Controlling the Complexity of Hierarchical Scheduling Frameworks – An MBSE Approach

Brandon C. Woolley [1] ***and Susan Mengel*** [2]

[1] Space and Airborne Systems Raytheon Company 2501 W. University, McKinney, TX 75071, USA.

[2] Department of Computer Science, Texas Tech University, Lubbock Texas, 79409, Email: susan.mengel@ttu.edu

Correspondence: E-mail: brandon.woolley@ttu.edu

Received 3 August, 2021; Revised 31 August, 2021; Accepted 31 August, 2021

Available online 31 August, 2021 at www.atlas-journal.org, doi: 10.22545/2021/00160

Hierarchical scheduling frameworks are a new scheduling paradigm where multiple system schedules are integrated (one-within-another). HSFs presents a multi-layered complexity problem that system engineers are struggling to contain. A promising trend in the aerospace and defense industry is to employ Digital Engineering's Model-Based Systems Engineering (MBSE) to deal with the complexity of HSFs. MBSE permits the abstraction of application-specific details that can radically speed up system design exploration. Thus, this paper investigates how the output from an HSF algorithm can be converted into an MBSE modeling language that enables architectural exploration for resource allocation. The Unified Modeling Language (UML) Modeling and Analysis of Real-Time and Embedded Systems (MARTE) Profile is the chosen unified modeling language of MBSE. The modeling language is used with an HSF application for demonstration purposes. The approach in this paper seeks to limit tool use by combining an inline verification method (Genetic Algorithm) with a new MBSE workflow.

Keywords: Hierarchical scheduling frameworks, MBSE, UML MARTE, complexity.

1 Introduction

Digital Engineering (DE) is at the center of the governments' new approach for developing next-generation mission-critical systems that can be adapted in months, not years [1]. This approach uses DE to connect all aspects of a mission system's lifecycle – design, testing, production, and operations from end to end – that renders each, in digital detail, in a virtual environment before the first physical system is built. A part of Digital Engineering, Model-based systems engineering (MBSE) has received a lot of attention recently. Friedenthal and Oster [2] state Model-Based System Engineering is an approach to systems engineering where the system model is the central artifact of the systems engineering process. MBSE is used to comprehend system complexity and provide design space exploration [3,4]. MBSE is applied to many different facets of the systems developed to handle complexity [5,6]. The focus is on establishing a single authoritative source of truth for the entire development process. The DE approach efficiently links

MBSE, including code development, virtual integration, and test, ultimately facilitating the digital twin or virtual model of a product, process, or service. The creation of Digital Twins [7] for each system in a virtual environment enables validation and learning before installation on a given platform. The use of MBSE and DE allows a program to assess the system's architecture design and impacts (including system safety and security) to changing requirements and identify any potential risks.

A potential application for DE is embedded software written more than 30 years ago, which needs to be migrated to modern architectures that include stringent safety and security requirements. For example, Department of Defense programs face a challenging migration problem because legacy software uses older single processing environments which must be upgraded and re-architected to accommodate current requirements and modern multicore architectures. Previous work [8] by the authors created a hierarchical scheduling framework (HSF) integrated with a genetic algorithm to help with this migration and demonstrated the efficacy of our approach. To benefit the defense industry, we must answer the question: how can this previous research be applied to enable the migration of legacy embedded systems' software tasks to multicore architectures? For legacy systems, MBSE would not exist. Thus, this paper investigates and applies the strategies and best practices for the embedded systems MBSE approach. It offers a streamlined approach to model and manage the resource effects of HSFs in large complex systems of architectures. We will show how the previously formulated genetic algorithm's solution can be converted into a universal modeling language model that can then be applied to and incorporated into an extensive system-of-systems model.

2 Conventional Embedded Schedule Analysis Tools

Handling the complexity associated with HSFs is an intimidating task. With the emergence of multicore architectures, some modeling tools are created with built-in extensibility to handle the added complexities of hierarchical scheduling problems. TIMES, Cheddar, and MAST are the most well-known tools available because they are open source and extensible. These tools were early attempts to address the complexity of timing but failed to achieve full model integration.

TIMES [9] is a schedulability analysis and synthesis tool used to model real-time system behavior. Using a set of tasks and their execution times, priorities, and resource constraints as inputs, TIMES generates and validates a scheduler using worst-case execution times. TIMES supports both non-preemptive and preemptive scheduling policies. For further validation, the TIMES tool provides a method for use (to agree with "model checkers") like UPPAAL [10] that uses an MBSE Framework for modeling schedules.

Cheddar [11] is a simulation framework that allows the system designer to verify if a real-time system meets its timing constraints. Cheddar uses real-time scheduling theory as the basis for its analytical portion of the simulation engine. Cheddar also supports multiprocessor systems and is extensible to simulate specific systems. Cheddar's extensibility is important because it shows how to incorporate the multicore and guest operating system environment.

MAST [12] is a modeling and analysis suite for real-time applications that supports distributed and multiprocessor systems and is extensible to accommodate hierarchical scheduling. MAST includes the ability to model and analyze hard real-time schedules with worst-case execution times and soft real-time event-driven schedules. One crucial consideration for MAST is the tool's ability to accept a system design generated by a UML tool which would allow for easy integration of a modeling and analysis phase within an engineering workflow. MAST-2 later created the MARTE2MAST update so that MARTE models could be extracted to/from the analysis software, which is of interest to this research because MARTE is a UML profile that provides the necessary analysis packages for schedulability. MAST Requires a system model from a UML MARTE Model; however, legacy systems were not defined using any modern modeling language. Our approach starts with processor tasks and defines the HSF to be consumed by a model.

A new design environment called Gaspard2 is proposed by Piel et al. in [13]. Gaspard2 is based on the UML MARTE Profile. Within the newly proposed environment, the authors aim to address the high-level model description and transform it into SystemC for model execution and simulation. The

authors are inspired by the need to wrangle in the complexity of multiprocessor system-on-a-chip designs and recognize that new design methodologies and tools are required to address these issues. They illustrate the effectiveness of the new tool with a simple case study and example. Their underlying method is inefficient and tends to use many transformation chains in and out of the model during model refinement.

A methodology and toolset called the COMPLEX UML/ MARTE framework is proposed by Herrera et al. in [14]. These methods and tools enable the specification of a design space that consists of a set of possible architectural mappings, a range of values of platform attributes, and a set of platform architectures. The authors wish to automate the design space exploration (DSE) by building and linking several tools together using and reusing the UML/MARTE model. They model several use cases to prove the efficacy of the proposed methodology and framework. The framework provides several illustrations of the model artifacts used in the tradeoffs for each scenario examined.

Over the last several years, many software tools and frameworks have been developed or extended to grapple with the complexity and analysis problem associated with HSFs. Unfortunately, they all require a heterogeneous tool workflow that creates another management and usability problem. We simplify the process by making the model the focus for managing the complexities associated with HSFs. We streamline the model creation process by reducing the number of tools and do not require multiple import/export cycles.

