The influence of CMMI on establishing an architecting process

The Influence of CMMI on Establishing an Architecting Process

Eltjo R. Poort[1], Herman Postema[1], Andrew Key[2], and Peter H.N. de With[3]

[1] LogicaCMG, P.O. Box 159, 1180 AD Amstelveen, The Netherlands
{eltjo.poort,herman.postema}@logicacmg.com
[2] LogicaCMG, Stephenson House, 75 Hampstead Road, London NW1 2PL, United Kingdom
andrew.key@logicacmg.com
[3] Eindhoven Univ. of Technol., P.O. Box 513, 5600 MB Eindhoven, The Netherlands
P.H.N.de.With@tue.nl

Abstract. A large IT company is creating a generic architecting process. Since the company has set an objective to achieve Maturity Level 3 of the Capability Maturity Model Integration (CMMI), the process needs to comply with the relevant requirements set by the CMMI. This paper presents the elicitation of such requirements, and the resulting set of requirements. It analyzes their potential impact on generic architecting processes found in literature. It turns out that many key architectural concepts are at best loosely defined in the CMMI. CMMI is strong in support of the development-related architecting activities, but gives only indirect support for other architecting activities, particularly in a product development context.

1 Introduction

The setting of this paper is a large IT company, in which it was established that an institutionalized architecting process would help control technical risks in projects and products. At about the same time, a company-wide objective had been set to achieve CMMI Maturity Level 3. This made it necessary to obtain insight into the requirements that architecting processes need to fulfill in order to comply with CMMI Maturity Level 3 [1]. This paper documents the process of establishing these requirements. Apart from this paper, we will elaborate on the establishment of this architecting process in a separate paper that is still under development.

As references we have chosen two generic processes found in literature: Architecture Based Development [1], because its scope is close to our purpose and because it represents one of the better known approaches to architecting in both industry and academia, and Hofmeister *et al.* [2], because their model represents the commonalities between five industrial approaches.

First, in Sect. 2 we will present the organizational context and scope of a generic architecting process. In Sect. 3, the CMMI Process Areas that are relevant to such an architecting process will be identified, and their requirements on architecting processes extracted. In Sect. 4 follows a discussion on the impact of the CMMI requirements on

[1] *CMMI Maturity Level 3* is abbreviated to *CMMI Level 3* in the rest of this paper.

S. Overhage et al. (Eds.): QoSA 2007, LNCS 4880, pp. 215–230, 2007.

generic architecting processes found in literature, and on the coverage of architecting processes by CMMI. We will finish up with some conclusions and further work to be done.

2 Architecting Process Context and Scope

2.1 Organizational Context

The analysis described in this paper was done by and for an IT Corporation of 40,000 people across 41 countries. The company has a diverse business portfolio, consisting of services centered on business consulting, systems development and integration and IT and business process outsourcing.

One of the company's Technical Directorate's activities is controlling technical risks in the various IT projects and products. It was felt that technical risk control could be enhanced by developing and institutionalizing a process that would provide guidance for making technical decisions: in short, an architecting process. Two of the authors of this paper work within the company's technical directorate at group and subsidiary levels, and have terms of reference that include management of technical risks.

The decision to institute an architecting process coincided with the setting of a maturity objective by the company's executive management. Encouraged by benefits experienced through local CMMI driven process improvement, management set an objective to achieve CMMI Maturity Level 3 for relevant organizational units throughout the whole company. This meant that the architecting process to be developed would be subject to the requirements set by the CMMI.

2.2 Scoping an Architecting Process

The terms Architecture and Architecting are used in a great variety of meanings in the IT world. Rather than risking a non-converging discussion on the meaning of the terms, it was decided to scope the architecting process in terms of a set of business goals and usage scenarios. The details of this work and the resulting process description will be the subject of a separate paper. For the purposes of this paper, a high-level summary is provided:

Business Goals. The business goals for the architecting process were established as *Consistency in Delivery*, *Risk Management*, *Customer Satisfaction* and *Knowledge Incorporation*.

Usage Scenarios. The process will be used for architecting activities in the following scenarios: *Responding to a Request for Proposal (RFP)*, *Software Development Project*, *System Integration Project* and *Product Development*.

The business goals and usage scenarios were analyzed to determine the scope of the architecting process. Apart from literature and the existing experience of the authors, additional input for the analysis came from other stakeholders, specifically the company's sales community, quality assurance community and technical community, obtained in a workshop.

The most significant elements in the outcome of this analysis are listed below.

