Using the Software Process Improvement approach for defining a Methodology for Embedded Systems Development using the CMMI-DEV v1.2

García, I. and Herrera, A.

Postgraduate Department
Computer Science Faculty, Technological University of the Mixtec Region
Carretera a Acatlima km. 2.5. 69000 Oaxaca. Mexico.
{ ivan@mixteco.utm.mx; andreai@mixteco.utm.mx }

Abstract. Software process improvement holds a significant promise to reduce cycle times and provide greater value to all development activities involved in the software process development. While these methods appear to be well suited for embedded systems development, their use has not become an organized practice. In the same way as that of software development, the embedded systems development could be failing due a bad management in the development process. CMMI-DEV v1.2 is a process improvement maturity model that has been developed by the Software Engineering Institute at Carnegie Mellon. CMMI-DEV v1.2 defines "what" processes and activities need to be done and not "how" these processes and activities are done. In this paper we introduce the SPIES methodology that integrates the CMMI-DEV v1.2 Level 2 process areas to specify a process for developing embedded systems. This methodology incorporates the TSP principles to support the lack of management and improve the process specification. To illustrate this approach, we describe an experimental system in which it has been applied to develop and manage a traffic light system.

Keywords: Software process improvement, embedded systems, effective development process, improvement models, product focused improvement.

1 Introduction

Over the last 20 years, software's impact on embedded system functionality, as well as on the innovation and differentiation potential of new products, has grown rapidly. The consequence of this is an enormous increase in software complexity, shorter innovation cycle times, and an ever-growing demand for extrafunctional requirements (such as software safety, reliability, and timelines, for example) at affordable costs [15]. Moreover, these embedded systems' complexity is increasing, and the amount and variety of software in these products are growing. This creates a big challenge for embedded system development that was evident in the 2008 Embedded Market Survey, where respondents listed their three biggest concerns related to this task as

Advances in Computer Science and Engineering.
Research Computing Science 45, 2010, pp. 127-144

Received 30/04/10
Final version 19/05/10
Accepted 14/05/10

meeting schedules; the debugging process; and increased lines and complexity of the code. According to [19], of the same respondents a 59% also indicated that they are not using a formal development method or technique in their current embedded projects. As shown in Figure 1, meeting schedules is still the number one concern for developers. In fact, that concern actually increased by about 10% over last year. But, the most significant lesson is that meeting schedules is narrowly related to software development method, specifically to the planning process.

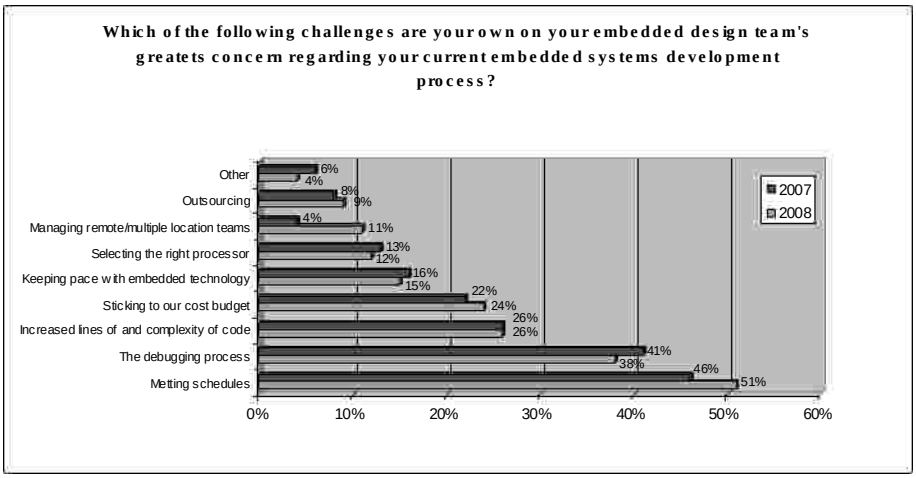


Fig. 1. Results of the 2008 embedded market study.

One potential reason for the poor adoption of development techniques is the gaps that exist when planning a detailed development lifecycle when the project begins. Traditionally, the gaps in the path of project planning and project monitoring and control to a solid product are overcome by experience and iteration. However for embedded systems, these gaps are more evident due to the need to obtain a realistic plan on affordable costs. This often results in embedded systems that exceed the established costs, overlap the schedule, and differ from the original functionality; necessitating extra time and resources to fulfill the initial requirements. Moreover, the limited availability of resources is preventing the introduction of new product features and applications, especially in areas where high-performance embedded systems are required [24]. Then, if there exists problems to develop conventional software products, are there problems related to develop the embedded software? In the same way, the commonly used software development process is a difficult task to perform when the developed software has to be run on an embedded system. Furthermore, due to cost, availability or developing reasons, the target system may not always be ready for the final phases (integration or testing time). These factors cause heavy bottlenecks when multiple developers work on the to-be-developed embedded system and need to concurrently access the development cycle for doing their tasks.

Just as we said, all these trends pose an urgent need for advanced embedded systems development techniques. However, and according to [14], the state of the art in embedded systems development is far behind other application areas. Graff et. al

studied seven European firms to determine the state of the practice in embedded software engineering [5]. One of the key findings in the study was that systems engineering decisions are largely being driven by hardware constraints, which then impact software efforts two stages later in the lifecycle when software requirements at the component level are developed. To optimize the timeliness, productivity, and quality of embedded systems development, companies must adapt software engineering technologies that are appropriate for specific situations. Unfortunately, the many available software development technologies do not take into account the specific needs of embedded systems development. In fact, developers do not tend to develop products with standardizations. For example, in Alcatel Shangai Bell, the project scales have been different from several person/months to 150 persons/months. Project outlay is different from thousands USD to million USD, many of which were cancelled or failed during the development because of delay, over cost, out of control and customer unsatisfied. Problems being encountered in other enterprises can be found in this company too.

