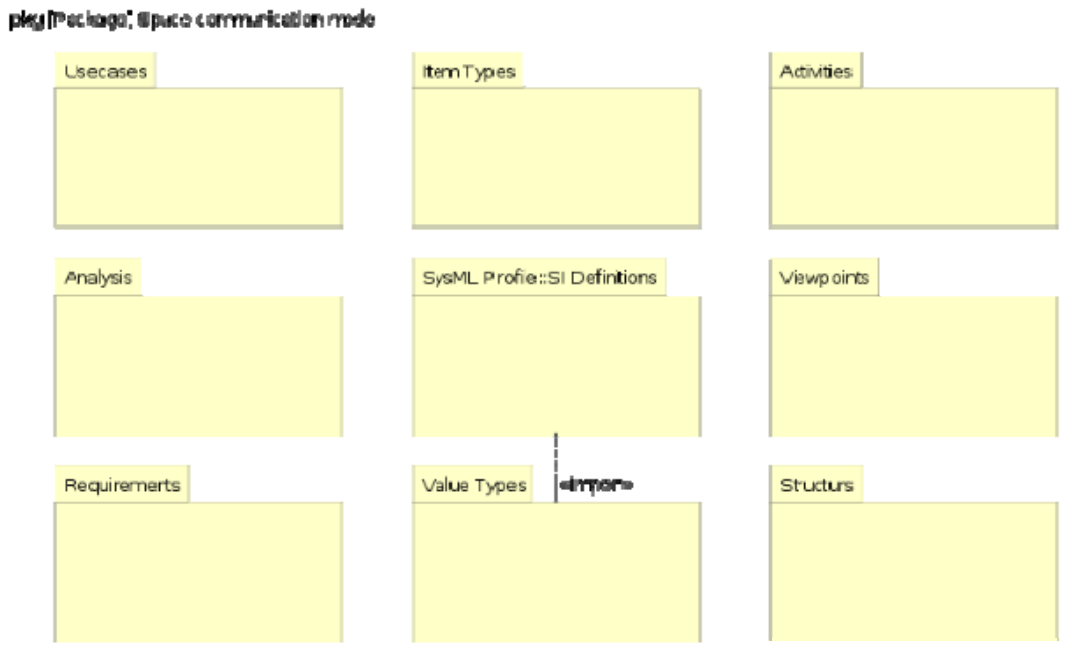

Figure 4.1 - [Package] Space communication model

In SysML, model libraries are packages containing reusable elements. In this example we have two model libraries "Value Types" and "Item Types". Required Value Types are defined in this user defined package, prior to use with corresponding unit and dimension properties, tagged with the "Value Type" stereotype; they are available to multiple models while Basic types are defined in the SI definition package and imported by this user defined model library. The same concept applies to Item Types package. The item types package capture the types of things that flow in the system. The existence of these Items is independent of where they flow and how they are used. For example in PTM system source packets, telemetry packets, transfer frame and channel access data unit (CADU) are defined as Item types. Several nested packages are contained within each of these packages and can be captured in different diagrams.

4.1.1 Telemetry Data Flow

As mentioned in the previous section Item Type package contains entities that flow in the system between components. The description of various data types in PTME system is mentioned in this section.

Multiple application processes running in different onboard computers generate a data structure unit called **"Source Packet",** that can be fixed or variable in length. The source packet consists of two major fields:

- The **packet header** which provides the control information during end-to-end transmission such as the identification of the source, sequence numbering and Data Field length.

- The **packet data field** contains the source data and can be variable in length.

Received telemetry data from several source applications will be multiplexed onto the Telemetry data channel. The data unit at this intermediate stage is called "**Telemetry Packet**". If the data field exceeding a valid length (65536 octets); segmentation is done on source packets before generating the Telemetry Packets by the application software. The Telemetry Packet header field must provide control information to reconstruct the source packet along with the source packet header field, such as Identification of Telemetry Packet segmentation sequence and the length of the Data Field of Telemetry Packet which is contains the various segment of original Source Packet Data Field.

Figure 4.2 shows the structure of telemetry packet and different fields using blocks. The value property indicates the Value Items which is defined for each individual block. In this figure the length of each field is defined by the length value. Telemetry packets are received at Telemetry Encoder and will be encapsulated in **Transfer Frames** which is the link protocol data unit. Telemetry packets are synchronously inserted into the transfer frame data field, with a pointer to the header of the first packet. Block definition diagrams are used to illustrate the telemetry packet and transfer frame data structure.

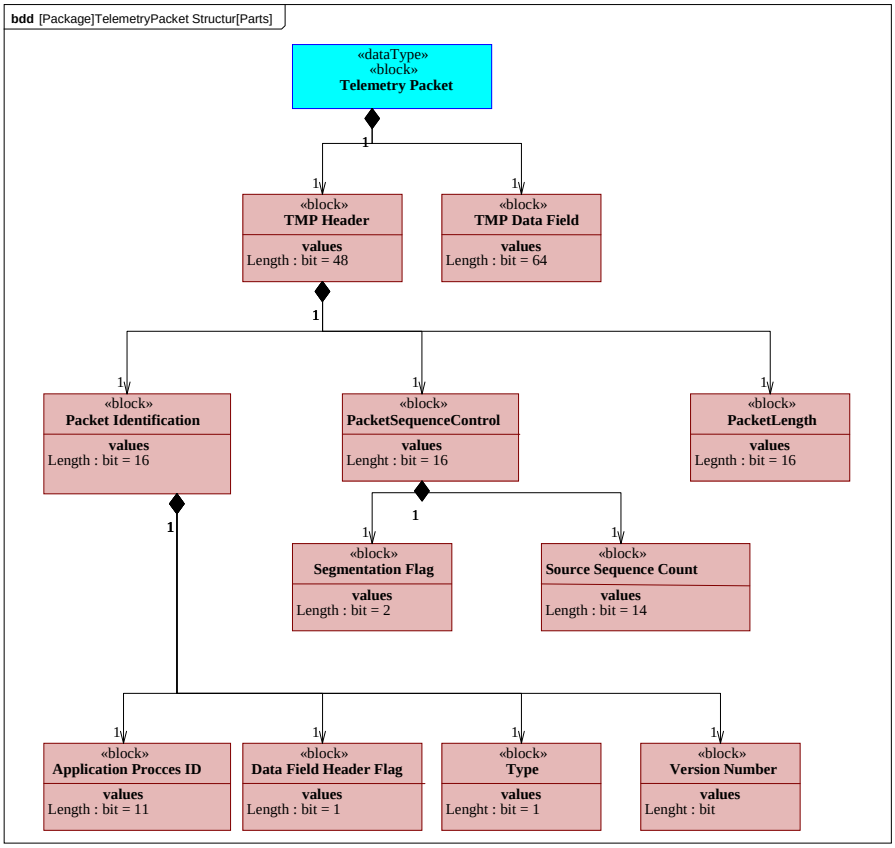

Figure 4.2 - [Package]TelemetryPacket Structur[Parts]

As shown in figure 4.3, the Transfer Frame consists of the primary header and the data field. Generation of Optional fields such as Secondary Header and Transfer Frame Trailer are indicated by means of flags in Primary Header. The supported Transfer Frame lengths are 223,446,892 or 1115 octets with four different trailer options, which are selected by four input pins on ASIC. Here we define constraints for the length parameter both on Transfer Frame and the data field inside the Transfer Frame. These constraints are indicated as yellow blocks in the diagram (figure 4.5). The Transfer Frames will be synchronized then optionally encoded, randomized and modulated. The data unit that consists of the synchronizer and codeblock or un-coded Transfer Frame is called the channel access data unit (CADU). The CADU is then modulated and transmitted over space links as bit streams.

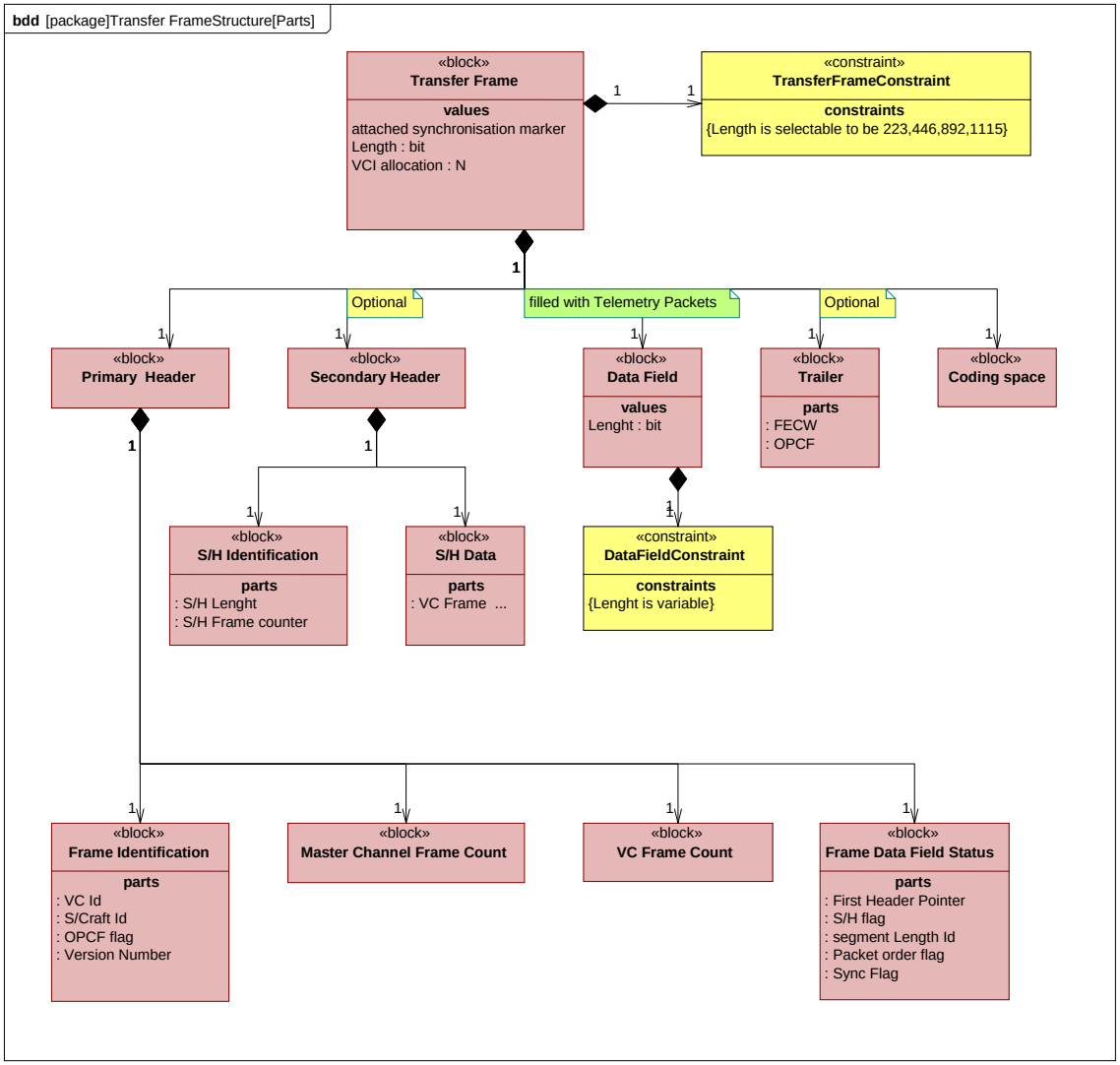

Figure 4.3 - [package] Transfer Frame Structure [Parts]

4.2 Operational domain (top-level functionalities)

As mentioned earlier, the functionality of the system can be described with use cases which are illustrated in SysML “Use Case” diagram. At this step we have the requirements specification from the stakeholders. Now the task is to identify the main functionalities, while considering the system as a black-box. Figure 4.4 shows the Top-Level use cases, and actors who are the users of the system by associations. For each use case there is a text-based description that corresponds to the functional requirements to provide additional information. These use case specification can be captured in the model as a single or multiple comments.

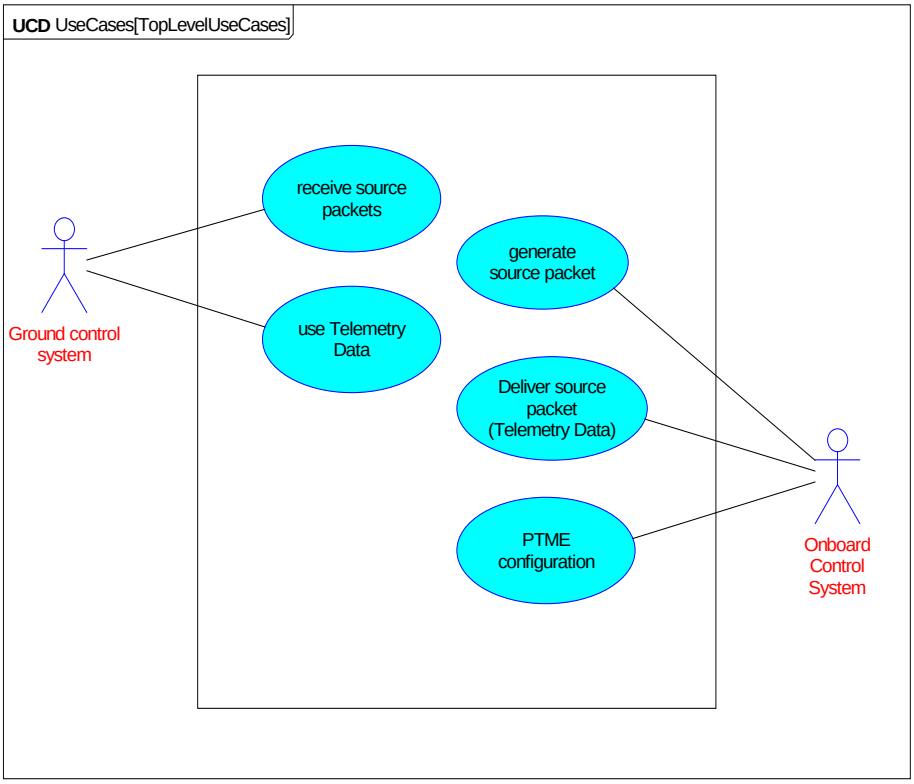

Figure 4.4 - UseCases [TopLevelUseCases]

The top-level use cases are as follows:

- **Generate source packet**: On-board Control System is responsible to packetize the telemetry data received from the onboard data sources.
- **Deliver source packet**: Telemetry data is transferred to the ground for processing and use.
- **PTME configuration**: configuration of the operational mode of the on-board system.
- **Receive source packets**: ground control system receives telemetry data from the onboard control system.
- **Use Telemetry Data**: Telemetry data is processed and analyzed by a ground user.

The use-case diagram in figure 4.5 shows the next level of use case hierarchy for delivering the source packets. It depicts the major functionalities which are provided by the Onboard Control System when delivering the telemetry data. At this stage, the onboard control system is divided into two parts with separate functionalities: the data handling software, which is executed by the onboard computers, and the Packet Telemetry Encoder. The data handling software is responsible for multiplexing source packets to generate data to be inserted in the data field of the telemetry packets and for conducting segmentation in case of long source packets. Generating Channel Access Data Unit is a use case for the Packet Telemetry Encoder, which includes the generation of the Transfer Frames. Operational configuration, encoding, randomization and modulation are performed on Transfer Frames before sending the output. The <<include>> dependency is used to decompose use cases.