3 Using MBSE and UML MARTE to Handle Complexity

This research focuses on modeling frameworks used for integrating and modeling schedules for real-time embedded systems. We investigate a novel approach to modify the output of the genetic algorithm to accommodate MBSE integration. This investigation simplifies HSF's complexity by converting them into a universal modeling language model and making them consumable and reusable. There are many MBSE software tools created to facilitate the modeling of system performance. Many of these tools have been deprecated or bought by modeling companies that charge licensing fees.

Ribeiro, Ribeiro, and Soares, in [15], combine SysML with UML to describe an embedded system architecture. They are motivated by a need to manage system/software design complexity of real-time embedded systems design. They use a simple case study to illustrate the effectiveness of their approach.

Hagner, Huhn, and Zechner showed, in [16], how to integrate and use the MARTE profile for real-time properties in a model-based development process. MARTE is used to provide a timing analysis of a railway automation system that can be handled in UML or SysML. MARTE improved architectural design by making inherited and derived requirements more explicit. The MARTE profile helped capture and visualize timing requirements and used annotations for scheduling analysis before implementation, which enabled automated timing analysis to allow for the exploration of various design alternatives. Through a simple use case, the authors demonstrate imports, exports, and reimports to an external timing analysis tool back into the MARTE Model. The outcome provided insight into the use case and suggested tool development for a model to analyze metamodel integration/translation. This work exports from the model to an external schedulability analysis tool and reimports the results back into the MARTE Model. Our approach removes the need for an external tool to build the schedule to meet the hierarchical scheduling frameworks' timeline.

A new Co-Design methodology presented by Koudri, Aulangnier, and Vojitisek, in [17], uses a semi-agile method defined by their MOPCOM (Model of Programmable Components on MARTE) research project. Their research seeks to provide a complete model-driven architecture tool to design system-on-a-chip applications. They are motivated by addressing the component design complexity for system composition and certification constraints for safety and assurance. The methodology used in their approach uses SysML combined with the MARTE profile that is refined to platform and allocation modeling. The authors propose three levels of abstraction and then map an example application onto an example hardware architecture.

The complexity of electronic systems is constantly increasing, and this requires new powerful design strategies. Mura, Panda, and Prevostini, in [18], analyze two UML Profiles, SysML and MARTE, from a Model-Driven Architecture paradigm perspective. To handle the impending on-slot of complexity associated

with electronic systems' design, the authors wished to study how specific UML profiles can alleviate the complexity and become enhanced by the code generation techniques. They use a Wireless Sensor Network case study as a reference for applying the MDA Paradigm using SysML and MARTE. On one hand, they found that the SysML profile was better at the generation of SystemC code than MARTE. On the other hand, MARTE was better at describing the timing interaction between objects within the model.

Bicchierai, Buccir, and Carnevali, in [19], provide an approach to integrate formal methods within an industrial software process using the UML MARTE Profile to manage the documentation process required by MIL-STD-498. They integrate formal methods into the development life cycle using a model-based approach. Their work seeks to alleviate the impact of formal methods on the software development process by using a model-based approach to manage the complexity and documentation. The authors provide a case study to test and illustrate the process's effectiveness and feasibility. The case study showed the idiosyncrasies and intricacies of the experiment.

3.1 Applying MARTE for MBSE Integration

Driven by the need for modeling interconnected embedded elements at a system and system of system's level, Zimmerman, Bringman, and Gerlach, in [20], provide extensions to the UML MARTE Profile for distributed systems applications. Their approach allows for holistic modeling and simulation to facilitate the exploration and verification of system behavior. Through discussion, illustration, and explanations, the authors detail their process with an exemplar target architecture.

With the position that the UML MARTE profile is difficult to understand and use, Hagner and Huhn, in [21], set out to create a more straightforward form of the profile that focuses on scheduling analysis. The authors extended a palette within a tool to support scheduling analysis for developers. They used an empirical methodology with novice subjects (students) and measured their ability to perform the analysis. The results showed increases in efficiency over those without the palette plugin.

The Clock Constraint Specification Language (CCSL) (a companion language for the MARTE Profile) used in scheduling applications is modified by Zhang, Dai, and Mallet, in [22]. They propose a less restrictive modification to CCSL for use in periodic scheduling applications. Their proposal targets bounded and periodic schedule descriptions. Their proposal also targets customized arbitration policies within both simulations and bounded model checking contexts. The authors wish to provide a greater exploration of periodic schedules that sufficiently meet the schedule constraint conditions using an algorithm to find the best answer automatically. This work illustrated the effectiveness of their approach by linking and using the model with the CCSL and the encoding tool, resulting in several simulation results. This approach informs our research by shortening the cycle from tool-to-tool for verification and validation with an inline application.

Driven by the need to solve and manage complex automotive applications in a modeling language, Espinoza, Richter, and Gerard, in [23], investigate the application of UML MARTE to the EU's TIMMO (Timing Model) Project for AUTOSAR (AUTomotive Open System ARchitecture). For this investigation, the authors first identify the critical timing requirements for AUTOSAR then identify the key MARTE characteristics that apply to TIMMO. The result is a set of recommended UML types for adoption for TIMMO and MARTE.

In [24], Andre et al. discuss overlaying the IP-XACT Standard for IP composition onto the UML MARTE Profile. The overlay leads to extensions and formalism with timing aspects associated with the standard. They want to utilize the timing features of MARTE because the IP-XACT has time representation issues. They create a metamodel for transition and profiling from IP-XACT into UML MARTE. They detail the example with illustrations and reuse scenarios. The method outlined in this paper informs our research as an example of how to solve our problem for model integration by way of a metamodel. In this paper, we create an HSF Pattern and use Rapid Modeling Tools (RMT) for model creation and integration.

4 Hierarchical Scheduling Framework

From our previous work, in [8], we define our HSF model, a task $\boldsymbol{\tau_i}$ is defined as the basic unit of execution and is characterized by worst-case execution time (WCET) defined as $\boldsymbol{C_i}$, a relative deadline $\boldsymbol{D_i}$, and a period of $\boldsymbol{T_i}$. A sporadic task will produce a series of jobs with inter-arrival times separated by $\boldsymbol{T_i}$ time units. The cost of each job is $\boldsymbol{C_i}$, and the relative deadline is $\boldsymbol{D_i}$ which occurs after the job's arrival time. The tasks are sporadic with implicit deadlines where $\boldsymbol{D_i = T_i}$ for all $\boldsymbol{\tau_i}$ (also known as Liu and Layland [25] task systems). From [8] we define the following:

Definition 1. *For $\boldsymbol{n}$ number of tasks, the tasks $\boldsymbol{\tau_i}$ are organized into tasksets $\boldsymbol{\Gamma}$ that are then assigned to the specific partition $\boldsymbol{\pi_i}$.*

Definition 2. *Partitions are gathered into partitionsets sets $\boldsymbol{\Pi}$ that are in the end assigned to the required number of cores $\boldsymbol{m_i}$.*