- Analysis of the business goals and experience indicates that *architectural decisions* are critical to the success of projects in terms of cost and timing of delivery. The process should therefore give guidance on how to identify and make architectural decisions. This matches requirements from CMMI about decision analysis and resolution, and with recent publications about the status of architectural decisions [3, 4, 5].
- Many architectural decisions are made during the *sales* phase of projects; the architecting process has to facilitate that process.
- A certain level of *reviewing and control* has to be facilitated by the process. This is the convergence of the architecture assessment practices from literature [6, 7], and the responsibilities of the authors to control technical risks. Not only are reviewing and control necessary parts of the process, it also has to be facilitated by a certain level of standardization in *documentation* of architectures.
- The involvement of architects in the *implementation* phase of solutions is essential in order to assure that the selected solution will be adequately implemented *conforming* to the architecture. The architecting process has to facilitate this.
- To contribute to the business goal of knowledge incorporation, the process should support a structure for organizational *learning from experiences*. Learning points may be both process-related (like best practices) and product-related (like best architectural constructs).
- The objective is to implement a process that gives guidance on aspects of architecting that are *not specific* to particular types of applications, e.g. not just software development, but also system integration, ERP implementations, and embedded system development. This means its concept of "architecture" covers both *software* and *system architecture*. For such a generic process to be useable, it must be accompanied by a set of guidelines for *tailoring* the process to the specific needs and characteristics of the usage environment. This is also required by CMMI Generic Practice 3.1 "Establish a Defined Process".

The result of all this is an architecting process description under development that focuses on requirements analysis, architectural decision making, shaping, selection and evaluation of the best-fit solutions, documenting and implementing architectures and controls like architectural governance and reviewing.

At the time of writing this paper, it is being considered to extend the scope of this process to better support the product development scenario, by adding e.g. reusable asset harvesting and product roadmapping.

The scope of what is meant by an "architecting process" in this paper is documented as a list of requirements[2] in Table 1. In Sect. 4.1, we will identify a number of generic architecting processes in literature that are similar in scope.

The scope of the architecting process has been determined by the analysis of the business goals and usage scenarios, with limited consideration of CMMI. We will now focus on the influence of CMMI in more detail.

[2] A note on the tagging of requirements in this paper: the reader will notice the use of mnemonic, hierarchical tagging [8]. The use of dots indicates a hierarchical grouping, with an implicit traceability to higher level requirements.

Table 1. Scope of architecting process: high-level requirements

rq.arch A process giving guidance on architecting technical solutions.
rq.arch.decision Guidance on how to make architectural decisions.
rq.arch.sales Facilitating the sales process.
rq.arch.documentation Standardization of architectural documentation.
rq.arch.controls Guidance on architectural controls.
rq.arch.conform Assuring conformance with architecture during the implementation process.
rq.scalable Scalable over business unit sizes (20 - 2000) and project/programme sizes (80 K - 500 M), and over a broad range of size and complexity of solutions.
rq.generic Flexible / generic to work in diverse applications.
rq.generic.tailoring Be accompanied by a set of tailoring guidelines.
rq.accessible Simple, accessible to all.
rq.accessible.terminology Terminology familiar to company staff.
rq.cmmi CMMI Maturity Level 3 compliant.
rq.learning.product Bottle product experiences and make available to architects in controlled manner.
rq.learning.process Support a structure for organizational process learning.

3 Architecting and CMMI

The Capability Maturity Model Integration (CMMI) is a process-improvement model developed by the Software Engineering Institute (SEI) of the Carnegie Mellon University. It is scoped towards the development, acquisition and maintenance of systems or services.

The "staged representation" of the CMMI consists of five maturity levels. With increasing maturity level, the process capabilities increase, resulting in a higher probability that development or maintenance targets will be realized. Each maturity level consists of a number of Process Areas (PAs). Each PA consists of a small set of "goals" followed by a collection of practices that must be performed in order to realize the goals. A process complies to a certain maturity level if the goals and practices of all PAs of that level are satisfied. The PAs are customarily referred to by a set of fixed tags; all level 2 and 3 PAs and their tags are listed in Table 2.

Goals and practices of a PA are divided into specific ones and generic ones. Specific goals and practices directly refer to the PA itself, whereas generic goals and practices represent mechanisms to institutionalize the specific goals and practices. These practices are called generic because they apply to multiple PAs.

CMMI Maturity Level 3 requires that for all PAs belonging to Level 2 and Level 3 a "defined process" is established. A defined process is tailored from the organization's "standard process" according to a set of tailoring guidelines. In addition, a defined process has a maintained process description, which implies that all (generic and specific) practices are described. For more information, the reader is referred to [9] or [10].

This section starts with an exploration of what a *CMMI Compliant Architecting Process* actually means. This is followed by a discussion on the use of architectural

concepts in the CMMI. We then proceed to identify the Process Areas that have a significant contribution to architecting according to the scope set out in Sect. 2.2. We call this set the *Architecting Significant Process Areas (ASPAs)*.

3.1 CMMI-Compliant Architecting Process

The boundaries (scope) of the architecting process are determined in Sect. 2.2. Because of the structure of the CMMI, the practices related to this process may be distributed over a number of Process Areas.

The CMMI Level 3 coverage of the architecting process can be obtained by analyzing every Level 2 and Level 3 specific practice to determine whether or not the practice is inside the scope of the architecting process. The generic practices of Level 2 and Level 3 will always be in scope because they apply to all PAs. This analysis will be performed further on in this paper.

