Towards a Process Maturity Model for Evolutionary Architecting of Embedded System Product Lines

Jakob Axelsson
School of Innovation, Design and Engineering
Mälardalen University
SE-721 23 Västerås, Sweden
+46 31 59 00 00
jakob.axelsson@mdh.se

ABSTRACT
Many companies developing embedded systems and software as part of a product line struggle with how to improve their architecting practices to deal with increasing complexity. As the amount of legacy systems from previous products increases, the architecting becomes more and more evolutionary. This paper develops a process maturity model for evolutionary architecting, that can be used by an organization to improve its practices. The model is based on the Capability Maturity Model Integration (CMMI) which is instantiated to suit the architecting needs. Through this instantiation and simplification, it becomes feasible also for a small architecting team to systematically improve its maturity without dealing with the full CMMI. It is shown how the resulting maturity model addresses a number of issues previously collected from industrial case studies. The method is evaluated by performing maturity evaluations at several companies.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management – *software process models*.

General Terms
Management, Measurement, Standardization.

Keywords
Architecture, embedded systems, evolution, maturity model.

1. INTRODUCTION
In many companies developing technical products, such as the automotive industry, process automation, or telecommunication, embedded systems and software play an increasingly important role. The embedded systems have developed into a large number of computers with distribution networks and millions of lines of software. This increasing complexity leads to soaring developing costs, and many companies strive to curb this trend by reusing software and hardware between products. Often, a product line approach is applied, where the same platform is used as a basis, with modifications to fit individual products and customers.

With a multiplicity of products and variants, the architecture is becoming very important and is a source of increasing interest for companies developing embedded systems. The decisions made by architects in the early phases influence many decisions made later on, and the architecting decisions are difficult to change further down the process [10]. With a poor architecture, downstream development activities will thus become much more expensive and time consuming.

We have previously done in-depth studies of the current architecting practices at a few automotive companies [13], [14]. The issues we found were later validated also in other industrial areas where embedded software and systems play an essential role. Among the issues, we saw a lack of processes for architecture development, and the organizations had an unclear responsibility for architectural issues. Also, there was a lack of long-term strategy to ensure that legacy does not negatively impact future decisions, and a lack of methods to evaluate the business value when choosing the architecture. In short, the organizations rely on the performance and knowledge of individuals instead of on processes and methods.

These findings are typical signs of immature organizations that rely on fire-fighting by individuals rather than fire-prevention through a well-defined and repeatable process. The basis for systematic process improvement is weak. These companies often state that they never again expect to start from fresh in their architecting, since it will be too expensive and complex. Instead, they will continue to refine their existing products. Some of the companies have tried to do major revisions of their architecture, but have failed spectacularly and been forced to revert to evolution of their existing solution.

1.1 Purpose and Contribution
Based on this information collected from industry, we find it plausible to assume that a mature organization would work with architecting of embedded systems and software mainly through stepwise refinement rather than large leaps. We call this an *evolutionary* architecting approach, in contrast with the *revolutionary* approach focusing on large but rare changes. Since the small steps will be carried out often and have short duration, the process for doing the changes can be analyzed quantitatively and the data can be used for continuous improvement. In a revolutionary approach, there will never be enough relevant data for systematic evaluation of how well the process works.

The purpose of this paper is to device a way for a company to improve the maturity of its architecting practices. We hypothesize that a maturity model similar to the Capability Maturity Model Integration (CMMI) could be a useful basis for process improvement in the architecting area. We choose CMMI since it

ECSA 2010, August 23-26, 2010, Copenhagen, Denmark.

is, to our knowledge, the best established model of this kind. Maturity levels help characterize a process and set out a strategy for its improvement. Using such a model, the organization can stepwise change its practices towards a better defined process.

So why not use CMMI as it is? The main reason is the complexity of CMMI as such. The architecting teams are usually small (about 10 persons in a product development organization counting thousands of engineers is typical). This has a benefit in that the teams can use relatively light-weight processes and tools and move rapidly. The drawback in this context is that a full CMMI implementation is beyond the resources of such a team. Also, the current CMMI v. 1.2 is not very strong on architecture (although the next version plans to improve this [6]).

