VU Research Portal

Improving Solution Architecting Practices

Poort, E.R.

2012

document version
Publisher's PDF, also known as Version of record

Link to publication in VU Research Portal

citation for published version (APA)
Poort, E. R. (2012). *Improving Solution Architecting Practices*. [PhD-Thesis - Research and graduation internal, Vrije Universiteit Amsterdam].

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:
vuresearchportal.ub@vu.nl

Download date: 13. Mar. 2024

6 The Influence of CMMI on Establishing an Architecting Process

In 2006, we started out to create a generic architecting process for Logica. Since the company had set an objective to achieve Maturity Level 3 of the Capability Maturity Model Integration® (CMMI®), the process needed to comply with the relevant requirements set by the CMMI. This chapter 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 CMMI 1.3 is much stronger in support of architecting activities than CMMI 1.1 (the version for which we have done this analysis previously), but a few possible improvements remain.

6.1 Introduction

The setting of this chapter is the establishment of an institutionalized architecting process in Logica. We had established that such a 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-DEV Maturity Level 3 [1]. The required analysis to obtain this insight was originally done using CMMI version 1.1 and published in [Poort et al., 2007]. This chapter updates the analysis to CMMI version 1.3. There are now three CMMI constellations: Development, Service and Acquisition. Our work pertains to CMMI for Development (CMMI-DEV) [CMMI Product Team, 2010].

[1] *CMMI-DEV Maturity Level 3* is mostly abbreviated to *CMMI Level 3* in the rest of this chapter

As references we have chosen two generic processes found in literature: Architecture Based Development [Bass and Kazman, 1999], 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., 2007], because their model represents the commonalities between five industrial approaches.

First, in §6.2 we will present the organizational context and scope of a generic architecting process. In §6.3, the CMMI process areas that are relevant to such an architecting process will be identified, and their requirements on architecting processes extracted. In §6.4 follows a discussion on the impact of the CMMI requirements on 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.

6.2 Architecting Process Context and Scope

6.2.1 Organizational context

The organizational context of this study was described in Chapter 1. One of the company's Technical Board'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.

The Technical Board's 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.

6.2.2 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. For the purposes of this chapter, 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*.

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 solutions. 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 [Bosch, 2004, Tyree and Akerman, 2005, van der Ven et al., 2006].

- 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 [Obbink et al., 2002, Clements et al., 2002], and the responsibilities of the Technical Board 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 good practices) and product-related (like good 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

Table 6.1: Scope of architecting process: high-level requirements.

rq.arch	Give guidance on architecting technical solutions.
rq.arch.decision	Give guidance on how to make architectural decisions.
rq.arch.sales	Facilitate solution shaping during the sales process.
rq.arch.doc	Standardize architectural documentation.
rq.arch.controls	Give guidance on architectural controls.
rq.arch.conform	Assure conformance with architecture during the implementation process.
rq.scalable	Be scalable over business unit sizes (20 - 2000) and project/programme sizes (80K€ - 500M€), and over a broad range of size and complexity of solutions.
rq.generic	Be flexible / generic to work in diverse applications.
rq.generic.tailoring	Be accompanied by a set of tailoring guidelines.
rq.accessible	Be simple, accessible to all.
rq.accessible.terminology	Use terminology familiar to company staff.
rq.cmmi	Be CMMI Maturity Level 3 compliant.
rq.learning.product	Bottle product experiences and make them available to architects in a controlled manner.
rq.learning.process	Support a structure for organizational process learning.

embedded system development. This means its concept of "architecture" covers not only software, but the wider scope of *solution 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".

In summary, we need an architecting process description 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.

The scope of what is meant by an "architecting process" in this chapter is documented as a list of requirements[2] in Table 6.1. In §6.4.1, we will identify a number of generic architecting processes in literature that are similar in scope.

[2]A note on the tagging of requirements in this chapter: the reader will notice the use of mnemonic, hierarchical tagging [Gilb, 2005]. The use of dots indicates a hierarchical grouping, with an implicit traceability to higher level requirements.

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.

6.3 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. CMMI-DEV is the CMMI constellation intended for solution development.

