A Project-Product Lifecycle Management approach for improved systems engineering practices

Amira Sharon, Valeria Perelman, and Dov Dori
Faculty of Industrial Engineering and Management
Technion – Israel Institute of Technology
Haifa 32000, Israel
amirash@techunix.technion.ac.il tel. +972525250965
valeriya@tx.technion.ac.il tel. +972502191823
dori@ie.technion.ac.il tel. +972545224270

Abstract. Developing and sustaining complex systems requires collaboration of multidisciplinary teams, coordination of processes, methods and tools, allocation of resources, and utilization of adequate facilities within enterprises. Engineering and management of a system comprises three intertwined domains: the product, the project and the enterprise. Despite the obvious links between them, each domain is distinct and uses its own ontology and toolset. This conceptual separation hinders effective handling of the project and product lifecycle activities within the enterprise. The unified Product-Project Lifecycle Management (PPLM) collaborative research project aims to establish, implement, and evaluate a methodology for managing and engineering complex enterprise-wide systems. The resulting software environment will serve as a test bed for proof-of-concept of the PPLM vision and approach. The comprehensive PPLM methodology with its software implementation is expected to enhance enterprise-level systems engineering and management by unifying the product and the project within which it is developed. The paper presents and illustrates the PPLM vision and framework.

Introduction

Systems engineering and management comprises three intertwined domains: the product, the project, and the enterprise. To demonstrate the relations between these three subsystems, consider testing. Testing activities focus initially on verifying the performance of software and hardware components. They continue with testing large subassemblies and end with the entire operational system and its supporting environment—the "whole system" concept. What needs to be developed, tested, and delivered is determined by the product requirements, its architecture, and design. When each component should and can be developed and tested is stated in the project plan, which is dynamically re-evaluated, and re-scheduled, depending on parameters such as resource availability, risks, and technological breakthroughs. Whether carrying out the mission is feasible, and at what cost and risk, is determined by the responsible enterprise, its size, structure, commitments, and other factors. The question is how to successfully manage one or more products, each developed via a dedicated project within an enterprise with limited resources.

Successful delivery of a large-scale system is predicated upon quality management of the system engineering process throughout the project and product lifecycles. Close collaboration is required between the program management, which focuses on the project, and the system engineering group, which focuses on the product.

Systems engineering management is concerned with issues such as schedules, supplier coordination through contracts, employee motivation, incentives, and teamwork. At the same time, it must handle such subjects as product requirements, alternative product architectures, product functionality, product architecture selection, detailed product design, verification and validation. As Figure 1 shows, the "program management" overlaps in a number of issues with the product's "systems engineering"

process. This inherent, inevitable mix of project on one hand and product on the other hand mandates unification of the project management and the system engineering of the product, which is the expected outcome of that project.

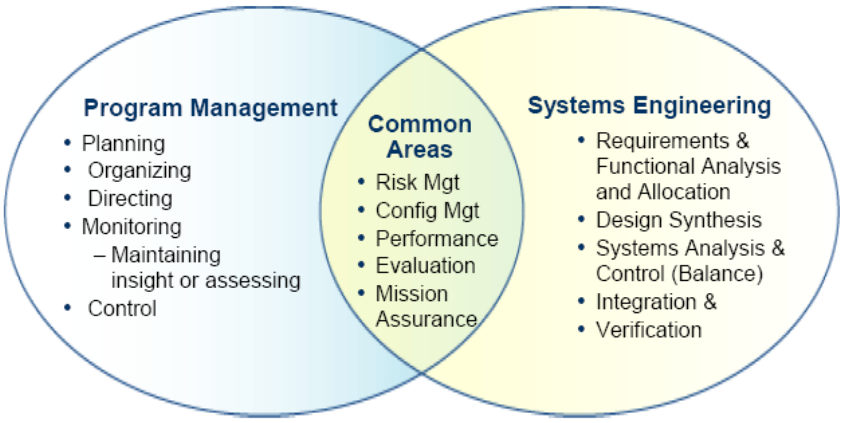

Figure 1 – System Engineering Management Scope

Source: Systems Engineering Cost/Risk Analysis Capability Roadmap Progress Review. NASA. April 6, 2005. All rights reserved.

The enterprise responsible for the product usually runs multiple projects in parallel, each involving many sub-contractors, each of which is assigned with a responsibility for some sub-product or an associated service. The enterprise is therefore the third aspect of system engineering management. The enterprise establishes the relationships between the products and projects it owns and runs to maximize its efficiency and effectiveness.

ISO/IEC 15288 and the Capability Maturity Model Integration (CMMI) explicitly refer to the enterprise as an aspect of the overall lifecycle view. ISO/IEC 15288 establishes a common framework for describing the lifecycle of a framework for systems created by humans. It defines a set of processes, shown in Figure 2, and their associated terminology. This standard provides a comprehensive and integrated framework for managing the entire lifecycle framework of complex systems as well as systems of systems.

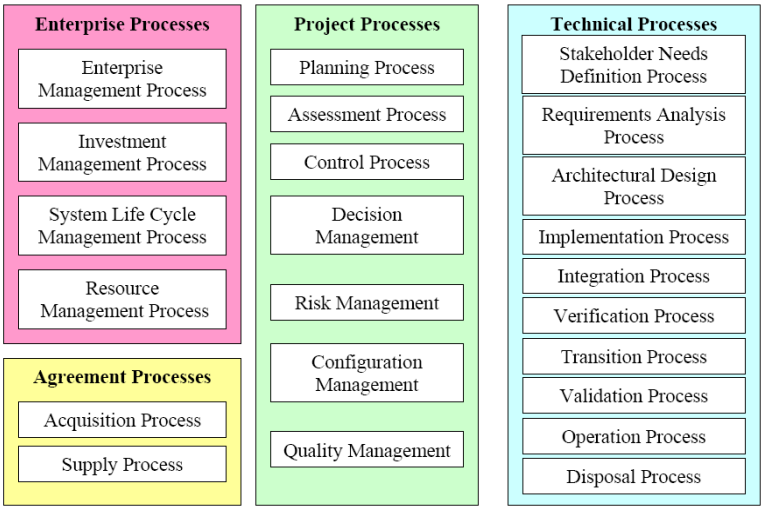

Figure 2 - The system life cycle processes

Source: ISO/IEC 15288, Figure D.8

The ISO/IEC 15288 lifecycle model is expressed in terms of 25 processes, their 123 outcomes, and 208 activities. Each process is defined by a title followed by the purpose of the process, giving a high

level description of the overall process goal. An outcome is an observable result of the successful achievement of the purpose of the process. The activities provide a structural decomposition of a process. However, ISO/IEC 15288 "does not detail the life cycle processes in terms of methods or procedures required to meet the requirements and outcomes of a process." The actual implementation requires complex collaboration of multidisciplinary teams, coordination of processes, methods and tools, allocation of resources, and utilization of adequate facilities within enterprises (Cook 2003), (Hitchins 2003).