Our research aims to facilitate an alternative methodology for embedded systems development. We establish a formal development method oriented to embedded application fields which we called SPIES (Software Process Improvement for Embedded Systems). The SPIES methodology consists of three essential elements: activities, assets and tools. SPIES is supported by the CMMI-DEV v1.2 [27] to guide the whole project development.

The rest of this paper is organized as follows. Section 2 summarizes the related work of methodologies for developing embedded systems. Section 3 provides a brief description of CMMI-DEV v1.2. Section 4 formally describes the SPIES phases. A case study is presented in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

According to our literature review, much efforts have been paid to establish "what activities to implement" instead of "how to implement" these activities for embedded systems development. The current problem with embedded systems development is not a lack of standards or models, but rather a lack of effective strategy to successfully implement these standards or models. However, according to the MOOSE (Software Engineering *M*eth*o*dol*o*gies for *E*mbedded Systems) project [18] we should consider other factors which could affect the project success:

- Coordinating all subprocesses (p.e. mechanical engineering, electrical engineering) to develop quality products is one of embedded system development's most challenging aspects.
- Systems engineering was mostly hardware driven – that is, from a mechanical or an electronic viewpoint. Consequently, software development started when hardware development was already at a stage where changes would be expensive.

Thus, since MOOSE results many approaches have been developed to incorporate formality to the embedded systems development process. In the following, we summarize the most significant work.

- The research by [29] was one of the first approaches that addressed conceptual models for relating together embedded systems' product and process characteristics. The purpose of the models was to describe in detail the characteristics of embedded systems for enhancing ISO 15504:1998 (SPICE) [10] conformant assessment methods to cover embedded systems' software process assessment. The analysis of embedded systems was based on the domain expertise of the PROFES (PROduct Focused improvement for Embedded Software processes) partners, a literature survey and assessments performed at several industrial sites.
- Years later, the enhancement of embedded products focused its efforts in the components reuse. The Koala component model [20], for example, was used for embedded software in consumer electronics devices, which allows late binding of reusable components with no additional overhead, but it does no take a development lifecycle into account and lacks a formal model.
- Later, PECOS [6] attempted to enable component-based technology for a certain class of embedded systems known as "field devices". The main features of this model are the data-flow-oriented programming style and the explicit incorporation of non-functional requirements. However, PECOS merely supports the non-functional properties of memory consumption and real-time, but does not provide any formal method supports, and only supports the data type interface.
- Since 2004, Miller and Smith have successfully used a prototype Test-Driven Development (TDD) [17] embedded system test framework, called Embedded-Unit, in their undergraduate classes and research work. In particular, the framework provides a stable, easy-to-learn environment through which students could solve problems generated at both the hardware and software levels. The TDD approach to support embedded system customer acceptance tests through Matlab-Fit and Embedded-FitNesse is best described as showing "potential".
- The Simplified Parallel Processes (SPP) [12] that provides a Software Process Improvement (SPI) solution referring CMMI [26] Level 2 and 3 for middle size enterprises. This research illustrates the development of a software processes management tool, called "Future", based on SPP to provide technical development specification for embedded systems. However, there doesn't exist a detailed explanation of SPP nor any real environmental validation or data of successful implementation.
- The REMES model for embedded systems by Seceleanu et. al [24] introduces a formal modeling and analysis of embedded resources such as storage, energy, communication, and computation. This model is a state-machine based behavioral language with support for hierarchical modeling, resource annotations, continuous time, and notions of explicit entry and exit points that make it suitable for component-based modeling of embedded systems. However, the analysis of REMES-based systems is only centered

around a weighted sum in which the variables represent the amounts of consumed resources.

- The research by [14] presents a formal model for specification, verification, and composition of component-based embedded software which they called ESCM (Embedded Software Component Model). Li et. al describe how components are specified from the syntactical view, functional view, QoS view and synchronization view. The refinement rules for functionality, QoS, and synchronous behavior are defined for the verification purpose and a lightweight method is provided for the purpose of composition. This approach uses five contracts to give a formal specification of ESCM from the four separated levels view. ESCM' strict specification is required if one developer wants to safely reuse a component.
- Sentilles et. al refine the component-based approaches through the development of Save-IDE [25]. Save-ID is an Integrated Development Environment for the development of component-based embedded systems that supports efficient development of dependable embedded systems using a dedicated component model, formal specification and analysis of component and system behaviors already in early development phases. In fact, the main contribution of Save-ID is related with its effort to establish a software development process, designated SaveCCT – SaveComp Component Technology, with three major phases: design, analysis and realization.

As Karsai et. al note in [13]: *"the development of software for embedded systems is difficult because these systems are part of a physical environment whose complex dynamics and timing requirements have to be satisfied"* Furthermore, the projects often lack an effective software development methodology [7]. Our contribution is more related with this last approach, defining and implementing a SPI effort to improve the development process establishing *effective practices* acquired from the good experiences of software commercial models, but covering the hardware elements implementation too.