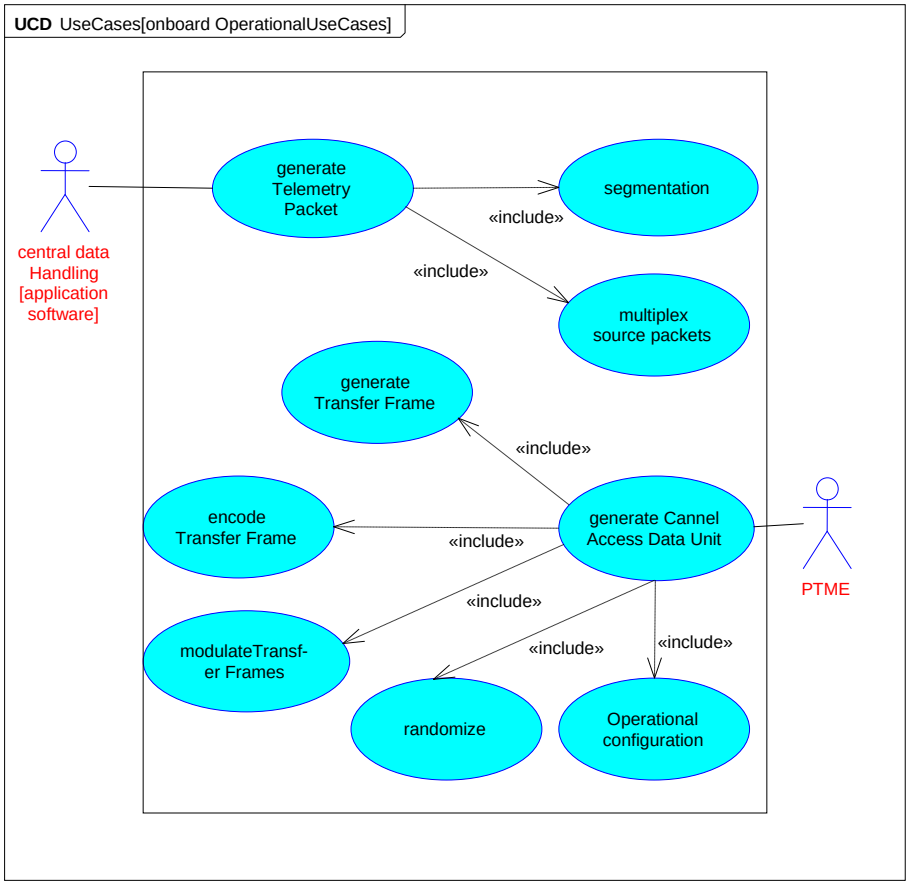

Figure 4.5- UseCases [onboard OperationalUseCases]

To describe the interaction between the different part of the system and external systems such as users, we need to perform a scenario analysis for each use case and then model these scenarios using either activity diagrams or sequence diagrams. Addressing the failure scenarios is also critical.

4.3 System context

To make the analysis and design of the system efficient, understanding of the environment in which the system will operate is important. Hence, it is essential to create models of the elements that interact with the PTME. Figure 4.6 is block definition diagram that shows the main system and the relevant entities in the environment.

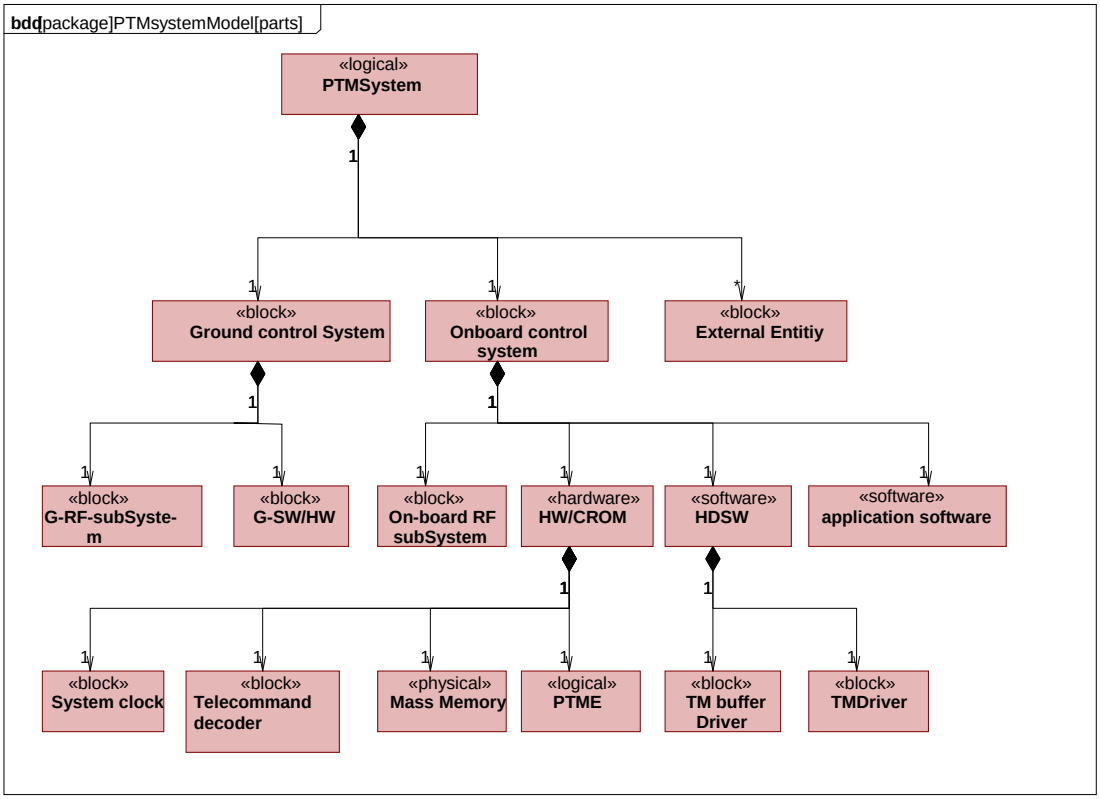

Figure 4.6- [package]PTMsystemModel[parts]

The multiplicity of the relations is 1-1; there is one Ground Control System inside the PTM system, one telemetry driver within the hardware driver software (HDSW), and so on. To show how the parts are connected and interact with each other, we use the Internal Block diagram shown in figure 4.7.

At the space side, the Telemetry data are first processed by the application software, and then either transmitted directly to the telemetry encoder via the telemetry hardware driver or stored in a mass memory for later transmission. The telemetry buffer driver which is also part of the hardware driver software is connected to the mass memory to store data. The packet telemetry encoder is connected to transmission hardware for the radio links to the ground communications circuits and mission control centres.

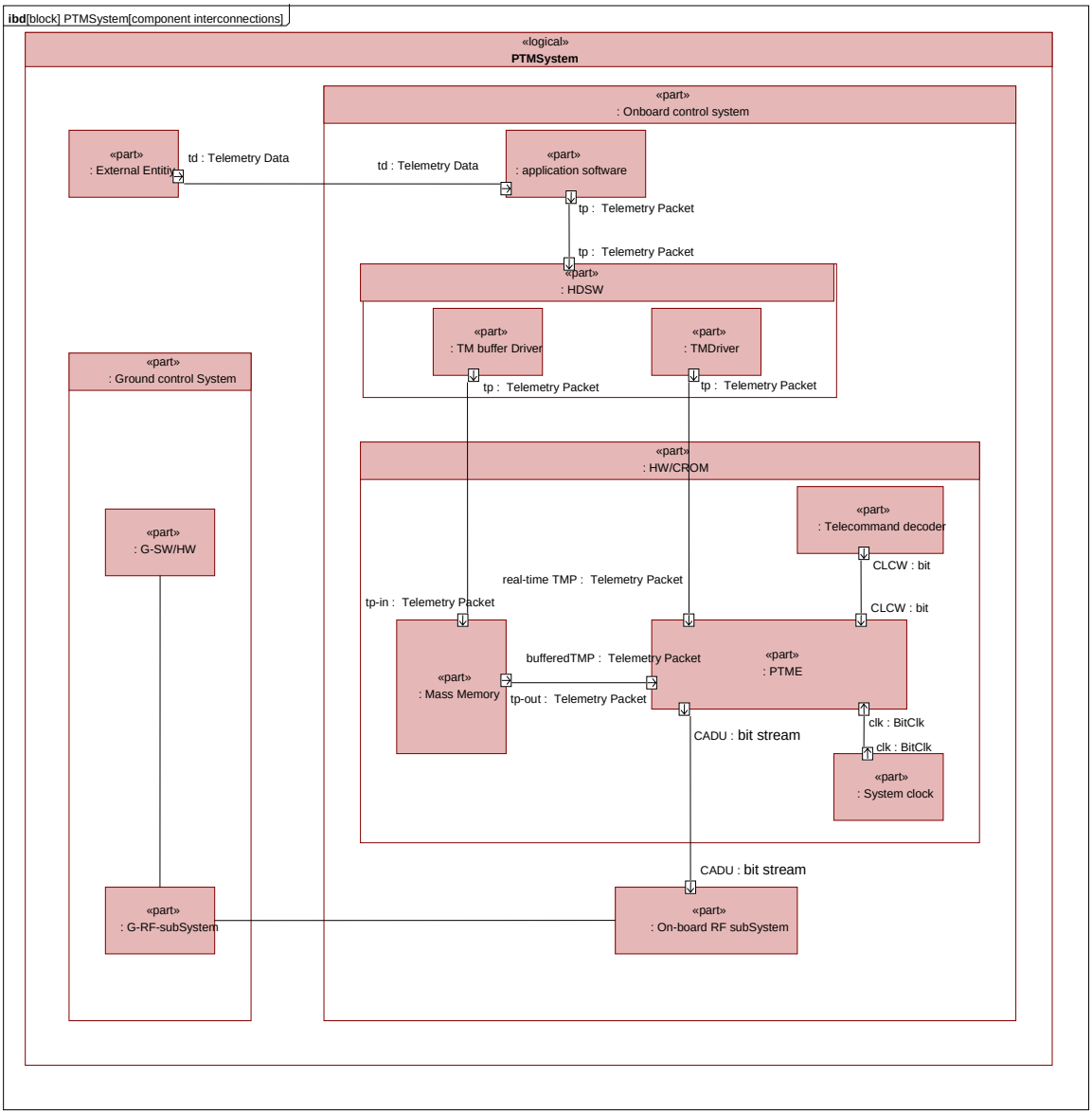

Figure 4.7 - [block] PTMSystem[component interconnections]

4.4 Modelling the requirements

Use case diagram can only provide a high level view of the system requirements. To specify requirements in detail, SysML provides a new feature called requirement diagrams. These are used to keeping track of requirements during development. This is done by specifying relations, either between different requirements, or between requirements and other model elements.

The requirements diagram is useful for stakeholders to express their problems, because it is simpler to understand from the picture than reading the structured documents which are provided by requirements management tools. The requirement

diagrams for the PTME depict the requirements described in the requirement specification documents contained and managed in the DOORS tool. Figure 4.8 shows the requirements hierarchy for the PTME system.

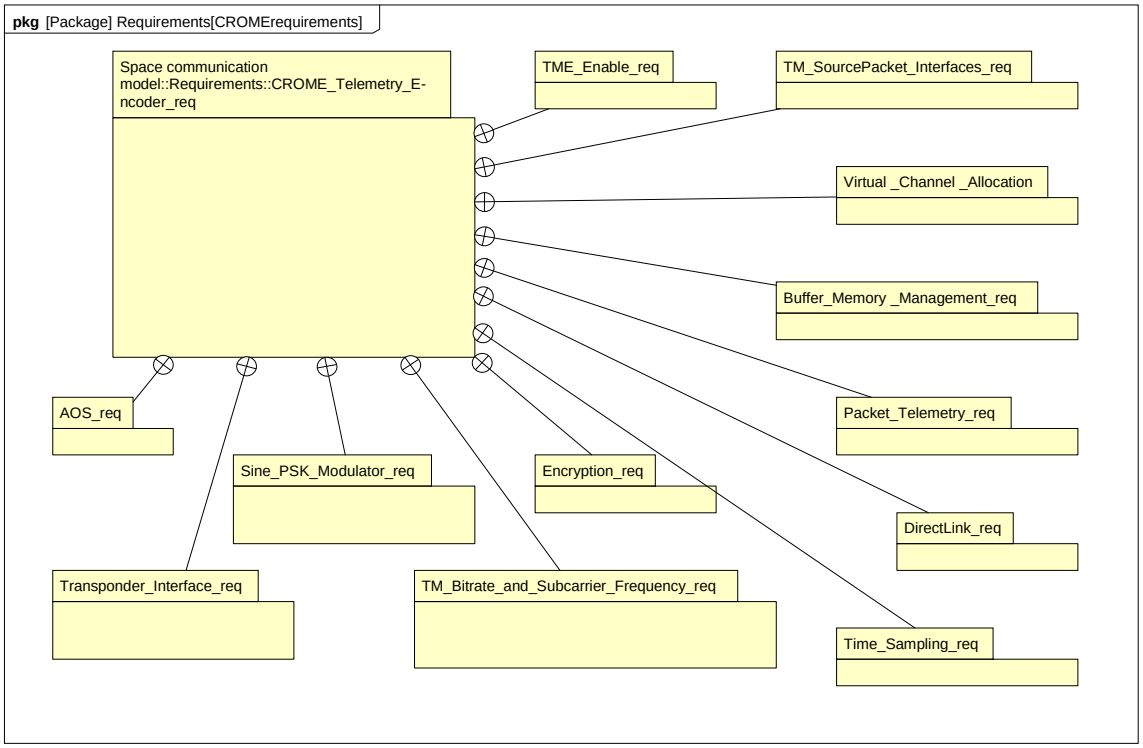

Figure 4.8 - [Package] Requirements [CROMErequirements]

As figure 4.8 shows the CROME requirements which are divided into packages. Each package contains the related requirements and their relationships. It is possible to stereotype the requirements to specify the category which could be from various types.

4.5 Structural Modelling

The system structure is created by decomposing the system (here the PTME) into different parts, and then defining the parts using block diagrams. System architecture can be divided into two parts, the logical architecture and the physical architecture. The purpose is to decompose the system into logical and physical components. The interactions among these components are needed in order to satisfy the system requirements. Physical components can be further divided into different types; hardware, software and so on. The logical components are allocated to the physical components. The design constraints are imposed on the physical architecture [7].

4.5.1 PTME module (component specification and functionalities)

The packet telemetry encoder module comprises the following sub modules: input interfaces, bandwidth allocation table, external shared memory, telecommand decoder interface, PTME internal bus and internal components (PTME-Internal). Internal components are implementing embedded encoders, randomizer and modulators. The PTME decomposition and block hierarchy captured in figure 4.9 using the SysML block definition diagram. This diagram shows the functional blocks such as physical components and the interfaces to the external components. The texts beside the arrows show the role of the corresponding block. For example the Bandwidth Allocation Table (BAT) is responsible for virtual channel selection. The specification of each block is described in details hereafter.