The Capability Maturity Model Integration (CMMI) also presents a distinction of project management processes from product processes, as shown in Figure 3. It defines relationships required between product data and project data for establishing quality processes across a project, a division, or an entire organization.

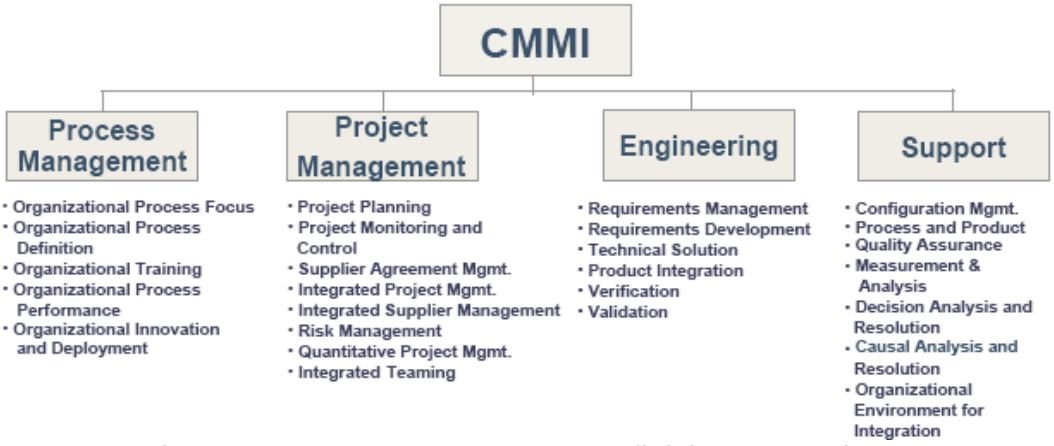

Figure 3 – CMMI Process Areas. ©. All rights reserved.

CMMI SP 1.4 advocates maintaining "bidirectional traceability among the requirements and the project plans and working products". The requirements, as well as the project plans, are certainly changing along the product-project life cycle. Therefore, a mechanism is required for continuous tracking and updating of relationships between evolving requirements and progressing project plans. Since requirements data is maintained separately from project plans, it is difficult to assess the impact of changes in requirements on the project plans, activities, and work products. A fusion of these separate data into a combined model and knowledge base will provide for establishing the relationships between product requirements and product plans, and for a continuous updating mechanism, which, in turn, will enable ongoing assessment of the impact of changes in the product on its cost and delivery date.

CMMI SP 1.5 advocates also identifying "inconsistencies between the project plans and work products and the requirements". Appropriate techniques to document and maintain traceability of the intricate relations among project plans, activities, work products, and requirements are definitely necessary. In the absence of a systematic approach and supporting software environment, inconsistencies between the project plans, work products and the requirements are bound to emerge, and no adequate ability to resolve them is in place.

Our basic assumption is that the identified information gap between the two interdependent fields of product engineering on the one hand and project management on the other hand is a root cause for the frequently failing projects. The majority of the products we use operate in accordance with our expectations: cars drive, aircrafts fly, etc. This indicates that humans do have the knowledge to "make things work." Still, too many projects fail to meet the time, budget and human resource boundaries. Why is it so difficult to construct more realistic plans?

Project and system engineering managers are usually highly qualified experts, so we can assume

that the project planning is carried out professionally. The discrepancies are, therefore, not within the execution of project planning, leaving three other possibilities: the process of project planning (i.e., the methodology) is faulty, some input data is missing or inconsistent, and the customer's requirements change after the project has started. Requirement changes seldom lead to project failure, as timelines and budget are likely to be updated accordingly. The process of project planning is not too complex either if we consider that there are only three main parameters—budget, resources and schedules—that must be accounted for. A variety of automatic tools have been developed to resolve this problem, ruling out the possibility that faulty planning is to blame for project overruns. The conclusion is that the main problem is due to lack of alignment between product engineering and its project management.

The Project-Product Lifecycle Management (PPLM)

The combined Project-Product Lifecycle Management (PPLM) approach is intended to help the system engineering manager to properly tailor the specific product to be delivered by its specific project within a specific enterprise, as illustrated in Figure 4.

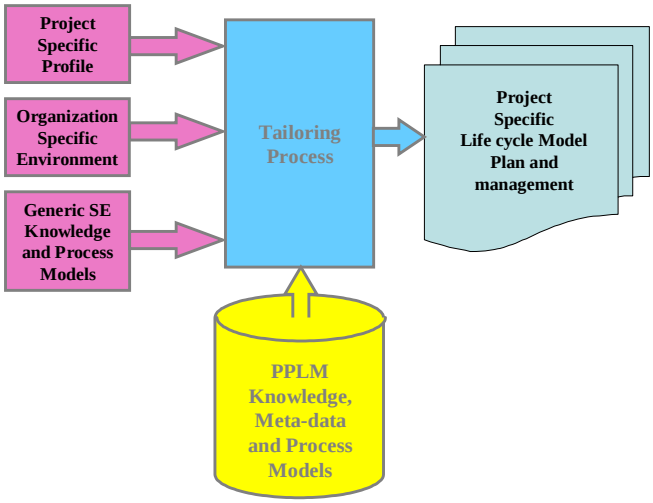

Figure 4 – PPLM guidance for project tailoring

PPLM is a combined research effort aimed at developing a methodology and a software environment for fusing the product to be developed with the project as they are executed by the enterprise. More specifically, the research is intended to develop an underlying holistic conceptual model based on a shared ontology with a software environment for an integrated project and product lifecycle support. The shared ontological foundations will facilitate associating concepts from the three domains. The resulting comprehensive model and supported software will enhance product and project lifecycle management capabilities, yielding significant cuts in time-to-market, project risks, and product malfunctions.

To make the framework useful and beneficial, analysis mechanisms to evaluate the integrated model will be defined. This entails the provision of execution capabilities for evaluation of project constraints and assertions related to the cost, time limits, resource allocation, and "what-if" scenario analyses.

The system engineering management processes and activities are usually performed by practitioners using specific methods and tools in the project management domain. Databases of work breakdown structure (WBS) elements and properties, as well as project plans containing resources and time constraints, are usually maintained separately from product data. Overall project risk management suffers from this separation as well. Furthermore, the product data itself is spread among different databases, each dealing with a specific system engineering view. For example, product requirements

are usually managed via specific tools enabling their traceability to other product data and the generation of the System Requirements Document (SRD). The dynamic system view is usually constructed in a separate database, in which the product processes are modeled to enable the creation of the operational concept document (OCD). The system's architecture is defined by the components and interfaces between them. This view is frequently handled using specific tools for managing interfaces and producing various system interface documents, including Interface Requirements Specification (IRS) and Interface Control Document (ICD). Finally, test- and evaluation-related data of the system is often spread among a host of different tools, not necessarily traced back to the Test & Evaluation (T&E) plan.