This paper contributes by providing an adaption and simplification of CMMI specifically for evolutionary architecting which can be used by industry as a tool for process improvement and for status assessment, without investing in penetrating and tailoring the full CMMI and performing detailed appraisals.

As a secondary contribution, we also see the model as a suitable tool for academic researchers in the area of system architecture. By performing a small appraisal using our model at a company, an initial and objective summary of its current status can be rapidly collected and used when analyzing its practices.

1.2 Related Work

The relation between CMMI and architecting is discussed in [6] which describes how an IT company made its architecting process CMMI Level 3 compliant due to a company requirement. A set of requirements on the architecting process is derived from CMMI, and it is also investigated how two generic processes from literature meet these requirements. However, the actual architecting process of the company is not described, and hence it is hard to determine whether it is evolutionary or revolutionary, and if it deals with embedded system product lines. Also, only a subset of CMMI process areas is included in the analysis.

When it comes to evolutionary architecting, one of the papers in the area is a case study describing how change requests to the architecture is handled in an automotive company [1]. It provides valuable information on the nature of the evolutionary process and its relation to revolutionary architecting.

Two separate papers [2], [11] describe the application of CMM(I) in small organizations. They conclude that these organizations often lack both resources and funds to invest in CMMI appraisals. Also, they note that since CMMI as such is written to suit a large organization, the small ones have to tailor CMMI to suit their needs, which makes it even more resource consuming. These findings support our initial assumption that CMMI is too heavy to use for a small architecting team and that a specialization of the maturity model could reduce the entry threshold.

In [9], a maturity model for requirements engineering is described which is loosely based on CMM. Although it addresses another area than architecture, this paper has inspired us in developing a less strict model than CMM(I), focusing more on internal process improvement than on external appraisals.

Instantiating CMM(I) generically to a certain type of activity is somewhat different from applying it at a specific company. A similar approach is reported in [5] which describes how a CMM appraisal is made of the software development process RUP.

In [8] an Architecture Alignment Model is presented where one component is architecture maturity. However, the focus is on IT systems rather than embedded systems, and the paper does not provide a detailed way of establishing an organization's maturity. The organizational integration of management and IT departments is emphasized, but many other aspects of architecting are missing.

From the IT systems area comes also the Enterprise Architecture Maturity Model Framework (EAMMF) [4], [12]. As CMMI, it contains five levels ("stages") with similar definitions. At each level, a number of practices ("core elements") are defined, and a number of goals ("critical success attributes") exist across the levels. Strangely, EAMMF does not make any reference to CMM(I), despite these similarities. EAMMF is not as formal as CMMI and probably the threshold for applying it is lower. However, the specific practices are very clearly directed towards the problem of integrating an organization with its IT tools. This makes it hard to use EAMMF as a basis for our problem, which is to integrate embedded systems into a product.

1.3 Overview of Paper

The remainder of the paper is structured as follows. In the next section, we describe the evolutionary architecting process for embedded systems and software. Certain characteristics of this activity are important to understand, because they determine how CMMI should be instantiated. Then, in Section 3, we summarize the theoretical framework, which is the CMMI specification. In Section 4, we develop the maturity model for architecting. The following section presents an initial evaluation by performing informal appraisals at a few companies, and some findings in the study are discussed further. In the final section, the conclusions are summarized together with ideas for future work.

2. Evolutionary Architecting

To better explain the activities involved in evolutionary architecting, we put it in the context of a V-model for the overall embedded system and software development (Figure 1). Often, development starts with the design of functions (expressed from an external or customer perspective). Based on this, systems are designed that together implement (i.e. express from a technical perspective) the functionality, and these systems are refined into components. Later, verification and validation is performed on the component, system, and function levels. Since the system-level development activities depend on input from the architects, it is important that the architecting process is predictable in terms of delivery time and quality, since otherwise the overall product development schedule will be delayed.

In this process, the architects get input primarily from the function developers in terms of the requirements on those functions. These requirements are complemented by needs from other stakeholders. The architects then try to design a high-level technical solution, which focuses on the distribution of functionality onto the different systems, and on the interfaces. When doing so, they also take into account architectural quality attributes, which are properties of the architecture itself which they strive to maintain. The architectural solutions are modeled, and from the model pre-requisites are derived and passed to the system developers.