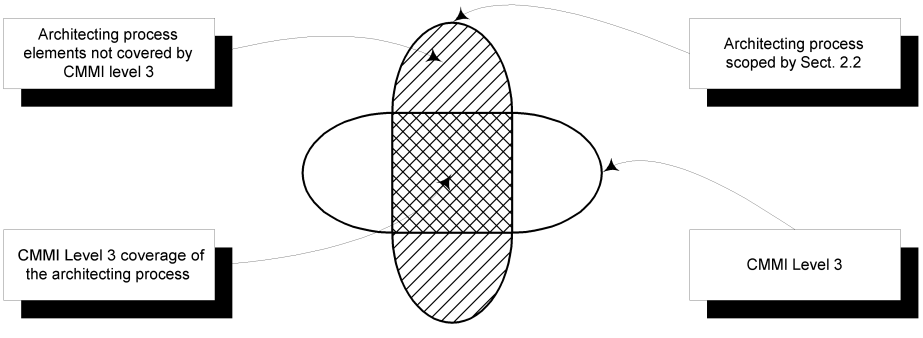

Fig. 1. CMMI coverage of the architecting process

Figure 1 illustrates the CMMI coverage of the architecting process. As can be derived from the figure, the architecting process may include elements that are not covered by CMMI Level 3. These may for example be elements that are beyond the scope of system development (like architectural roadmapping) or elements that are considered critical for a successful architecting process but cannot be found in the CMMI.

Summarizing the above information, it can be stated that a CMMI Level 3 compliant architecting process:

- has a maintained description of all specific and generic practices that are in scope of the architecting process (the square box in the figure)
- has a maintained description of guidelines to tailor the process to the specific needs and characteristics of the usage environment
- is consistently deployed inside the company in the context of the user scenarios referred to in Sect. 2.2.

The scope of this paper is the determination of the practices that should be part of the maintained description mentioned in the first two items. These practices will be presented as a list of requirements imposed on an architecting process description. In Sect. 3.3 we will present the elicitation of these requirements, but first we will have a more general look at the use of architecture concepts in the CMMI.

3.2 Architecture Concepts in the CMMI

The word "architecture" is used extensively in the CMMI. It appears in 7 out of 25 Process Area descriptions [9]. The CMMI is a collection of industry best practices and not a formal theoretical model. Effort was put in making the model consistent and unambiguous, but many parts are still subject to different interpretations.

Architecture itself is not defined in the CMMI glossary. The word is mostly used informally to denote a number of concepts, some of which are related to our architecting process, and others are not.

Architecting the company processes: the CMMI describes how to set up and maintain a company's processes in order to best achieve its business goals. The design and overview of these processes and their alignment with the company's IT resources require architecting skills. This type of architecting is relevant in the OPD and OPF processes. It is outside the scope of this paper as defined in Sect. 2.2.

Architecting the product: the bulk of the CMMI PAs describe how to improve systems and software engineering processes. Architecting is an essential part of those processes, especially RD, TS, PI, PP, REQM, DAR and RSKM.

Furthermore, several architecture-related terms are defined in the CMMI glossary:

Functional Architecture is defined as "The hierarchical arrangement of functions, their internal and external (external to the aggregation itself) functional interfaces and external physical interfaces, their respective functional and performance requirements, and their design constraints."

Process Architecture is defined as "the ordering, interfaces, interdependencies, and other relationships among the process elements in a standard process". This concept is on a different level than the "architecture" in the architecting process as described in this document.

Shared Vision is defined as "a common understanding of guiding principles including mission, objectives, expected behavior, values, and final outcomes, which are developed and used by a group, such as an organization, project, or team."

A significant finding is the fact that "product architecture", though used extensively, is not defined in the CMMI glossary.

These considerations show that the concepts and terms relevant to architecting are generically defined (e.g. Shared Vision) or not defined at all (e.g. Product Architecture) in the CMMI. Hence the terms provide little guidance in themselves. The word architecture is used in many different ways, making it inadequate as a basis to establish direction for an architecting process. We have therefore selected a different approachto

establishing the CMMI requirements on an architecting process, which will be the subject of the following section.

3.3 Process Areas Relevant to Architecting

Our approach to establish which requirements CMMI imposes on architecting processes is to first identify which PAs are relevant for the process, and then to extract requirements on the process from the practices in their descriptions. An analysis of the CMMI Level 3 PAs against the architecting process scoped in Sect. 2.2 results in a set of PAs that have a direct and significant contribution to the objectives of this process. As discussed before, these PAs are called *Architecting Significant Process Areas* (ASPAs).

The PAs of the CMMI are grouped into four categories:

Process Management. These PAs contain the activities related to defining, planning, implementing, monitoring, evaluating and improving all other processes. The architecting process is subject to these process management PAs in order to assure the required level of capability.

Project Management. These PAs cover the project management activities related to planning, monitoring and controlling the development or maintenance project. The architecting process is generally performed in the context of a project.

Engineering. These PAs cover the development and maintenance activities that are shared across engineering disciplines (e.g. systems engineering and software engineering). The architecting process falls mainly within these PAs.

Support. These PAs cover the activities that support all other PAs like establishing measurement programs, verification of compliance, and effective decision making. The architecting process is also subject to these PAs.

Table 2 identifies the categorized set of Level 3 PAs and indicates which PAs have been qualified as an ASPA. It should be noted that *all* PAs of the CMMI contribute to the objectives of the architecting process. Their contribution may be direct because the PA is actually part of the architecting process, or indirect because the PA is establishing the context and preconditions for a successful architecting process.