The system engineering data gathered as the product evolves through its lifecycle is generally traced neither to the system engineering management plan (SEMP) nor to the project plans. The multiplicity of sources, even within each domain alone, prevents free flow of vital information, at the required situation, in an appropriate format, to the right people, so they can make the right decisions. Had the relevant project and product data been appropriately managed under a unified umbrella, the various data elements could be converted into valuable information at critical decision points. Figure 5 depicts the desirable combined timeline based on the currently separate product, project and enterprise time lines.

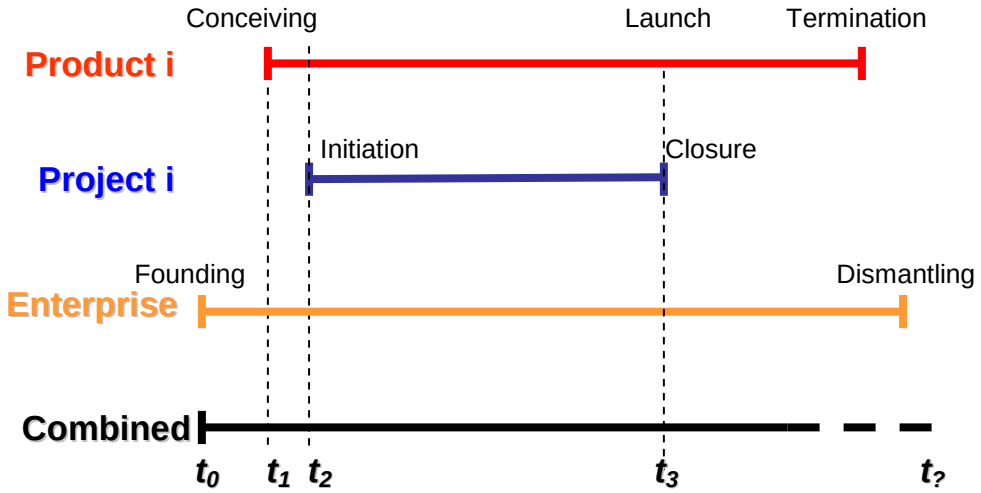

Figure 5 – The combined timeline of project and product lifecycles with the enterprise lifecycle

The combined timeline can become valuable since system engineering management of large complex systems, as well as systems of systems, is usually performed by incremental and iterative intertwined processes. Each of the separate product-project timelines at the upper level is only a projection of a deep structure of relevant timelines. Figure 6 shows the hierarchy structure of both product and project and the corresponding relationships, through the hierarchy decomposition.

The picture is even more complex when examining the lifecycle of systems at each level of the hierarchy in terms of their baselines, documents, and reviews, as depicted in Figure 7.

The effort to combine the domains into a unified lifecycle includes defining the different data elements, relationships required at each stage, and the information that can be extracted from the existing data. This can be achieved by capturing relevant data and dependencies found in a repository, e.g., a relational database tool. For example, when some project data will be combined with the relevant product data, the calculated lifecycle cost would be more realistic, providing tangible benefits to all the stakeholders.

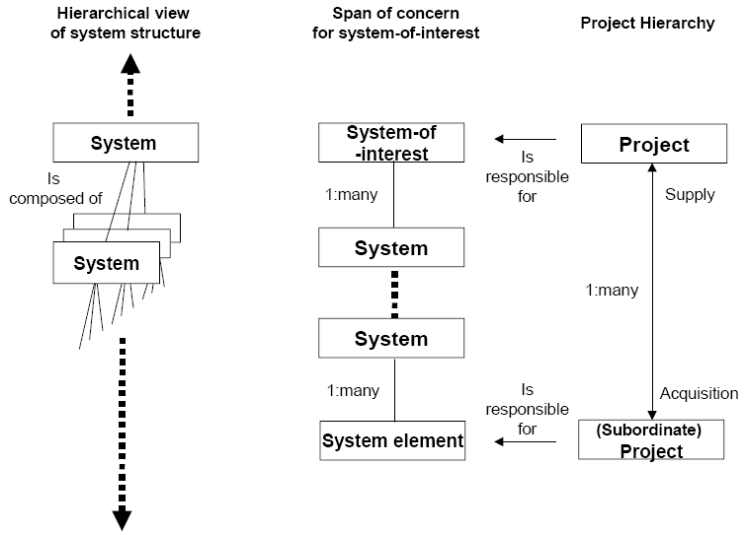

Figure 6 – System and project hierarchies
Source: ISO/IEC 15288, Figure D.4

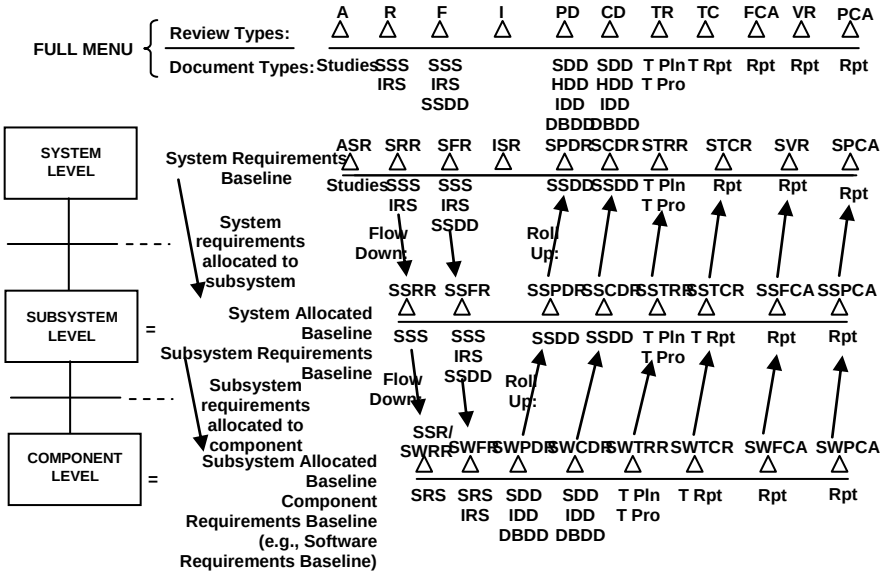

Figure 7 – Technical Baselines, Documents, and Reviews
Source: John O. C., "Systems Engineering and Software Engineering Processes, Products, and People from a Standards Perspective", Hampton Roads Area and Southern Maryland Chapters of INCOSE Systems Engineering Seminar, Slide 23, November 2003.