When a totally new system is developed, the architects would typically try to collect the functional requirements of all functions at the same time, and take the total mass of functionality as input when designing the architectural solution. However, when working with product lines and platforms, most of the

functionality will already exist in the platform, and the architects can focus on dealing with the few change requests (CR) that capture added or changed functionality in the new products to be derived from the platform. We call this the *evolutionary architecting process* (EAP).

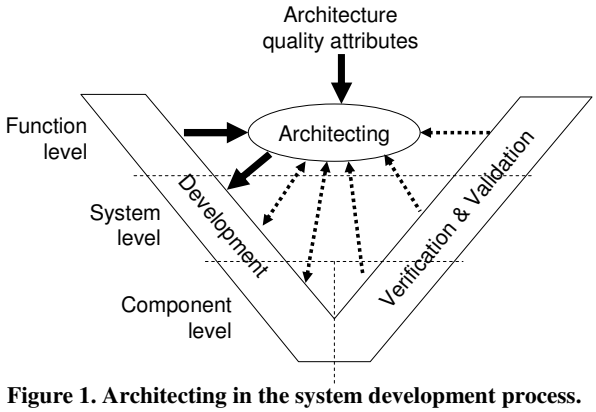

Figure 1. Architecting in the system development process.

In the EAP, the existing architecture for the platform, and the requirements that led to that architecture, are important input in the architecting process. The organization must find a way to deal with the architecture description of the platform as an asset that exists between instantiations of the process. It is also worth noting that several instances of the EAP can be active at the same time, dealing with different CRs, and therefore some co-ordination is necessary. Each CR is limited in scope, so the planning for each request does not need to be very detailed.

3. Overview of CMMI

The theoretical framework used in this research is the CMMI for Development, version 1.2 [3] which is based on "best practices" in software and systems engineering. In this section, a brief summary of the relevant parts is provided.

3.1 Maturity Levels

CMMI defines a sequence of *maturity levels*, where each level provides a set of process areas that characterize different organizational behaviors. This approach offers a systematic and structured way to improve the processes one stage at a time, from an ill-defined state to a state that uses quantitative information to determine and manage improvements. The levels are:

1. *Initial*. Processes are usually ad hoc and chaotic. Success depends on the people in the organization.
2. *Managed*. Work is planned and executed in accordance with policy (i.e., guiding principles established by senior management) by skilled people with adequate resources. Relevant stakeholders are involved and progress is monitored and controlled. The work is evaluated for adherence to the process descriptions.
3. *Defined*. Processes are well characterized and understood, and are described in standards, procedures, tools, and methods. Standard processes are established and improved over time, and are adapted by projects through tailoring guidelines. Processes are described more rigorously than at level 2.
4. *Quantitatively managed*. The organization establishes quantitative objectives for quality and process performance and uses them as criteria in managing processes. Special causes (i.e., unique and disruptive events) of process variation are identified and their sources are corrected to prevent future occurrences. Process performance is more predictable than at level 3.
5. *Optimizing*. The organization continually improves its processes based on a quantitative understanding of the common causes of variation. Process performance is quantitatively predictable, whereas at level 4 the predictability is qualitative.

3.2 Process Areas, Goals, and Practices

CMMI defines 22 *process areas*. At Level 1, no process areas are defined, and then more process areas are added at each level.

Each process area relates to a number of goals that are required to be met. There are both *specific goals* that relate to only a particular process area, but also two *generic goals* that relate to many process areas. The generic goals relate primarily to how the process improvements can be institutionalized. The generic goals associated with a maturity level should be applied to all process areas that are relevant at that level, even if those process areas were first introduced at another level.

For each goal, a number of practices are defined, that are expected (but not required) to be implemented. There are both *specific practices* that relate to a certain specific goal, and *generic practices* that relate to a certain generic goal.

4. Evolutionary Architecting Maturity Model

In this section, we present the Evolutionary Architecting Maturity Model (EAMM). When developing EAMM, we have axiomatically assumed that everything in CMMI is correct and relevant, unless we can find good reasons for changing it. The reasons for changing are primarily based on the characteristics of the evolutionary architecting process and organization described in Section 2. Also, the terminology has been updated to suit the architecting activities. In a few cases, additions to or reinterpretations of CMMI have been made. Since architecting is an internal activity, formal appraisals are not focused. Instead, self-assessment is a more relevant tool for the architects.