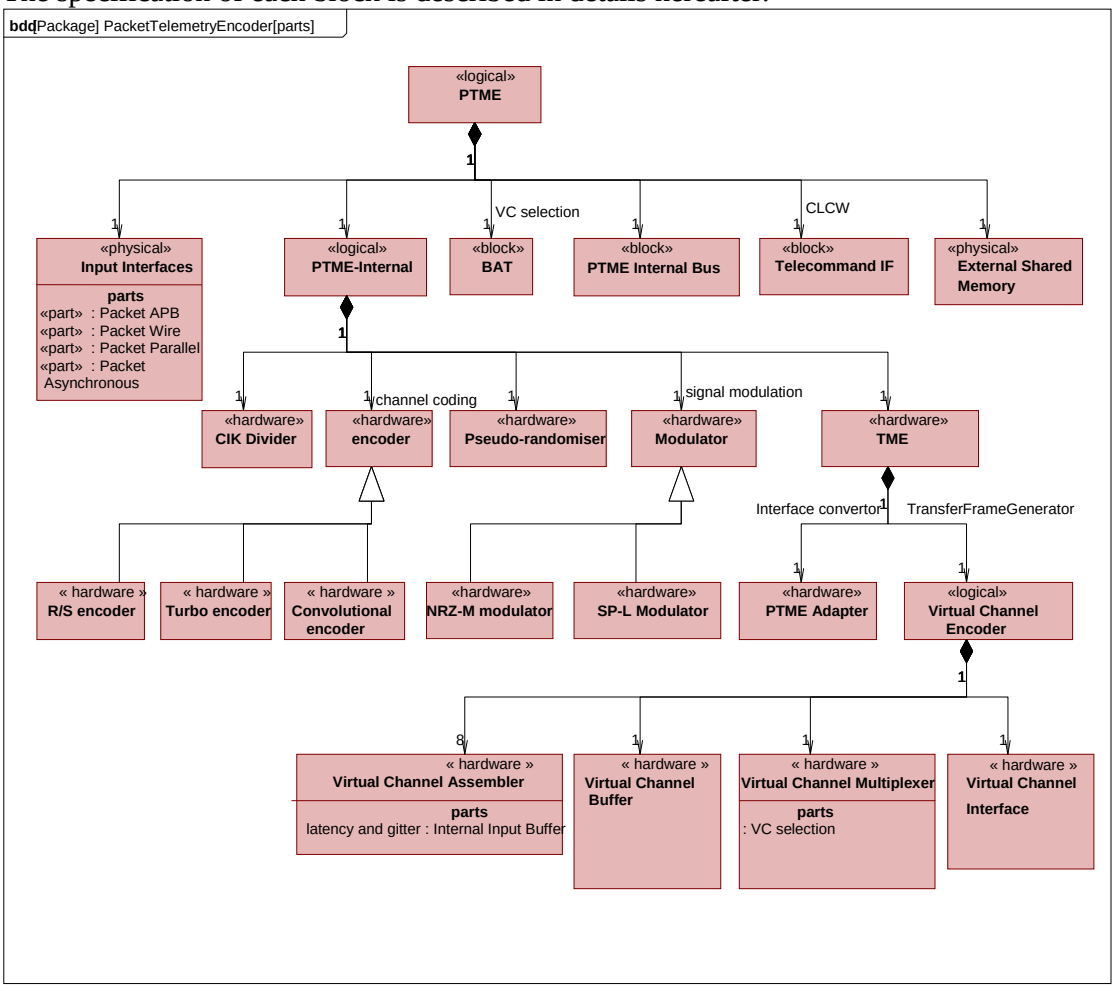

Figure 4.9 - [Package] PacketTelemetryEncoder[parts]

The interactions between the parts are shown in the internal block diagram in figure 4.10. The ports (flow ports in this diagram) are defined and the flow Item is specified. As it is shown in figure 4.10 PTME Internal Bus (PIB) resides between the PTME Internal part and the External Shared Memory, because every communication between these two parts is through this internal bus. The Packet

Telemetry Internal Bus can be configured at initialization time for logical pointer addressing or physical addressing, depend on **gPhysicalAddress** parameter value.
The packet telemetry encoder has several flow ports. Two input ports for receiving the telemetry packets on different virtual channels. The other input port is connected to the Telecommand Decoder to retrieve part of the Command Link Control Word which is inserted into of the transfer frame trailer field via the telecommand interface. The third input port is connected to the system clock. There is one output port for transmitting the CADUs.

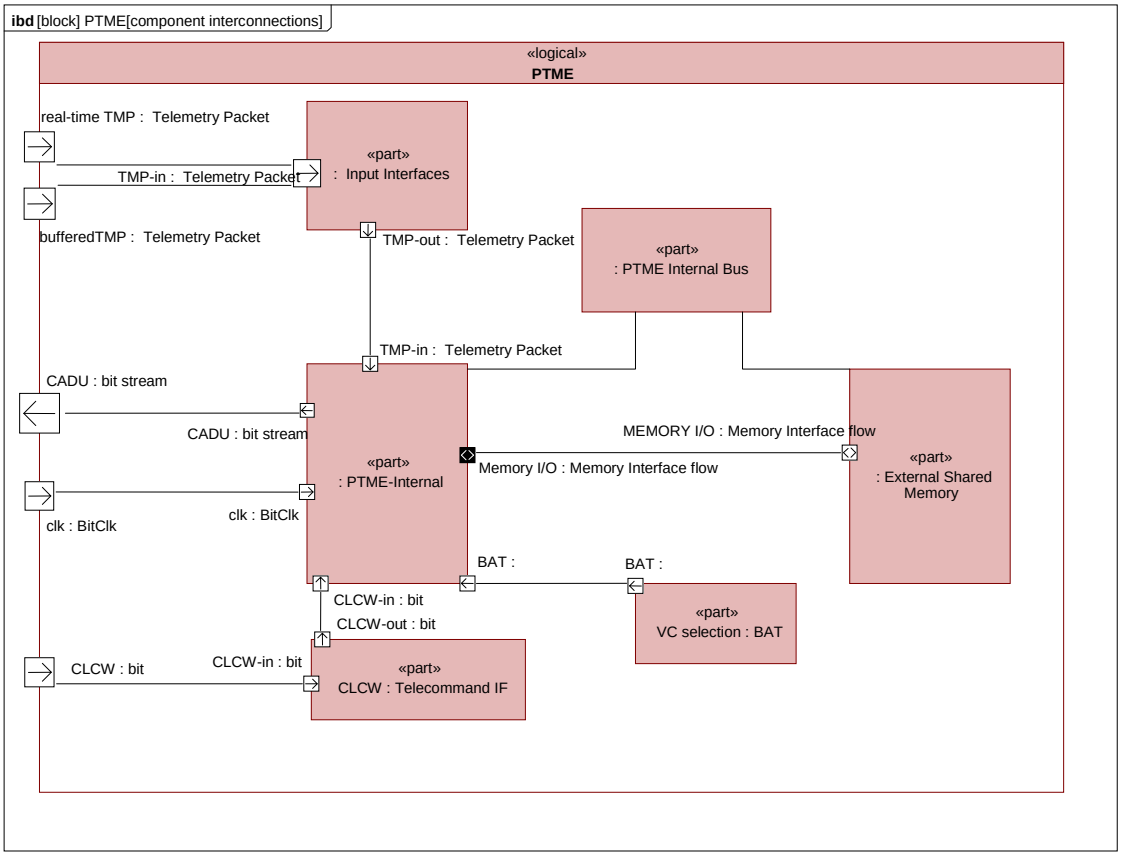

Figure 4.10 - [block] PTME[component interconnections]

4.5.1.1 PTME Internal components

Packet telemetry encoder internal module compromises several sub modules such as clock divider, telemetry encoder (TME), channel encoders, modulators and randomizer. Telemetry encoder and clock divider are always instantiates by PTME module and the rest of modules such as subsequent encoders and modulators instantiates separately depend on PTME configuration at initialization time. Figure 4.11 shows the internal block diagram of the packet telemetry encoder. Parts and interconnections between the components are captured. The flow ports and Item flow types is represented along with connections.

Clock divider: The Clock Divider (CD) generates different clock enable signals for the different encoders and modulators to control the bit rates. The source for the bit

rate frequency is always the dedicated bit rate clock input BitCLK. It is able to divide the bit rate clock frequency by means of the different clock enable signals.

Reed-Solomon Encoder (RSE): If enabled in initialisation time configuration this block generates and inserts the Reed-Solomon check symbols. RSE encoder implements the encoding procedure according to the channel coding standard.

Turbo Encoder (TE): The output of this block is turbo codes at a nominal bit rate. The rate is 2, 3, 4 or 6 times the frame bit rate. This functionality is optional and can be enabled by setting the enable parameter at initialisation time.

Convolutional Encoder (CE): This block generates Convolutional codes. The output rate is twice as high as the frame bit rate. This functionality is optional and can be enabled by setting the enable parameter at initialisation time.

Split-Phase Level modulator (SP): If this option is enabled, Split-Phase modulation is performed on both the Transfer Frame and the Attached Synchronisation Marker, being output as a bit stream from preceding encoders.

Non-Return-to-Zero Mark modulator (NRZ): This functional block performs NRZ_Mark modulation on the Transfer Frames being output. This functionality is optional and can be enabled by setting the enable parameter at initialisation time. Both data and the Attached Synchronization Marker (ASM) are affected by the coding. When the encoder is not enabled, the bit stream is by default non-return-to-zero level encoded. The Non-Return-to-Zero Mark encoder (NRZ) and Split-Phase Level modulator (SP) implements signal modulation according to the AD5 standard.

Pseudo-Randomiser (PSR): The functionality of this block is to multiplex the generated Transfer Frame from previous processes and the Reed-Solomon codes (if available) with pseudo-random bit pattern. The output is synchronised with each Transfer Frame. This functionality is optional and can be enabled by setting the enable parameter at initialisation time.

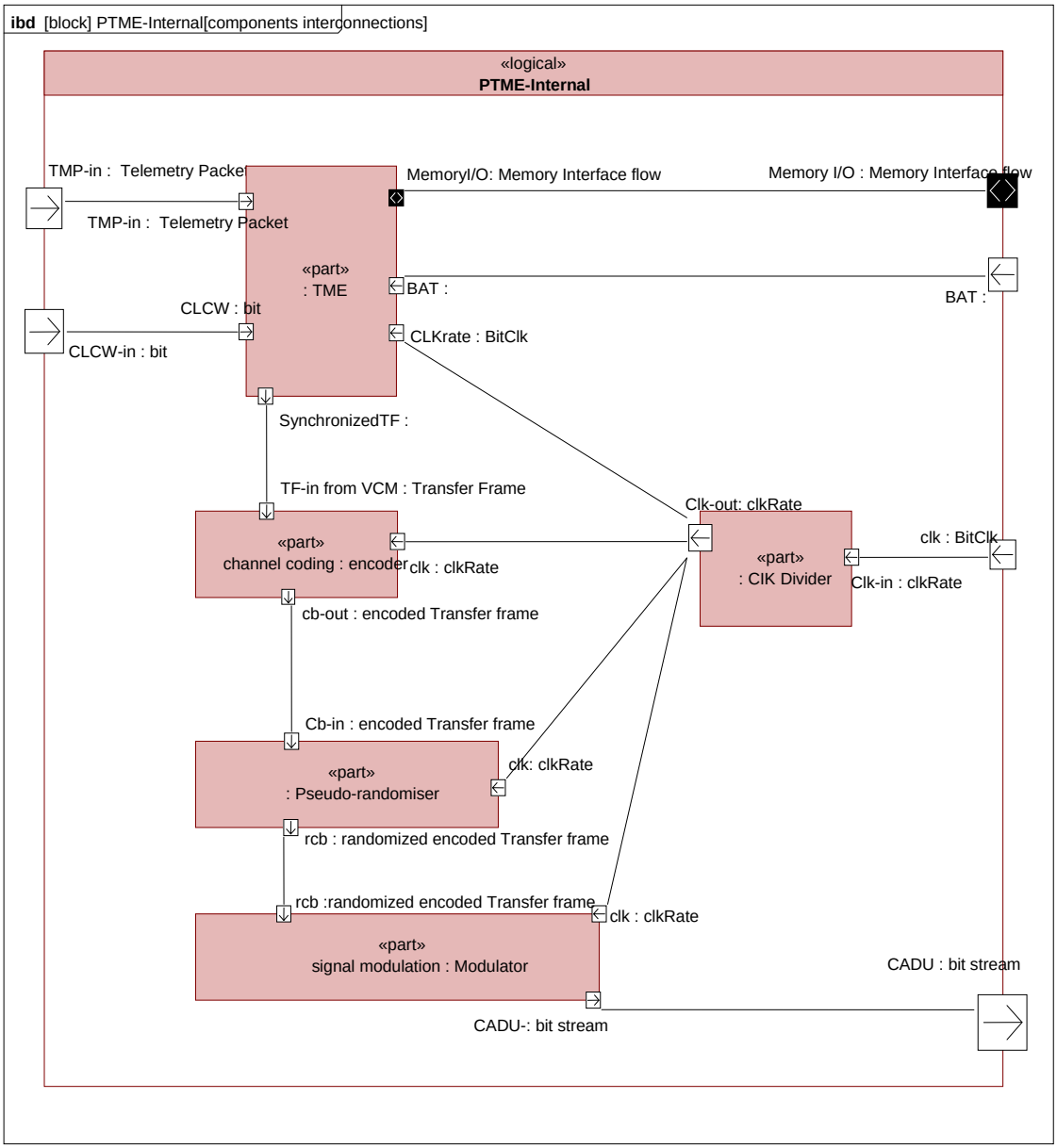

Figure 4.11 - [block] PTME-Internal[components interconnections]

Telemetry Encoder: Telemetry encoder, the core component of packet telemetry Encoder receives Telemetry Packets and generates telemetry transfer frames. The output of the TME is synchronized transfer frames. Telemetry Encoder comprises six modules which are mentioned here after. Figure 4.12 depicts the interactions between the components of the telemetry encoder.

Virtual Channel Encoder (VCE): VCE initiates Virtual Channels (VC), Virtual Channel Multiplexer (VCM), and Virtual Channel Buffer (VCB). The main functionality of this block is generating Transfer Frames with attached

synchronization marker. Virtual channels are logical components and up to 8 virtual channels are supported by the telemetry encoder and decoder on the physical channel. Each VC is identified by the virtual channel identifier (VCID) and spacecraft identifier (SCID). They are configured individually during operation and at system initialisation (power up). The VCs can handle packets and data blocks with data fields of up to 65536 octets and they are independent of the packet format and any data structure can be used. Two physical components are resides in VC module. Each VC comprises one virtual channel assembler and one virtual channel interface.

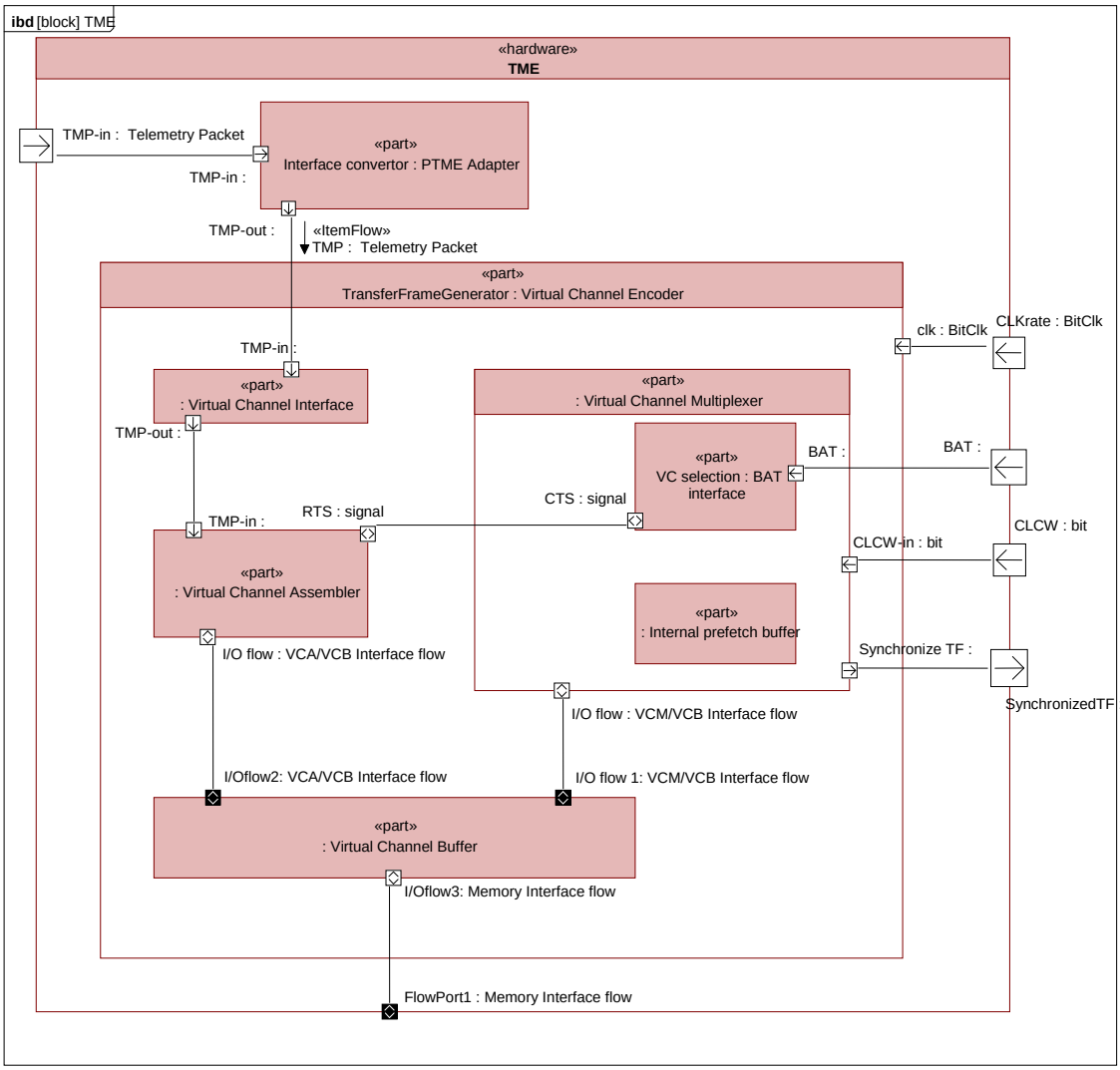

Figure 4.12 - [block] TME[interactions]

Virtual Channel Assemblers (VCA): For each Virtual Channel there is a Virtual channel assembler that receives the telemetry packets over virtual channel Interface which is connected to the various types of input interfaces depending on how the virtual channel assemblers are allocated. Data received from serial input interfaces are assembled into octets which are internally forwarded to the parallel interface. This data will be transferred over the PTME Internal bus (PIB) to the Virtual

Channel Buffer to be stored in the external buffer memory. There is also a local buffer in the VCA to allow for latency and jitter on the PIB.