3 The Capability Maturity Model Integration for Development

According to SEI, "CMMI is a process improvement maturity model for the development of products and services. It consists of best practices that address development and maintenance activities that cover the product lifecycle from conception through delivery and maintenance. This latest iteration of the model as represented herein, integrates bodies of knowledge that are essential for development and maintenance. These, however, have been addressed separately in the past, such as software engineering, systems engineering, hardware and design engineering, the engineering "-ilities," and acquisition" [26]. The prior designations of CMMI for systems engineering and software engineering (CMMI-SE/SW), are superseded by the title "CMMI for Development", to truly reflect the comprehensive integration of these bodies of knowledge and the application of the model within the organization.

CMMI-DEV v1.2 provides a comprehensive integrated solution for the development and maintenance activities applied to products and services.

The CMMI-DEV official report indicates that: *"CMMI for Development v1.2 is a continuation and update of CMMI V1.1 and has been facilitated by the concept of CMMI "constellations" wherein a set of core components can be augmented by additional material to provide application-specific models with highly common content. CMMI-DEV is the first of such constellations and represents the development area of interest"*[1].

To improve software development practices, practitioners, projects, and organizations must move from ad hoc practices to explicit software development practices. Using CMMI-DEV v1.2 [27] and the IDEAL model [16] organizations can do just that. The IDEAL Model is used as the approach for this improvement process. The steps of the IDEAL Model are outlined below.

- *Initiating:* Laying the groundwork for a successful improvement effort.
- *Diagnosing:* Determining where you are relative to where you want to be.
- *Establishing:* Planning the specifics of how you will reach your destination.
- *Acting:* Doing the work according the plan.
- *Learning:* Learning from experience and improving your ability to adopt new technologies in the future.

Using the IDEAL model and CMMI-DEV v1.2, a process improvement team would improve its organization's software development practices. We believe that this approach can be applied with a high level of success to the embedded systems development. CMMI-DEV v1.2 is composed by 22 process areas which are divided in four categories and five maturity levels and/or capability levels to guide the process improvement. Table 1 shows that SPIES covers basically process areas of CMMI-DEV v1.2 Level 2, it means establishes a "managed process".

Table 1. CMMI-DEV v1.2 process areas for Level 2

Process area	Category	Maturity Level	Capability Level				
			1	2	3	4	5
Requirements Management	Engineering	2	Target Profile 2				
Project Planning	Project Management	2					
Project Monitoring and Control	Project Management	2					
Supplier Agreement Management	Project Management	2					
Measurement and Analysis	Support	2					

[1] The differences between v1.2 and v1.1 are explained in detail on the SEI Web site (www.sei.cmu.edu) in CMU/SEI 2006 TR-008.

Process and product quality assurance	Support	2				
Configuration Management	Support	2				

To achieve maturity level 2, all process areas assigned to maturity level 2 must achieve capability level 2 or higher. However, individual process areas can complete all practices and to achieve capability level 2. The relationship between process areas and SPIES' product life cycle are shown in the following section.

4 The SPI for Embedded Systems Methodology

According to [22], a methodology consists of a language to specify elements and relationships among system's components, and a whole process (activities, products, inputs, outputs, metrics, entry criteria, exit criteria, roles, and more), which indicates what parts of language to use, how to use them, and when to use them. SPIES specifies an integrated set of activities (adapted from CMMI-DEV v1.2) to guide developers during all development lifecycles to develop robust, capable and secure systems. Figure 2 shows the elements that interact to establish a SPI methodology for embedded systems development. The development process is designed as a top-down approach with an emphasis on continuous improvement as exposed in [28] [30] and [1].

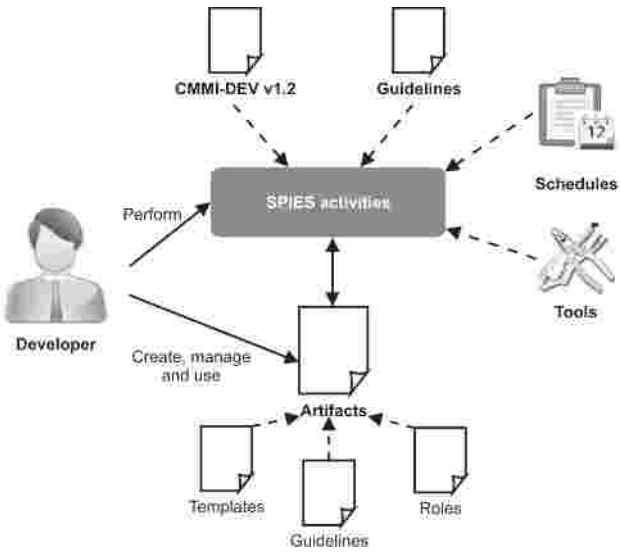


Fig. 2. The nature of SPIES methodology.

SPIES is organized by phases which are composed by more specific activities. The iterative approach of SPIES ensures that the development process be tested in each phase and not until the end of the project. As shown in Figure 2, our methodology uses an artifacts repository (or repository of assets) to introduce the process improvement in each phase. This repository contains templates for each activity (products), guidelines to perform activities, and role assignation. On the same way that CMMI-DEV v1.2, SPIES recommends for each activity (as possible) the use of an specific tool for project planning, estimation and effort process, requirements modeling, system validation, and more. For example, SPIES establish a set of activities to perform within the Requirements Specification phase using the Rhapsody Tool [23] from Telelogic (see Section 4 for a detailed explanation). SPIES use the basic idea of TSP [9] to establish a set of documents to guide developers in project management. The variation for incorporating this repository of knowledge to each phase ensures us high levels of success.

As we said, SPIES is extracted from process areas and specific practices in CMMI-DEV v1.2 level 2, which are simplified for embedded software demands. Figure 3 shows the relationship between CMMI-DEV v1.2 (at process areas level) and SPIES. Our methodology is composed by three layers and eight phases. Layers are dependent and related progresses. Every developer knows when to do an activity according the SPIES specification.