In the remainder of this section, we will present each maturity level of EAMM, and discuss in more detail the contents of the process areas. Space does not permit a detailed description of how CMMI goals and practices are incorporated into EAMM, but instead a short summary of each level and process area is given.

4.1 Level 0: Incomplete

In the EAMM, we have included a Level 0. A company at this level does not work with product lines at all, but each product has its own architecture and the ambition for reuse is low. There is no organizational responsibility for architecture across the products, and no defined process.

4.2 Level 1: Initial

A company who fulfills the requirements that it is working evolutionary based on product lines and has an organization responsible for architecting the products is at least at EAMM Level 1. At this initial level, the processes are usually ad hoc and chaotic, and success is highly dependent on the skills of the

people. Many of the companies mentioned in Section 1.2 show all the signs of being not much higher than this level.

4.3 Level 2: Managed

At EAMM Level 2, an organizational policy is established where the roles and responsibilities of the architects are formalized with respect to other parts of the development organization.

There is a need at this level to define what quality attributes should serve as guiding principles for the architects' work. These should not be connected to any specific function, and should relate primarily to the product line architecture rather than to the architecture of individual products.

A key factor in evolutionary architecting is to maintain the architecture descriptions that are used and updated when architecting each CR. These cross-project assets must be defined clearly to allow efficient management, and the CMMI does not provide clear guidance on how to manage data between projects.

EAMM defines six process areas at this level. In CMMI, a seventh is included, namely Supplier Agreement Management (SAM). It is removed from EAMM since architects do not usually deal with suppliers, although the suppliers (or the purchasing organization) should be included among the stakeholders.

Requirements Management (REQM). The organization performs systematic requirements management to ensure that the requirements are linked to the elements of the architecture description. This is to prevent that a later CR leads to an architecture update which is in conflict with old, but still valid, requirements.

Configuration Management (CM). The organization performs systematic configuration management of the architectural description to ensure its integrity. This is to ensure consistency between different versions of the common architecture used in different products, but also to avoid that architects working in parallel on different CRs make conflicting decisions. Also, changes made late in the development process must be fed back into the architecture description to ensure that future CRs are based on a correct view of the current design.

Measurement and Analysis (MA). The organization has a measurement capability that is used to support management information needs. It has defined what metrics should be used and how these should be measured. Usually, two kinds of metrics are used: technical metrics that relate to the quality attributes that will be used for technical decision making, and process related metrics that will be used for process improvement activities.

Project Planning (PP). When a new CR arrives to the architecture organization, a brief plan is expected to be created, identifying what areas need to be investigated, the expected duration and resource needs of the investigation, who is responsible for it, and which stakeholders should be included.

Project Monitoring and Control (PMC). The project monitoring and control area follows the processing of all on-going CRs. Some organizations may implement this through regular progress reports at an Architecture Change Control Board.

Process & Product Quality Assurance (PPQA). To ensure that the architecting process has the expected performance and the results meet the quality standards, objective evaluations are made. At regular intervals, it is monitored that the formal process description is followed, and that the resulting architectural prerequisites delivered to the system-level are evaluated with respect to quality and schedule.

4.4 Level 3: Defined

At EAMM Level 3, processes are institutionalized in the organization, and they are improved over time and adapted for each CR through tailoring guidelines. EAMM defines 11 process areas at this level, which are the same as in CMMI.

Requirements Development (RD). In Level 2, the procedures and tools for managing and storing requirements are defined. In the requirements development area at Level 3, the actual development of requirements is addressed, including collection, refinement, and analysis. For architecting, each CR is analyzed to identify the relevant stakeholders and elicit their needs, resulting in a set of architecturally significant requirements expressed from the customer's perspective. The requirements are analyzed to ensure that they are necessary and complete, and in case of conflicts between new or existing requirements, the necessary trade-offs are made. The requirements are validated to ensure that they really correspond to the stakeholders' intentions. It is important to state that it is not the role of the architects to develop all requirements, but only to gather those that are relevant to the architectural decisions.