Virtual Channel Interface (VCI): The actual input interfaces that are used to receive the Telemetry data in form of octets from the onboard data- handling system placed on ASIC for the virtual channels are placed outside telemetry encoder module and Virtual Channel Interface resides between this input interfaces and corresponding virtual channel assembler. The interfaces are situated outside the VCE.

Virtual Channel Buffer (VCB): Any accesses to the external shared memory from internal modules are via the VCB over PTME internal bus. The VCB acts as an arbiter on the PIB to allocate the access bandwidth dynamically to each in proper way. This is done by multiplexing the different read and writes accesses to the common external buffer memory interface. VCA communicates to the virtual channel buffer to manage the allocated memory space.

Virtual Channel Multiplexer (VCM): One to eight Virtual Channel Assemblers together with one Virtual Channel Multiplexer are used to generate Transfer Frames according to the ESA Packet Telemetry standard.
VCM include following Interfaces:
Interface for retrieving dynamic part of the Command link Control Word (CLCW) per Transfer Frame generated. This interface provides two Synchronous bit serial channels to allow direct connection to two Packet Telecommand Decoders. The VCM provides a mechanism for multiplexing the CLCW from two Telecommand Decoders. The retrieved data will form part of the OPCF field in the Transfer Frame Trailer.

VCA/VCM Interface: It resides between Virtual Channel Multiplexer and each of the eight Virtual Channel Assemblers. Ready-To-Send signal is an input from VCA to VCM which is sent over Virtual Channel Request Interface. This signal is asserted each time VCA is ready to transfer a packet which is going to be used by VCM to generate the complete Transfer Frame. The output signal From VCM to VCA called Clear-To-Send, which is asserted corresponding to the selected VCA.

An internal pre-fetch buffer in VCM is used to allow for latency and jitter on the PTME internal bus. The depth of this buffer can be increased by means of the ***gPreLength*** constant at initialization time.

VCM and VCAs operates at a maximum clock frequency of 12.5 MHz. the maximum operating bit rate is 12.5Mbit/s, which is proportional to the clock frequency used. The VCM operates on the falling BITCLK edge while VCA operates on the rising BITCLK edge. Both VCA and VCM feature Built-In Self Test that can be automatically activated after reset.

4.5.1.2 Input Interfaces

Depending on the design specification, telemetry data is either transmitted from application software directly to the Telemetry Encoder or stored in the mass memory for later transmission. The configuration is done during the operation. In

real-time transmission Telemetry Encoder receives telemetry packets on serial or parallel interfaces connected to the separate logical components, the virtual channels (VC). Typically, this connection is via a Space-Wire link from the processor to the CROME ASIC. The transmission from the mass memory to Telemetry Encoder is through a Packet-Wire link to the CROME ASIC. The supported interface types are Space-Wire (via CROME ASIC), Packet-Wire interface (PW), Packet-Asynchronous (PA), Packet-Parallel (PP) and Packet-APB (PAPB).

4.5.1.3 Bandwidth Allocation Table

The Bandwidth Allocation Table (BAT) is used for the arbitration of the downlink bandwidth when several Virtual Channels are trying to access the memory. The size of the BAT and the number of entries are selectable at initialization time. The values are in the range of the implemented Virtual Channel Identifiers. Two built-in algorithms are provided by the Virtual Channel multiplexer (VCM) to select which Virtual Channel to output the next Transfer Frame. The selection algorithms are reprogrammable during the operation.

Bandwidth allocation, in this mode the VCM uses adaptive frame ordering. The bandwidth allocation table entries indicate the virtual channel to be selected. All virtual channels will therefore have a guaranteed minimum bandwidth allocated when the bandwidth allocation algorithm is selected. In this table for each virtual channel one property can be set. The property is set when the ready to send signal is asserted by the corresponding virtual channel assembler. The VCM scans the BAT in a round robin fashion to select a virtual channel assembler that has the ready to send (RTS) flag asserted. Scanning continues until there is no virtual channel assembler with the RTS flag asserted. If scanning process reaches the VCA with asserted RTS, the VCM will send a request to corresponding VCA. This is done by sending clear to send signal (CTS) to that VCA. Otherwise the VCM selects one VCA and send the request.

Priority selection, in this mode each VC is assigned a priority which is programmable and VCM selects a VCA with the highest priority that can produce a normal Transfer Frame. If priority selection is enabled, the Virtual Channel with identifier zero will have the highest priority and the Virtual Channel with the highest identifier value will have the lowest priority.

4.5.1.4 External Shared Memory

Different virtual channels share a common external memory. The telemetry data received on each VC is stored temporarily in a buffer in this memory. The sizing and splitting of the external memory between VCs is almost fixed at initialisation time. The parameters concerning memory partitioning are: the overall memory size, the number of areas into which the memory is split, the number of areas that can be allocated to each VC. The number of memory areas allocated to the Virtual Channels also can be done dynamically via an external interface.

4.5.1.5 Telecommand Interface

The packet telemetry encoder is connected to the telecommand decoder to retrieve part of the **command link control word**, which is included in the transfer frame trailer field. The command link control word (CLCW) is transmitted as part of the Operation Control Field (OPCF) in a Transfer Frame Trailer. CLCW has two parts with 16 bits length each. The static part is generated by the Telemetry Encoder or retrieved from external data interface. Static part of the Command Link Control Word contains: Control Word Type, Version Number, and Status Field, COP in Effect, Virtual Channel Identifier and Reserved Field. The dynamic part is retrieved from the Telecommand Decoder via the external data interface and contains the following parameters: Number of Radio Frequency Available, Number of Bit Lock, Lock Out, Wait, Retransmit, FARM B Counter, Report Type and Report Value.

4.6 Modelling behaviour

The activity diagrams in the following figures characterize continuous flows and parallel behaviour of the packet telemetry encoder and its subsystems. Figure 4.13 captures the main activities which are performed when generating the channel access data units in the packet telemetry encoder. The description of the main functions that are visualized using SysML diagrams are mentioned thereafter. Rounded-cornered boxes in a SysML activity diagram represent actions which are needed to perform a particular activity. The lines that are used to link the actions are control flows that define the sequence of actions. The things that flow (object flows) can be represented by object nodes (blocks) connected to actions or activity parameter nodes which are placed in the boundary of an action box.

Some activities that contain other activities are represented as a “CallBehaviourAction”. This is used when we want to make a reference from one activity to the related activities which specify it in more detail. This capability will also give us the possibility to reuse the activity. The fork sign indicates the “callBehaviourAction”. The texts beside the arrows show the input to the following action or activity. An activity will not execute until the required input is available. Another constraint to start an activity is receiving the required events if it is mentioned in the specification. Examples of required signals (events) are shown in figure 4.19.

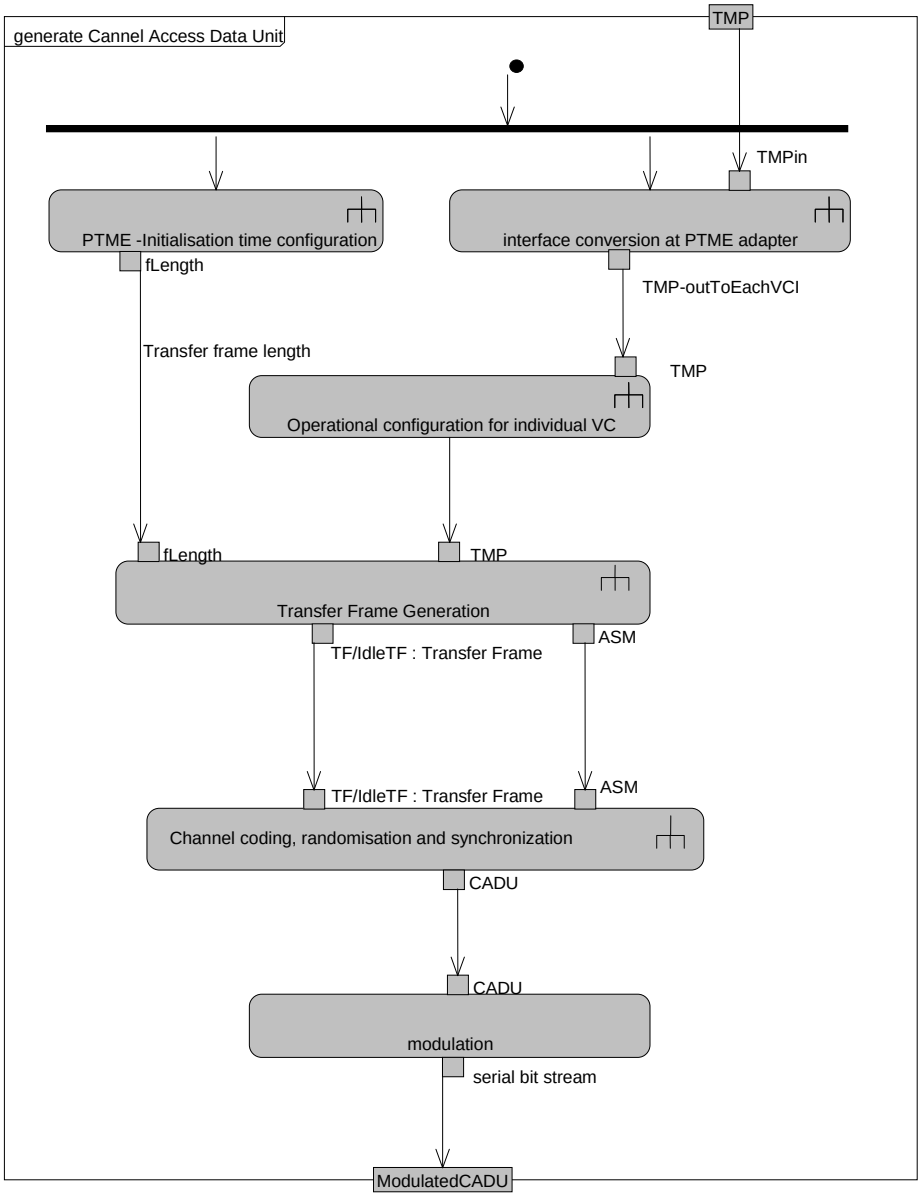

Figure 4.13 - [Activity] generate Cannel Access Data Unit [main activities]

4.6.1 Packet Telemetry Encoder initialisation time configuration

Implementation options are selected at initialisation time. The packet telemetry encoder is very flexible and adaptable to various mission needs. The following parameters can be programmed during the initialisation time to provide a variety of services by telemetry encoder. Figure 4.14 shows the actions which are included in PTME- initialisation time configuration activity.

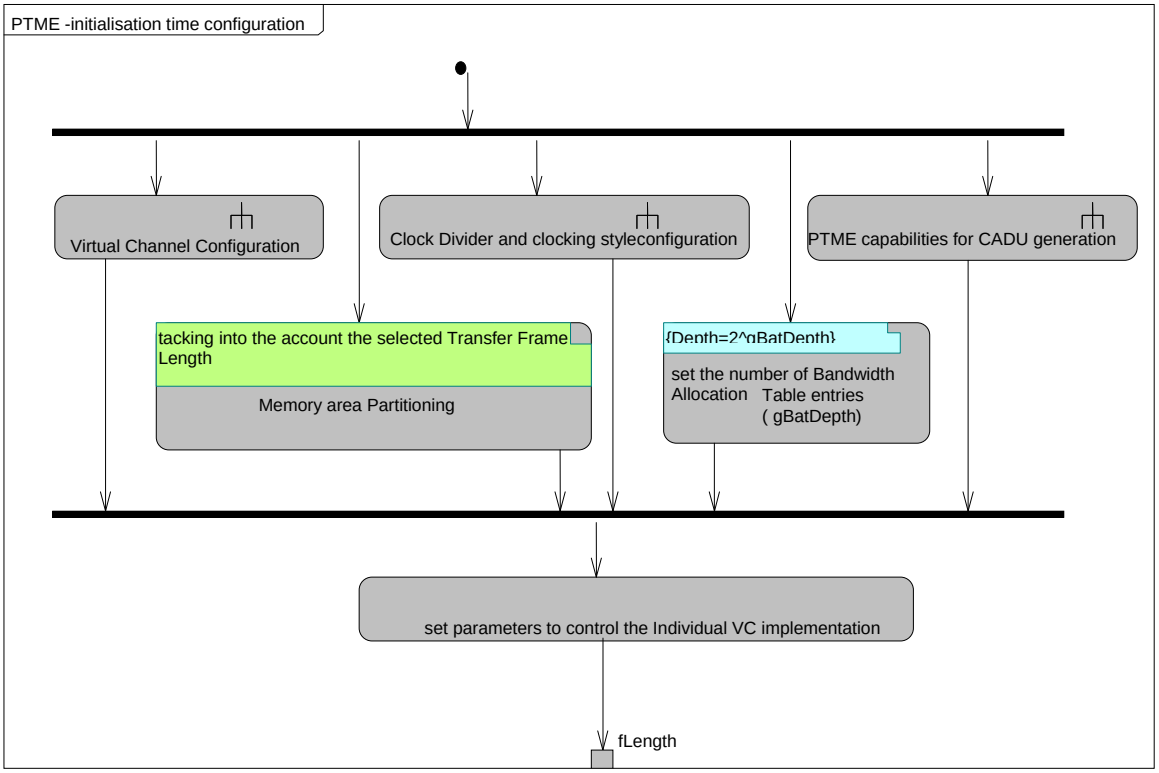

Figure 4.14 - [Activity] Initialisation time configuration

Virtual channel configuration, configuration of each individual virtual channel is done by programming a set of parameters and constants. For example, the number of virtual channels to be implemented is set with the **gNumberOfVCs** constant at initialisation time between one and eight.
The support for generation of Idle transfer frame is selectable, either on a separate virtual channel or any of the implemented virtual channels. Each virtual channel must be assigned an identifier, either automatically sequentially beginning at 0 or via external input. This options are selectable depend on the value of the **gFlexVCID** constant.