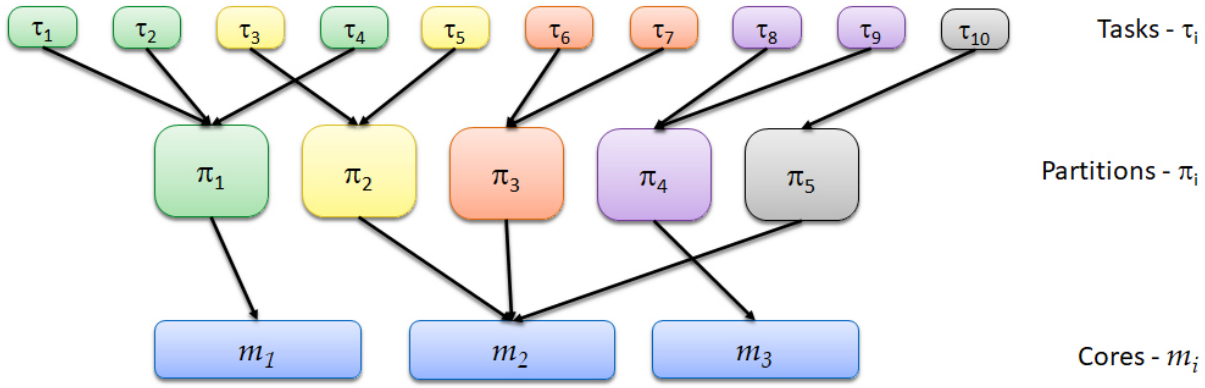

Figure 1: Task → Partition → Core Allocations [8].

The HSF seeks to allocate tasks to partitions and partitions to cores efficiently while distributing the total utilization across all cores, as illustrated in Figure 1. The tasks are assigned to partitions with guest operating systems that employ a fixed priority pre-emptive rate monotonic scheduler. The partitions are assigned to cores that use a static cyclic scheduler (also known as the partition scheduler). Figure 1 shows the hierarchical relationship of the scheduling problem we solved. Hierarchically combining both schedulers presents a complex problem. A modeling approach is needed to handle the associated complexity. In the following sections, we will show how to migrate the output of the HSF to be incorporated into a model.

5 The UML MARTE Profile

The Object Management Group (OMG) publishes the Universal Modeling Language (UML) Specification [26] as well as the UML Modeling and Analysis of Real-Time and Embedded systems (MARTE) Profile [27]. The MARTE profile is an extension of UML that supports the real-time modeling of embedded real-time systems. MARTE, shown in Figure 2, is comprised of three primary packages and one annexes package. First, the foundational package "MARTE_Foundations" defines the foundational concepts that support development from design to analysis phases. These concepts are further specified in the other packages.

The second package, "MARTE_DesignModel" is the model design package. It is comprised of high-level model constructs for embedded applications. The third package, "MARTE_AnalysisModel" is the model analysis package. It provides generic models for quantitative analysis.

Figure 2: UML MARTE Profile.

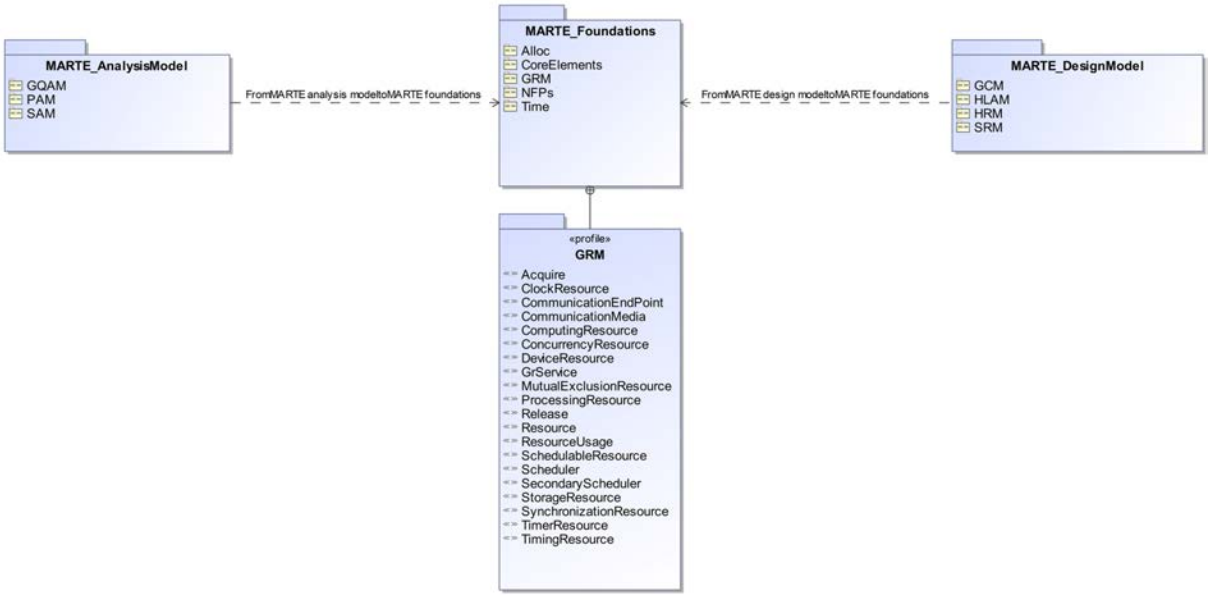

Figure 3: UML MARTE GRM Profile.

5.1 MARTE GRM Meta-Model Description

From MARTE Specification [27], the Generic Resource Model (GRM): "provides foundational modeling constructs that are later refined to support design (SRM & HRM) as well as analysis (GQAM, SAM & PAM) models." SRM and HRM are the Software and Hardware Resource Modeling packages used in detailed resource modeling. GQAM, SAM, and PAM are the Generic Quantitative Analysis, Schedulability Analysis Model, and Performance Analysis Model packages used in quantitative analysis modeling.

For our approach, we will target the UML MARTE Profile's GRM like [24] because it provides the foundational model forms for integration. As an MBSE project progresses through its lifecycle, it will mature in detail and fidelity. Therefore, our metamodel for transition will use the package and stereotypes found in GRM (shown in Figure 3).