Technical Solution (TS). The Technical Solutions process area is where the architectural decisions are made and the architectural descriptions are written. It is expected that solution alternatives are produced and evaluated based on a set of criteria. The architectural description of the parts that change is written, including the interfaces.

Product Integration (PI). In architecting, product integration (PI) takes place at the level of architectural descriptions. In the TS process area, a description is produced of what updates are needed in the architecture as a result of a CR. In the PI process area, it is ensured that this update is consistent with the already existing architectural description where it is expected to fit in. This includes ensuring that the interfaces are compatible, and to integrate the updated architectural description.

Verification (VER). Verification is to ensure that the technical solution meets the stated requirements, i.e. in this case that the architectural description meets the architectural requirements. In many cases, peer reviewing (or other structured reviews such as FMEA) among architects and system developers is used.

Validation (VAL). Validation is done to ensure that the technical solution meets the customer needs. As for verification, this is usually done through reviews. However, whereas the verification is done by having architects as peers, validation reviews should be done by functional developers and other stakeholders to ensure that their needs are correctly understood. In addition, the effects on architectural quality attributes need to be validated.

Decision Analysis and Resolution (DAR). The purpose of Decision Analysis and Resolution (DAR) is to analyze possible decisions using a formal evaluation process that evaluates identified alternatives against established criteria. In EAMM, the primary use of this is when evaluating solution alternatives.

Integrated Project Management (IPM). The organization ensures that each CR follows a standard processes. It also ensures that relevant stakeholders are involved in the processing of a CR.

Risk Management (RSKM). The organization performs risk management as part of dealing with each CR. This includes identification, analysis, and mitigation of risks.

Organizational Process Definition (OPD). As part of moving to EAMM Level 3, the organization develops a standardized architecting process to deal with CRs. The interfaces to other roles and organizations (primarily function developers and system developers) are clearly defined.

Organizational Process Focus (OPF). To ensure continuous improvement of the processes, the organization needs to identify improvement opportunities. This is done by a periodical assessment of the current processes.

Organizational Training (OT). One group to train is the architects, but since these are relatively few, it is likely that tuition and coaching by a more experienced architect is more suitable than formal courses. Another group is the different stakeholders.

4.5 Level 4: Quantitatively Managed

At EAMM level 4, the organization uses quantitative analyses to establish a stable architecting process with predictable behavior.

Organizational Process Performance (OPP). The organization has defined what metrics it should use to measure the process performance of its architecting processes. Typically, this will include how *efficiently* (i.e. how much resources and time is consumed) and how *effectively* (i.e. what the quality of the result is) it can deal with architecture CRs. The organization has also established objectives for quality and process performance, and has made measurements to establish the current status. Finally, it has developed prediction models that it uses to estimate key metrics for a CR based on its characteristics.

Quantitative Project Management (QPM). Based on the metrics defined in OPP, QPM measures the progress of individual CRs to collect statistical data about the various sub-processes. The organization puts up objectives and takes corrective actions if these are not satisfied. Statistical methods are applied on the collected data to identify causes of variation, with focus on special causes (rare events) that need to be removed in order to reach a stable process performance.

4.6 Level 5: Optimizing

At EAMM level 5, the organization continually improves its processes and architectural assets based on a quantitative understanding of the common causes of variation. It also strategically manages the architecture by identifying future bottlenecks and planning for refactoring at suitable times.

Causal Analysis and Resolution (CAR). Based on the data collection put in place at level 4, the organization can start to systematically detect defects and deviations from expected performance in its work, and analyses the root causes.

Organizational Innovation and Deployment (OID). With defined processes and measuring capability in place, the organization can start to work systematically with a strategic architecting processes to remove emerging bottlenecks before they hinder the execution of CRs. At lower maturity levels, the organization's behavior is reactive, and the aim is to resolve each CR as well as possible. However, such organizations often run into cul-de-sacs where the architecture's resources are suddenly exhausted and a major revision is needed. A mature organization instead pro-actively avoids such situations by analyzing the long-term consequences of each decision.

To achieve this, the organization needs to identify which the limiting factors are in the architecture. It must also monitor the rate of change over time in these factors based on CRs, in order to predict the most appropriate time for an architecture refactoring. CRs are prepared to initiate the refactoring, and those CRs can then be processed using the ordinary EAP.