The "staged representation" of the CMMI-DEV 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 [Gibson et al., 2006]. Each maturity level consists of a number of process areas (PAs). Each process area consists of a small set of *goals* followed by a collection of *practices* to be performed in order to realize the goals. In order to satisfy a process area, an organization must have visibly implemented the achievement of the process area *goals* in its processes. Before goals can be considered to be satisfied, either the process area *practices* as described, or acceptable alternatives to them, must be present in the planned and implemented processes of the organization. [CMMI Product Team, 2010] contains the goals and practices for all process areas, accompanied by information to help CMMI users understand them.

A process complies to a certain maturity level if the goals and practices of all process areas of that level are satisfied. The process areas are customarily referred to by a set of fixed tags; all level 2 and 3 process areas and their tags are listed in Table 6.2.

Goals and practices of a process area are divided into specific ones and generic ones. Specific goals and practices directly refer to the process area 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 process areas.

CMMI Maturity Level 3 requires that for all process areas 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 [CMMI Product Team, 2010].

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

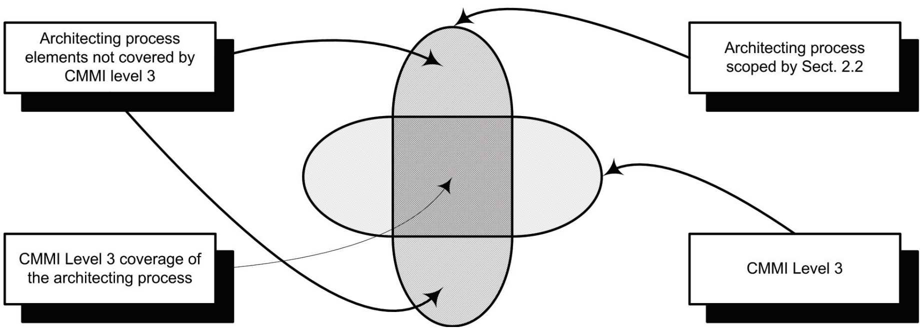
Architecting process elements not covered by CMMI level 3
Architecting process scoped by Sect. 2.2
CMMI Level 3 coverage of the architecting process
CMMI Level 3

Figure 6.1: CMMI coverage of the architecting process.

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 §6.2.2. We call this set the *architecting significant process areas (ASPAs)*.

6.3.1 CMMI-compliant architecting process

The boundaries (scope) of the architecting process are determined in §6.2.2. 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 process areas. This analysis will be performed further on in this chapter.

Fig. 6.1 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 §6.2.2.

The scope of this chapter 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 §6.3.3 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.

6.3.2 Architecture concepts in the CMMI

The word "architecture" is used extensively in the CMMI. It appears in 12 out of 22 process area descriptions [CMMI Product Team, 2010]. 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 defined in the CMMI-DEV 1.3 glossary. Apart from its defined usage, the word is also used in the concept of "process architecture" to denote designing of company processes. This type of activity is outside the scope of this chapter as defined in §6.2.2, and we have filtered out this usage in our analysis.

Several architecture-related terms are defined in the CMMI glossary:

- *Architecture* is defined as: "The set of structures needed to reason about a product. These structures are comprised of elements, relations among them, and properties of both." The glossary explicitly points out the role of quality attributes in the context of architecture. This definition is quite close to our definition of solution architecture in §1.2.1 on page 3, except that it lacks the keyword "fundamental".
- *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 requirements, and their design constraints"
- *Definition of required functionality and quality attributes*: "A characterization of required functionality and quality attributes obtained through "chunking," organizing, annotating, structuring, or formalizing the requirements (functional and

non-functional) to facilitate further refinement and reasoning about the requirements as well as (possibly, initial) solution exploration, definition, and evaluation." This term refers to the beginning of architecting activities, and is within the scope of our architecture process.

- *Nontechnical requirements* are defined as: "Requirements affecting product and service acquisition or development that are not properties of the product or service." This term coincides with our definition of "Delivery requirements" in §2.3.2 (page 19).

- *Quality attribute* is defined as: "A property of a product or service by which its quality will be judged by relevant stakeholders. Quality attributes are characterizable by some appropriate measure." The CMMI glossary explicitly links quality attributes to architecture.

- *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 project or work group."

One other architecture-related term is used extensively, but not defined: *design*. Since a design is definitely a structure needed to reason about a product, one could argue that it falls under the CMMI definition of architecture. We include guidance about design in CMMI-DEV in our analysis wherever it falls within our scope as defined in §6.2.2.