PPLM Approach and Framework

Since a major PPLM challenge is to unify data from different sources through a systematic model-based approach, the selection of the supporting model notation must be both formal and intuitive. The leading characteristics we looked for were (1) ability to represent large amounts of data in simple, hierarchically organized diagrams, (2) expressiveness of the notation, allowing the definition of a common PPLM conceptual metamodel, (3) formalism and clear semantics of the modeling notation as a basis for simulation and execution. In spite of the formalism requirement, since the PPLM potential users come from a wide range of disciplines and user profiles, such as project management and enterprise management, the notation should be simple and intuitive to all the potential PPLM users. Among the notations examined were SysML, UML and OPM (Dori 2002).

OPM was selected as the methodology for PPLM as it was found to have all the required

characteristics, and these are implemented in OPCAT (Dori 2003), its supporting software environment. OPCAT includes features and modules that are likely to enable the reduction of the development efforts, such as API, basic animation module, and integration with files in the CSV format, which is commonly used by various project management tools such as MS Project.

Figure 8 shows the top level framework of the PPLM approach; the Project-Product Lifecycle Management (PPLM) process requires both product and project data, obtained by Product Management PPLM Derivative process and Project Management PPLM Derivative process, respectively. Each one of these two processes is performed accordingly by the suitable management team, as a derivative of the overall management process performed over the entire project-product lifecycle. The Product-Project Management PPLM Derivative processes should yield the data required by the PPLM processes by extracting it as required from the appropriate Management Tool Set, provided that both domains follow the PPLM Framework methodology and guidelines, enabling data to be extracted from the common tools used in each domain. For example, using a predefined template, the PPLM processes will use the Origin Requirements data extracted from the appropriate tool in order to establish the relationships among those requirements and the WBS components, also received through a predefined template, from the specific WBS Tool. The PPLM Outputs will be introduced back to the management teams, to be used as part of the Product-Project Management PPLM Derivative processes.

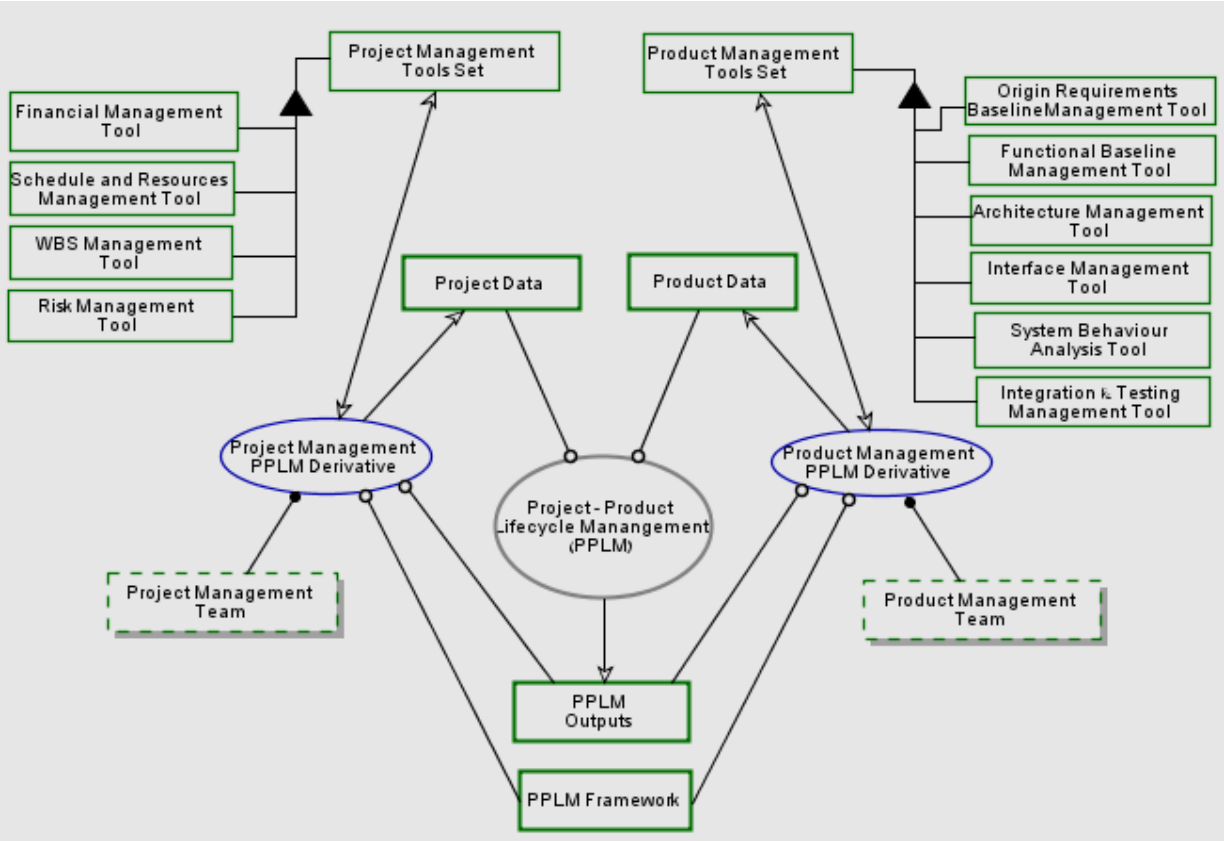

Figure 8 – PPLM Framework

The PPLM research outputs will include the specific methodology, guidelines and templates required, based on identified PPLM ontology. The PPLM methodology assumes iterative incremental development over the lifecycle span. The incremental strategy determines stakeholders' needs and expectations and defines the system requirements. It then performs the rest of the development in a sequence of increments. The increments successively incorporate parts of the planned capabilities, until the system is complete. Figure 9 shows the increments for the system level. Each increment contains

five phases: Definition, Development, Deployment, Operation & Maintenance, and finally Replacement or Dismantling.