As stated before an ASPA has a direct contribution and this contribution should also be significant. This is the case for all Engineering PAs, one Project PA (Risk Management, RSKM) and one Support PA (Decision Analysis and Resolution, DAR). Both RSKM and DAR are actually part of the architecting process and contribute significantly to its objectives. The architecting relevance of the set of ASPAs is shortly explained below. Where relevant, underpinning references to the CMMI text have been added in [braces].

REQM *Requirements Management.* The role of architecting in Requirements Management focuses around the impact of requirements and their traceability to the architecture. [Specific Practice 1.1 *Obtain an Understanding of Requirements* describes the process of the acceptance of requirements according to objective criteria. "Does not break the architecture" is an important criterion to assess requirements, implied in the example criterion "Appropriate to implement". It is also implicit in the impact

Table 2. Categorized Process Areas and their Architecting Significance

Tag		Process	Project	Eng	Supp	ASPA
OPF	Organizational Process Focus	X				N
OPD	Organizational Process Definition	X				N
OT	Organizational Training	X				N
PP	Project Planning		X			N
PMC	Project Monitoring and Control		X			N
SAM	Supplier Agreement Management		X			N
IPM	Integrated Project Management		X			N
RSKM	Risk Management		X			Y
IT	Integrated Teaming		X			N
ISM	Integrated Supplier Management		X			N
REQM	Requirements Management			X		Y
RD	Requirements Development			X		Y
TS	Technical Solution			X		Y
PI	Product Integration			X		Y
VER	Verification			X		Y
VAL	Validation			X		Y
CM	Configuration Management				X	N
PPQA	Process and Product Quality Assurance				X	N
MA	Measurement and Analysis				X	N
DAR	Decision Analysis and Resolution				X	Y
OEI	Organizational Environment for Integration				X	N

analysis mentioned in SP1.3 *Manage Requirements Changes*. SP1.4 *Maintain Bidirectional Traceability of Requirements*: traceability to architectural components is implied, as is traceability to architectural decisions.]

RD *Requirements Development.* This process area is where a system's functional architecture is defined, and where the requirements are analyzed and developed. Architecting is important here both as a source of new requirements and as a means to structure requirements. ["Analyses occur recursively at successively more detailed layers of a product's architecture". Specific Goal 2 *Develop Product Requirements* identifies the selected product architecture as a source of derived requirements. SP2.1 *Establish Product and Product-Component Requirements* prescribes that "architecture requirements addressing critical product qualities and performance necessary for product architecture design" be developed, and that "requirements that result from design decisions" be derived. SP2.3 *Identify Interface Requirements* prescribes the definition of interfaces as an integral part of the architecture definition.]

TS *Technical Solution.* This process area covers the core of architecting: developing a solution that fulfills the requirements. [TS specific goals are SG1 *Select Product Component Solutions*, SG2 *Develop the Design* and SG3 *Implement the Product Design*. SP1.1 *Develop Detailed Alternative Solutions and Selection Criteria* prepares architectural decision making by identifying alternatives and selection criteria. SP1.2 *Evolve Operational Concepts and Scenarios* prescribes the use of scenarios to help assess alternative solutions for usability.

SP1.3 *Select Product Component Solutions* and SP2.4 *Perform Make, Buy or Reuse Analyses* are about making design decisions and documenting them, including rationale. SP2.1 *Design the Product or Product Component* establishes the product architecture. It describes architecture definition, driven by the architectural requirements developed in RD SP 2.1. It identifies elements of architectures, such as coordination mechanisms, structural elements, standards and design rules. It also mandates architecture evaluations to be conducted periodically throughout product design. SP2.2 *Establish a technical data package* and SP3.2 *Develop Product Support Documentation* are about documenting, giving guidance on where the architecture definition and the rationale for key decisions are documented. SP2.3 *Design Interfaces Using Criteria* supplies requirements to the interface design process.]

VER *Verification.* Verification is an essential part of the architecting process because its purpose is to ensure that the work products of this process meet the specified requirements. Typical work products of the architecting process are the architecture and design documents and the architecture and design itself. Means for verification may be peer reviews (for documents) and architectural assessments. Verification activities should be prepared, performed, the results analyzed and corrective actions identified.

VAL *Validation.* Validation is in fact a variant on verification but its objective is to demonstrate that a (work) product fulfills its intended use (i.e. that it meets user needs). Regarding the architecting process, the work products and means for validation are similar to verification.

DAR *Decision Analysis and Resolution.* Key to architecting is decision making [3, 4]. The DAR process area prescribes a formal evaluation process for decisions of this kind: evaluation criteria should be established, alternatives should be identified, evaluation methods selected, alternatives evaluated and a solution selected. There should also be guidelines establishing which decisions should be subject to this formal evaluation process.

RSKM *Risk Management*. Better risk management is one of the business goals of the architecting process. The inherent risk in a requirement is an important factor in determining whether or it is an architectural requirement. [A requirement that, when not fulfilled, heavily "impacts the ability of the project to meet its objectives" (SP1.1 Determine Risk Sources and Categories), has a good chance to be considered architectural. The RSKM process area prescribes how to deal with such risks: risk parameters should be defined (SP1.2), a risk management strategy should be established (SP1.3), the process should give guidance on how risks are identified and analyzed (SG2), and mitigated (SG3). Insofar as architectural requirements involve risks, they should be treated the same way.]