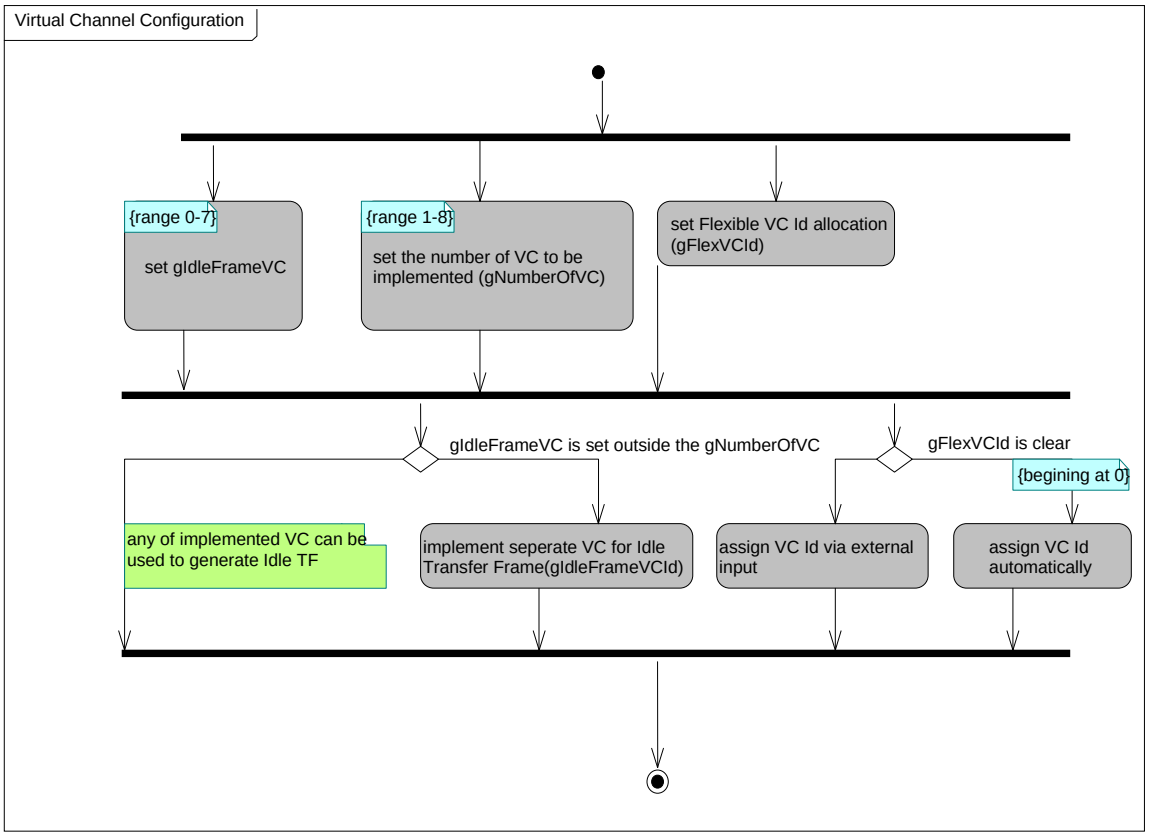

Figure 4.15 - [Activity] set the parameters related to all Virtual Channels

Memory Partitioning, as described earlier virtual channels share a common external buffer memory. The splitting and allocation can be done at initialisation time by setting some constant variables, as it is shown in figure 4.16.

- set the amount of memory to be shared by all VCs (gMemoryDepth)
- set the number of areas into which memory is partitioned(gAreaDepth)
- set the maximum number of memory areas allowed for any VC (gGroupDepth)
- automatic/manual external memory assignment (gGroupInterface)

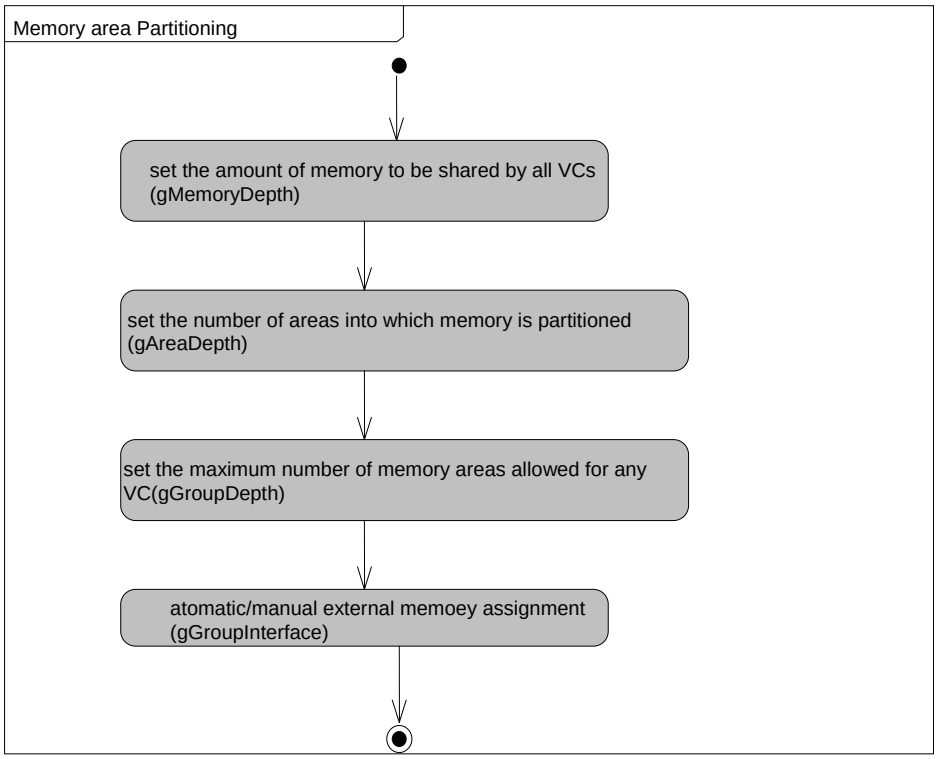

Figure 4.16 - [Activity] Memory Partitioning and allocation inside VCE

CADU generation capabilities: Different options that can be set at initialisation time are as following:

- Support for alternative Attached Synchronisation Marker (ASM) generation
- Support for Secondary Header generation
- Support for Operation Control Field (OPCF) generation
- Support for Frame Error Control Word (FECW) generation
- Support for time strobe generation can be selected
- Support for implementing 223 octet based and/or 239 octet based frame lengths, The Transfer Frame length and trailer type are selected by four input pins instead of using a mode register.
- The size of internal buffer in VCM and
- The interface to the Bandwidth Allocation Table.

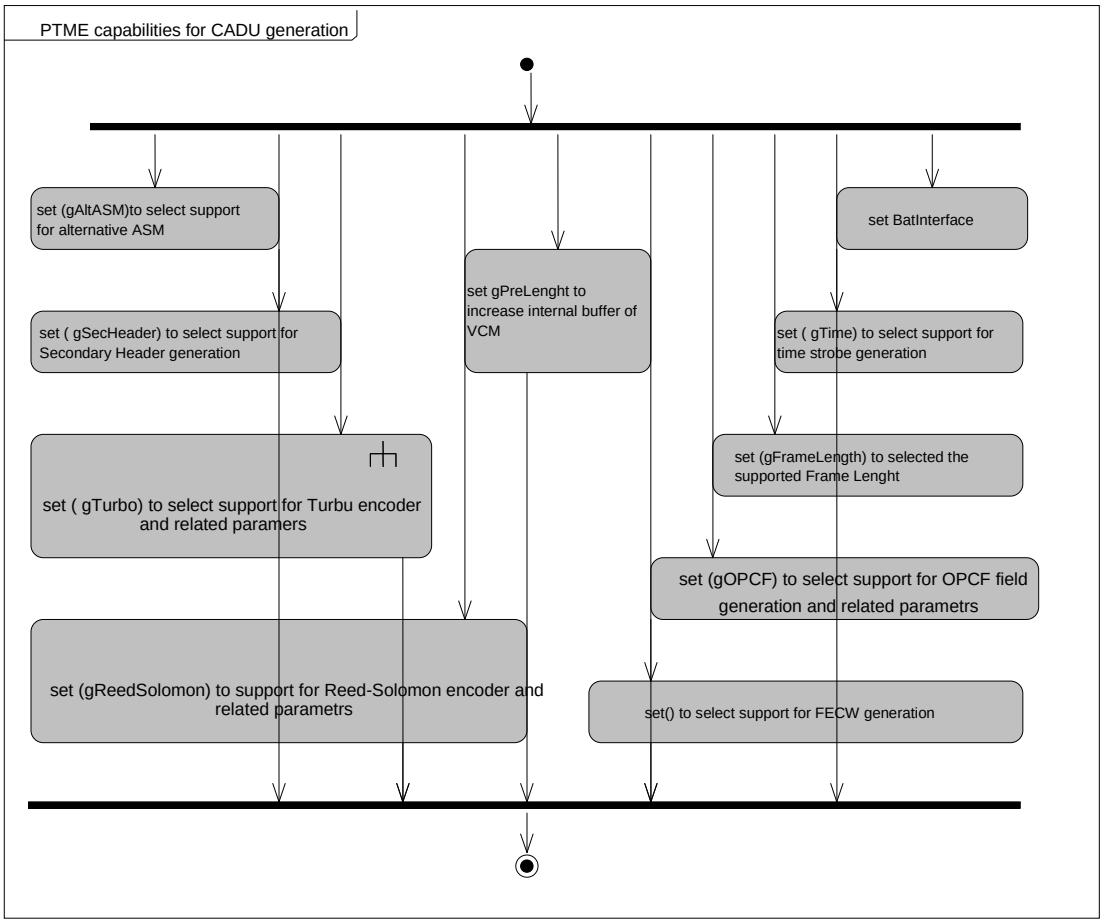

Figure 4.17 - [Activity] Configuration of PTME capabilities

BAT configuration: As mentioned earlier, the Bandwidth Allocation Table (BAT) is used for the arbitration of the downlink bandwidth. The following interfaces can be selected: no interface, asynchronous memory type interface, or BAT is implemented outside the TME (and PTME).

Set parameters to control the individual VC implementation:
The following parameters are configurable at initialisation time for each Virtual Channel, corresponding Virtual Channel Assembler and Virtual Channel Interface.

- **gLenght**: the Internal input buffer size for VCA. The default size is one octet and the value of this parameter is interpreted as an increase of the internal input buffer size.
- **gPacket**: Support for packet handling
- When Packet generation is supported, possibility for packet insertion abort (**gAbort**) and Support for Idle Source Packet insertion (**gIdle**) can be selected. It is also possible to generate First Header Pointer dynamically if the Dynamic FHP input pin is asserted.
- **gReady**: Support for external buffer memory availability indication.

4.6.2 Individual Virtual Channel Operational configuration

The virtual channel encoder module is the functional module responsible for generating the transfer frames. The required configurations before the mission are mentioned earlier regarding the memory allocation, number of Transfer Frame Data fields and so on for different Virtual Channels. The Virtual Channel Encoder block consists of two functional blocks, the Virtual Channel Assembler and the Virtual Channel Multiplexer.

Some configurations are also done for generating the Transfer Frame from telemetry packets during the mission. These configurations are called operational configuration. Operational configurations implement the design option which is selected at initialisation time. They are done for each individual virtual channel in corresponding virtual channel interface, virtual channel assembler(as mentioned earlier for each VC there is one VCI and one VCA) and virtual channel buffer and. The virtual channel interface keeps track of the number of octets received and the packet boundaries in order to detect the start and the end of a packet. This information is used by the virtual channel assembler to calculate the first header pointer. The virtual channel interface also provides the input interfaces which are resides outside the virtual channel encoder module with information whether the virtual channel block is busy processing the input data or if there is a space in the assigned external buffer memory for a new packet (or segment).

Following parameters are generated dynamically by the virtual channel encoder (VCE) module for each virtual channel during the mission. The encoder should be reset after a configuration change.

- Assign the memory allocation identifier to each VC, The identifier of the memory area allocated to a virtual channel is generated by virtual channel buffer. Then the length of transfer frame data fields and the number of octets for a frame, allocated to a Virtual Channel is set.
- All VCAs in a PTME are sized according to the same external buffer memory size, the number of memory areas and the number of areas that can be grouped together for a single VCA.
- Time strobe periodicity
- VCA input buffer size
- Spacecraft Identifier
- Support for generation of First Header Pointer Dynamically
- Set the threshold for External buffer memory allocation to the Virtual Channel
- Set Idle packet version and insertion timeout values.

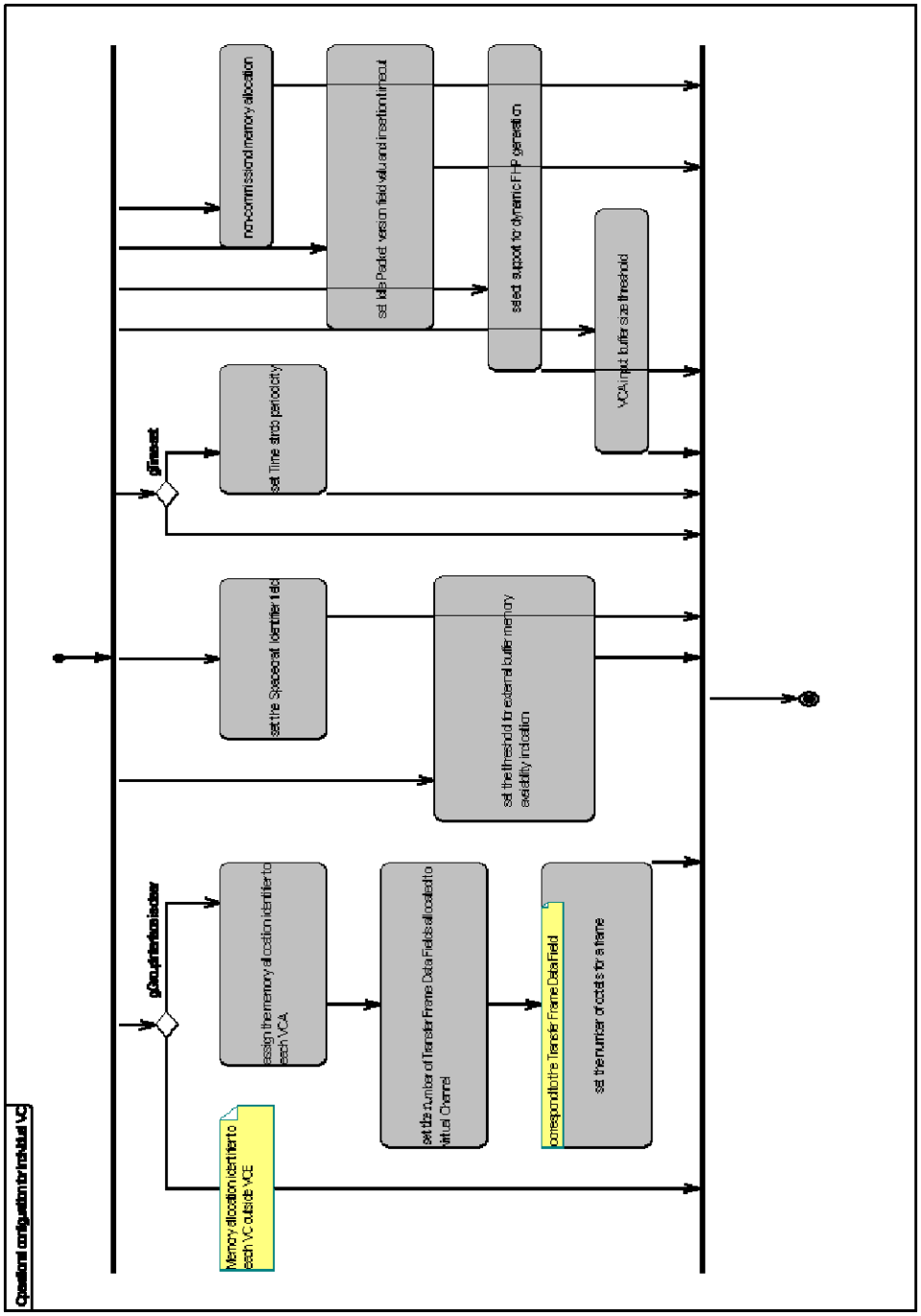

Figure 4.18 - [Activity] TransferFrameGeneration operational configuration

Following parameters are set via the corresponding input ports:

- **DynamicFHP** input port: Dynamic or static First Header Pointer calculation
- **MaxFramePtr** input port: Each VCA can then be allocated an individual number of frames that can be stored in memory. number of Transfer Frame Data Fields allocated to a Virtual Channel
- **MaxOctetPtr** input port: the number of octets that can be stored in one such frame. Note that there must be at least three frames allocated to each. number of octets allocated per Data Field for a given Virtual Channel
- **Rdy Threshold input port:** Threshold for external buffer memory availability indication
- **PktVersion input port:** Idle Packet version field value (000, 100).