- The *management layer* provides the needed process to control the whole project development. This layer begins with the realization of a project plan and closes with the lessons learned in the final stage. These lessons are stored in the knowledge repository as needed effort, time, resources, people and more. This layer is composed by two process areas: Planning (PLA) and Product Continuous Improvement (PCI).
- The *engineering layer* provides the activities related to develop the complete system. The TSP artifacts are used in each SPIES phase (included in the Project Plan in previous layer) and determine when developers can begin the next phase through entry and exit criteria. The layer is composed by six process areas: Requirements Specification (RES), Product Design (PDS), Product Development (PDE), Product Integration (PIN), Product Validation (PVAL), and Product Delivery and Maintenance (PDM).
- The *support layer* provides help to achieve the expected quality through establishing activities for configuration and managing contracts and measuring them continuously. The layer is composed of three process areas: Configuration Management (CMA), Product and Process Quality Assurance (PPQ), Contract Management (CMG) and Measurement and Analysis (MAN).

Development process specification of Figure 3 belongs to CMMI-DEV v1.2' engineering processes category (Requirements Specification, Product Design, Product Integration, and Product Validation). Therefore, concrete technical development specification (Product Development) is necessary for applications. Specification defines phases of processes, activities and subactivities, entry and exit criteria, measures, etc; templates that illustrate formats for different documents, and how to

fulfill them; guidelines that specify how to realize and tailor specification and phases. Thus, the development process for embedded systems includes technical development specifications and their templates, guidelines, and more assets. In addition, hardware related processes are emphasized here.

The addition of the PLA process for embedded systems improve the conventional methods commonly used for embedded systems development in industrial environments; (for example the Model-Driven Architecture [21]; standard IEC 61508 [11]; the UML-based Rapid Object-Oriented Process for Embedded Systems, which is based on the spiral process model [2]; and different approaches based on a V-Model [3]) to establish realistic plans.

Across the eight phases of SPIES the information flows in the form of processes assets. The knowledge repository manages all information about the project (from project plans and contracts to improvement information – effort, time, and more) to continuously improve the embedded system development process.

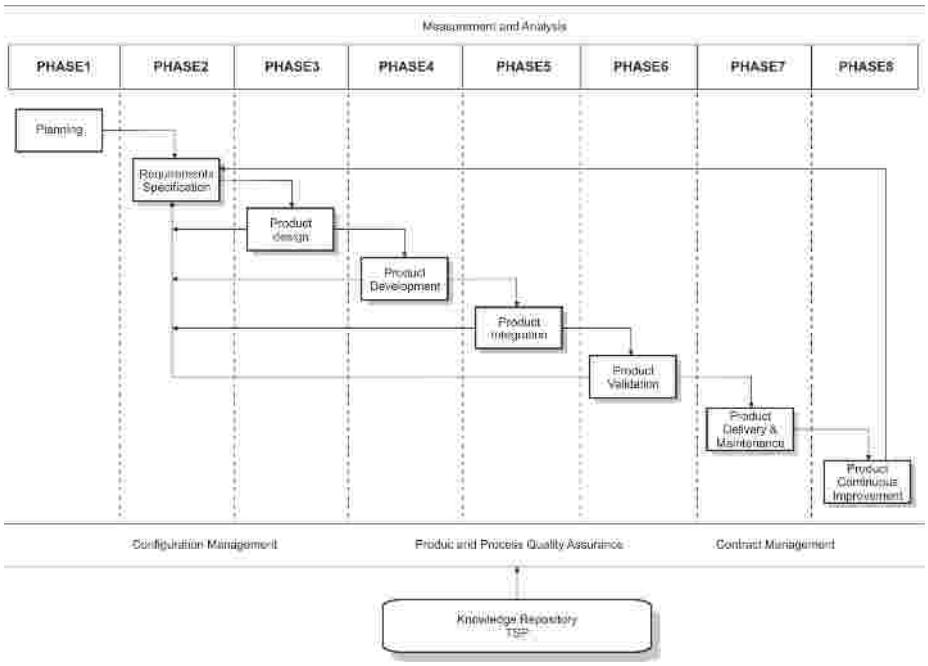


Fig. 3. SPIES phases and process.

According to Hansen et. al [8] *"model can be found to solve problems but there is a gap between theory and practice"*. Thus, we think that a suite of general SPI solutions can be outlined for different enterprises which look for establishing a "quality development process". We try to unify "theory" and "practice" through the use of a knowledge repository using the TSP practices and templates as support and guiding developers through the CMMI-DEV v1.2 effective practices.

4.1 The SPIES structure

The SPIES structure enables developers to easily tailor specifications for their own necessities. SPIES has, from PHASE1 to PHASE8, 16 process areas and about 25 templates. When applying SPIES it is suggested modifying properly according to concrete situations (such as measures, capability levels, entry and exit criteria, etc.). In SPIES step by step is very important through an iterative approach. Figure 4 provides a detailed explanation of SPIES processes (CMA, PPQ, CMG, PPQ, and MAN are not shown because their activities are implicit in the other 11 processes).

