Software Architecture Review and Assessment (SARA) Report

Version 1.0

Revision History

Date	Version	Description	Author
03 May 1999	0.1	Creation at the Burlington meeting	Philippe Kruchten
04 May 1999	0.2	Add material from the 3 subgroups on: - Social aspects - Workflow - Foundations	Rick Kazman, A. Ran H. Obbink, Ph. Kruchten R. Hilliard, T. Mikkonen
09 September	0.3	Add material from the 3 subgroups on: Gr1, Analysis framework for ASR's section 8.1 Gr2, Architecture Description Document Chapter 6 Gr3, Review typology section 5.3	Alexander Ran, Herman Postema, Wojtek Kozaczynski, Henk Obbink Ed. Kahan, Juergen Mueller
04 October	04	Added revised material from Juergen on Architecture impact matrices	Henk Obbink
Jan 26, 2000	05	Includes new material on workflow, foundations, methods & techniques	Rick Kazman, Ed Kahan
Jan 26, 2000	06	Adds section on pragmatics, reformats	Rick Kazman
Jan 27, 2000	07	Cleaning up the document	Henk Obbink, Steve Heise, Ed Kahan
28 February 2000	07a	Revision to Section 4 Conceptual Framework	Rich Hilliard
May 10-11, 2000	07b	Munich review: modified Conceptual Framework, Review input and outcomes, Workflow, Pragmatics	R. Hilliard, A.Ran, L. Dominick, N. Werner, F. Paulisch, R. Kazman, T. Mikkonen, H. Obbink
May 11, 2000	08	Create master document and subdocuments Propose alternate strategy for section on method and techniques	Ph.Kuchten, A Ran, R. Hilliard
September 06, 2000	09	Reviewed the version 0.8 and the suggested changes are in the file. Sections 1-7 have been changed.	H..Obbink, A.Ran, W.Kozaczynski, Lutz Dominique, H. Postema
August 27, 2001	0.95	Amsterdam meeting General clean up; Define template for technique;	Ph. Kruchten, R. Hilliard, H..Obbink, A.Ran, W. Tracz, W.Kozaczynski, H. Postema
November 1, 2001	0.96	Technical Presentation Edit	Will Tracz

November 21, 2001	0.96a	Revision of Chapter 4	A.Ran
January 28th, 2002	0.97	Incorporated templates Dealt with several comments from a reviewer, a colleague of Henk Fixed external references. Added Siemens's example	Philippe Kruchten
4 February 2002	0.98	Fixed Glossary and definitions.	Rich Hilliard
6 February 2002	1.0	Issue version 1.0 for ICSE Workshop	Philippe Kruchten

Table of Contents

Report on Software Architecture Review and Assessment

1 Acknowledgments

Many people representing multiple companies worked on this report. Below is the list of the major contributors:

Henk Obbink	Philips
Philippe Kruchten	Rational Software
Wojtek Kozaczynski	Rational Software
Herman Postema	SIOUX
Alexander Ran	Nokia
Lutz Dominick	indatex
Rick Kazman	SEI
Rich Hilliard	ConsentCache
Will Tracz	Lockheed Martin Systems Integration - Owego
Ed Kahane	IBM

2 Objectives

The objective of this document is to provide concrete, practical, experience-based guidance on how to conduct architectural reviews. This includes guidelines on:

- what steps to follow,
- what questions to ask,
- what information to collect and document,
- what documentation templates to use, and
- tips on how to manage the social, managerial, and technical issues that arise when reviewing an artifact as important and complicated as a software architecture.

This document can be used for an external review or an internal (or self-assessment) review.

The international working group on Software Architecture Review and Assessment (SARA) has produced the document. It is a summary of the group's findings and conclusions on the review and assessment of software architectures (and system architectures, where those systems are software intensive). The intention of the document is to represent the collected best practices of a wide group of industrial architects and consultants. As such, it is rooted in practice. It is intended to report on what actually *works* for evaluating architecture quality. This document serves as a growing repository covering all aspects of reviewing and analyzing software architectures.

Note: This document does not describe how architectures are designed, how they are developed into working systems, or how they are justified to management or stakeholders.

3 Document Structure

The next section of this document (Section 4) provides a conceptual framework for architectural reviews. The concepts introduced and explained in this section are used throughout the remainder of the document.

Sections 5 and 6 describe the inputs to and the outcomes of architectural reviews respectively.

An architectural review is a structured activity and as such should follow a well-defined workflow. Section 7 describes a typical workflow and a set of typical review activities. These activities refer to common review methods and techniques, which are described in Section 8.