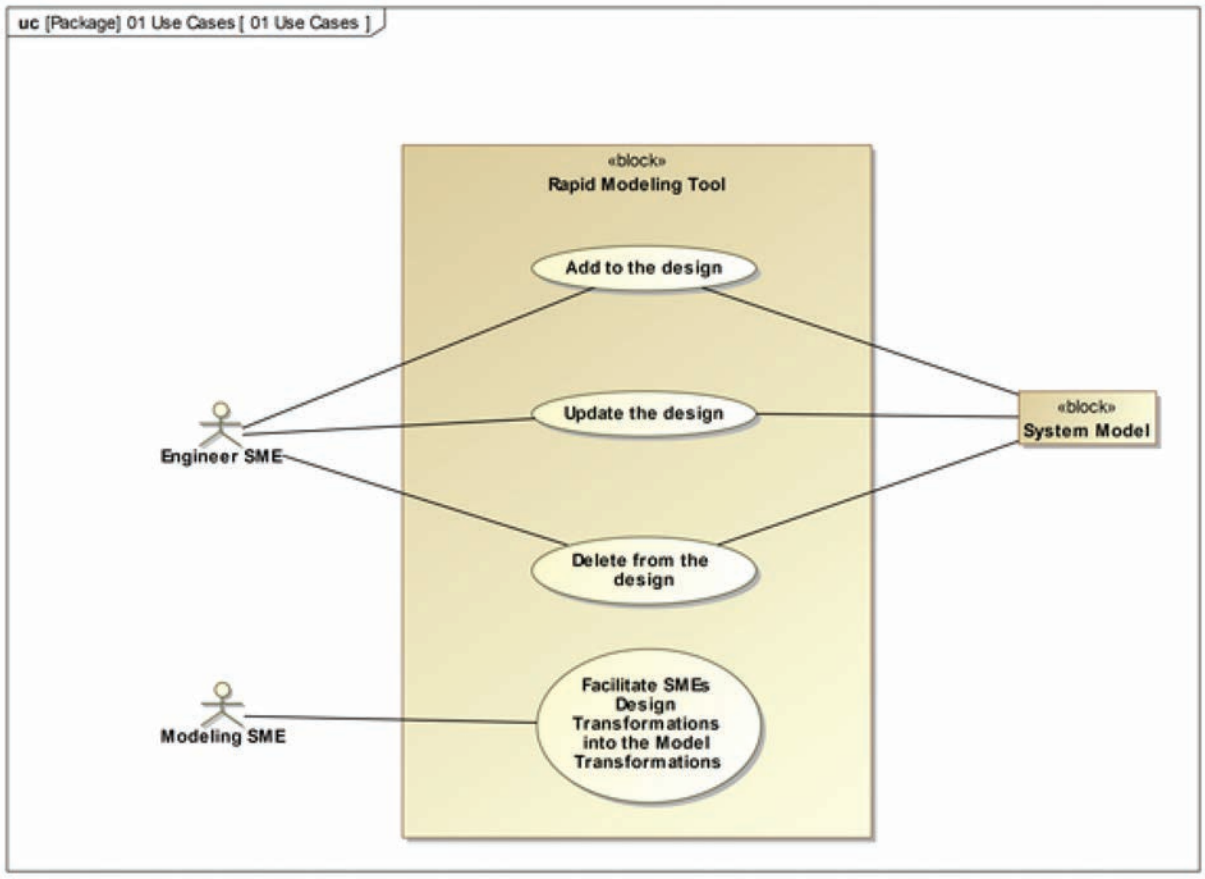

Figure 4: RMT Example Use Cases [28].

6 Rapid Modeling Tools

In a recent contribution to the MBSE GIT repository, Connelly and Cole, in [28], of Georgia Tech Research Institute created Rapid Modeling Tools (RMT) repository to help infrequent contributors to MBSE. The authors are motivated by the need to ingest large data elements from non-modelers and to save skilled engineers time by increasing the speed of model integration. RMT goes beyond a simple CSV import to support full model integration within domain engineering processes.

In Figure 4 [28], RMT's Use Cases illustrate the engineer, modeler, and the system's interaction.

The RMT, shown in Figure 5 [28], Consists of two primary components for model tool integration: Ingrid and Player Piano. Ingrid is written in Python, which translates a spreadsheet using a modeling pattern to create a JSON platelet that the Player Piano Cameo plugin interprets to build the models.

6.1 Ingrid Nerdman

The motivation for the Ingrid Nerdman software tool is to reduce the impact of the big data problem and to provide rapid, dynamic, and flexible model instantiation. Ingrid Nerdman employs change detection before altering the model baseline. This tool surpasses standard Excel or CSV import by supporting full model integration. It uses an adaptable model pattern to support model integration.

The Ingrid Nerdman process uses a pattern sheet representing a user's metamodel of the system being created. Figure 6 [28] illustrates an example of the pattern sheet. The column titles represent specific

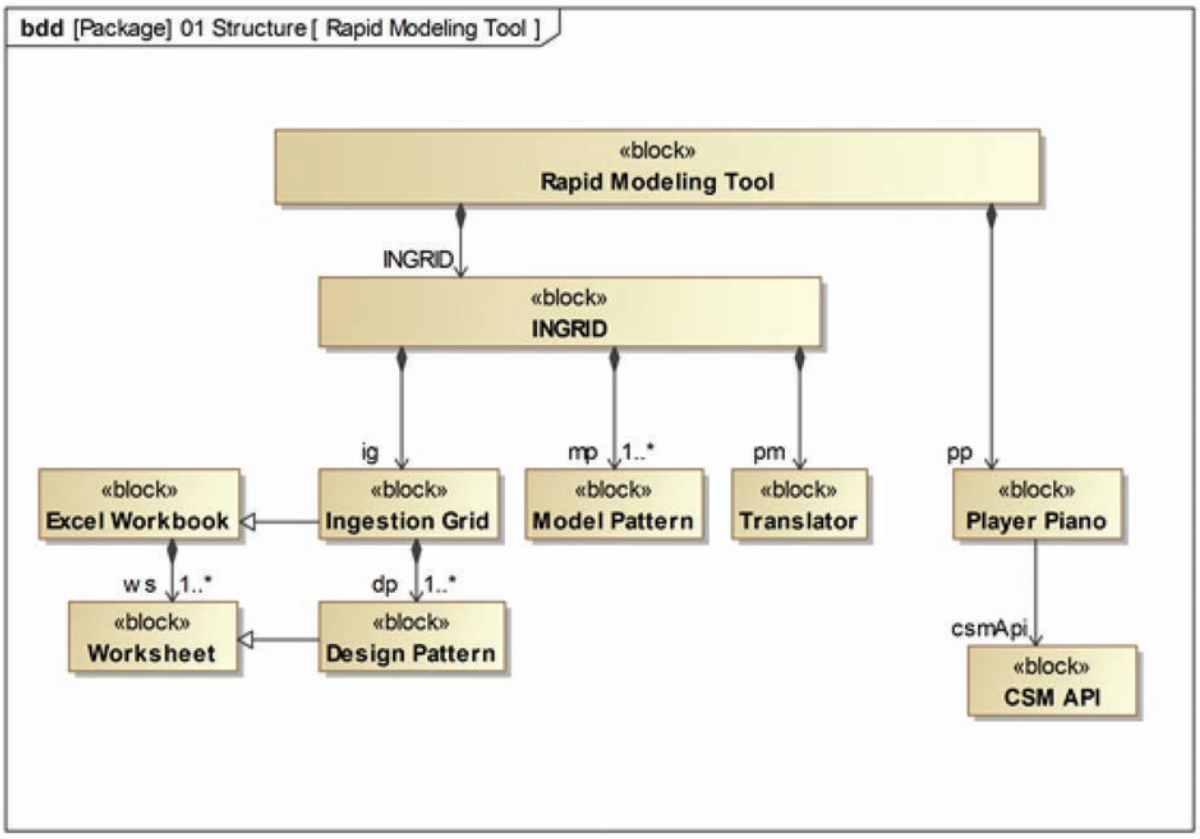

Figure 5: RMT Component Structure [28].

Excel Representation

Component/Composite Thing	**Position**/component	**Part**/Atomic Thing	composite owner (hidden)	A_composite owner_component (hidden)
Spacecraft	ME	Main Engine	spacecraft qua me context	A_spacecraft qua me context_me

Figure 6: Example RMT Composition Excel Representation [28].

model concepts. In Figure 6 [28], the columns show a composition modeling concept depicted in Figure 7 [28].