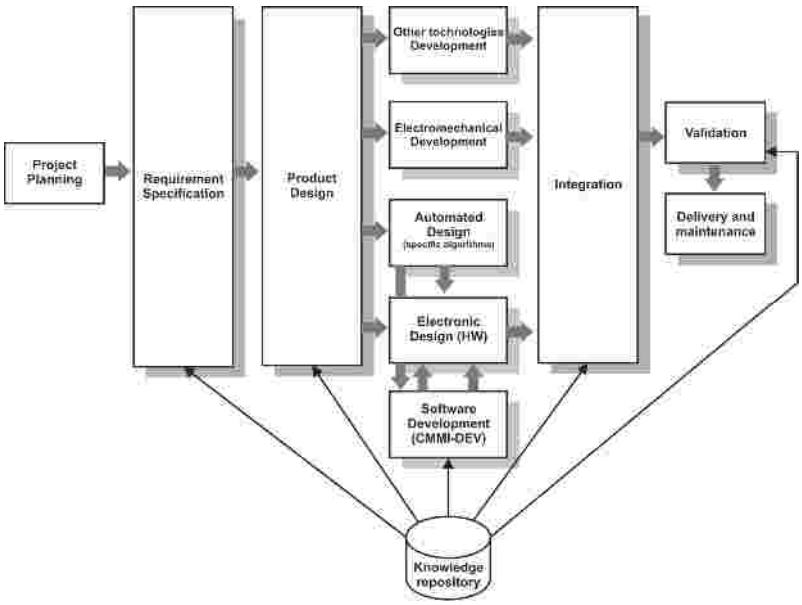


Fig. 4. SPIES process areas.

The contents of SPIES process can not be modified, while the modification of the previous projects assets (as documents, templates, guidelines, estimations, etc.) is enabled. The TSP model provides all templates from project initialization (planning) to project closuring (delivery and maintenance). We provide a brief description of SPIES processes:

- **Project planning.** This process provides guide to establish and maintain plans that define project activities. Planning includes estimating the attributes of products and tasks, determining the resources needed, producing a schedule, identifying and analyzing project risks, and considerations for choosing the right microprocessor for an embedded application. Iterating through these SPIES' activities may be necessary to establish the project plan.
- **Requirement specification.** This process provides activities needed for requirements development and requirements management. Depending on the modeling languages used to describe requirements, developers can use

graphical languages such as UML or the Systems Modeling Language (SysML) to model requirements. SPIES recommends activities to use Rhapsody Tool from Telelogic.

- **Product design.** This process provides activities for creating a system's effective design based on the previous requirements and focused on two issues: functional design and architecture design. Functional designs cover an embedded system's functionalities without considering any technical implementation details. SPIES indicates to developers how to define the system's architecture on the basis on the requirements and functional design. The obtained architecture definition consists of various views covering the architecture's different aspects.
- **Other technologies development.** The quality enables performing the objective or subjective evaluation of the performed functionality. It is possible that the embedded system should use other technologies which affect the planned functionality. Thus, this process provides practices to develop embedded systems without technological dependencies.
- **Electromechanical development.** This process provides activities to enable developers to manage and synchronize information in elements of electrical, mechanical and software designs. SPIES performs automatic controlled reviews to identify potential problems among disciplines and use collaboration tools to quickly solve incidences in the process.
- **Automated design.** This process provides practices to integrate all system's components into an unified model. The process considers third-party requirements during the design of an embedded system and its software. A whole simulation is carried out.
- **Electronic design.** This process contains activities to use tools for designing and producing electronic systems ranging from printed circuit boards (PCBs) to integrated circuits. This is sometimes referred to as ECAD (electronic computer-aided design) or just CAD.
- **Software development.** This process contains activities to refine or extend platform-independent models in the platform-specific design, to support efficient code generation for the target execution platform. SPIES support the concurrent development with the hardware and other components of the embedded system through design and development phases.
- **Integration.** This process provides activities to integrate the components in several steps before the embedded system is tested. The integration phase of the development cycle must have special tools and methods to manage the complexity. The process of integrating embedded software and hardware is an exercise in debugging and discovery.
- **Validation.** The last validation phase may include extensive field testing and validation, before the embedded system is ready for delivery and maintenance.
- **Delivery and maintenance.** The majority of embedded system designers (around 60 percent) maintain and upgrades existing products, rather than design new products. Developers should use activities to use the existing documentation and the embedded system to understand the original design well enough to maintain and improve it. This process requires tools that are

especially tailored to reverse engineering and rapidly facilitating "what if" scenarios.

- **Measurement and analysis.** The measurement process is a prerequisite for all previous processes and for the successful process improvement. In this context, the SPIES measurement should be used for two purposes: evaluating conformance of an embedded system to the quality specification, and evaluating process-system relationships. This conceptual methodology includes measurement explicitly and applies it for these two purposes. SPIES defines criteria to which an embedded system development focused on SPI needs to comply. These criteria are depicted in Figure 5, together with an overview of the relationships between the criteria and the proposed layers.

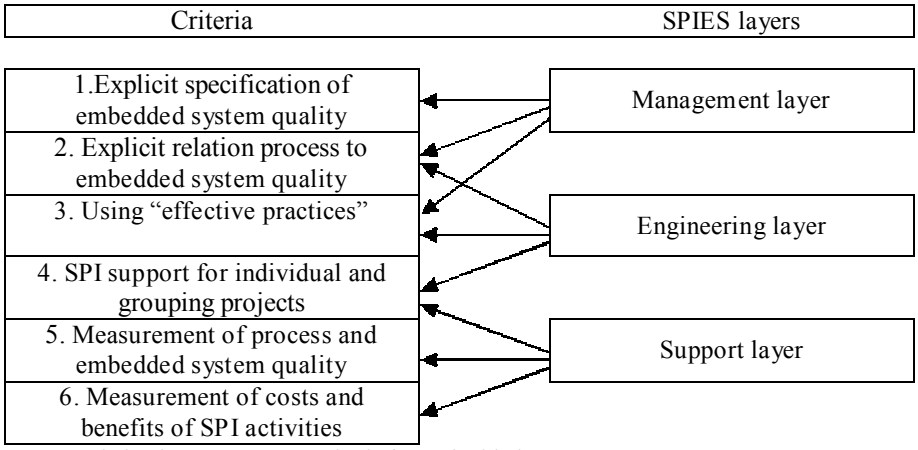


Fig. 5. Relation between SPIES criteria for embedded systems.