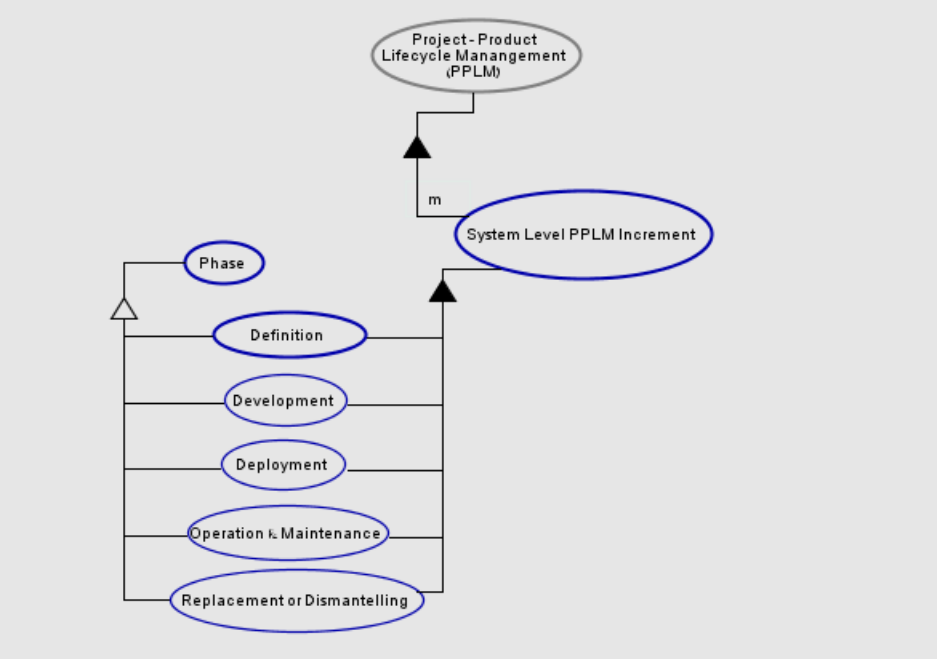

Figure 9 – The PPLM process unfolded

The PPLM Outputs, referred to in Figure 8, are traceability reports and relationship matrices among data elements of the project and of the product. Theses PPLM Outputs will be introduced back to the management teams to be used as part of the Product-Project Management PPLM Derivative processes, to enhance availability, transparency and reliability of the product-project development progress. For example, both teams will have a specific "traceability picture" of the Origin Requirements that are not addressed by any WBS element. These data items should be adjusted, synchronized and maintained during the entire lifecycle span in response to ongoing changes. As another example, once the product tree (Bill of Materials, BOM) is established, the PPLM Outputs will include a report of the product elements that are not covered by the WBS. Again, this data should also be adjusted, synchronized and maintained during the entire lifecycle span.

The establishment of relationships among data elements of the project and of the product is extremely valuable for determining and maintaining the Test and Evaluation Management Plan (TEMP) for the system. The uncertainties and risks to be reduced by the appropriate test and evaluation management plan activities can be integrated with the product data to define and maintain the exact configuration increments features, enabling continuous evaluation throughout the lifecycle. Furthermore, simulation can be executed in order to analyze cost and duration for a given functionality of each increment. An outcome could be optimized scheduling for the completion of requirements and design. The combined product-project model described using a formal, unambiguous visual modeling language will make it possible to construct simulation and testing of its functional and partially also its non-functional requirements, visualize possible system scenarios, and check some of the quantitative and qualitative characteristics.

The research is pursuing the following roadmap to attain its objectives:

1. Develop the **PPLM methodology**, including underlying **ontology** and a **conceptual model** based on this ontology that addresses all the relevant product, project, and enterprise aspects. The methodology will fuse the product to be developed with the project that creates it, including its milestones and deliverables, as they are executed by the enterprise. This development entails

conceptually defining, architecting, designing and modeling a comprehensive and adaptive PPLM methodology, which will integrally incorporate the product to be designed, manufactured, delivered, used, serviced, and disposed of, with the project, including all its significant artifacts, such as resources and their allocation, timetable, limits and milestones, evaluation of project constraints, and assertions related to its cost and risks.

2. Develop the **PPLM software deployment and execution environment** as a framework for formal design and continuous maintenance of the combined product and project lifecycles. The PPLM software environment will provide a novel framework for formal yet accessible design, execution, and maintenance of the integrated, synchronized and reciprocally affected product and project lifecycles across the organization. The conceptual ontology-based executable model will be verified for consistency, accuracy, and abilities to simulate a complex, software-intensive product lifecycle with the objectives of identifying risks associated with meeting deadlines, resource allocation, and bottlenecks.

3. Provide a **proof-of-concept** to the viability of the PPLM methodology and software environment as an effective executable infrastructure for jointly managing complex project-product pairs. This objective will be achieved by applying the PPLM approach to selected industrial case studies that represent different application domains. Feedback and lessons learnt from these industrial pilots will serve to improve the PPLM methodology and software deployment environment.

PPLM Illustrative Example

The following small-scale example illustrates a project-product combined model, which includes a project plan with milestones, product requirements, functions and a top-level architecture. The illustrated methods will demonstrate the evaluation of the PPLM environment: (1) product validation using the executable capabilities of the PPLM framework carried out at the conceptual top level architecture of the model, and (2) Forecasting the project progress and meeting milestone dates by analysis of the currently non-supported requirements.

The project, based on (Rosenbluth 2001), aims to develop a modern braking system for automatic vehicles. The system consists of an antilock braking system (ABS), a traction control system (TCS) and brake assisting system, called BAPlus, with an advanced feature for approximation of the distance to other vehicles that involves a radar component. The functional and business requirements, as well as the technical constraints, are detailed below.

Required braking system functions:

F 1 Preventing wheel locking,

F 2 Traction control – preventing loss of traction, and

F 3 Break assisting – detecting panic brake or emergency situation, in which another vehicle is too close and automatic assistance in reduction of the stopping distance.

System requirements:

R 1 The whole system weight shall not be greater than 2 kilograms.

R 1.1 Derived Requirement: ABS System shall weigh at most 1.5 kilograms.

Business requirements:

R 2 Cost reduction

R 2.1 The derived system technical requirements: The brake actuator and the wheel sensor system (WSS) shall be shared by the ABS, TCS, and BAPlus sub-systems.

R 2.1.1 ABS and TCS shall be interfaced.

R 2.1.2 ABS and BAPlus shall be interfaced.

R 2.2 The total cost of the braking system shall not exceed $800,000

R 2.2.1 ABS sub-system cost shall not exceed $500,000

Technical constraints:

R3 TCS and BAPlus shall have identical type of connectors that will interface with the WSS components of the ABS sub-system. Note: we suppose that the interfaces and the system architecture

conceptual modeling phases were carried out before the exemplified evaluating process invocation. Hence, the current constraint was defined beforehand.

ABS Project Schedule – Gantt Diagram

The project's Gantt describes the project working plan, as shown in Figure 10. All the review tasks are marked with black diamonds, symbolizing milestones. Black stripes refer to the work packages (WPs), whereas the red and the blue ones are the project critical path and regular project task durations, respectively. The arrows among the stripes depict dependencies among tasks, such as Finish-to-Finish (FF) and Finish-to-Start (FS).