These considerations show that a number of key concepts and terms relevant to architecting are defined in the CMMI. The following section will discuss the identification of the CMMI requirements on an architecting process.

6.3.3 Process areas relevant to architecting

Our approach to establish which requirements CMMI imposes on architecting processes is to first identify which process areas 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 process areas against the architecting process scoped in §6.2.2 results in a set of process areas that have a direct and significant contribution to the objectives of this process. As discussed before, these process areas are called *architecting significant process areas* (ASPAs).

The process areas of the CMMI are grouped into four categories:

- *Process Management*. These process areas contain the activities related to defining, planning, implementing, monitoring, evaluating and improving all other processes. The architecting process is subject to these process management process areas in order to assure the required level of capability.
- *Project Management*. These process areas 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 process areas 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 process areas.
- *Support*. These process areas cover the activities that support all other process areas like establishing measurement programs, verification of compliance, and effective decision making. The architecting process is also subject to these process areas.

Table 6.2 identifies the categorized set of Level 3 process areas and indicates which process areas have been qualified as an architecting significant process area. It should be noted that *all* process areas of the CMMI contribute to the objectives of the architecting process. Their contribution may be direct because the process area is actually part of the architecting process, or indirect because the process area is establishing the context and preconditions for a successful architecting process.

As stated before an architecting significant process area has a direct contribution and this contribution should also be significant. This is the case for all Engineering process areas, two Project Management process area (Risk Management and Requirements Management) and one Support process area (Decision Analysis and Resolution). Risk Management, Requirements Management and Decision Analysis and Resolution are actually part of the architecting process and contribute significantly to its objectives. The architecting relevance of the set of architecting significant process areas 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 (SP)1.1 *Understand Requirements* describes the process of the acceptance of requirements according to objective criteria. “Consistent with

Table 6.2: Categorized Level 2 & 3 Process areas and their architecting significance.

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

architectural approach and quality attribute priorities" is an example criterion relevant to architecting. It is also relevant to the impact analysis mentioned in SP1.3 *Manage Requirements Changes*: "Requirements changes that affect the product architecture can affect many stakeholders." SP1.4 *Maintain Bidirectional Traceability of Requirements*: traceability to architectural components is mentioned: "Work products for which traceability may be maintained include the architecture, product components, development iterations (or increments), functions, interfaces, objects, people, processes, and other work products.". Traceability to architectural decisions is implied.]

RD *Requirements Development.* This process area is where functional and quality attributes requirements are elicited, analyzed and established. 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 to "develop architectural requirements capturing critical quality attributes and quality attribute measures necessary for establishing the product architecture and design". SP2.3 *Identify Interface Requirements* prescribes the definition of interfaces as an integral part of the architecture definition. SP3.2 *Establish a Definition of Required Functionality and Quality Attributes* prescribes the identification of architecturally significant quality attributes. Other important architecting activities are impact and risk assessment of the requirements, mentioned under SP3.4 Analysis and SP3.5 Validation.]

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 *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 mentions architecture evaluations to be conducted periodically throughout product design. SP2.2 *Establish a technical data package* gives 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.]

PI *Product Integration.* In this process area, the architecture is implemented in an actual integrated system and delivered. The architecting significance of the process area lies in the involvement of the architect in the implementation phase, and the architectural significance of the integration strategy [Product Integration has three Specific Goals (SG): *Prepare for Product Integration*, *Ensure Interface Compatibility* and *Assemble Product Components and Deliver the Product*. These goals should be achieved in line with the product architecture.

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 when placed in its intended environment (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 [Bosch, 2004, Tyree and Akerman, 2005]. 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. Many DAR requirements overlap with selection practices in Technical Solution. ["When competing quality attribute requirements would result in significantly different alternative architectures." is listed as a typical guideline for requiring formal evaluation.]