5 Testing the SPIES methodology

To evaluate the applicability of our methodology, we have successfully implemented a prototype with SPIES development, called the traffic light system, in our undergraduate classes and research work. During the original course, we gave students a complete course of embedded systems development. The traffic light system was designed to be used inside the faculty campus and includes three components: the sensor component for detecting vehicles and pedestrians, a Front Panel Display (FPD) to automatically configure the system, and the internal component to manage the traffic. It is important to say that as we are talking about a University environment, there is no such traffic as a common street. As illustrated in Figure 6, the traffic management component includes five modes: safe intersection mode, evening low volume mode, responsive cycle mode, fixed cycle time mode, and adaptive mode. The sensor can detect vehicles (priority vehicle and emergency vehicle) and pedestrians. This figure was obtained by RES activities related to Rhapsody tool.

We will briefly explain how the SPIES methodology is used. Our methodology begins with the planning process. Our students fulfilled the "Project plan template" to start the embedded system development. Managing embedded projects can be a conflict in disciplines, with the need to foster engineering creativity while wrapping the students in enough process to keep them on track. SPIES grouping contains effective practices for creating a project plan, tracking the plan against reality, managing contracts and delivering a project on time and budget. Figure 7 shows an example of two SPIES templates to generate a project plan.

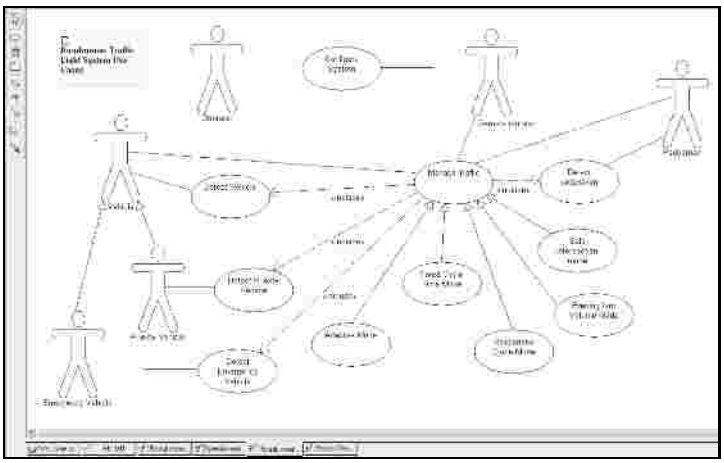


Fig. 6. The traffic light system.

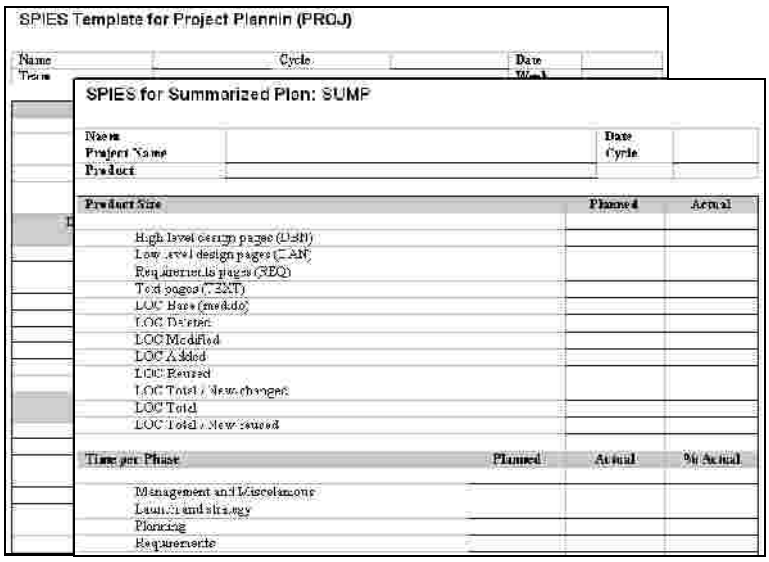


Fig. 7. An example of project plan generated by SPIES.

Table 2. An example of SPIES activities for RES process

RES 1.1	Identify and describe ideas for new system
RES 1.2	Prepare preliminary description of the new system
RES 1.3	Establish system requirements
RES 1.4	Identify interface requirements
RES 1.5	Establish operational and functional scenarios
RES 1.6	Establish a definition of required functionality
RES 1.7	Validate requirements
RES 1.8	Manage requirements changes
RES 1.9	Maintain bidirectional traceability of requirements

Student responsibility for collecting information and feedback from the experience plays an important role in this phase. All the collected information is updated in the knowledge repository to increase the set of effective practices and relate it to future projects. The role of RES is emphasized in the Product Design phase, where the system level architecture of the system implementation plan is refined with specifications for the software, electronic hardware, mechanical hardware, and more. RES 1.2, 1.3, 1.4 and 1.5 activities are supported by additional activities related to Rhapsody Tool. SPIES includes these activities to establish a formal process for obtaining and understanding the embedded system requirements. Figure 8 illustrates a set of general requirements, based on SPIES activities for modeling on Rhapsody, for our experimental project.