The following section (Section 9) addresses the non-technical aspects of architectural reviews. These include social, psychological, and managerial issues that the reviewers should understand and cope with.

Section 10 is a set of case studies and is followed by references and a glossary.

4 Conceptual Framework for Architectural Review

This section presents the conceptual framework for an architectural review used in the remainder of the document. It introduces key terms and concepts associated with the process of creating an architecture, the nature of reviews, their required inputs and expected outputs, and the role of software architecture reviews in the overall system lifecycle.

4.1 The Context of Architecture Design

The **architecture** of a software-intensive system is ***"the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution."*** [IEEE 1471]

The Figure 4-1 presents the architecture design context conceptual model used in this document. A "system" may be a single application, a subsystem of another system, a "system of systems," a product line or product family, etc. A system is designed to operate in a specific environment. That environment exerts influences on the system. These constraints can include developmental, operational, political, and social influences.

A system is designed for direct or indirect use by people that become stakeholders in the system's design, construction, and deployment. System stakeholders inhabit the environment of the system (at least in the sense of information and control flow). Stakeholders include:

- the system's clients,
- end users,
- developers,
- maintainers,
- component vendors,
- administrators,
- owners, and
- operators.

Stakeholders have different concerns that are addressed by the system. Some stakeholders may have concerns specific to system architecture. Some stakeholders may be concerned with the properties and quality of the architecture description. Their concerns may range from the very specific (e.g., functional and non-functional requirements) to the very general (e.g., needs, goals, preferences, business objectives, and opportunities).

The architect identifies, analyzes, and negotiates the stakeholders concerns and represents them by specific architecturally significant requirements (ASRs). The architect creates an architecture description prescribing the architecture to be realized by the system and that will address the concerns of the stakeholders when used in a specific environment. Once the system is built, the architecture description will document, at a conceptual level, the actual implemented system using abstractions to capture the essential properties of the system.

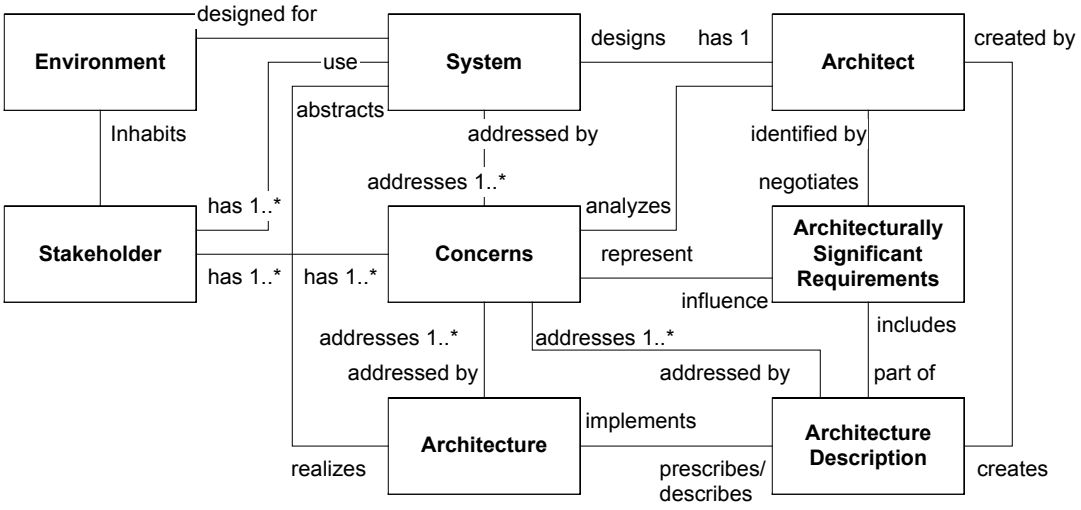

Figure 4-1. Architecture Design Context Conceptual Model (UML class diagram).

4.2 Main Concepts of Software Architecture for Software Architecture Review

When discussing architecture review we need to use a model of what the architecture is – a conceptual framework for software architecture. A software architecture is ***a set of concepts and design decisions about structure and texture of software that must be made prior to concurrent engineering (i.e., implementation) to enable effective satisfaction of architecturally significant, explicit functional and quality requirements, along with implicit requirements of the problem and the solution domains***.

For the sake of concreteness, in this section we summarize the main concepts of a software architecture conceptual framework as presented in [Ran2000]. While there are other possible models of software architecture, we use the one described below as an example that satisfies our goals. The conceptual framework consists of the eight elements that follow.