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 not it is an architectural requirement, as will be explained in Chapter 8. [A requirement that, when not fulfilled, heavily "affects the ability of the project to meet its objectives" (SP1.1 Determine Risk Sources and Categories), has a good chance to be considered architectural. Typical architectural risk sources listed are "uncertain requirements" and "Competing quality attribute requirements that affect solution selection and design". 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 architecting significant process areas yields the requirements imposed on the architecting process by the CMMI. These requirements are listed in Table 6.3. In agreement with the nature of the CMMI, this table is effectively a list of 73 best practices that support companies in creating and implementing an architecting process.[3]. These best practices are based on the *informative* text accompanying the architecting significant process areas in [CMMI Product Team, 2010], so strictly speaking an architecting process that does not fulfill the requirements can still be CMMI compliant, as long as the architecting significant process area goals are visibly fulfilled by alternative practices. For the purposes of this analysis, however, we have based the requirements on the architecting significant process area texts. The tags in Table 6.3 allow traceability to the process areas that the requirements originated from, and give the list a clear structure. The largest contributor is Technical Solution (TS) with 30 requirements, confirming our earlier observation that TS covers the core of architecting. The next largest contributor is Requirements Development (RD) with 21 requirements, indicating that an architecting process within our scope includes a substantial amount of requirements development practices. All other process areas provide only 4 or less requirements.

6.4 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.

6.4.1 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

[3]CMMI version 1.1 yielded 67 [Poort et al., 2007]), giving a quantitative indication of the improved support for architecting in version 1.3

Table 6.3: 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.doc	Translate stakeholder needs, expectations, constraints, and interfaces into documented customer requirements.
rq.cmmi.rd.prio	Establish and maintain a prioritization of customer functional and quality attribute requirements.
rq.cmmi.rd.fun-arch	Develop a functional architecture.
rq.cmmi.rd.recursive	Analyze requirements recursively.
rq.cmmi.rd.arch-req	Develop architectural requirements capturing critical quality attributes necessary for establishing architecture.
rq.cmmi.rd.tech	Develop requirements in technical terms necessary for product and product component design.
rq.cmmi.rd.drivers	Determine key mission and business drivers, and determine architecturally significant quality attributes based on them.
rq.cmmi.rd.part	Partition requirements into groups, based on established criteria, to facilitate and focus the requirements analysis.
rq.cmmi.rd.alloc	Allocate requirements and design constraints to product components and the architecture, and to functional elements.
rq.cmmi.rd.derive	Derive requirements that result from design decisions.
rq.cmmi.rd.if	Identify interface requirements.
rq.cmmi.rd.depend	Establish and maintain relationships between requirements.
rq.cmmi.rd.an	Analyze requirements.
rq.cmmi.rd.an.key	Identify key requirements that have a strong influence on cost, schedule, performance, or risk.
rq.cmmi.rd.an.perf	Identify technical performance measures that will be tracked during the development effort.
rq.cmmi.rd.an.scen	Develop and analyze 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 definition of required functionality and quality attributes.
rq.cmmi.rd.impact	Assess the impact of architecturally significant quality attribute requirements on product and development costs and risks.
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.ts.alt	Develop detailed alternative solutions to address architectural requirements.
rq.cmmi.ts.alt.cots	Identify candidate COTS products that satisfy the requirements.
rq.cmmi.ts.alt.techn	Identify technologies currently in use and new product technologies for competitive advantage.
rq.cmmi.ts.alt.reuse	Identify re-usable solution components or applicable architecture patterns.
rq.cmmi.ts.alt.crit	Develop the criteria for selecting the best alternative solution, typically addressing costs, schedule, benefits and risks.
rq.cmmi.ts.alt.crit.eval	Based on the evaluation of alternatives, assess the adequacy of the selection criteria and update them as necessary.
rq.cmmi.ts.alt.issues	Identify and resolve issues with the alternative solutions and requirements.
rq.cmmi.ts.alt.eval	Evaluate alternative solutions against criteria.
rq.cmmi.ts.alt.acq	Identify the product component solutions that will be reused or acquired.
rq.cmmi.ts.alt.doc	Establish and maintain the documentation of the solutions, evaluations, and rationale.
rq.cmmi.ts.scenario	Evolve operational concepts and scenarios.
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.doc	Document and maintain the design in a technical data package.
rq.cmmi.ts.doc.levels	Determine the number of levels of design and the appropriate level of documentation for each design level.
rq.cmmi.ts.doc.views	Determine the views to be used to document the architecture.
rq.cmmi.ts.doc.impl	Base detailed design descriptions on the allocated product-component requirements, architecture, and higher level designs.
rq.cmmi.ts.doc.rationale	Document the key decisions made or defined, including their rationale.
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.peer	Perform peer reviews on architecture and design documents.
rq.cmmi.ver.eval	Perform architecture evaluations.
rq.cmmi.ver.conform	Verify architecture conformance of the implementation.
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 formal decision 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.