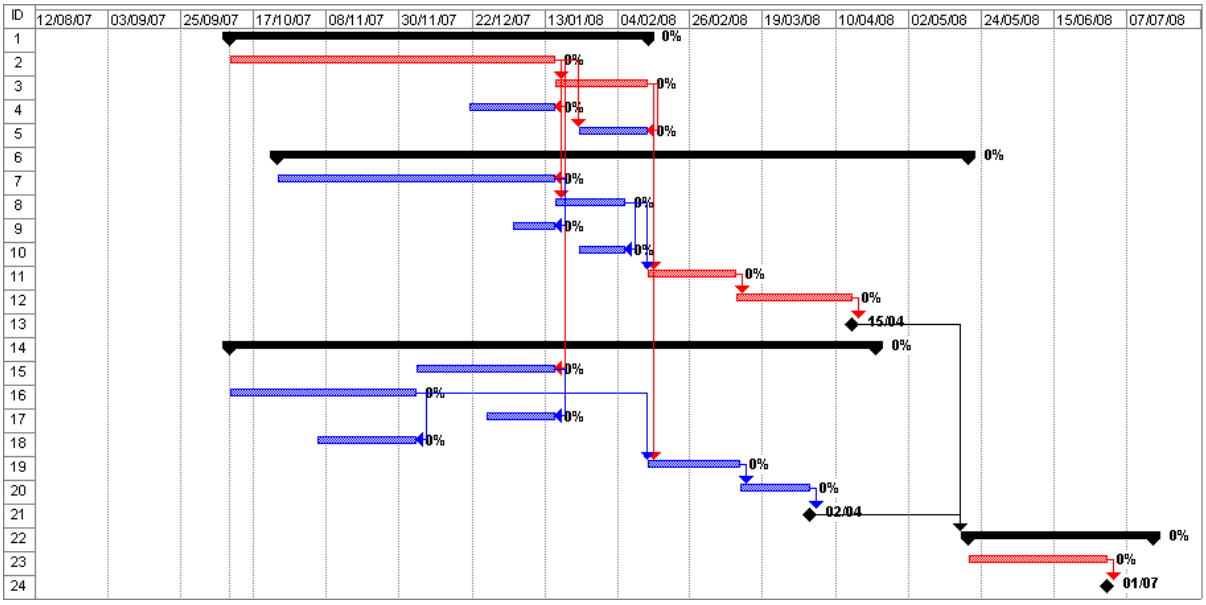

Figure 10 - Braking System Project Gantt Chart

The Braking System Architecture

The OPM System Diagram—the top level OPD in Figure 11—depicts the architecture of the system. Following OPM guidelines for top-down functional decomposition, the Brake Controlling process is the central process of the system, referring to the main function of the system. The Braking System process consists of three subsystems: ABS, TCS and BAPlus. The Braking System is the instrument for the Brake Controlling process and is part of the whole Car system, which in addition consists of four wheels and is characterized by the Velocity attribute. Both states of the velocity and of the wheels can be influenced by the Braking System Controlling process. The process can be invoked externally by the Operator or internally by the Emergency Braking Event, generated by the BAPlus subsystem. The event is generated as a result of BAPlus radar detecting that the vehicle is too close to another object, not shown at this level of detail.

The ABS subsystem OPD describing the ABS top level architecture is shown in Figure 12. The subsystem implements the required system function F1 – preventing wheel locking. Its central process is the ABS Antilocking function, which relates to the F1 function. Similarly, F2 and F3 are the functions related to the central processes of the TCS and BAPlus sub-systems, respectively.

ABS is composed of ECU and a set of four Hydraulic Modulators and four WSS components. Each one of the modulators actuates exactly one wheel of the Car; similarly each one of the WSS components monitors exactly one wheel. ECU is composed of the Microprocessor and ABS software running on the microprocessor and invoking the ABS Rule Set in accordance with the difference in the wheel speeds detected by the WSS components. The rules generate commands executed by the Hydraulic Modulators. For instance, the Hydraulic Modulator of some Wheel may assist in breaking

that Wheel if its speed was essentially higher than that of the other wheels. The ABS Antilocking process, which controls the car wheel speeds, is an operation (process feature) of the ABS subsystem. It is instrumented by the WSSs, Hydraulic Modulators and the Microprocessor. The additional attributes of the ABS subsystem are its cost and weight. We assume that all the parts (physical as well as informatical, or software components) are characterized by the cost attribute and its physical parts are also characterized by their weights.

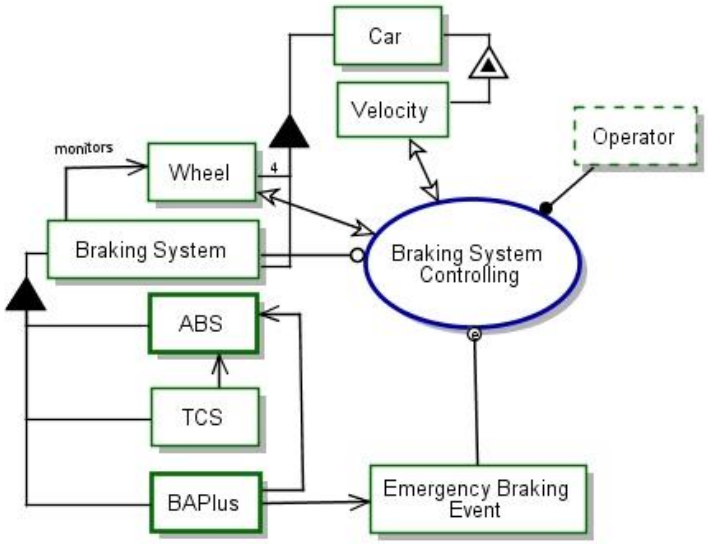

Figure 11 - OPM top-level diagram of the Braking System

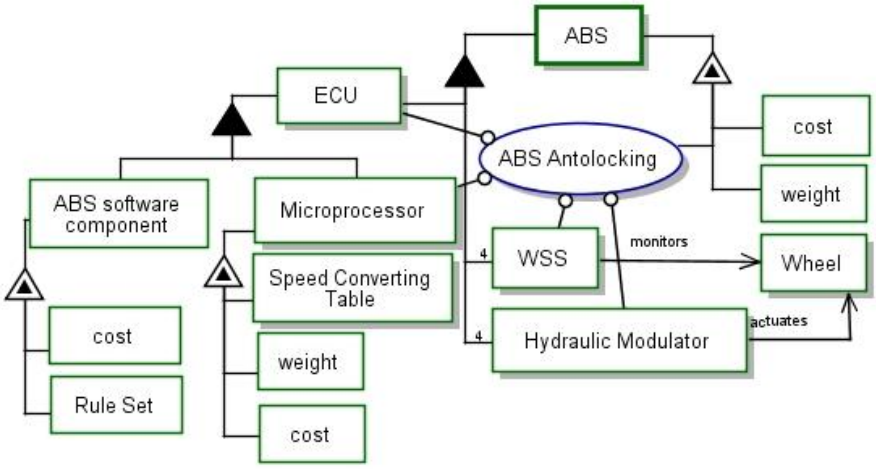

Figure 12 - ABS unfolded

Laying down the execution infrastructure

In the OPD depicted in Figure 13, few of the equations addressing technical constraints are modeled. These constraints relate to the requirements R1, R1.1 and R2.2. The model specifies calculating the weight and cost attributes of the system, subsystems and components.