4.6.3 Transfer Frame Generation

Depending on the design specification; different fields in the header of the Transfer Frame are varied during the operations. These fields are filled partly by the virtual channel assembler and are completed by virtual channel multiplexer. Figure 4.19 illustrates the activities for generation of transfer frame.

TME provides several different input interfaces to the virtual channels, connected to virtual channel interfaces and subsequently virtual channel assemblers on the telemetry encoder chip set. The telemetry data received from the onboard users through input interfaces can either be packets or a bit stream and are stored temporarily in pre-allocated external buffer memory slots via the virtual channel buffer and it is subsequently inserted into transfer frame. These data are inserted in data field of transfer frames. Virtual channel buffer multiplexes the read and write accesses on to a common external buffer memory interface.

The part of transfer frame which is called “Partial Transfer Frame” and generated by the virtual channel assembler consists of the last three octets of Transfer Frame Primary header fields, optionally the secondary header fields and the Data Field. The generation of Partial Transfer Frame which is only an internal structure, not defined in Packet Telemetry Standard is illustrated in figure 4.20. The output is stored in the external buffer Memory. In order to calculate the First Header Pointer (FHP), the virtual channel assembler retrieves the required data from corresponding virtual channel interface and put it in the FHP field in the header. The Partial Transfer Frame header, generated by VCA will be stored in the memory into the pre-allocated slots, together will Transfer Frame Data Field.

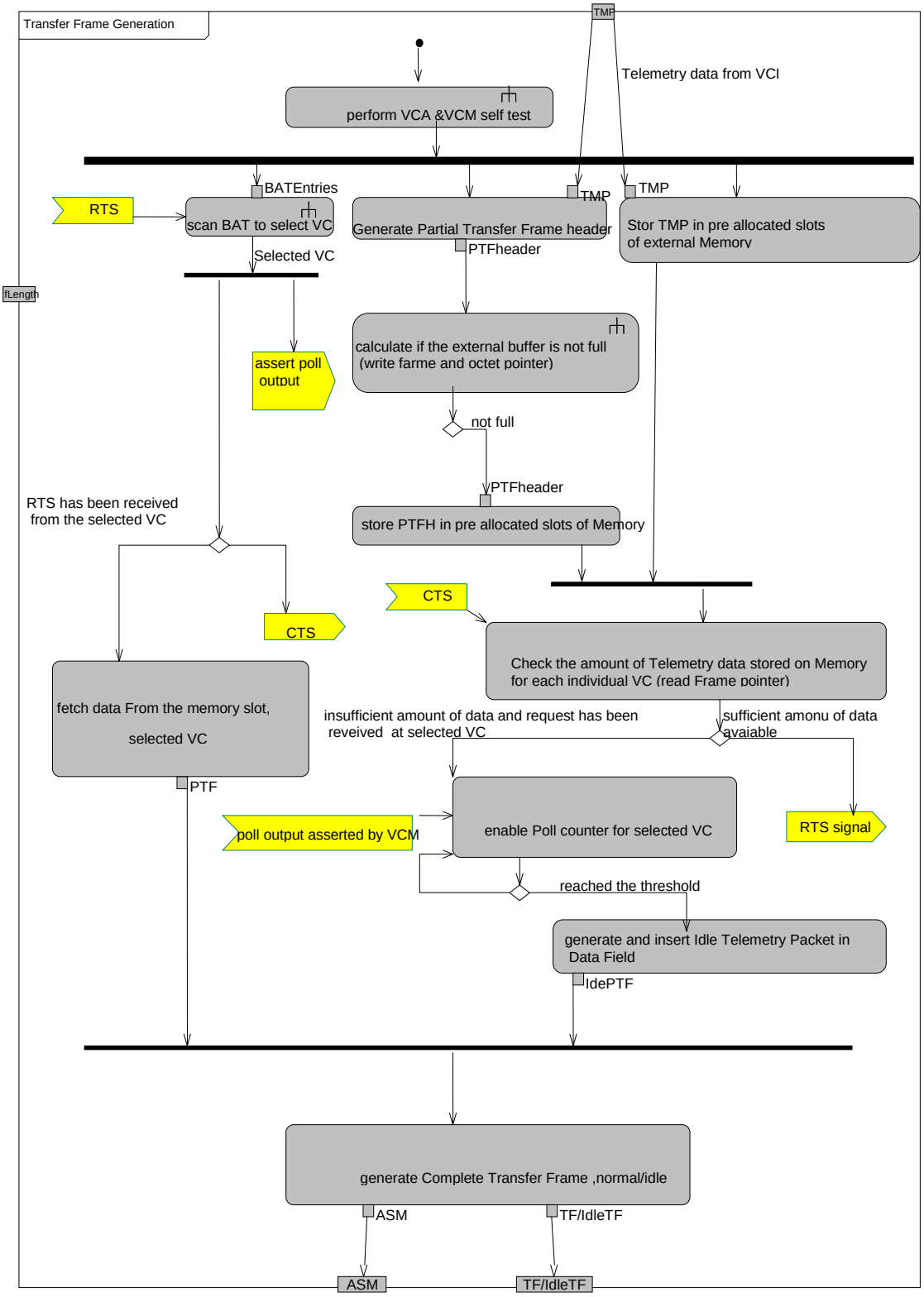

Figure 4.19 - [Activity] Transfer Frame Generation

Several parts of the system are responsible to perform this activity, so there are some concurrent activities and some sequential activities. These are separated depending on the interactions between the parts. The send and received events (signals in this example) which are indicated as yellow shapes are used to represent the synchronisation between the parts when they should interact. As mentioned earlier the execution of an activity may also depend on receiving the required input from other activity.

Each VCA informs the VCM when there is an enough data in the external buffer memory to fill the complete data field of the Transfer Frame. This is done by sending a ready to send signal (RTS). Meanwhile the Virtual Channel selection mechanism is performed by VCM selection algorithm as mentioned earlier. For selected VC it will check if RTS signal has been received or not. If so, it means that sufficient amount of data is provided by corresponding VCA, the data will be fetched by VCM to complete the Transfer Frame. In other case when no RTS signal has been received, depend on algorithm, the VCM will send the request to VCA to output the required Partial Transfer Frame and also assert a Poll signal. When VCA receives the request, first it will check if the stored data in its local buffer is sufficient to generate the Transfer Frame or not, which normally result in negative case, so the Poll counter will be enabled to count how many poll signals is received from VCM. When the number of polls reached the threshold, the VCA will generate the Idle Source Packets to fill the Transfer Frame data field and store them in memory until the data field is completed. The VCM fetches the Idle source Packets from memory and generate the complete Transfer Frame.

To configure Frame data Field Status parameters following input ports are used by VCA:

- SecHeader input port is only used as a data bit for the FDFS and does not affect the sizing of the Transfer Frame Data Field in any sense.

- The Sync input port is used as a data bit for the FDFS and is also used as one of the qualifiers for the Idle Source Packet insertion.

- The PktOrder input port is only used as a data bit for the FDFS.

- The SegmentLen input port is only used as data bits for the FDFS and does not affect the external buffer memory availability signaling.

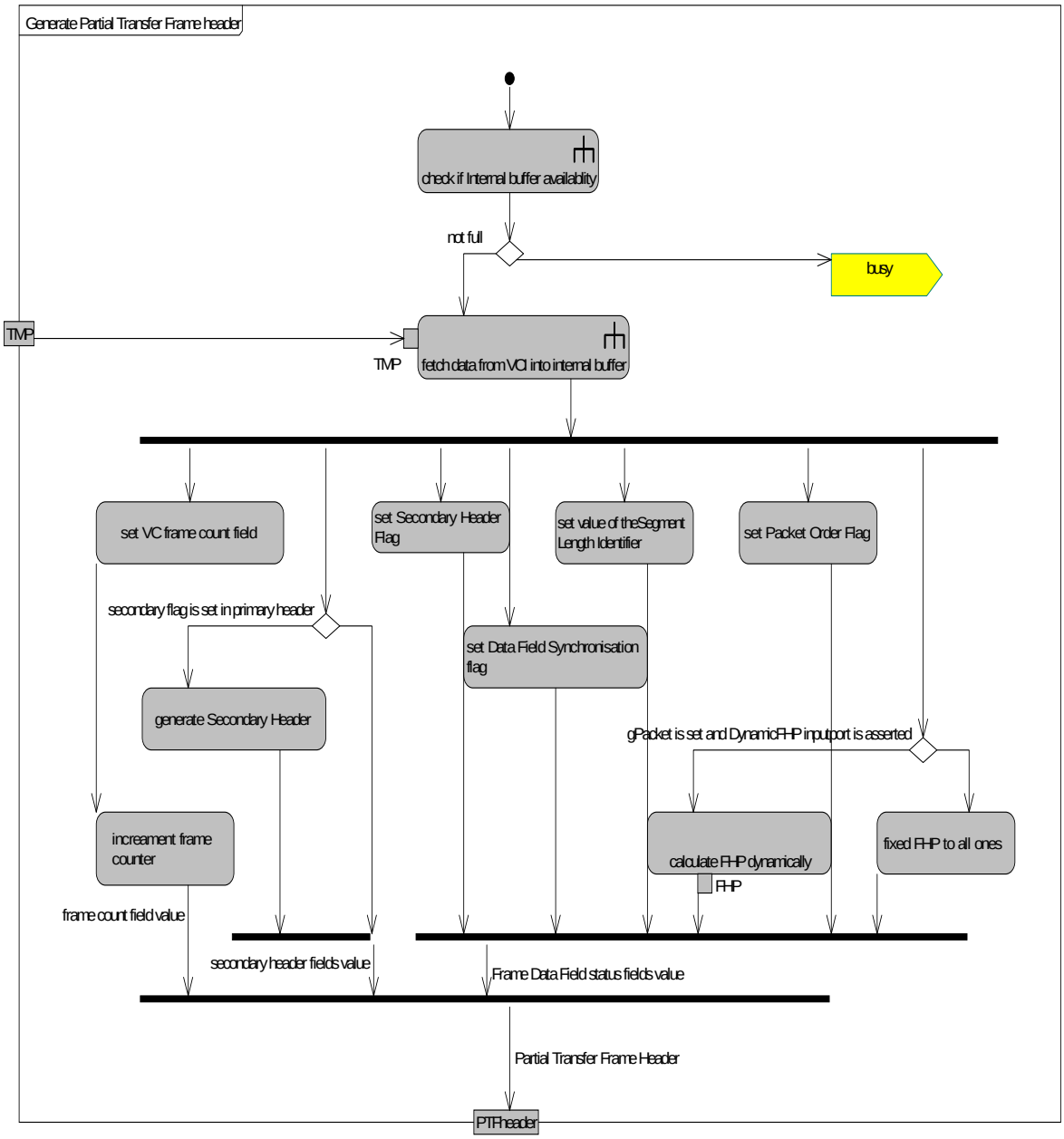

Figure 4.20 - [Activity] Generate Partial Transfer Frame header

Generate Complete Transfer Frame. Serial data generated by selected Virtual Channel Assembler is an input to the Virtual Channel Multiplexer. The VCM multiplexes the serial data, generates a Synchronizer Marker and parts of Transfer Frame Header and the optional Transfer Frame Trailer to complete the Transfer Frame. Figure 4.21 shows the activities for generating the complete transfer frame. Generation of the trailer which is a part of transfer frame is presented in a activity diagram which is shown in figure 4.22.

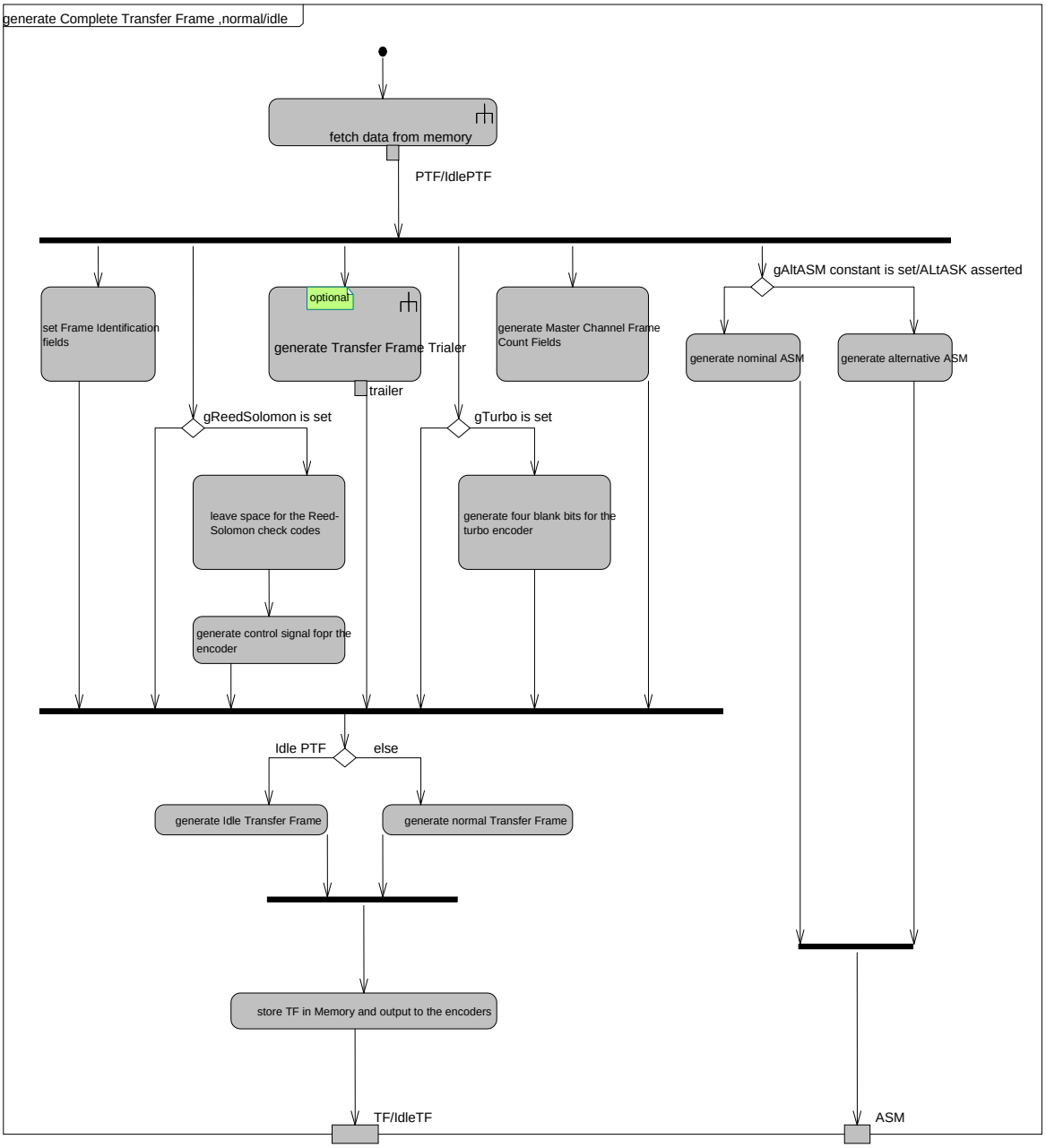

Figure 4.21 - [Activity] generate Complete Transfer Frame, normal/idle

Frame Identification Fields and Master Channel Frame Counter are parts of the Header that is generated by the VCM. The Master Channel Frame Count retrieved from the master Frame counter which is incremented by one for each frame that is output. The optional fields of the trailer are a 16-bit Frame Error correction Word (FECW) and an Operational Control Field (OPCF). Generation is selectable by means of setting constants at initialisation time and it is also possible to enable or disable the insertion at run time.