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 chapter on two generic architecture process models found in literature: one from a technical report and one from a journal 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 §6.2.2.

Architecture-Based Development (ABD)

This is the generic architecting process as developed by the Architecture group at the SEI. It is described in [Bass and Kazman, 1999], but aspects of it are present in most of the publications of the SEI Architecture group (e.g. [Bass et al., 2003]). It is used as a reference here because its scope is close to that determined in §6.2.2, and because it represents one of the better known approaches to architecting in both industry and academia.

The ABD process consists of six activities:

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 6.4 shows how the architecting significant process areas 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 architecting significant process areas related to this step are satisfied. The following explanation applies to this mapping:

- Requirements Development (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.
- Verification (VER) activities start from the Design step because, as discussed before, verification refers to the requirements produced during the Elicit step.

Table 6.4: 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

- 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, Technical Solution (TS) and Verification (VER) should be mapped onto the Maintenance step.

- Since Risk Management (RSKM) and Decision Analysis and Resolution (DAR) generally support all development and maintenance activities, they are related to all steps of the ABD process.

Generalized software architecture design model

[Hofmeister et al., 2007] 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).

Table 6.5: 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

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 excludes the realization of the architecture (it is design focused, as the name "generalized architecture design model" implies).

Table 6.5 shows how the selected set of architecting significant process areas 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 architecting significant process areas related to this activity are satisfied. The following explanation applies to this mapping:

- Unlike the ABD process, the generalized model excludes architecture realization from its scope. For this reason, PI cannot be mapped to this model.

- The Architectural Evaluation activity ensures that the architectural design decisions made are adequate. The candidate architectural solutions are measured against the architecturally significant requirements (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. *Validation* (VAL) in CMMI terms (against the user needs behind the requirements) is not part of the generalized model.

- Since Risk Management (RSKM) and Decision Analysis and Resolution (DAR) generally support all development and maintenance activities, they are related to all activities of the generalized model.

6.4.2 CMMI Coverage of architecting processes

As discussed in §6.3, the architecting process scoped in §6.2.2 may include elements that are not covered by CMMI Level 3. An analysis of the information in this section against the CMMI yields the following elements that are not or only indirectly covered.

rq.arch.doc Standardization of architectural documentation: the activity to document architecture and design information is part of the practices of Technical Solution (TS), including what kind of information should be documented and guidance on how it should be organized. In this way the CMMI guides standardization of documents. In CMMI 1.3, guidance on architecture documentation is significantly improved over CMMI 1.1, including e.g. the use of views [ISO 42010, 2011].

rq.arch.conform Facilitating conformance to architecture during the implementation process: The implementation phase as such is part of the practices of Technical Solution (TS) and Product Integration (PI), including references to Verification (VER) in order to verify the implementation once it is finished. CMMI 1.1 did not provide any explicit support in ensuring that the architecture and design will be adequately implemented during implementation; CMMI 1.3 gives more guidance, including the use of architecture evaluations and the role of quality attributes.

rq.learning.product Bottle experiences and make available for architects: the CMMI has many process areas 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 process area 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.

rq.arch.controls The requirements for controls like architectural governance and reviewing in CMMI is limited. Reviewing is covered in the Verification (VER) process area. Architectural governance, however, is largely missing. With architectural governance we mean activities to manage architectural resources like reference or enterprise architectures, or the architects themselves. The only reference made to resourcing architects is an example in Generic Practice 2.4: "Appointing a lead or chief architect that oversees the technical solution and has authority over design decisions helps to maintain consistency in product design and evolution." The only type of architectural asset discussed is a software product line's core asset base. The lack of architecture-specific governance guidance is a logical consequence of the fact that the CMMI model places such governance activities in Generic Practices: they are abstracted away from specific application areas.

rq.arch.sales CMMI-DEV offers no support for the sales process. Some examples in [CMMI Product Team, 2010] refer to the existence of a contract, but the definition of Customer is limited to "The party responsible for accepting the product or for authorizing payment." As we have seen in Chapter 3, a sales process can have significant impact on architecting activities. CMMI could be improved by acknowledging this impact and giving guidance on it.