5. Evaluation

In this section, we present an initial evaluation of the EAMM through informal appraisals at a few companies. The results and experiences gained from this and from developing EAMM are also discussed.

To be able to perform appraisals of how mature a company's architecting practices are according to EAMM, we have derived a set of 53 appraisal questions which is presented in an appendix. These were used for an initial validation through interviews with architects at three companies in the automotive domain. The results are summarized in Table 1.

Table 1. Summary of EAMM appraisals.

Level	No. of questions	Percentage of maximum score per level		
		A	B	C
1	3	100%	92%	92%
2	17	65%	37%	15%
3	22	66%	33%	16%
4	5	20%	0%	0%
5	6	67%	13%	17%
Total	**53**	**3.17**	**1.57**	**1.39**

In previous studies, we have done interviews and questionnaires with people at all three companies concerning what they consider to be issues or problems in their current architecting practices, as described in Section 1 above. In those studies, Company B and C have reported similar levels of identification with the issues, whereas Company A has reported better results. This is in-line with what the table shows from the EAMM appraisal.

The table also shows the expected trend that companies implement practices up to a certain level, and then the number of activities sharply decline. An interesting observation is that Company A insisted that they already do several of the practices at Level 5 and much fewer at Level 4. When digging deeper into this, we found that at this company there is a strong culture to deal seriously with all deviations and discover the root cause for them. However, it does not involve a statistical analysis of measurement data, but is done qualitatively.

5.1 Discussion

In this work we have defined EAMM by instantiating CMMI for evolutionary architecting. When doing so, we have axiomatically assumed that implementing CMMI provides benefits, being based on "best practices" as it is. By tailoring CMMI for architecting we hope to improve the cost-benefit-equation for an architecting team by removing some of the cost but still provide similar benefits.

EAMM contains many activities that are rarely performed by architecting organizations today. It would not be surprising if

more resources are needed as the organization proceeds up the maturity scale, although we do not know this for sure. The benefit of reaching high maturity levels would not be to have a leaner process, but a more predictable one with less variability. This is also important, because it would improve the overall development planning for function and system developers.

An expected benefit of EAMM is also improved quality of the resulting architecture. However, the positive effect of this is difficult to measure in practice, because the result of poor architecture quality is additional work downstream in the development process. The cost of rework ("the hidden factory") is thus located far from the architecting process and the cause-effect relationship is hard to establish.

Being a small organization is both a curse and a blessing when it comes to process improvement. On the one hand it can be hard to find the resources to implement the process improvement program (which is why we devised EAMM to reduce the cost). On the other hand, it is much easier to get everyone in the same boat in a small organization, and changes can be made much more rapidly. We believe this would increase the chances that an architecting organization can move beyond Level 3 (where many companies seem to get stuck when applying CMMI).

6. Conclusions

In this paper, we have presented a maturity model for architecting of embedded system product lines. The model is based on CMMI, which is simplified and adapted to architecting. By doing so, we hope to reduce the resource needs of implementing a process improvement program for a small organization, while still retaining the benefits of moving to higher maturity levels. Also, it makes it possible for an organization to improve its architecting practices without having to deploy CMMI companywide.

An initial evaluation shows that the model appears relevant for companies developing embedded systems, and that it addresses issues found in previous research. We conclude that CMMI can be a basis also for improving the architecting processes.

However, we would like to stress that EAMM is not CMMI. We have strived to create a light-weight model with no ambition for formal appraisal, but something that could serve as a tool for self-improvement by the architecting organizations.

6.1 Future Work

The results presented in this paper open several roads for future research. In the direction of process improvement, it would be interesting to do an action research project where the effects of process improvement following the maturity model are studied and evaluated. It would require a project over several years, with a rigorous monitoring of many parameters.

We also plan to perform appraisals of a larger number of companies, both in order to better understand the overall maturity of the embedded systems industry, and to see if there are any particular companies that stand out as role models for others.

Finally, we would also like to see if the model can be generalized to other areas, such as IT systems or non-product line systems.