The spreadsheet example in Figure 6 [28] represents a graph interpreted from a SysML modeling pattern shown in Figure 7 [28] used to create a new model or update an existing one.

The standard Ingrid Nerdman model creation flow, depicted in Figure 8, shows the automated generation of a model based on design patterns and the desired content to be modeled. The workflow begins with the definition of composition modeling patterns. The composition modeling patterns allow the user to express model elements and their linkages. Once these composition modeling patterns are defined, the user can use a spreadsheet to represent the input design. The spreadsheet is filled in with the design elements,

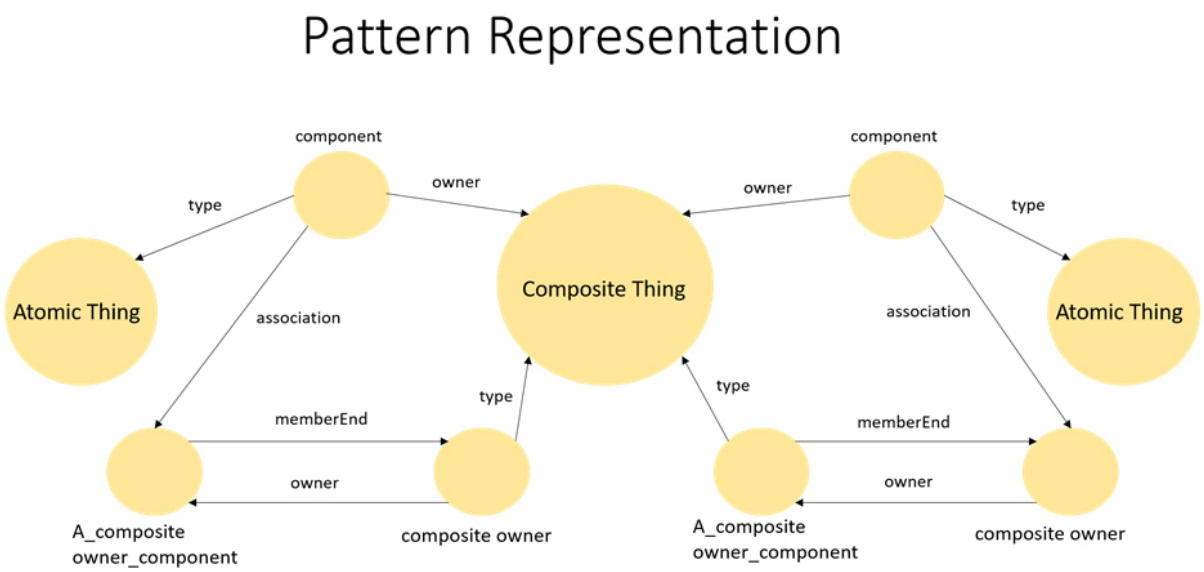

Figure 7: RMT Composition Pattern Representation [28].

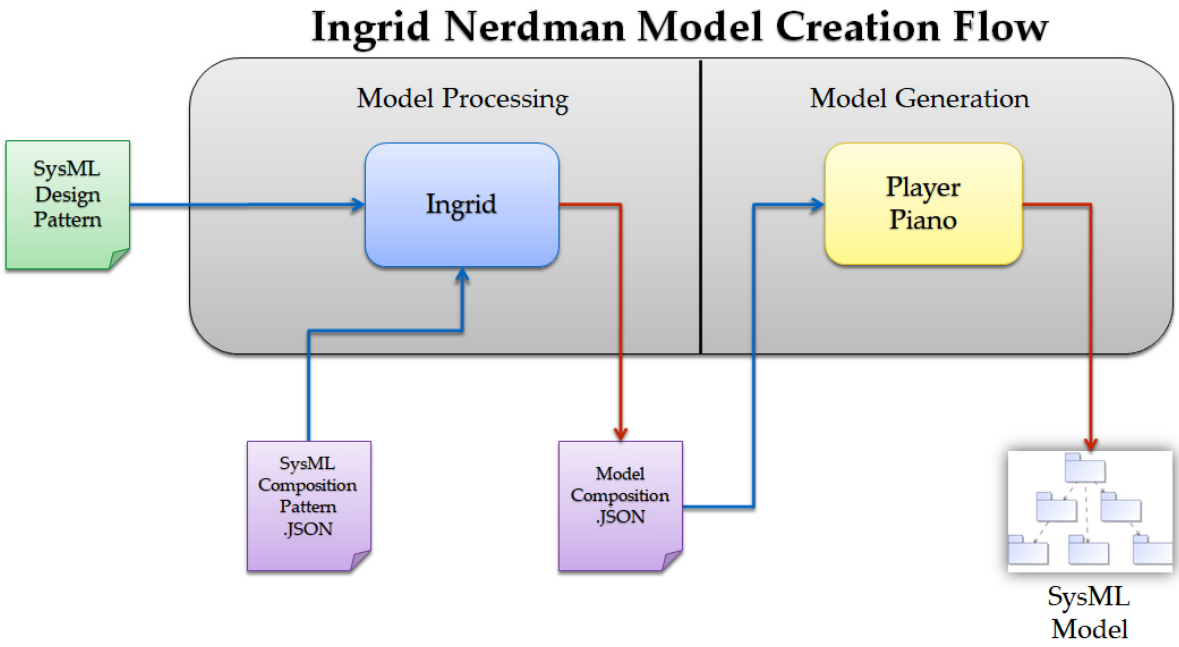

Figure 8: Standard Ingrid Nerdman Model Creation.

then composition modeling patterns and the design elements are both inputs into Ingrid Model Processing software. Ingrid takes these inputs and outputs a JSON model composition. The model composition is processed into Cameo via the player piano macro. The player piano processes the model composition file to transform it into the final model product.

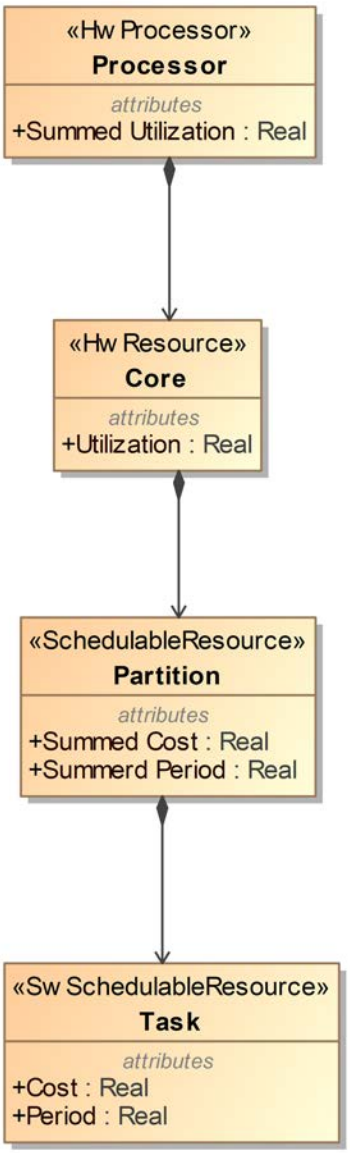

Figure 9: MARTE Metamodel Example of the Hierarchical Relationship.