An informal visualization of the overlap between CMMI and the architecting process is presented in Fig. 6.2. In this figure, CMMI process areas (circles) and architecting process requirements (ovals) are plotted onto the areas of Fig. 6.1. Elements in the overlapping square area are covered by both CMMI and our architecting process scope. Partly covered elements are plotted straddling the scope boundary lines.

A note on the meaning of the fact that some 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 that *are* covered by CMMI 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 §6.3 merely indicates that in the current organizational setting, the elements would contribute to achieving the business goals set.

6.5 Conclusions and Further Work

Our starting point in this chapter 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 process areas significant to architecting, and a list of

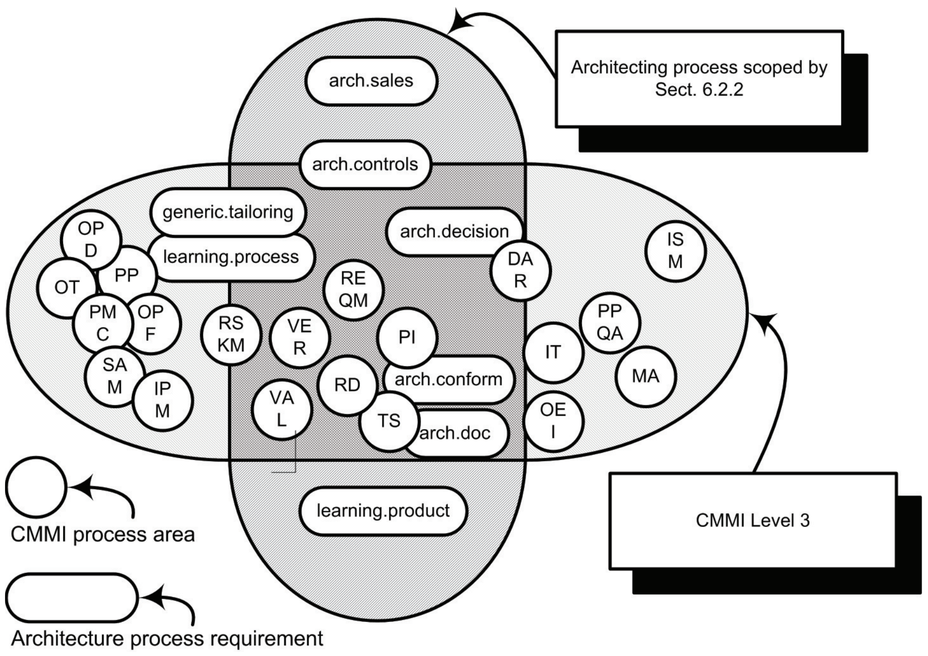

Figure 6.2: CMMI, architecting process and cross-section. See p.88 for PA abbreviations.

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 conclude that:

- Architecture is a well-defined concept in the CMMI-DEV 1.3.
- CMMI-DEV 1.3 provides considerable support in establishing an architecting process. However, in some areas of architecting, the CMMI only gives weak support. The weaker areas are architecture governance, facilitating the sales phase, and learning from architectural choices.

Besides these conclusions, other relevant findings worth mentioning are:

- Although the scope of this chapter was limited to CMMI Level 3, an investigation of the level 4 and 5 process areas shows that none of these are Architecting Significant according to our scope, with the possible exception of Causal Analysis and Resolution.
- Although architecting is generally viewed as an engineering activity, three process areas outside Engineering are crucial to a good architecting process: Requirements Management, Risk Management and Decision Analysis and Resolution. This may not seem significant if the placement of these process areas outside of Engineering is merely seen as a structural choice in the CMMI model. Organizations should, however, not make the mistake of not applying these processes to engineering, or not involving their (architecting) engineers in them.
- CMMI 1.3 is much improved over version 1.1 in terms of support for architecting.

Further work

As has been mentioned in the introduction to this chapter, the work described here was done in the context of designing a generic architecting process for a large IT company. This work gave rise to the remaining chapters in this Part.

The generalized architecture design model discussed in §6.4.1 returns in Chapter 8, where we will use it to illustrate the impact of our insight into the nature of architecture.

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 [US Department of Commerce, 2007]. This comparison could be subject of a future analysis.