For simplicity, we assume that the weight and cost of each system, subsystem, and component can be calculated as the sum of the weights or costs of their lower level parts plus an integration weight and an integration cost, whose values are predefined. For higher modularity, an abstract attribute can be defined, generalizing the behavior of the two attributes. Since this is less related to the current

discussion, this abstraction was not implemented in the example, but it is planned in the research.

The OPM executable will include well-defined basic executable processes for the OPM basic types (e.g., integer, float), such as Adding, which is the plus arithmetic operator, from which all the other calculations will be hierarchically composed. For example, the Averaging process operating on a set of integers will include adding them and Dividing the sum by the size of the set. In this example, we assume that the weight and cost attributes are of the double basic type (as defined in Java), and that the Weight Calculating and Cost Calculating processes use the Adding process.

The executable capability of the framework above makes it possible to execute the model. The executable model of this particular system will be embedded in the generic PPLM model and executed there as part of the large integrated project and product model. This will enable tradeoff analysis of various constraints. For example, if the resulting system is too heavy by 100 Kg, redesigning components for lighter weight will require four weeks in the project's critical path. Is the delay justified? Perhaps continue with the heavy parts now and in parallel redesign for lighter parts?

Figure 14 shows importing and invoke the Weight Calculating and Cost Calculating external processes from the current library, and checking the appropriate constraints.

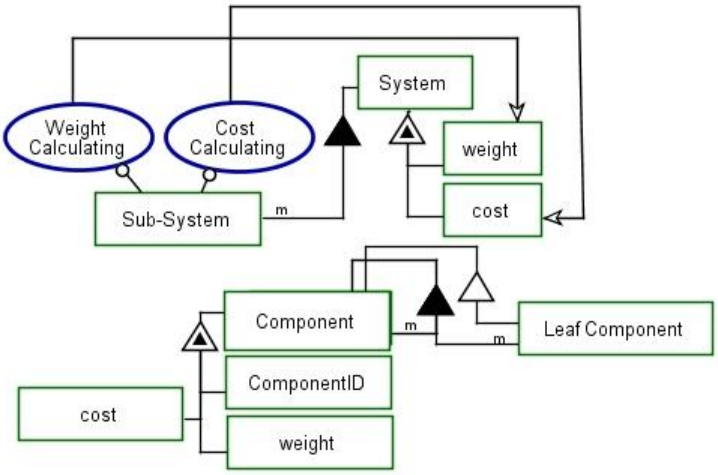

Figure 13 - Weight and Cost Calculating functions

Evaluation of quantitative non-functional requirements exemplified

The OPD depicted in the Figure 14 belongs to the PPLM model of the braking system. This model represents the ABS subsystem and its relation to one of the technical constraints. The technical constraint, represented by the OPM object R2.2.1, stands for the requirement R2.2.1—ABS sub-system cost shall not exceed $500,000. It is modeled with the corresponding validating process, involving the Cost Calculating process from the PPLM Parameter Calculations Library exemplified in Figure 13.

The assumption is that the braking system architecture is tagged using the PPLM ontology and the PPLM Parameter Calculations Library was defined using the same ontology. Thus, the matching of objects involved in the system process is done in correspondence with their PPLM roles as defined in the PPLM Parameter Calculations Library.

Execution of the validating process using the future OPM executable capabilities and including instances population with predefined start values could generate the actual cost of the ABS subsystem and determine whether requirement R2.2.1 is satisfied or violated by this subsystem. This type of the product model evaluation could be used by the system engineer and the technical management to comprehend the current state of the product and its implications at different phases of the project. This could be used in constructing prognoses of the further project evolution, as described in the following section.

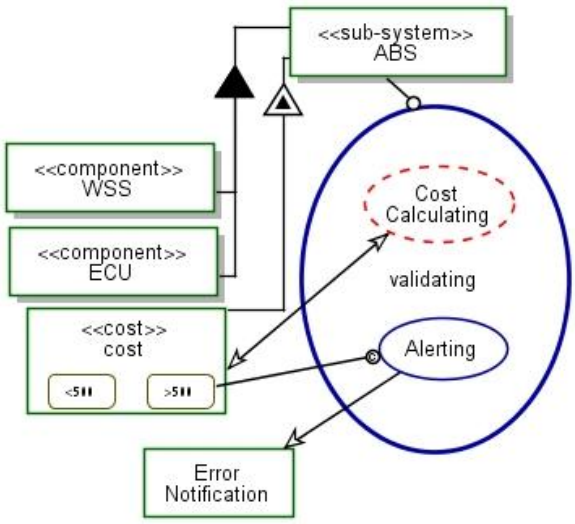

Figure 14 - Validating the R2.1.1 requirement in the PPLM model using the externally defined cumulative functions

Time vs. Performance tradeoff analysis

The following evaluation method example is a tradeoff analysis of delivery time vs. product performance. We assume that data depicted in the OPD of Figure 15 was constructed semi-automatically as a subset of the unified PPLM model for the braking system example. Since the example illustrates construction of the project prognosis, the time scale is essential. The assumption now is that the system engineering and project managers are currently executing task #12 – ABS & TCS integrating. They have used the PPLM model and invoked validating processes that had been defined for all the technical requirements related to the ABS and TCS subsystems. Based on the results of the processes and the original project plan and using the PPLM analyzing module, they find possible future influences of all the requirements that have not been satisfied by now.

Suppose requirement R2.1.1 – ABS and TCS shall be interfaced – was unsatisfied since the integration of the two subsystems TCS and ABS failed due to some architecture-level inconsistencies. This failure was detected through invocation of the OPM execution, simulation or animation modules on the unified PPLM model. The technical constraint R3 defines a strong relation between R2.1.1 and R.2.1.2 – ABS and BAPlus shall be interfaced. For the sake of simplicity, we assume that failure to integrate ABS with TCS means that the ABS interface was wrongly specified; thus, failure to satisfy the requirement R2.1.1 means failure to satisfy the equivalent requirement of the ABS integration with the BAPlus sub-system.

The PPLM model keeps information regarding all the complex dependencies: Relation between R2.1.1 and R2.1.2, relation between R2.1.1 and the implementing task #12, between R2.1.2 and task #20, etc. Using the above information, managers are able to plan the improvement actions, such as inserting new task to the project. Assume a new task named Fixing the ABS Portability was proposed. Execution/simulation of the updated relevant subset of the PPLM model will reveal the impossibility of arriving at the next milestone on time if the fixing task duration will be longer than X days. Thus, due to the well-defined relations between requirements and system components, between requirements and tasks, PPLM framework could forecast possible failure to meet the next milestone, ABS & BAPlus review task #20. The calculated Weight starting date of the task could be compared with the originally planned one. Note that without knowing the relations specified by the unified PPLM model, such forecasting should be carried out manually and be less accurate, especially for large scale systems.