7. REFERENCES

[1] Axelsson, J. Evolutionary architecting of embedded automotive product lines: An industrial case study. In Proc. Joint 8th Working IEEE/IFIP Conference on Software Architecture & 3rd European Conference on Software Architecture, Cambridge, UK, Sept. 14-17, 2009.

[2] Brodman, J. G. and Johnson, D. L. What small businesses and small organizations say about the CMM. In Proc. of the 16th International Conference on Software Engineering, pp. 331-340, Sorrento, Italy, May 16-21, 1994.

[3] CMMI for Development, Version 1.2. Software Engineering Institute, CMU/SEI-2006-TR-008, 2006.

[4] Kaisler, S. H., Armour, F., Valivullah, M. Enterprise architecting: Critical problems. In Proceedings of the 38th Hawaii International Conference on Systems Sciences, 2005.

[5] Manzoni, L. V. and Price, R. T. Identifying extensions required by RUP (Rational Unified Process) to comply with CMM (Capability Maturity Model) levels 2 and 3. IEEE Trans. on Software Engineering, Vol. 29, No. 2, pp. 181-192, February 2003.

[6] Philips, M. and Shrum, S. Process Improvement for All: What to Expect from CMMI Version 1.3. Crosstalk–The Journal of Defense Software Engineering, Jan. 2010.

[7] Poort, E. R.,Postema, H., Key, A., and de With, P. H. N. The Influence of CMMI on Establishing an Architecting Process. In Proc. of the 3rd Intl. Conf. on Quality of Software Architectures, pp. 215-230. Medford, USA, July 2007.

[8] van der Raadt, B., Soetendal, J., Perdeck, M., and van Vliet, Hans. Polyphony in architecture. In Proceedings of the 26th International Conference on Software Engineering, 2004.

[9] Sawyer, P., Sommerville, I., and Viller, S. Capturing the benefits of requirements engineering. IEEE Software, March/April 1999.

[10] Smith, P. G. and Reinertsen, D. G. Developing Products in Half the Time: New Rules, New Tools. John Wiley, 1998.

[11] Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P., and Murphy, R. An exploratory study of why organizations do not adopt CMMI. Journal of Systems and Software, Vol 80, pp. 883-895, 2007.

[12] United States General Accounting Office. Information technology: A framework for assessing and improving enterprise architecture management (version 1.1). GAO-03-584G, April 2003.

[13] Wallin, P. and Axelsson, J. A case study of issues related to automotive E/E system architecture development. In Proc. 15th IEEE Intl. Conf. on Engineering of Computer Based Systems, pp. 87-95, Belfast, Northern Ireland, March 2008.

[14] Wallin, P., Johnsson, S., and Axelsson, J. Issues Related to Development of E/E Product Line Architectures in Heavy Vehicles. In Proceedings 42nd Hawaii International Conference on System Sciences, Hawaii, January 5-8, 2009.

8. APPENDIX: APPRAISAL QUESTIONS

In this appendix, we present the appraisal questions used in EAMM. We also show in brackets which process area, special goal, and level of CMMI each question is derived from. The questions are grouped in a thematic order (general, requirements, architecture, quality assurance, project management, process), that we have found becomes natural during interviews. The questions are answered using a Likert scale with five levels: Never (1), Rarely (2), Sometimes (3), Usually (4), and Always (5). A

summary indicating the approximate maturity level of the organization can be calculated by using the following formula (where q_i is the number of questions at level i and s_{ij} is the answer between 1 and 5 to question j at level i):

$$\sum_{i=1}^{5}\sum_{j=1}^{q_i}\frac{s_{ij}-1}{4q_i}$$

To what extent does your organization… [PA, SG, Level]:

1. work with product lines, where different individual products share components or systems? [-, -, 1]
2. have a team responsible for developing and maintaining the architecture for all products? [-, -, 1]
3. make continuous additions and changes to an existing architecture rather than developing a new architecture from scratch for each new product? [-, -, 1]

---

4. collect requirements when handling an architecture change request? [REQM, 1, 2]
5. systematically collect the needs that are significant for the architecture for all stakeholders? [RD, 1, 3]
6. translate the stakeholder needs into a formal set of architectural requirements? [RD, 2, 3]
7. analyze architectural requirements to ensure that they are necessary and complete? [RD, 3, 3]
8. make trade-offs between conflicting architectural requirements, including old and new ones? [RD, 3, 3]
9. have routines for managing changes to architectural requirements? [REQM, 1, 2]
10. ensure traceability between requirements and architectural descriptions? [REQM, 1, 2]