An analysis of the texts of these ASPAs yields the requirements imposed on the architecting process by the CMMI. These requirements are listed in Table 3. In agreement with the nature of the CMMI, this table is effectively a list of 67 best practices that support companies in creating and implementing an architecting process. The tags allow traceability to the PAs that the requirements originated from, and give the list a clear structure. The largest contributor is TS with 31 requirements, confirming our earlier observation that TS covers the core of architecting. The next largest contributor is RD with 16 requirements, indicating that an architecting process within our scope includes

Table 3. Requirements imposed on Architecting Process by CMMI

rq.cmmi.reqm.arch	Use architectural fit as criterion when assessing requirements and changes.
rq.cmmi.reqm.trace	Maintain traceability between requirements and architectural components and decisions.
rq.cmmi.rd.fun-arch	Develop a functional architecture.
rq.cmmi.rd.recursive	Recursive analysis of requirements.
rq.cmmi.rd.arch-req	Develop architectural requirements.
rq.cmmi.rd.alloc-comp	Allocation of requirements to product components.
rq.cmmi.rd.alloc-fun	Allocation of requirements to functions.
rq.cmmi.rd.elicit	Elicit needs from stakeholders by proactively identifying additional requirements.
rq.cmmi.rd.derive	Derive requirements that result from design decisions.
rq.cmmi.rd.if	Identify interface requirements.
rq.cmmi.rd.trace	Document relationships between requirements.
rq.cmmi.rd.analyze	Analyze requirements.
rq.cmmi.rd.scenario	Develop operational concepts and scenarios.
rq.cmmi.rd.balance	Use proven models, simulations, and prototyping to analyze the balance of stakeholder needs and constraints.
rq.cmmi.rd.risk	Perform a risk assessment on the requirements and functional architecture.
rq.cmmi.rd.lifecycle	Examine product life-cycle concepts for impacts of requirements on risks.
rq.cmmi.rd.assess	Assess the design as it matures in the context of the requirements validation environment.
rq.cmmi.rd.measure	Identify technical performance measures.
rq.cmmi.ts.alt	Develop detailed alternative solutions to address architectural requirements.
rq.cmmi.ts.alt.crit	Develop selection/evaluation criteria for alternative solutions.
rq.cmmi.ts.alt.crit.assess	Assess adequacy of selection criteria after use.
rq.cmmi.ts.alt.req-alloc	Obtain a complete requirements allocation for each alternative.
rq.cmmi.ts.alt.scenario	Develop timeline scenarios for product operation and user interaction for each alternative solution.
rq.cmmi.ts.alt.eval	Evaluate alternative solutions against criteria.
rq.cmmi.ts.alt.issues	Identify and resolve issues with the alternative solutions and requirements.
rq.cmmi.ts.alt.select	Select the best set of alternative solutions that satisfy the established selection criteria.
rq.cmmi.ts.alt.alloc	Establish the requirements associated with the selected set of alternatives as the set of allocated requirements to those product components.
rq.cmmi.ts.alt.doc	Establish and maintain the documentation of the solutions, evaluations, and rationale.
rq.cmmi.ts.reuse	Identify the product-component solutions that will be reused or acquired.
rq.cmmi.ts.reuse.analyze	Perform make, buy or reuse analysis.
rq.cmmi.ts.scenario	Evolve operational concepts and scenarios.
rq.cmmi.ts.technology	Identify technologies currently in use and new product technologies.
rq.cmmi.ts.design	Establish the product architectural design.
rq.cmmi.ts.design.struct	Establish product partition into components.
rq.cmmi.ts.design.struct.if	Identify and document major intercomponent interfaces.
rq.cmmi.ts.design.struct.id	Establish product-component and interface identifications.
rq.cmmi.ts.design.state	Establish main system states and modes.
rq.cmmi.ts.design.if	Identify and document major external interfaces.
rq.cmmi.ts.design.crit	Establish and maintain criteria against which the design can be evaluated.
rq.cmmi.ts.design.method	Identify, develop, or acquire the design methods appropriate for the product.
rq.cmmi.ts.design.standard	Ensure that the design adheres to applicable design standards and criteria.
rq.cmmi.ts.design.fulfill	Ensure that the design adheres to allocated requirements.
rq.cmmi.ts.design.doc	Document and maintain the design in a technical data package.
rq.cmmi.ts.design.levels	Determine the number of levels of design and the appropriate level of documentation for each design level.
rq.cmmi.ts.design.impl	Base detailed design descriptions on the allocated product-component requirements, architecture, and higher level designs.
rq.cmmi.ts.rationale	Document the rationale for key decisions made or defined.
rq.cmmi.ts.if	Establish and maintain interface descriptions.
rq.cmmi.ts.if.crit	Design interfaces using criteria.
rq.cmmi.ts.implement	Implement design adhering to design decisions and architecture.
rq.cmmi.pi.seq	Guidance on determining the product integration sequence.
rq.cmmi.pi.if	Ensure interface compatibility of product components, both internal and external.
rq.cmmi.pi.if.review	Review interface descriptions for completeness.
rq.cmmi.pi.if.manage	Manage interface definitions, designs and changes.
rq.cmmi.ver.prepare	Prepare verification activities.
rq.cmmi.ver.review	Perform peer reviews on architecture and design documents.
rq.cmmi.ver.verify	Verify (part of) the architecture or design.
rq.cmmi.ver.analyze	Analyze verification results and identify corrective actions.
rq.cmmi.val.prepare	Prepare validation activities.
rq.cmmi.val.validate	Validate (part of) the architecture or design.
rq.cmmi.val.analyze	Analyze validation results and identify corrective actions.
rq.cmmi.dar.guid	Specify when a technical choice or design decision is architectural and subject to architecting process.
rq.cmmi.dar.rank	Evaluation criteria for alternative solutions should be ranked.
rq.cmmi.dar.evalmethod	Guidance on selecting evaluation methods for alternatives.
rq.cmmi.rskm	Guidance on handling architectural requirements as risks.
rq.cmmi.rskm.id	Identify architectural risks.
rq.cmmi.rskm.analyze	Analyze architectural risks.
rq.cmmi.rskm.mitigate	Mitigate architectural risks.
rq.cmmi.gen	Architecting process should be institutionalized according to CMMI's Generic Practices.