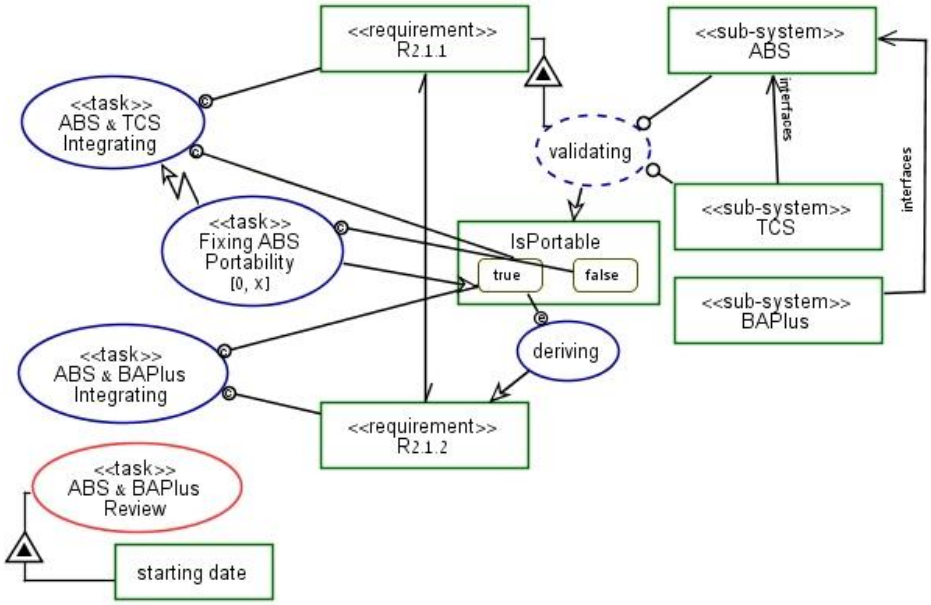

Figure 15 - Partial PPLM model of the Braking System

Discussion and Summary

The novelty of the proposed PPLM framework is its ability to serve as an instrument for ***improved simulation-based technology management and decision making processes***. The combined, integrated model will simultaneously express the function, structure and behavior of the three domains—the project, the product, and later on also the enterprise—via the same ontological foundations, making it possible to directly and precisely link entities in one domain to those in the other two, thereby attaining a holistic view of this highly complex and intricate ensemble.

The immense expected theoretical and practical value of the comprehensive PPLM methodology cannot be overestimated. At the level of systems engineering theory, the common ontology and the integrated conceptual model based on it will lay the foundations of a new field of scholarship resulting from the integration of the three domains—project, product, and enterprise—into an integrated conceptual model. On a less theoretical level, the novel PPLM approach will potentially pave the way for enterprises, large, medium or small, to the attainment of superior product quality and advanced project and product lifecycle management capabilities, yielding significant cuts in total cost, time to market, and risk.

PPLM is intended to serve as a prime engineering management vehicle to be used by the lead system engineering manager and her/his team, which will enable the following functions:

- **Adaptation:** Tailor a dedicated project plan for the specific product to be developed within the project by a specific enterprise through the use of predefined meta-model portions included in the PPLM library,
- **Executable simulation:** Using the execution capabilities of PPLM, simulate and adapt the project and product parameters for feasibility or even optimality of the project's designed timeline, resource allocation, and tradeoff of product functionality and deliverables,
- **Project-product management:** Concurrently monitor and control both the project and product evolution, take corrective actions in real time as needed if potential deviations are detected via the execution-based simulation or if actual deviations are detected via close monitoring of the interacting project-product lifecycles within the enterprise or in relation to subcontractors' deviations from their schedules, and

- **Change impact analysis:** Answer "what-if" questions that enable executives to assess the impact of changes requested by the customer or are about to be introduced to the product during its design due to technological, legislation, or economical considerations, in terms of impacting cost, risk, and time-to-market. What-if analysis based on possible architecture models can be conducted to help achieve objective and rationalized decisions regarding the appropriate architecture from the set of potential candidates. Such analysis can reveal contradicting requirements and find out corrective actions in order to improve the analyzed architecture. Finally, the system model analysis reduces the risk of developing a system with architecture that does not meet the requirements and the technical, legal, and/or environmental constraints.

References

ISO/IEC International Standard 15288, 2002.

Cook, S.C., Kasser, J.E., and Ferris, T.K.J. Elements of a Framework for the Engineering of Complex Systems, Proc. of the 9th ANZSYS Conference, Systems in Action, Melbourne, Australia, Paper No. 3000079 November 2003.

Hitchins, D.K. World class systems engineering – the five layer model. http://www.hitchins.co.uk, 2003.

Software Engineering Institute, CMMI http://www.sei.cmu.edu/cmmi

Dori, D. Object-Process Methodology – A Holistic Systems Paradigm, Springer Verlag, Berlin, Heidelberg, New York, 2002.

Dori, D. Reinhartz-Berger, I., and Sturm, A. Developing Complex Systems with Object-Process Methodology using OPCAT. Conceptual Modeling – ER 2003. Lecture Notes in Computer Science (2813), pp. 570-572, 2003.

John, O.C. Systems engineering and software engineering processes, products, and people from a standards perspective. Hampton Roads Area and Southern Maryland Chapters of INCOSE Systems Engineering Seminar, November 2003

Rosenbluth, W. Investigation and interpretation of black box data in automobiles: A guide to the concepts and formats of computer data in vehicle safety and control systems, pp. 72-80, ASTM Digital Library, http://www.astm.org/, 2001.

Biographies

Amira Sharon is a Senior Systems Engineer at IAI's ELTA Division and a Ph.D. candidate of Information Management Engineering at the Technion. She holds a B.Sc. in Mechanical Engineering (1992) and M.Sc. in Industrial Design (1997). Her professional career spans the areas of programming, systems engineering, and technical management of large scale projects. During the past seven years, she has been leading systems and software engineering methodologies development and implementation across IAI.

Valeria Perelman is a Ph.D. candidate of Information Management Engineering at the Technion. She holds a B.Sc. in Computer Science (2002) and M.Sc. in Information Management Engineering (2005), both at the Technion. Currently she also teaches following courses: Database Management, Software Specification and Design.

Dov Dori is Associate Professor and Head of the Information Systems Engineering Area at the Faculty of Industrial Engineering and Management, Technion, Israel Institute of Technology, and Research Affiliate at Massachusetts Institute of Technology. Prof. Dori has developed Object-Process Methodology (OPM), a holistic systems paradigm for conceptual modeling, presented in his 2002 book (by Springer). Prof. Dori is Fellow of the International Association for Pattern Recognition and Senior Member of IEEE and ACM. He authored six books and over 100 journal publications and book chapters.