FECW consists of the CRC checksum over the whole Transfer Frame, excluding the FECW and Attached synchronisation Marker and it is mandatory when Reed-Solomon encoding is not used.

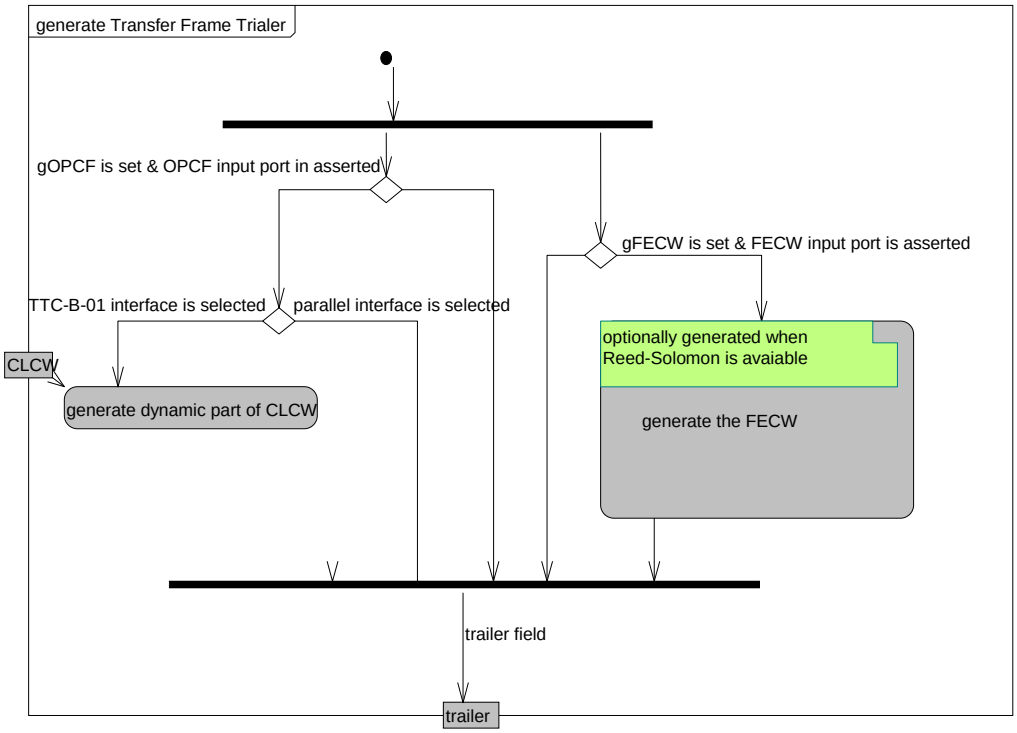

Figure 4.22- [Activity] generate Transfer Frame Trailer

The VCM will produce the alternative Attached Synchronization Marker (ASM) when the ***AltASM*** input port is asserted and the ***gAltASM*** constant is set at initialization time. Otherwise only the nominal ASM will be generated. The Spacecraft Identifier is set with the ***SCId*** input port. The individual Virtual Channel Identifiers can be set via the ***FlexVCId*** input port.

The VCM can generate blank space for insertion of Reed-Solomon check symbols by the Reed-Solomon Encoder and four blank bits for the trellis termination process in the Turbo Encoder. The output at this stage is the complete Transfer Frame with the Synchronisation Marker and optionally space for Reed-Solomon encoder in form of serial bit stream.

4.6.4 Channel coding, Synchronization and Pseudo-randomization

The standard prepared by the ECSS-E-ST-50-01C Working Group specifies several space telemetry channel coding schemes. (Synchronisation and channel coding)
The order of different encoding schemes and modulation is dependent on the implementation. Synchronization is combined with the different coding options. Synchronization applies to all telemetry channels, coded or uncoded by attaching an attached sync marker (ASM) to the codeblock or transfer frame and immediately

proceed. Correct decoding of codeblocks and processing of the transfer frames are depend on accurate synchronization.

The Virtual Channel Multiplexer produces and sends the 32 bit attached synchronisation marker before outputting the Transfer Frame. Either nominal or alternative ASM can be produced at a time. The alternative ASM is produced if selected at initialisation time. Otherwise the nominal ASM will be generated.

The ASM is not a part of the encoded data, so it shall not be presented to the input of the encoders or decoder. It is attached to the codeblockes. Synchronization of the Reed-Solomon or turbo codeblock (or transfer frame, if the telemetry channel is not Reed-Solomon coded or turbo coded) is achieved by using a codeblocks (or transfer frames) with an attached sync marker (ASM) between them. The data unit that consists of the ASM and the stream of fixed-length Reed-Solomon or turbo codeblock or transfer frame is called the channel access data unit (CADU)

The codeblocks or transfer frames are randomised with a standard pseudo-random sequence to ensure sufficient randomness in order to achieve minimum bit transition density. The presence or absence of pseudo-randomization shall be fixed at initialisation time. Randomization is applied after turbo encoding or Reed-Solomon encoding (if either is used), but before convolutional encoding (if used). Pseudo-randomization is combined with synchronization and with the different coding options. When using Reed-Solomon encoding the Transfer Frame‘s length is variable correspond to the interleaver factor.

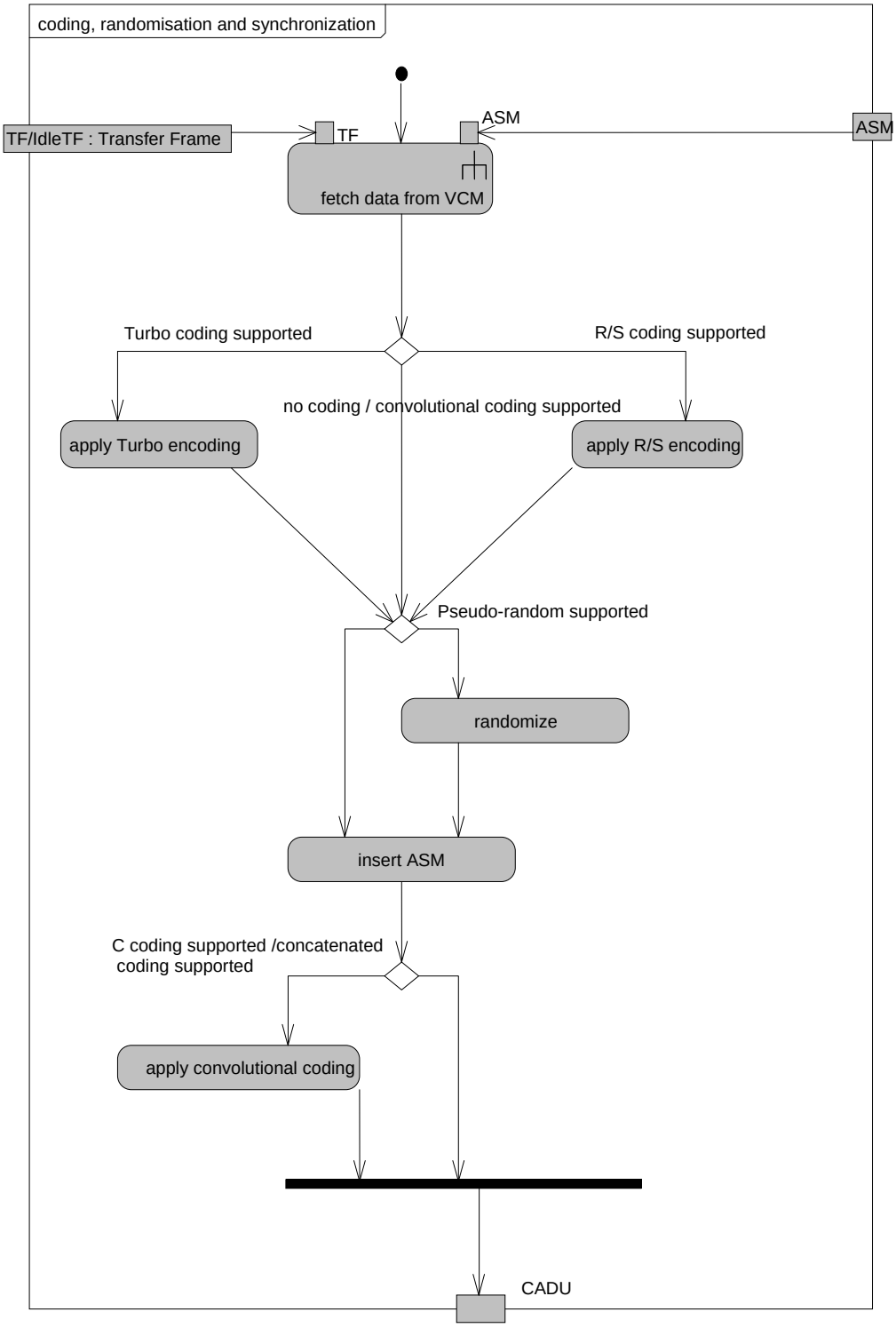

Figure 4.23 - [Activity] coding, randomization and synchronization

4.7 Requirements Analysis

In section 4.4 the partitioning of the system requirements is done, containment and derived relationships are captured. Complex requirements are decomposed into sub-requirements, using composite relationship. In this section the cross-cutting relations between the system requirements and other model elements are captured for traceability purposes. Now it is simpler for different stakeholders to express their problems by studying the requirements diagram. It is easier to make sure that the requirements are satisfied. The “trace” and “satisfy” relations are used the most. Some requirements may be traced to more than one block. On the other hand one block may satisfy more than one requirement. After defining the cross-cutting relations, there might be a need to refine the corresponding requirement. The new requirement is related to the original one using the refine relation. The assumptions which are made to create the refined requirements are captures as rationales and connect to the relation.

Figure 4.24 is an example of requirements diagram in which the cross-cutting relations are captured.

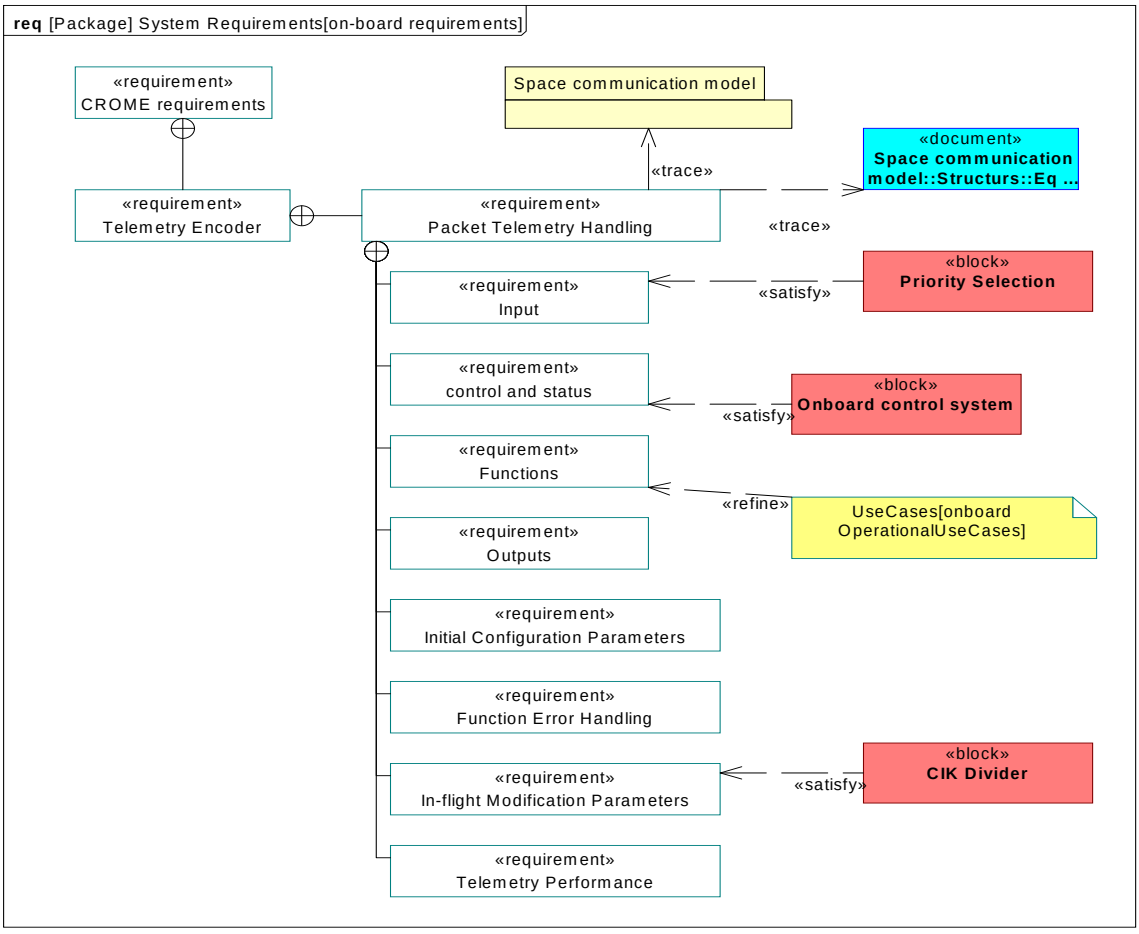

Figure 4.24 - [Package] System Requirements[on-board requirements]

4.8 Critical System Properties and Constraints

Value properties of system components are used to capture the non-functional requirements of the system. To define the Constraints on value properties the first step is to identify these properties. I could not find relevant information in documents to model the Telemetry Encoder Parametrics, so here I can't put sample diagrams. This part is left for the future work of modeling the PTME system.

5 Related work

5.1 Simulation-based design

As mentioned earlier, SysML provides graphical representations for modeling system requirements, behavior, structure and integration with engineering analysis. Two papers written by Peak et al. [17], [18] introduce SysML parametrics and illustrate how they can be used in simulation-based design. The papers introduce functional principles of SysML parametrics in general and its applications to engineering design and analysis in particular. In the first part SysML parametric concepts, analysis libraries are introduced. The definition of reusable equations and corresponding SysML constraints blocks and constraint properties to capture the analysis libraries are also discussed. In the second part the way to reuse the analysis building blocks (captured by SysML constraint block) in simulation-based design is mentioned along with some examples. Composable objects (COBs) are concepts that have been developed at Georgia Institute of Technology to support the execution of SysML parametric models. COB technology provides a conceptual foundation for SysML parametrics by combining object and constraint graph techniques. COB can be used together with SysML to capture engineering knowledge for further analysis in modular and reusable form. Engineering analysis models are captured in SysML and executed by analysis tools.