7 Our UML MARTE Profile For HSF

In this paper, we extend a software tool called Ingrid Nerdman that is based on SysML to use the UML MARTE profile and to map the HSF. The goal is to automate the HSFs creation with connections between objects and compose UML diagrams automatically. To adapt the IN SysML patterns to accommodate the MARTE profile, we need to create an HSF overlay/metamodel with a MARTE component and stereotype mapping. The UML diagram in Figure 9 shows how we model the HSF's metamodel element relationships using the OMG UML MARTE profile.

A MARTE metamodel is needed to decompose the relationships between the HSF elements. The metamodel is used to define the composition within IN tool. As shown in Figure 9, the metamodel created

Table 1: *Mapping HSF Elements to UML MART.*

HSF Element	UML MARTE Stereotype
Task	<<swSchedulableResource>>
Partition	<<SchedulableResource>>
Core	<<hwResource>>
Processor	<<hwProcessor>>

for the given processing architecture is composed of processors, cores, partitions, and tasks. From there, we can map the appropriate HSF elements onto UML MARTE. Table 1 describes the relationships chosen for this metamodel via HSF element selection to UML MARTE stereotypes.

Next, we can create the Ingrid composition pattern to accommodate the MARTE profile and its stereotypes. This pattern describes the HSF composition in UML MARTE and will be used to generate the model. A JSON pattern file is used to map the HSF from a CSV/XLSX input file that is used to create the composition model. At this time, we can update the genetic algorithm to output the HSF into a pattern consumable by the model processing module, Ingrid. Once the Ingrid JSON composition is created and the HSF output has been formatted, we can start using the model flow shown in Figure 10.

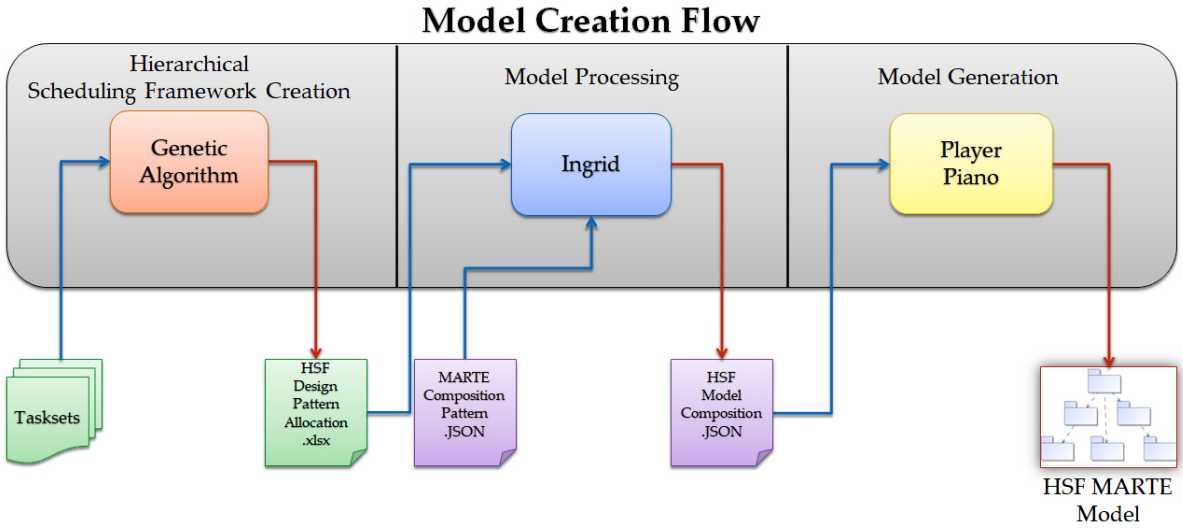

Figure 10: Model Creation Flow Using GA, Ingrid, and Player Piano.

The first step in the holistic model creation process, depicted in Figure 10, is HSF generation. During this phase of the model creation, the tasksets (from Definition 1) are input into the GA that outputs the HSF design pattern allocation. The HSF design pattern allocation provides an output consumable by model processing. The second step in the model creation process is Model Processing. During this phase of the model creation, the formatted HSF design pattern allocation output and the Ingrid MARTE JSON composition are input into Ingrid. Ingrid processes these inputs to generate a JSON HSF model composition. This composition describes the HSF task allocation. The third step in the model creation process is Model Generation. During this phase of the model creation, the HSF model composition file is input into the Cameo player piano macro (from RMT). The player piano processes the model composition file to create the final model product.

Figure 11: Simple 4 Core, 8 Partition, and 16 Task Example.

8 Results and Discussion

We verified the model creation process first by generating multiple use cases from simple to complex. The uses cases increased in complexity by increasing the number of tasks within a taskset.

To demonstrate the working of the model creation framework, we provided sixteen tasks in the taskset, which generated eight partitions, on four cores, within one processor. From the illustration in Figure 11, the model creation process successfully generated the model from a taskset through HSF. The image shows how MBSE can be used to wrangle the complexities associated with HSFs.

Multicore architectures come in various arrangements that can vary from 4, 8, 12, and 24 cores. As the number of cores increases, an exponential expansion is created that drives up the scheduling and HSF complexity. Imagine Figure 11 multiplied by two or even eight times in a cascading fashion. The creation and management of 32 different two-level hierarchical schedules can become a complex task. To show how these complex architectures can be handled via an MBSE representation, we created a taskset composed of 109 tasks. The GA allocated these tasks into 48 partitions and eight cores.

Figure 12 illustrates the complex use case developed to demonstrate our new model creation flow shown in Figure 10. Based on the results captured in the Cameo, our model creation process is a viable approach to represent HSFs in an MBSE environment.

