UML/MARTE Methodology for System design

April, 2014

Microelectronics Engineering Group
TEISA Dpt. , University of Cantabria
Authors: P. Peñil

Index:

Index of Figures:

The complexity of embedded, parallel systems and platforms requires design methodologies that, based on separation of concerns, enable the design teams to work in an efficient way. Separation of concerns enables the specialization of the design process; separate but collaborating sets of designers can deal with different system concerns (application modelling, HW/SW platform design, etc), improving the development process. Therefore, well-defined system concerns in the same model enable designers to focus on their designing domain, guaranteeing system consistency by using the same specification language, producing synergy among different design domains.

Support of this separation of concerns is covered in two steps. First, system models are divided into three sub-models, following the Y structure commonly applied in the latest design methodologies. In these flows, designs start with the definition of the two main starting points: HW platform and expected system functionality, and evolve to define how to support the functionality in the HW platform.

Following this structure, the system model is composed of different sub-models defined according to the features they must capture:

- The Platform Independent Model (PIM), which describes the functional and non-functional aspects of the system functions (e.g. application, functional code).
- The Platform Description Model (PDM), which describes the different HW and SW resources that form part of the system platform.
- The Platform Specific Model (PSM), which describes the system architecture and the allocation of platform resources.

Using these sub-models, the UML/MARTE system design activity takes charge of all modelling tasks required for initially defining the system under development, especially in the following aspects:

1. Data types
2. Modelling the code files.
3. Communication interfaces
4. Channel types
5. The system application, definition:
6. The functionality associated with each application
7. The concurrent structure of the application components
8. The communication media to interconnect the applications
9. Memory partitions
10. The allocation of the applications into memory partitions
11. The system platform, both at HW and SW resource level.

12. The system architecture:
13. Instantiating HW/SW platform components,
14. Defining the allocation of the memory partition into the platform sources
15. The environment that interacts with the system

However, the integration of all these aspects into the three sub-models is too complex to be done. This is because UML models are based on graphical descriptions and so the number of elements that can be described in a model must be limited in order to maintain the benefits of the visual methodology. As a result, the three models are also sub-divided into parts, which are called views. Each of the previous modelling tasks are dealt with by using a model view.

The next sub-sections describe the views, what they are used for and the process defined to create them.

Definition of model views

There are different model views:

- **Data View**: defines the kind of data types used for the information exchange among the system functionalities. The view is mandatory.
- **Functional view**: this view includes the specification of the interfaces provided/required by the application components in order to be connected amongst themselves. Additionally, the view includes the specification of the files that contains the implementation (functional source code) of each application component. The view is Mandatoty.
- **Application View:** includes the definition of the application components and the application structure. Additionally, the view includes the association of the functional files defined in the *FunctionalView* with each application component. The view contains a "System" component that is used for specifying the application structure. It includes application components interconnected by using the interfaces defined in the *FunctionalView* and the communication mechanism defined in the *CommunicationView*. Mandatory.
- **Communication view**: captures the set of communication channels used for interconnecting the different application components. Additionally, the view includes the mechanisms used for synchronizing threads and processes. The view is optional if no communication media are considered.
- **Memory Space view**: defines the memory partitions that model the system processes as well as the allocation of application components onto these processes. The view is mandatory.
- **HW Resource view**: provides a description of the HW platform resources. The view is mandatory.

- **SW Platform view**: provides a description of the SW platform resources. The view is mandatory.
- **Architectural view**: defines the platform architecture and the mapping of system processes onto platform resources. Additionaly, this view includes the association of threads with processors. The view is mandatory.
- **Verification view**: defines the environment components that interact with the system. The view is not mandatory.

The **PIM** includes the views:

- Data View
- Functional view
- Application view
- Communication view
- Memory Space View

The **PDM** includes the views

- HW Resource view
- SW Platform view

The **PSM** includes the view:

- Architectural view

Modelling process

The following figure shows the proposed steps that are covered by the system specification methodology defined in this document. These steps guide the designer through the generation of the complete model required to perform the further synthesis and parallelization activities. The numbers in the boxes in the figure follow the proposed modelling step order.

The system specification methodology starts by defining the Data view where the designer models the data types required for the application and communication modelling. Then, four different views can be specified independently:

1. Data View: defines the data types used for specifying the arguments of operations and services of the application components
2. HW Resource view: describes the components of the HW platform. Each component specification must provide its identifier and its type, and the parameter values which define it.

3. SW Platform view: describes the components of the SW platform. Each component specification must provide its identifier and its type, and the parameter values which define it.
4. Verification View: defines the environment responsible for exciting the system. In a Test-Driven development methodology, the Verification view would be the first view to be defined. The following steps are recognized in this activity to achieve the previous objectives:
5. Identify the different environment subsystems interacting with the system and define how they interact with the system.
6. Model the functional elements that define the behaviour of the environment subsystems
7. Instantiate the system and the environment subsystems in order to define the environment-system structure.