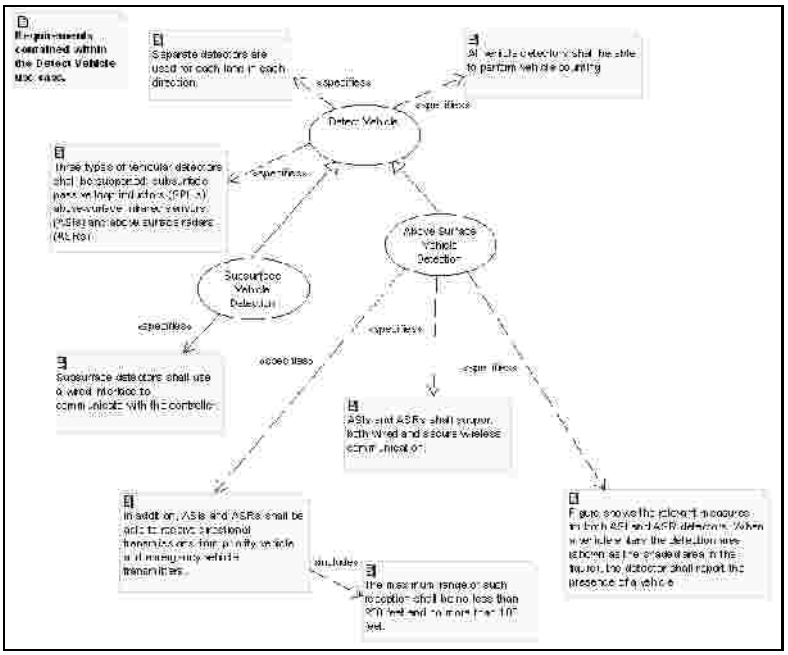
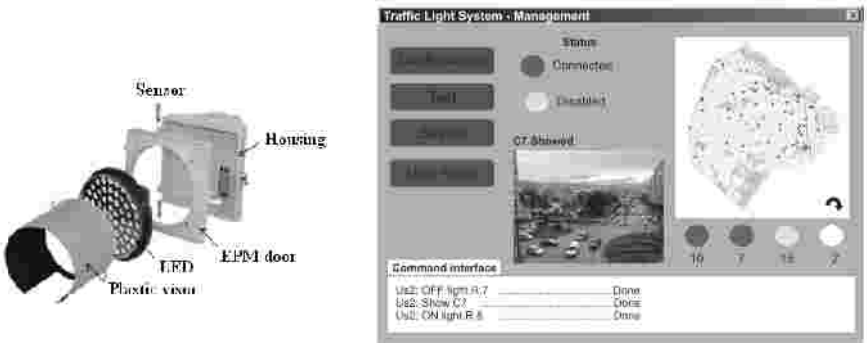


Fig. 8. SPIES in requirements modeled with Rhapsody.

In the traffic light system described above, we only give a general explanation of how SPIES works in specific process areas (specifically RES) and omit other processes specification which are given in Section 4.1 because of the length. A photograph of the traffic light system can be seen in Figure 9. It is developed with our SPIES methodology and software architecture and was developed in our group in collaboration with postgraduate students and research partners.



a) b)

Fig. 9. a) The traffic light component. b) The management traffic light system.

Fortunately, using the engineering layer (discussed earlier in Section 4), we can shield the embedded software from hardware dependencies. Hence, the embedded software could be designed by following the SPIES activities and by modeling and simulation in Matlab/Simulink and Rhapsody tool, while abstracting from hardware details. Finally, the code is generated in C++ and deployed.

5 Conclusions and Future Research

The methods used, tools and techniques for embedded systems development do not only vary between companies and research institutes, but also within companies and universities themselves. Frequently, there is a general high-level approach present, but not much is standardized on a more detailed level. Thus different projects often use different tools and notations. Besides, a relevant factor mentioned in [4] when applying SPI approach to embedded systems development is the limited management support that disenables the successful factor. We think that the combination of two quality models (CMMI-DEV v1.2 and TSP) to improve the development cycle can support a formal process to avoid this lack of management. It is true that SPI depends on a company's capabilities (human, infrastructure and resources) and business strategies. But, the process development relies on standardization management too.
CMMI-DEV v1.2 concentrates on project management and pay attention to technical course. We think that the standardization on technical processes for embedded systems is a crucial element to improve the quality of the final product.

In this paper, we presented a methodology for developing embedded systems using the SPI approach. We define the SPIES methodology and phases for the development process specification for embedded systems supported by TSP model. The SPIES methodology is based on effective practices and cheap compared with buying tools separately (in fact our methodology is Open Source). We briefly show experience about the implementation of a traffic light system using SPIES methodology.
One of the observed disadvantages of SPIES is related to the previous knowledge on quality models that could be difficult to use. The repository knowledge can solve this problem through feedback from users.

As future work, we will refine effective practices through experimentation giving a formal architecture style for the construction of an automated software that implements the practices reflected in SPIES.

We will implement our methodology in enterprises, and modify it according to users' lessons learned. Finally, more experiments in academia will be defined and carried out at one or more of the industrial partners to test SPIES in real-world environments.