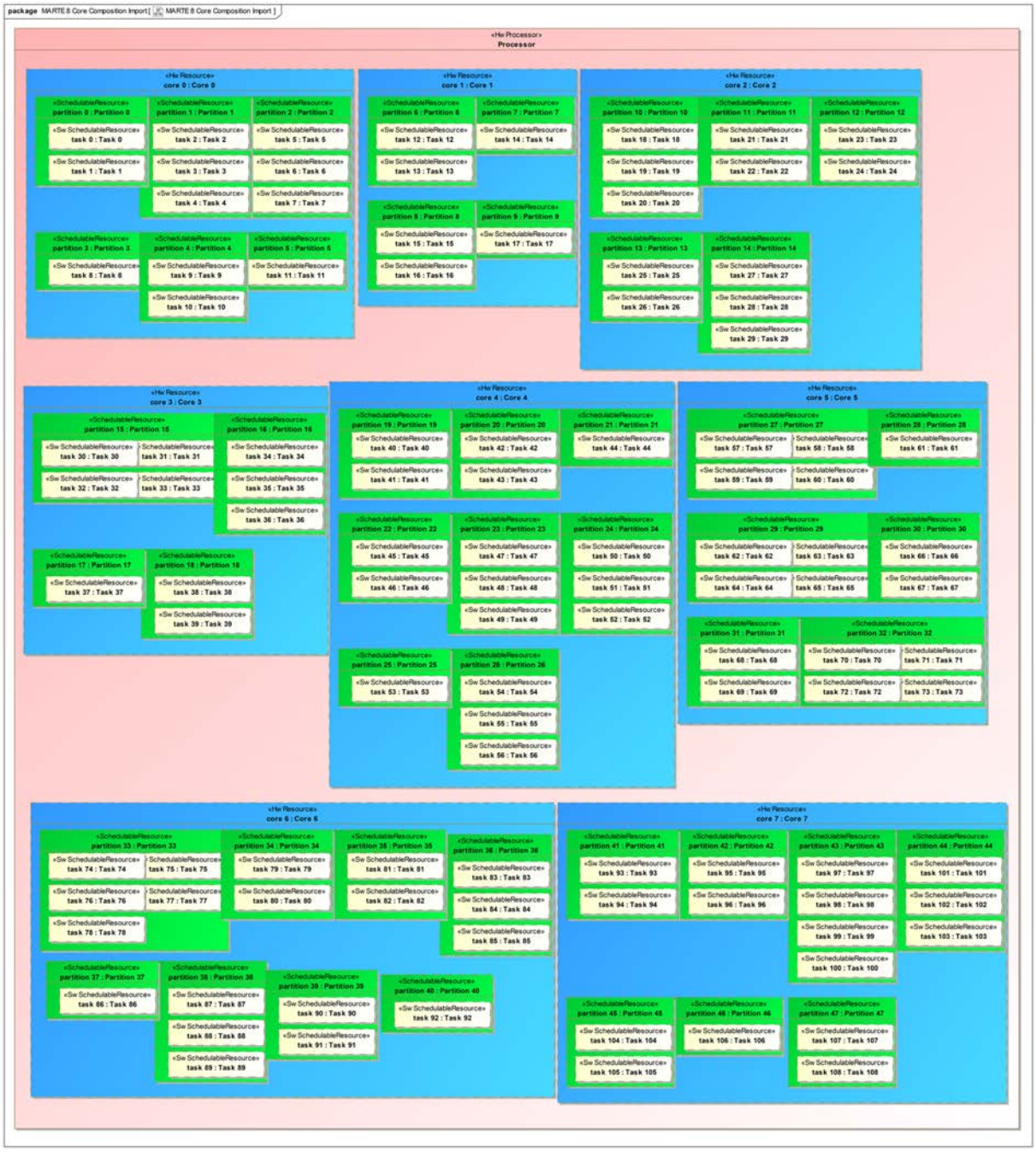

Figure 12: Complex 8 Core, 48 Partition, and 109 Task Example.

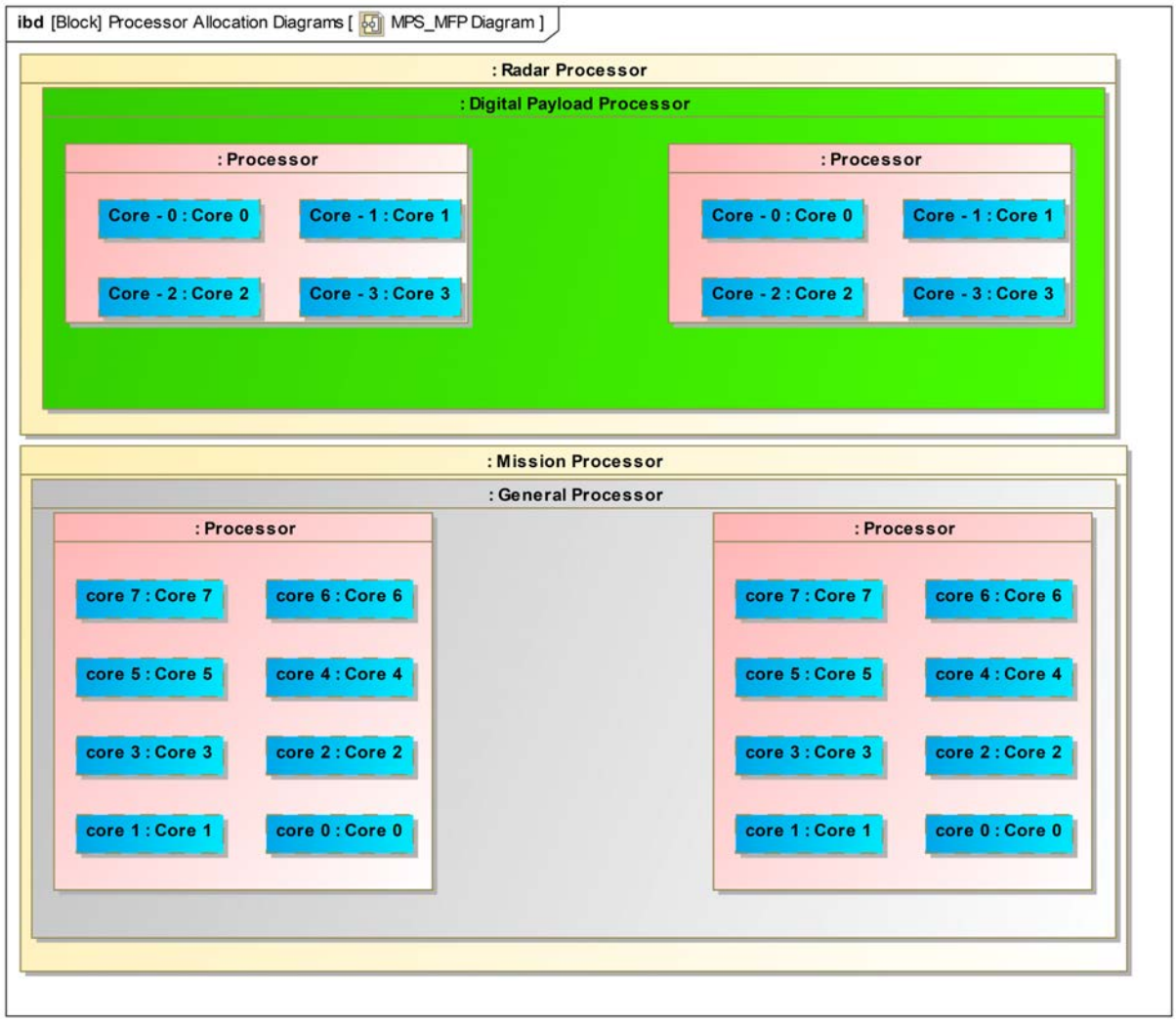

Figure 13: Example Radar and Mission Processor Allocations.

9 Conclusions

Due to the evolving complexity of systems, new approaches to architect, design, develop, and test must be made. The method for this in next-generation systems is to use Digital Engineering so that a digital twin can be created and evaluated. It must be incorporated into a model-based system engineering paradigm for a legacy system to be migrated to new architectures. Our hierarchical scheduling framework (HSF) algorithm is constructed to integrate into an MBSE environment using the newly formulated model creation workflow.

The HSF & GA + This new MARTE MBSE Flow provides a quick way to prototype compositional representations and model resource allocations on potential system designs. An illustration of an example design trade is shown in Figure 13.