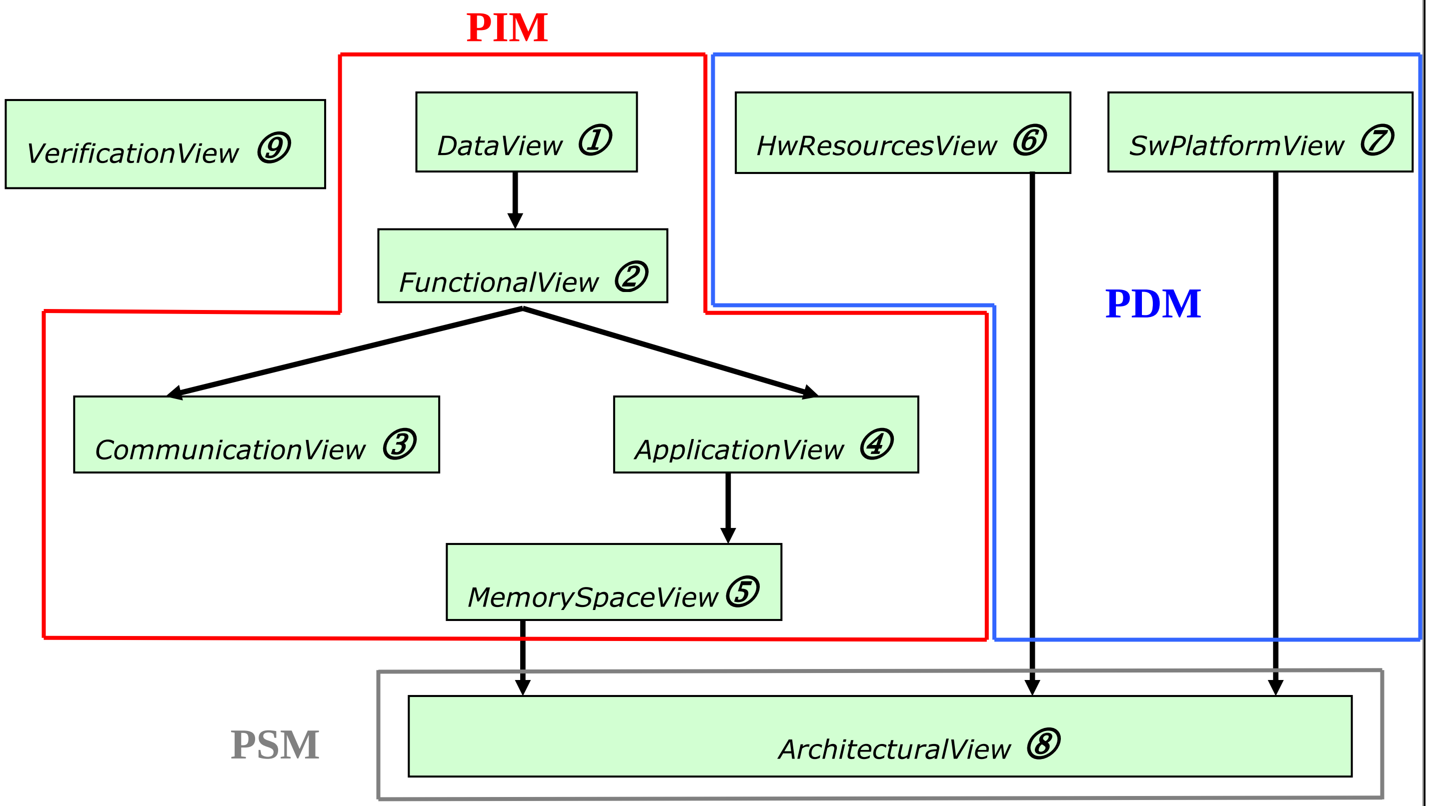

Figure 1 UML/MARTE view modelling activity

After defining the Data View, two model views can be defined in a cooperative way:

8. Functional View: defines the interfaces used by the applications to communicate, defining the operations that these interfaces have available and the arguments of these operations. The arguments are typed by the data types defined in the Data

view. In addition, the functional elements that define application functionality are modelled.

9. Application view, which models the concurrency in the system in four ways:
10. Modelling the application components.
11. Associating the corresponding functional elements defined in the Functional view with each application component.
12. Define the application structure that consists of:
    - 12.1. The specific structure of each application component defining the internal parts (application instances and the communicating elements defined in the Communication view that interconnects these applications)
    - 12.2. The System *top* of the application structure
13. Communication view: defines the type of channels used for communicating the application components.

In the next step, the components of the Application view are mapped to memory partitions. This task is dealt with in the Memory Allocation view where:

14. The memory partitions are defined.
15. The allocation of the *top* application instances defined in the Application view are mapped to the memory partition instances.

The architectural view defines the HW platform architecture, by using instances of the HW components defined in the HW Platform view and the SW platform architecture by using instances of SW components defined in the SW Platform view. Then, the SW instances are allocated to HW instance resources. The second allocation process that takes place is the allocation of the instances of memory partitions defined in the Memory allocation view.

With the Architectural view, the system design is completed and the transformation process from UML/MARTE can be done.