5.2 Model-based Requirements engineering

System requirements are derived from user requirements which need to be described in detail, analyzed, documented and managed throughout the system development life cycle. This process is called Requirements Engineering which is an important phase and the most critical process when developing a system.

Requirement management process includes capture, analysis, refinement and evolution management [11]. In UML world “Requirements Modeling” which is part of Requirements management is done using a new profile called SysML. Bruno Traverson in [11] proposed a modeling approach for requirements, based on three levels of description. These levels are Product Vision to specify the goals of the system by capturing the problems, Functional Perimeter to describe the system functions through Use cases and the third level is Requirements. For the third level, requirements are divided into three, functional, extra-functional and constraints. Unlike functional requirements which are specific processing rules or behavior, extra-functional refers to quality properties that the system must have as the IEEE 830.1198 [19] call them “System Attributes”. These properties are performance, security, usability and so on. In this paper the goal is to explore the link between requirements expressions in SysML (SysML requirements Diagram) to Requirements Management objectives.

In [19-20] Pietro Colombo describes the integration of Problem Frames concept which is an approach to requirements modeling with SysML. Following the PFs approach the first task is to understand and represent the various problem domains,

the connections between domains and system under development and its connection to the problem domain by means of context diagrams. These diagrams are defied in SysML. Next is to add requirements to the context diagram by means of problem diagram. According to [21] “a PF is a description of recognizable class of problems”. So far no tool has implemented Problem Frame and there is no support by notation, these issues limit the use of this concept in industry.

In [22] Mihai Fonoage describes the mechanism for selecting the components and having automatic design by defining structured representation of requirements and component then introducing an automatic process for the architectural design. The proposed methodology is part of the Requirements-Driven Design Automation framework that has been developed for component based system development [23]. The goal is to reduce the development time and cost by detecting requirement specification errors before the design stage. A methodology for requirements specification, representation of functional requirements by corresponding models and automatic validation for completeness and consistency are described in [1]. SysML has been adopted for the first phase and the validation is done with rule-based system which is implemented in Prolog.

5.3 Modelling, validation and verification

Unsatisfied properties are detected by verification and validation to assess the design. In [24] a unified approach for verification and validation of System and software engineering design models is proposed. The unified paradigm is expressed in SysML. Formal verification, program analysis and software engineering are three major techniques that have been combined and used to achieve the objective. Model checking is performed by formal verification technique. The flow of data and control is analysed by program analysis and software engineering refers to measurement techniques for quality attributes in software. In [25] the verification of system’s safety requirements are discussed by combining SysML as a non-formal method and formal methods of the discrete event system theory which are recommended by safety-related standards such as IEC61508. Requirements refinement and allocation is captured in SysML diagrams and what they have proposed to proof these relationships in some formal way for verification is to modify the SysML meta-model and extend it by means of stereotype to introduce logic properties.

In [26] the lack of formality of SysML modelling language for the verification of requirements before the system implementation is addressed. The proposed approach for verification is to use SysML to describe the structure of the system and the requirements modelling. Petri nets and Temporal logic (LTL) are used to formalize the system behaviour and requirements for automatic formal requirements verification. Temporal logic is used to formalize the SysML requirements diagram and instead of SysML Activity Diagram, Petri nets are used to provide a formal description of the component behaviours. In [27] the validation of embedded real-time systems (ERTS) with energy constraints is discussed. SysML is considered in combination with MARTE as specification language. MARTE is used to support for quantitative notations and to overcome the lack of formality for validation SysML Activity Diagram in translated into Time Petri Net with Energy constraints. Energy consumption and execution time are two critical constraints of ERTSs that

are represented by MARTE profile and validated by Time Petri nets in this proposed approach. Following this approach, the properties of the system can also be verified. Further research needs to be done in order to automate the generation of Time Petri nets from SysML Activity diagram and also to cover other diagrams such as sequence diagram and state machine diagram.

5.4 SysML Profile for System on chip (Soc) Design

In [8] a SysML profile with suitable extensions for modelling Systems-on-chips (SoCs) is proposed. The transformation of the model into SystemC code is also discussed. SystemC is an IEEE standard language, based on C++ and it is used for describing systems at the very high level of abstraction. SystemC codes are executable and this feature provides the possibility to simulate the system [28].

SysML block definition diagram, Internal Block Diagram and parametric Diagram has been used to model the structure and express the constraints for further analyses. The allocation feature of SysML described within Activity Diagram to allocate behaviour to parts (Different components of the system). According to the paper the next step is to define basic entities of corresponding SystemC like modules, ports and processes and map these SysML to SystemC as follows:
“SysML parts to SystemC modules, SysML flow port to SystemC ports and SysML allocations to SystemC processes.” The procedure is followed by creating a header file and building the implementation files while translating the SysML model into SystemC code. The later is done by generating an XMI file through the SysML tool. This file serves as the textual description of SoC that must be transformed into SystemC code.

6 Summary of the PTME SysML model

The purpose of this section is to summarize the SysML model of the Packet Telemetry Encoder (PTME). Lessons learned and the conclusions of the case study are presented in the next section.

The overall goal of the case study was to assess the usefulness of SysML for specification of hardware-based designs. This was done by developing a functional design specification of the PTME in SysML. The input to this modeling exercise was a set of text-based design and requirements specifications, providing detailed descriptions of the PTME functions at various levels of abstraction.

SysML includes nine diagram types. Six of these were used in the PTME model: the use case diagram, the block definition diagram, the internal block diagram, the activity diagram and the requirement diagram and package diagram. The model consist of two package diagram, two use case diagrams, four block definitions diagrams, three internal block diagrams, eleven activity diagrams and one requirement diagram.

The model of the system is organized into several packages all captured in one package diagram. The packages include use cases, requirements, structural components, various value and item types, and activities.

The use case diagrams provide a high-level view of the PTME functionalities. The use case modeling has been done at two levels. One use case diagram describes the top-levels functions in the entire packet telemetry system from the viewpoints of two actors, the ground control system and the on-board control system. The other use case diagram shows the main functions in the on-board system with the application software running on the on-board computers as one actor, and the PTME as another actor.

A block is a SysML concept used to represent the system components, subsystems and other entities in the PTME system. Value properties are used to describe the attributes of blocks. For example, the transfer frame is modeled by a block, which has a length as an attribute which is indicated as a value property of the block. Four block definition diagrams are used to capture the context and the structure of the system. The environment in which the PTME will operate is illustrated in one block definition diagram. The purpose is to capture the system context. The structure of two data types, the telemetry packet and the transfer frame are shown in two block definition diagrams. One block definition diagram describes the PTME and its parts.

Internal block diagrams are used to capture the interactions between different parts and subsystems in the model. One such diagram is used to illustrate the interconnections between the entities of the PTME system and the environment, capturing the data flow between different components. There is one internal block diagram that shows the interactions between the PTME components and one that describes the interactions between the internal parts of the PTME.

The input data to the PTME is a telemetry packet and the output is a modulated channel access data unit. The internal parts of PTME include the telemetry encoder, modulators, randomizer, clock divider and other encoders.

The behavioural aspects of the PTME system are captured in Activity diagrams showing the interactions among the different parts of the system that are defined in the internal block diagrams. Eleven activity diagrams are used to capture the various actions involved in producing a channel access data unit. These diagrams are related and when we follow the associations between the activities, functions are divided into sub functions at the next level. For example when describing the creation of Transfer Frames, six activities are shown, each containing other activities that are captured in separate activity diagrams.

Requirements modeling enables text based requirements to be captured by graphics in requirement diagrams, tables or tree structures, which can be traced to other model elements using the satisfy, derive, verify and refine relationships. There is a very large number of requirements for the PTME system. Translating all of these requirments and their relations into requirements diagrams was beyond the scope of this work. However, to illustrate how textual requirements can be expressed in SysML, I created one package diagram that shows the requirement hierarchy for the CROME ASIC. This diagram illustrates how cross-cutting relations between different requirements can be captured.

7 Conclusions

This thesis presents an overview of the SysML modelling language and a case study where I developed a partial design specification of a packet telemetry encoder in SysML. The purpose of the case study was to assess the usefulness of SysML for specifying requirements and documenting the high-level functional design of hardware units. The main conclusions and lessons learned are as follows:

- I found the SysML language intuitive and easy to learn and use. The book by Friedenthal et al. [7] is easy to read and very useful for both beginners and advanced users.

- The Artisan real-time studio modelling tool has a user friendly interface and its documentation is well written. The tool supports modelling with both SysML and UML. It is also possible to synchronize the tool with DOORS to import and export requirements. However, this feature was not explored due to a problem with file access rights and the synchronization module.

- The main challenge in the case study was to understand the requirements and the functional design of the PTME system from the documents that were provided as input for the case study. Once these were understood, it was a relatively easy task to create the SysML model.

- The PTME model contains only a few use case diagrams. They were useful for providing a high-level view of the system, but were not essential for capturing the requirements of the PTME system.

- Modelling the structure of the PTME system using block definition diagrams and internal block diagrams was straight forward and consumed relative little time.

- Capturing the behavioural aspects of the system was the most time-consuming part of the work. For each block diagram, several actions were modelled using activity diagrams. The activity diagrams provided an efficient means for modelling data flow and interactions between different blocks.

- SysML follows a profile based approach to extend the language, this feature enhance the capability of system to add more domain-specific stereotypes to customize the language and reuse purposes. To expand the profile it’s very important to create meaningful stereotypes. Knowledge and experiences of system engineering are needed to define these new constructs.

- Allocation of the system requirements and mapping them to each other as well blocks and activities enhances traceability. This capability of SysML is used to keep track of changes either in the requirement’s specification or the component models. The language does not provide any guidance for how to map requirements to model elements, so the modeller should preferably be an expert in the field of requirements engineering. Sometimes it is awkward to capture requirements as a block in a Requirements diagram. The alternative is to use the tables and matrixes. There, we can capture the relation just by adding more columns.

SysML supports specification, design, analysis, early verification and validation of software-intensive systems. The structure, the behaviour and the requirements are captured separately, and then the analyses are performed. The purpose is to figure out the relations between the diagrams and consequently different accepts of the system (structure, behaviour and requirements). The result will help the engineers to perform the verification and validation before the actual implementation.

The limitation of study is due to the time. I could not cover all the elements that interact with Packet Telemetry Encoder. So some parts of the modelling have been left behind. Creating SysML Parametric Diagram has not been covered, because I could not find the relative information in the design specification of PTME. Another capability of the SysML which I couldn't cover when doing the case study is using the view and view point concepts.

8 References

[1] M. Mura, L. G. Murillo, M. Prevositini,"Model-based Design Space Exploration for RTES with SysML and MARTE", *Forum on Specification and Design Languages* (FDL´08), Stuttgart, Germany, IEEE CS Press, 2008, pp.203-208, doi: 10.1109/FDL.2008.4641446

[2] http://www.aadl.info

[3] http://www.omgmarte.org

[4] http://www.artisansoftwaretools.com/community/standards/marte/

[5] http://www.modaf.org.uk/2background/55/what-is-modaf-and-why-use-it

[6] M. Hause, "An Overview of UPDM – A Unified Profile for DoDAF/MODAF," http://www.mil-embedded.com/articles/id/?3653

[7] S. Friedenthal, A. Moore, R. Steiner, *A Practical Guide to SysML*, Morgan Kaufmann Publishers, 2008

[8] M. Prevostini, E. Zamsa,"SysML Profile for SoC Design and SystemC Transformation", Faculty of Informatics, University of Lugano, Switzerland, May 11, 2007; http://www.prevostini.ch/tech_report/Tech_Report_003.pdf

[9] *IEEE Std. 1471-2000, Recommended Practise for Architectural Description for Software –Intensive Systems*, IEEE, 2000.

[10] *ISO/IEC 10746, Open Distributed Processing- Reference Model, Part 1-4*, 1995.

[11] B.Traverson, "Linking Requirements to Enterprise Viewpoint Specification Using Correspondence Rules", 11th Int. IEEE EDOC Conference Workshop, 2008, pp. 254-259.

[12] E. W. Dijkstra," On the Role of Scientific Thought", *Selected Writings on Computing: A Personal Perspective*, Edsger W. Dijkstra, Springer-Verlag, 1982

[14] M. Hause, "Cross-Cutting Concerns and Ergonomic Profiling Using UML/SysML", INCOSE Int'l Symposium, 2006

[15] http://www.papyrus-uml.org

[16] http://www.telelogic.com/products/tau/index.cfm

[17] R.S. Peak, R.M. Burkhart, S.A Friedenthal, M.W. Wilson, M. Bajaj, I. Kim, "Simulation-Based Design Using SysML—Part 1: A Parametrics Primer", INCOSE Int'l Symposium, San Diego, 2007.

[18] R.S. Peak, R.M. Burkhart, S.A Friedenthal, M.W. Wilson, M. Bajaj, I. Kim, "Simulation-Based Design Using SysML— Part 2: Celebrating Diversity by Example", INCOSE Int'l Symposium, San Diego, 2007.

[19] *IEEE Std. 830-1998 Recommended practise for Software requirements specification*, IEEE, 1998.

[20] P. Colombo, V. Del Bianco. L. Lavazza, A. Coen-Porisini, "A Methodological Framework for SysML:a Problem Frames-based Approach," 14th Asia-Pacific Software Engineering Conference. IEEE press,2007, pp. 25-32, doi:10.1109/ASPEC.2007.56

[21] M.Jackson, *Software Requirements and Specification: : a lexicon of practice, principles and prejudices*, Addison-Wesley, 1995.

[22] M. Fonoage, I. Cardei, Ravi Shankar, "Mechanisms for Requirements Driven Component Selection and Design Automation", 3rd IEEE Systems Conference, 2009, pp. 1-6, doi: 10.1109/SYSTEMS.2009.4815761.

[23] I. Cardei, M. Fonoage, R. Shankar,"Framework for Requirements-Driven System Design Automation", 1st IEEE Systems Conference,USA, April 2007, pp. 1-7, doi: 10.1109/SYSTEM.2007.374671.

[24] L. Alwneh, M. Debbabi, F. Hassaine, Y. Jarraya, A. Soeanu, "A Unified Approach for Verification and Validation of Systems and Software Engineering Models", 13th Int. Symp. and Worshop of Computer Based Systems (ECBS´06), IEEE CS Press, 2006, pp. 409-418, doi: 10.1109/EBCS.2006.17

[25] D. Evort, J-F. Petin, G. Morel, "Combining SysML and Formals Method for Safety Requirements Verification". Insight Journal of INCOSE 11, no. 3, 2008: http://hal.archives-ouvertes.fr/docs/00/34/48/94/PDF/INSIGHT_EVROT_Vfinale.pdf

[26] M. V. Linhares, R. S. de Oliveira, J-M. Farines, F. Vernadat, "Introducing the Modelling and Verification process in SysML". IEEE Int. Conf. On Emerging Technologies and Factory Automation, 2007, pp. 344-351, doi: 10.1109/EFTA.2007.4416788.

[27] E. Andrade, P. Maciel, G. Callou, B. Nogueira, "A Methodology for Mapping SysML Activity Diagram to Time Petri Net for Requirement Validation of Embedded Real-Time Systems with Energy Constraints", IEEE 3rd Int'l Conf. on Digital Society, 2009, pp: 266-271, doi: 10.1109/ICDS.2009.19

[28] W. Raslan, A. Sameh, "Accelerating High-Level SysML and SystemC SoC Designs", The 2007 IP Based Electronic System International Conference, IP-SoC 2007, Grenoble, France, December 2007: http://www.design-reuse.com/articles/17562/high-level-sysml-systemc-soc-designs.html

[29] OMG, "UML profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE)", http://www.omg.org/spec/MARTE/1.0/

[30] http://www.omgsysml.org

[31] http://www.incose.org

[32] http://www.artisansoftwaretools.com

[33] http://www.ilogic.com

[34] http://www.architectureframework.com