a substantial amount of requirements development practices. All other PAs provide only 4 or less requirements.

4 Discussion

In this section, we will discuss our results in conjunction with two generic architecting process models found in literature, and we will discuss the coverage of architecting processes in CMMI.

4.1 Generic Architecting Process Models in Literature

The CMMI imposes requirements on processes used by organizations. So if an organization were to institutionalize an architecting process based on a model found in literature, what would that organization have to do to make their architecting process CMMI level 3 compliant?

Although this analysis of CMMI's influence on architecting processes was based on an initial scope set out in the context of a particular company setting, the results of the analysis should be relevant for other generic architecting processes. This section explores that relevance. We examine the impact of the CMMI requirements derived in this paper on two generic architecture process models found in literature: one from a technical report and one from a recent conference paper. Please note that the architecting process models treated here differ significantly in scope: one focuses on design and analysis and the other focuses on architecture playing a central role throughout the software development lifecycle process. Also note that the models only roughly overlap the architecting process scope set out in Sect. 2.2. A discussion on how exactly these processes match or mismatch this scope will be presented in a separate paper.

Architecture-Based Development (ABD). This is the generic architecting process as developed by the Architecture group at the SEI. It is described in [1], but aspects of it are present in most of the publications of the SEI Architecture group (e.g. [11]). It is used as a reference here because its scope is close to that determined in Sect. 2.2, and because it represents one of the better known approaches to architecting in both industry and academia.

The ABD process consists of six steps:

1. Elicit the architectural requirements.
2. Design the architecture.
3. Document the architecture.
4. Analyze the architecture.
5. Realize the architecture.
6. Maintain the architecture.

Table 4. ASPAs Mapping onto ABD Steps

	Elicit	Design	Document	Analyze	Realize	Maintain
REQM	X					
RD	X	X				
TS		X	X	X	X	X
PI					X	
VER		X	X	X	X	X
VAL	X	X	X	X	X	
RSKM	X	X	X	X	X	X
DAR	X	X	X	X	X	X

Table 4 shows how the ASPAs map onto these steps. In order to make the ABD process CMMI Level 3 compliant, each of these steps should be implemented in such a way that the practices belonging to the ASPAs related to this step are satisfied. The following explanation applies to this mapping:

- RD is not only mapped onto the Elicit step but also onto the Design step. This is because the establishment of the "functional architectural structure" as part of this step is actually a practice that is part of RD.
- VER activities start from the Design step because, as discussed before, verification refers to the requirements produced during the Elicit step.
- The ABD process defines that each step includes validation (VAL) activities. For the Elicit step this refers to the validation of behavioral and quality scenarios.
- The Maintenance step is not well defined and scoped in the ABD process description. The existing text refers to means to prevent that the architecture drifts from its original precepts due to poor maintenance. This may include activities to extract the architecture of the as-built system, verify its level of compliance with the architecture of the as-designed system and performing the required corrective actions. In this respect, TS and VER should be mapped onto the Maintenance step.
- Because RSKM and DAR generally support all development and maintenance activities, they are related to all steps of the ABD process.

Generalized Software Architecture Design Model. In [2], Hofmeister *et al.* compare five industrial approaches to architectural design, and extract from their commonalities a general software architecture design approach. The approach involves three activities:

1. *Architectural analysis*: define the problems the architecture must solve. This activity examines architectural concerns and context in order to come up with a set of Architecturally Significant Requirements (ASRs).
2. *Architectural synthesis*: the core of architecture design. This activity proposes architecture solutions to a set of ASRs, thus it moves from the problem to the solution space.
3. *Architectural evaluation*: ensures that the architectural design decisions made are adequate. The candidate architectural solutions are measured against the ASRs.