4.2.1 Scope

In most circumstances an architecture is intended for more than one instance of a system. The architectural scope characterizes the class of systems for which the architecture is applicable. There is usually a hierarchy of scopes that play an important role in an application domain, a corporation, a company, a product division, etc. One should create a software architecture for a specific architectural scope, consistently extending the architecture from its enclosing scope and providing context and constraints on the architectures created in its sub-scopes.

4.2.2 Concerns

A software architecture can only play an essential (and independent) role in software development if it addresses specific system lifecycle concerns that are not addressed by other software development functions and activities. These concerns may not necessarily be evident from system requirement documents or any other system descriptions. The architect should identify and document architectural concerns and their owners (i.e., stakeholders).

4.2.3 Requirements

The architect should refine architectural concerns into architecturally significant requirements (ASR) that are specific in terms of desired system properties. Furthermore, the architect should define how achieving these properties influences or constrains the architecture.

4.2.4 Component Domains

Software exists in multiple forms (e.g., source or object code components, executable components, executing components). Each kind of components forms its own component domain. There are relationships, but oftentimes there is no direct correspondence between components in different component domains. The component partition in each component domain addresses different ASRs. The architect identifies relevant component domains for a given system based on its architecturally significant requirements.

4.2.5 Structures

In each component domain a partition into components with their relationships forms an architectural structure. Components of architectural structures are concrete system elements (files, threads, executables, source code modules). Different architectural structures may be interdependent, however they are not different views of the same entity.

4.2.6 Views

Architectural views are models of architectural structures or other elements of a software architecture. Architects use views throughout the architecture lifecycle for the specific purpose of understanding and analyzing the different aspects of system development and performance. The major characteristic of a view is its purpose and utility. Views are not complete descriptions of the software architecture. Oftentimes views omit otherwise important architectural information in order to be useful for their specific purpose. The major issue with the views is consistency with the system rather than completeness. Architectural views are a necessary part of any architecture and serve as a way to design architectural structures and to understand different aspects of system design.

4.2.7 Texture

Certain design decisions that are only visible within relatively fine-grained components are nevertheless extremely expensive to revise. Consequently such decisions are architecturally significant even though they are concerned with fine-grained elements. This happens when the implementation of the decision cannot be localized, but must be replicated creating recurring uniform microstructure or texture. The texture of software is created by recurring (uniform) microstructure of its components. There are certain aspects of software functionality that are hard to localize using common programming languages and techniques. Such functionality cannot be implemented once and then used in different components; rather it must be implemented multiple times. This fact raises the importance of choices and consistency in implementation of such functionality so that it becomes a major part of software architecture. Decisions that affect texture of software have significant impact on the system and they are as hard to revise as decisions regarding the structure. Consistency of the texture is very often a problem since the decisions appear to be local to a component. It is not easy to identify the common concerns present in the implementation of different components without concentrating on the texture in the architecture of software.

Well-designed software has consistent texture. Software components need to observe policies for security, flow-control, overload control, fault detection and handling; they must rely on infrastructure for communication, coordination, state maintenance, execution tracing, etc. In order to achieve consistency in component design the architecture should provide the necessary information. Examples of software texture that must be designed and regulated by rules and standards include uniform component model realization, error reporting, exception identification and handling, and execution tracing mechanisms.

4.2.8 Concepts

From the perspective that considers software architecture to be an approach to dealing with complexity, the most important architectural decision is the selection of concepts used to reason about the system. The existence of architectural concepts is often taken for granted. However architectural concepts are not found in the application domain or in the implementation domain, but need to be invented in order to simplify design, construction, and representation of complex software. All other parts of a software architecture directly depend on the invention (selection) of architectural concepts.

According to this model the purpose of an architecture is to enable satisfaction of architecturally significant requirements. The content of an architecture is a set of concepts and design decisions about the structure and texture of the software – the architecturally significant decisions (**ASD**). The architect must make these architecturally significant decisions prior to concurrent engineering/implementation because they influence many design decisions associated with every component.

The purpose of an architecture review is to understand the impact of every architecturally significant decision (ASD) on every architecturally significant requirement (ASR).

In an ideal world, ASD and ASR are clearly documented including their relationships and cross influences. In the real world this is seldom the case. Therefore architecture reviews need to utilize available input to reconstruct this information. The following sections provide some information useful for identifying ASR and ASD.