References

1. Basili, V., McGarry, F., Pajerski, R. & Zelkowitz, M. "Lessons learned from 25 years of process improvement: The Rise and Fall of the NASA Software Engineering Laboratory" *Proc. of the 24th International Conference on Software Engineering* (ICSE 2002), pp. 69-79, 2002.
2. Boehm, B. "Guidelines for Verifying and Validating Software Requirements and Design Specification" *Proc. of the European Conference of Applied Information Technology* (Euro IFIP), North-Holland, pp. 711-719, 1979.
3. Boehm, B. "A Spiral Model of Software Development and Enhancement" *Computer*, 21(5):61-72, 1988.
4. Graaf, B., Lormans, M. & Toetenel, H. "Software Technologies for Embedded Systems: An Industry Inventory" *Proc. of the 4th International Conference on Product Focused Software Process Improvement* (PROFES 2002), LNCS 2559, pp. 453-465, 2002.
5. Graaf, B., Lormans, M. & Toetenel, H. "Embedded Software Engineering: The State of the Practice" *IEEE Software*, 20(6): 61-69, 2003.
6. Genßler, T., Christoph, A., Winter, M., Nierstrasz, O., Ducasse, S., Wuyts, R., Arévalo, G., Schönhage, B., Müller, P. & Stich, C. "Components for embedded software: the PECOS approach" *Proc. of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems* (CASES 2002), ACM press, pp. 19-26, 2002.
7. Greene, B. "Agile methods applied to embedded firmware development" *Proc. of the Agile Development Conference* (ADC 2004), pp. 71-77, 2004.
8. Hansen, B., Rose, J. & Tjørnehøj, G. "Prescription, description, reflection: the Shape of the Software Process Improvement Field" *International Journal of Information Management*, 24(6): 457-472, December 2004.

9. Humphrey, W. "Introduction to Team Software Process", Addison-Wesley, Reading, MA, 2000.
10. ISO/IEC TR 15504:1998(E). *Information Technology – Software Process Assessments. Parts 1-9*. International Organization for Standardization: Geneva, 1998.
11. International Electrotechnical Commission. IEC 61508, Functional Safety of Electrical/Electronical/Programmable Electronic Safety-Related Systems, 1998.
12. Jun, D., Rui, L. & Yi-min, H. "Software Processes Improvement and Specifications for Embedded Systems" *Proc. of the 5th ACIS International Conference on Software Engineering Research, Management & Applications* (SERA 2007), IEEE Computer Society, pp.13-18, 2007.
13. Karsai, G., Sztipanovits, J., Ledeczi, A., "Model-integrated development of embedded software" *Proceedings of the IEEE*, 91(1): 145-164, 2003.
14. Li, C., Zhou, X., Dong, Y. & Yu, Z. "A Formal Model for Component-Bases Embedded Software Development" *Proc. of the International Conference on Embedded Software and Systems* (ICESS 2009), IEEE Computer Society, pp. 19-23, 2009.
15. Liggesmeyer, P. & Trapp, M. "Trends in Embedded Software Engineering" *IEEE Software*, 26(3): 19-25, 2009.
16. McFeeley, B. "IDEAL: A User's Guide for Software Process Improvement" *CMU/SEI-96-HB-001*, Software Engineering Institute, Carnegie Mellon University, 1996.
17. Miller, J. & Smith, M. R. "A TDD Approach to Introducing Students to Embedded Programming" *Proc. of the 12th Annual Conference on Innovation and Technology in Computer Science Education* (ITiCSE 2007), ACM Press, pp. 33-37, 2007.
18. MOOSE project. Available at: http://www.mooseproject.org/ [On line]. 2009.
19. Nass, R. "An insider's view of the 2008 Embedded Market Study" Available at: http://www.embedded.com/design/210200580. January, 2008.
20. Ommering, R. V., Linden, F. V. D., Kramer, J. & Magee, J. "The Koala Component Model for Consumer Electronics Software" *IEEE Computer*, 33(3): 78-85, 2000.
21. Petrasch, R. & Meimberg, O. *Model Driven Architecture – Eine praxisorientierte Einführung in die MDA*, Heidelberg, Germany: dpunkt, 2006.
22. Powel, B. *"Real-time UML Workshop for Embedded Systems"* Elsevier, Boston USA. 2007.
23. Powel, B. *"The Telelogic Harmony/ESW Process for Real-Time and Embedded Development"* Telelogic White Paper. Telelogic, 2008.
24. Seceleanu, C., Vulgarakis, A. & Petterson, P. "REMES: A Resource Model for Embedded Systems" *Proc. of the 14th IEEE International Conference on Engineering of Complex Computer Systems* (ICECCS 2009), IEEE Computer Society, pp. 84-94, 2009.
25. Sentilles, S., Petterson, A., Nyström, D., Nolte, T., Petterson, P. & Crnkovic, I. "Save-IDE – A Tool for Design, Analysis and Implementation of Component-Based Embedded Systems" *Proc. of the 31st International Conference on Software Engineering* (ICSE 2009), IEEE Computer Society, pp. 607-610, 2009.

26. Software Engineering Institute. *CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, V1.1)*. Continuous Representation. CMU/SEI-2002-TR-011, Software Engineering Institute, Carnegie Mellon University. 2002.
27. Software Engineering Institute. *CMMI for Development (CMMI-DEV V1.2)*. CMU/SEI-2006 TR-008, Software Engineering Institute, Carnegie Mellon University. 2006.
28. Solingen, R. V. *"Product Focused Software Process Improvement – SPI in the embedded software domain"* Eiddhoven University of Technology, The Netherlands. 2000.
29. Taramaa, J., Khurana, M., Kuvaja, P., Lehtonen, J., Oivo, M., & Seppänen, V. "Product-Based Software Process Improvement for Embedded Systems" *Proc. of the 24th Euromicro Conference*, IEEE Computer Society, pp. 905-912, 1998.
30. Trienekens, J., Kusters, R. & Solingen, R. V. "Product Focused Software Process Improvement: Concepts and Experiences from Industry" *Software Quality Journal*, 9(4): 269-281, 2001.