A designer can create the trade, in Figure 13, in less than an hour. Typically, this would take Days or Weeks to compile this type of trade. Since the HSF GA already determined the fungibility of the tasks, partitions, and cores for schedulability automatically, the modeler does not need to perform this allocating step. The tasks and partitions are within each of the core elements; we have hidden their detail brevity.

We successfully converted and adapted the output of our HSF GA algorithm into the UML MARTE profile utilizing RMT's Ingrid Nerdman. Then we tested our approach on both complex and straightforward use cases to verify the approach's scalability. This method allows for an automated MBSE process to be applied to HSFs. The automation and integration into an MBSE workflow provide the transition of large legacy programs to modern multicore architectures.

Authors' Contribution: This paper was produced from B.W.'s dissertation. S.M. is the advisor of B.W.'s committee. S.M. worked with B.W. to edit and review the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Raytheon Company.

Conflicts of Interest: The authors declares no conflict of interest.

References

[1] Digital Engineering Strategy. In: Office of the Deputy Assistant Secretary of Defense for Systems Engineering, 2018.

[2] Friedenthal, S., Oster, C. (2017). *Architecting Spacecraft with SysML: A Model-Based Systems Engineering Approach*. CreateSpace Independent Publishing Platform.

[3] Batista, L., Hammami, O. (2016). Capella based system engineering modelling and multi-objective optimization of avionics systems. Paper presented at: 2016 IEEE International Symposium on Systems

[4] Fei, X., et al. (2020). A Model-Based System Engineering Approach for aviation system design by applying SysML modeling. Paper presented at: 2020 Chinese Control And Decision Conference (CCDC).

[5] Deschner, C. (2020). Enhanced model-based engineering for centrally managed configuration management in product lifecycle management. Paper presented at: SHS Web of Conferences.

[6] Masior, J., et al. (2020). Beyond Model-Based Systems Engineering towards Managing Complexity. *Procedia CIRP*, 91:325-329.

[7] Madni, A. M., et al. (2019). Leveraging Digital Twin Technology in Model-Based Systems Engineering. *Systems*, 7(1).

[8] Woolley, B., et al. (2020). An Evolutionary Approach for the Hierarchical Scheduling of Safety- and Security-Critical Multicore Architectures. *Computers*, 9(3).

[9] Amnell, T., et al. TIMES: A Tool for Schedulability Analysis and Code Generation of Real-Time Systems. In: Vol 2791. Berlin, Heidelberg: Springer Berlin Heidelberg; 2004:60-72.

[10] Hilbrich, R., Goltz, H.-J. (2011). Model-based Generation of Static Schedules for Safety Critical Multi-Core Systems in the Avionics Domain. Proceeding of the 4th international workshop; New York, New York, USA.

[11] Nana, L., et al. (2004). *Cheddar: a flexible real time scheduling framework.* Vol XXIV. New York, New York, USA: ACM.

[12] Gonzalez Harbour, M., et al. (2001). MAST: Modeling and analysis suite for real time applications. 13th Euromicro Conference on Real-Time Systems.

[13] Piel, É., et al. (2008). Gaspard2: from MARTE to SystemC simulation. Paper presented at: proc. of the UML MARTE Workshop.

[14] Herrera, F., et al. (2014). The COMPLEX methodology for UML/MARTE Modeling and design space exploration of embedded systems. *Journal of Systems Architecture*, 60(1):55-78.

[15] Ribeiro, Q. A., et al. (2017). A Technique to Architect Real-time Embedded Systems with SysML and UML through Multiple Views. Paper presented at: International Conference on Enterprise Information Systems.

[16] Hagner, M., et al. (2008). Timing Analysis using the MARTE Profile in the Design of Rail Automation Systems. Paper presented at: Embedded Real Time Software and Systems (ERTS2008); 2008-01-29; Toulouse, France.

[17] Koudri, A., et al. (2008). Using MARTE in a co-design methodology. Paper presented at: UML Workshop.

[18] Mura, M., et al. (2008). Executable models and verification from MARTE and SysML: a comparative study of code generation capabilities. Paper presented at: Proceedings of MARTE Workshop, Munich, Germany.

[19] Bicchierai, I., et al. (2013). Combining UML-MARTE and Preemptive Time Petri Nets: An Industrial Case Study. *IEEE Transactions on Industrial Informatics*, 9(4):1806-1818.

[20] Zimmermann, J., et al. (2008). Holistic system modeling and refinement of interconnected microelectronic systems. *Modeling and Analyzis of Real-Time and Embedded Systems with the MARTE UML profile.5.*

[21] Hagner, M., Huhn, M. (2008). Tool support for a scheduling analysis view. Paper presented at: MARTE workshop March 14, 2008.

[22] Zhang, M., et al. (2018). Periodic scheduling for MARTE/CCSL: Theory and practice. *Science of Computer Programming*, 154:42-60.

[23] Espinoza, H., et al. (2008). Evaluating MARTE in an industry-driven environment: TIMMO's challenges for AUTOSAR timing modeling. *Modeling and Analyzis of Real-Time and Embedded Systems with the MARTE UML profile.17.*

[24] André, C., et al. (2008). Modeling spirit IP-XACT in UML marte. Paper presented at: Conf. on Design, Automation and Test in Europe, MARTE Workshop; March 14, 2008.

[25] Liu, C. L., Layland, J. W. (1973). Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM (JACM).*

[26] Unified Modeling Language (UML), Specification, Version 2.5.1. In: Object Management Group (OMG), 2017.

[27] UML Profile for Modeling and Analysis of Real-Time and Embedded (MARTE), Specification, Version 1.2. In: Object Management Group (OMG), 2019.

[28] Connelly, S., Cole, B. Ingrid Nerdman. 2019; Release 1.0. Available at: https://github.com/gtri/rapid-modeling-tools. Accessed (June 10, 2021), .

About the Authors

Mr. Brandon Woolley has a BS in Computer Science from UTSA and MS CSE from SMU. He is an Associate Director at Raytheon Technologies and has been with the company for more than 21 years. He is the Chief System Security Architect providing architectural vision and technical direction for RTX's Advanced Concepts and Technology Division for next gen platforms. He is the primary author of multiple patents related to embedded systems security. Mr. Woolley is a recipient of multiple technical and innovation awards within Raytheon for accomplishments related to high assurance secure computing.

Dr. Susan Mengel received her Ph.D. from Texas A&M University in 1990. She is an Associate professor at Texas Tech University. She has played strategic leadership roles in numerous transdisciplinary projects involving the delivery of innovative software and data models in sleep management, student retention and advising, computer education, nutrition, speech therapy, cardiovascular disease, and cybersecurity. She helped to establish the Master's in Software Engineering degree program at Texas Tech University, served as Associate Editor for Computing for the IEEE Transactions on Education, served on the Steering Committee of the ACM/IEEE Computing Curriculum, and served as the Outreach Chair FY19 of the SWE Outreach Committee. She currently serves on the Texas Tech Institutional Review Board for the Protection of Human Subjects and is a faculty advisor for the TTU Collegiate Chapter of SWE.