---

11. produce architectural descriptions according to a well-defined format? [TS, 2, 3]
12. clearly define the interfaces between different parts of the architecture? [PI, 2, 3]
13. develop alternative architectural solutions and evaluate them based on well-defined criteria? [TS, 1, 3]
14. have routines for releasing different versions of the architecture descriptions? [CM, 1, 2]
15. have routines for handling changes to the architecture descriptions? [CM, 2, 2]
16. ensure that the the released architecture description really corresponds to the final as-built product? [CM, 3, 2]
17. make decisions based on an evaluation of alternatives using established criteria while architecting? [DAR, 1, 3]
18. have well-defined quality attributes that the architecture should meet? [MA, 1, 2]
19. regularly collect measurement data on the quality attributes of the architecture? [MA, 2, 2]
20. identify and analyze improvement opportunities in the architecture? [OID, 1, 5]
21. systematically trigger redesign activities to address architecture improvement possibilities and avoid future bottlenecks? [OID, 2, 5]

---

22. regularly review the quality of the architectural descriptions produced by the architecting process? [PPQA, 1, 2]
23. have well-defined procedures and criteria for how to verify that the architecture fulfils its requirements? [VER, 1, 3]
24. perform reviews to ensure that the architecture fulfils its requirements? [VER, 2, 3]
25. have well-defined procedures and criteria for how to validate that the architecture fulfils stakeholder's needs? [VAL, 1, 3]
26. perform reviews with stakeholders to ensure that the architecture fulfils the stakeholder's needs? [VAL, 2, 3]

---

27. regularly follow up the progress of the processing of each change request? [PMC, 1, 2]
28. quantitatively measure the progress and result of each change request? [QPM, 1, 4]
29. take corrective actions when the processing of a change request deviates from its plan? [PMC, 2, 2]
30. estimate the amount of work associated with a new change request? [PP, 1, 2]
31. perform planning of the work associated with a change request? [PP, 2, 2]
32. explicitly define the specific process to use for a certain change request by tailoring a standard process? [IPM, 1, 3]
33. ensure that all stakeholders are involved when dealing with a change request? [IPM, 2, 3]
34. have a defined risk management strategy for the architecture development? [RSKM, 1, 3]
35. identify and analyze risks during architecture development? [RSKM, 2, 3]
36. define and implement risk mitigation plans during architecture development? [RSKM, 3, 3]

---

37. have a standardized architecting process description? [OPD, 1, 3]
38. regularly evaluate how well the defined architecting process is followed? [PPQA, 1, 2]
39. have routines for tracking and resolving deviations from the defined architecting process? [PPQA, 2, 2]
40. systematically analyze the root cause of defects and deviations in the architecting process? [CAR, 1, 5]
41. take actions to remove the root cause to avoid recurrence of defects and deviations in the architecting process? [CAR, 2, 5]
42. periodically assess its architecting process improvement needs? [OPF, 1, 3]
43. plan and implement process improvements based on the identified needs? [OPF, 2, 3]
44. have well-defined performance metrics that the architecting process should meet? [MA, 1, 2]
45. regularly collect measurement data on the performance of the architecting process? [MA, 2, 2]
46. have defined objectives for the metrics used to measure process performance and quality for architecting? [OPP, 1, 4]
47. have models to estimate future values of the metrics used to measure process performance and quality for architecting? [OPP, 1, 4]
48. regularly summarize and present the current performance of the architecting process? [OPP, 1, 4]
49. perform statistical analysis on the measurement data from the architecting process to identify sources of variation? [QPM, 2, 4]
50. identify and analyze changes in the architecting process that can lead to better values for the processes' quality and performance metrics? [OID, 1, 5]
51. systematically update the architecting processes based on statistical data? [OID, 2, 5]
52. provide training in the architecting process for architects? [OT, 1, 3]
53. provide training in the architecting process for stakeholders, including function and system developers? [OT, 2, 3]