It should be noted that this generalized model is of a higher level of abstraction than the ABD process discussed before, and that its scope is explicitly limited to the Design step of architecting.

Table 5 shows how the selected set of ASPAs map onto these activities. In order to make a process based on this generalized model CMMI Level 3 compliant, each of these activities should be implemented in such a way that the practices belonging to the ASPAs related to this activity are satisfied. The following explanation applies to this mapping:

- Unlike the ABD process, the generalized model has limited its scope to the design step of the architecture. For this reason, PI and VAL cannot be mapped to this model.

Table 5. ASPAs Mapping onto Generalized Architecture Design Model Activities

	Analysis	Synthesis	Evaluation
REQM	X		
RD	X		
TS		X	
PI			
VER			X
VAL			
RSKM	X	X	X
DAR	X	X	X

- The Architectural Evaluation activity ensures that the architectural design decisions made are adequate. The candidate architectural solutions are measured against the ASRs. Although the result is called the *validated* architecture, this activity is *verification* (VER) in CMMI terms because it refers to the requirements (ASRs) produced during the Architectural Analysis activity.
- Since RSKM and DAR generally support all development and maintenance activities, they are related to all activities of the generalized model.

4.2 CMMI Coverage of Architecting Processes

As discussed in Sect. 3, the architecting process scoped in Sect. 2.2 may include elements that are not covered by CMMI Level 3. An analysis of the information in this section against the CMMI results in the following elements that are not or only indirectly covered.

rq.arch.documentation. Standardization of architectural documentation: the activity to document architecture and design information is part of the practices of TS, including roughly what kind of information should be documented. In this way the CMMI guides standardization of documents. Concrete standards, however, are not provided.

rq.arch.conform. Facilitating conformance to architecture during the implementation process: The implementation phase as such is part of the practices of TS, including references to VER in order to verify the implementation once it is finished. However, the CMMI does not provide any explicit support in ensuring that the architecture and design will be adequately implemented during implementation (e.g. by involvement of the architects).

rq.learning.product. Bottle experiences and make available for architects: the CMMI has many PAs that deal with establishing an infrastructure for organizational learning and improvement. Because the CMMI is a process framework, this is strongly focussed on the process dimension (like the architecting process), not on the product dimension (like architectural solutions). Only at Level 5 the PA Organizational Innovation and Deployment (OID) addresses improvements on processes and (process and product related) technologies. Product related technologies may also be interpreted as architectural solutions.

Reuse development. Although not indicated as a requirement in Sect. 2.2, reusable assets like product components, source code libraries and technical documents may be developed and maintained in the context of the usage scenario Product Development. Although the CMMI addresses the concept of reuse in a number of PAs, the development and management of a reusable base of core assets is not adequately covered. The SEI has an area of work called Software Product Lines in the context of its Product Line Systems Program to cover this. Although this includes some process related support, no CMMI-like reference model exists.

Roadmapping. Like reuse development, the development and evolution of architecture and technology roadmaps has not been indicated as a requirement in Sect. 2.2 although it is relevant in the context of the usage scenario Product Development. Roadmapping is not supported by the CMMI because it is mainly a product planning (pre-development) activity.

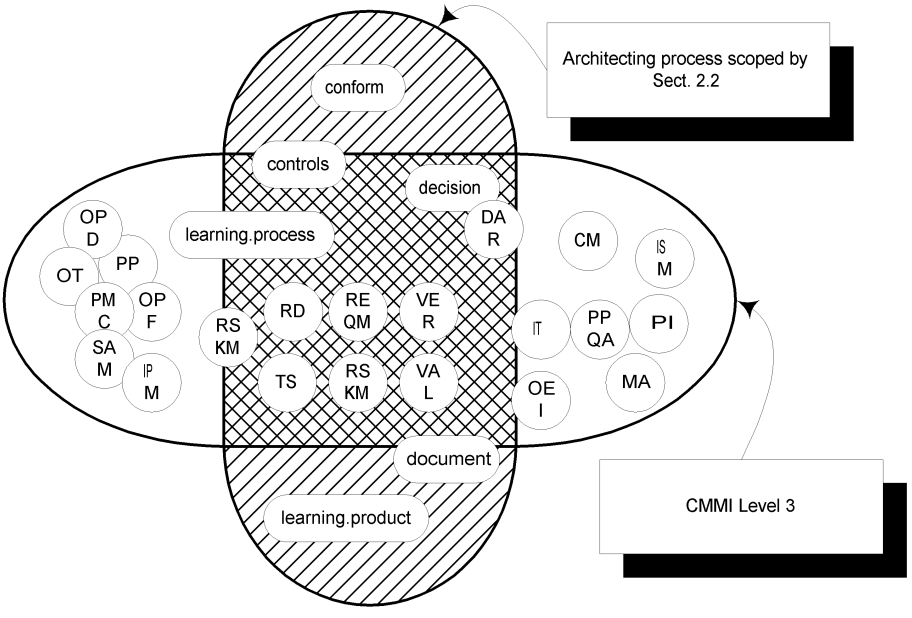

Fig. 2. CMMI, architecting process and cross-section

An informal visualization of the overlap between CMMI and the architecting process is presented in Fig. 2.

A note on the meaning of the fact that these elements are not covered by CMMI. We have not made any statement on the relative merits of these elements. One could argue that this lack of coverage is a shortcoming of CMMI; conversely, one could argue that, given the success of CMMI, how do we know that the elements in the square aren't by themselves good enough for an optimal architecting process? The current state of affairs does not allow us to answer this question in a general sense; the analysis in

Sect. 2.2 merely indicates that in the current organizational setting, the elements would contribute to achieving the business goals set.

5 Conclusions and Further Work

Our starting point in this paper was a large IT company with a need to institutionalize a generic architecting process that is compliant with CMMI Maturity Level 3. To this end, we have studied and discussed the relation between architecting and CMMI, resulting in the identification of PAs significant to architecting, and a list of requirements to make a generic architecting process compliant with CMMI Maturity Level 3. Furthermore, we have compared our findings with two well-known process models from literature.

We have concluded that:

- Architecture is not a well-defined concept in the CMMI; the word is used in many meanings, most of which are not defined in the glossary.
- CMMI implicitly provides considerable support in establishing an architecting process. However, in some areas of architecting, the CMMI only gives indirect support. The weaker areas are documentation, facilitating the implementation of the architecture, and learning from architectural choices.
- In product development contexts, there are two activities generally associated with architecting that are insufficiently supported by the CMMI: architectural roadmapping and the exploitation of reusable assets.

Besides these conclusions, other relevant findings worth mentioning are:

- Although the scope of this paper was limited to CMMI Level 3, an investigation of the level 4 and 5 PAs shows that none of these are Architecting Significant according to our scope. This resonates with remarks made informally by Grady Booch [12].
- Although architecting is generally viewed as an engineering activity, two PAs outside Engineering are crucial to a good architecting process: RSKM and DAR.

Further Work. The work described in this paper was based on CMMI version 1.1. Since August 2006, CMMI version 1.2 exists. This is the ”CMMI for Development”. There are now three CMMI variants: Development, Service and Acquisition. Since support for CMMI 1.1 will be dropped in time, we will update the work in this paper to CMMI 1.2 for Development in the coming months.

As has been mentioned previously in this paper, the work described here was done in the context of designing a generic architecting process for a large IT company. Since this other work also yielded some interesting insights, we will describe it in more detail in a separate paper, which will also contain a comparison of our developed architecting process to the generic architecting processes discussed in Sect. 4.1.

An architecting process that complies with a maturity model also begs a comparison with Architecture Maturity Models (AMMs), such as the IT Architecture Capability Maturity Model (ACMM) developed by the US Department of Commerce [13]. This

comparison could be subject of a future analysis. Conversely, the development of architecting enhancements to the CMMI would be an attractive idea for CMMI-compliant companies that wish to enhance their architecting maturity levels, but would rather not introduce another maturity model on top of CMMI. This could be another interesting avenue to explore.

References

1. Bass, L., Kazman, R.: Architecture based development. Technical Report CMU/SEI-2000-TR-007, SEI (April 1999)
2. Hofmeister, C., Kruchten, P., Nord, R., Obbink, H., Ran, A., America, P.: Generalizing a model of software architecture design from five industrial approaches. In: WICSA 2005. 5th Working IEEE/IFIP Conference on Software Architecture, Pittsburgh, Pennsylvania, pp. 77–86 (2005)
3. Bosch, J.: Software architecture: The next step. In: Oquendo, F., Warboys, B.C., Morrison, R. (eds.) EWSA 2004. LNCS, vol. 3047, pp. 194–199. Springer, Heidelberg (2004)
4. Tyree, J., Akerman, A.: Architecture decisions: Demystifying architecture. IEEE Software 22(2), 19–27 (2005)
5. van der Ven, J.S., Jansen, A., Avgeriou, P., Hammer, D.K.: Using architectural decisions. In: Hofmeister, C., Crnkovic, I., Reussner, R. (eds.) QoSA 2006. LNCS, vol. 4214, Springer, Heidelberg (2006)
6. Abowd, G., Bass, L., Clements, P., Kazman, R., Northrop, L., Zaremski, A.: Recommended best industrial practice for software architecture evaluation. Technical Report CMU/SEI-96-TR-025, SEI (1997)
7. Clements, P., Kazman, R., Klein, M.: Evaluating Software Architectures. Addison-Wesley, Reading (2002)
8. Gilb, T.: Principles of Software Engineering Management. Addison-Wesley, Reading (1988)
9. CMMI Product Team: Capability maturity model integration (cmmi(sm). Technical report, SEI (2002) staged representation, version 1.1, CMU/SEI-2002-TR-012, ESC-TR-2002-012
10. Chrissis, M.B., Konrad, M., Shrum, S.: CMMI: Guidelines for Process Integration and Product Improvement. Addison-Wesley, Reading (2003)
11. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice, 2nd edn. Addison-Wesley, Reading (2003)
12. Booch, G.: Observations from the road (May 2006), `http://www-03.ibm.com/developerworks/blogs/page/gradybooch?entry=observation_from_the_road`
13. US Department of Commerce: It architecture maturity model `http://www.osec.doc.gov/cio/arch_cmm.htm`