4.3 Finding and Structuring ASR

It is generally accepted that a software architecture directly affects system-wide properties such as availability, reliability, security, etc. Well-structured software architectures also support requirements for change, reusability, interoperability with other systems, etc. Of course, if all different requirements were supported by the same architectural structure, it would be impossible to satisfy them independently. For example, requirements concerning performance and reliability interact since software execution structure affects both kinds of properties.

Often system requirements may be grouped so that requirements in different groups may be addressed by different (and at least partly independent) software structures established by partitions of software in different component domains. Such partitions exist simultaneously and often are independent of each other. For example, the architect can address:

- **performance requirements** by partitioning software into execution threads of varying priority (or utility), specifying thread scheduling policies, regulating use of shared resources, etc.,
- **change and reuse requirements** by partitioning software into modules—substitutable, unit-testable components having well-defined boundaries, predictable interaction with the environment, and minimal, well-specified dependencies on other modules,
- **portability requirements** by defining software layers and establishing conformance of layers and their interfaces to existing standards, and
- **requirements for independent re-start or independent failure modes** by partitioning the software into a set of separately loadable and executable processes.

The architect must group architecturally significant requirements so that requirements in different groups may be satisfied independently, while requirements within each group may interact and even conflict.

A good rule of thumb for finding independent requirements is to group them by the software lifecycle interval/phase they address. For example, very often requirements that address software development and change can be satisfied almost independently from requirements that address run time behavior, or software upgrade. Some of the lifecycle times on can consider are:

- write (or design) time,
- build time,
- configuration time (when software is configured for delivery on a specific platform),
- upgrade time,
- start-up time (initialization),
- run-time, or
- shutdown time.

These are "significant intervals" in the lifecycle of a software system. The number of architecturally significant requirements related to each of these significant intervals should be small (i.e., from one to three). The architect may need to order the ASR in each group by significance. The focus of an architecture review should be limited to one, or at most two, significant intervals and thus only ASR related to these intervals need to be considered.

Note that stakeholders have concerns with respect to the product, not only the software, and these concerns need be translated into architecturally significant requirements. See [Jazayeri 2000], section 4.4.5.

4.4 Finding the Architecturally Significant Decisions (ASDs)

Examples of well-understood and documented ASD are difficult to find. Current industry practices at best provide documentation for the system under review in terms of a few architectural models and facts about selected platforms and technologies that reflect multiple decisions and trade-offs. In most cases it is the task of the review team to understand what ASD have been actually made. This task cannot be undertaken in its entirety by the review team during the time of a typical, 3-day architecture review. Therefore they must scope the discovery of ASD according to the focus and objectives of the review. Fortunately, if they have identified and structured ASR as suggested in the previous section then they can provide some suggestions on the scoping of the ASD discovery.

It is important to note that once the review's focus is confined to a given significant interval, the review team only needs to consider the architecture in the corresponding component domain. Models found in the architectural documents may represent system concepts, structure, and texture in the relevant component domain. If models are missing, then the review team may need to reconstructed them and identify the ASD implied by the models. This process is illustrated in Figure 4-2.

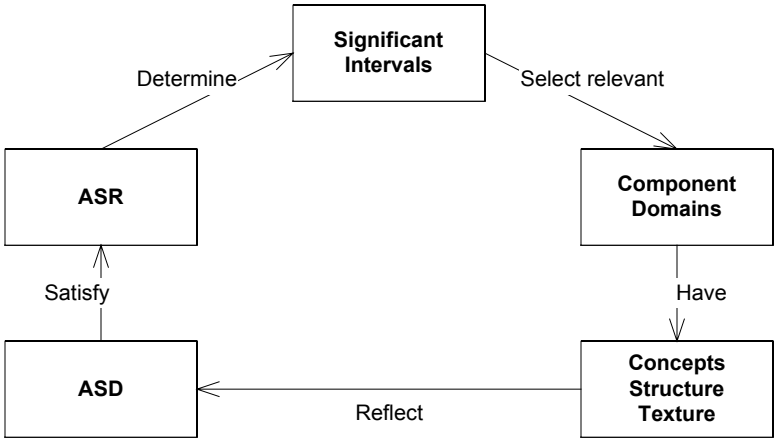

Figure 4-2. Selecting ASR - Discovering ASD.

For example the focus of an architecture review can be on some aspect of performance, such as capacity. The ASR of interest could be defined in terms of the number of serviced requests per unit of time. The system behavior in high workload situations must be specified. The following questions could then be answered:

- What should happen when the rate of actual requests falls far below system capacity?
- What should happen when the actual rate of requests exceeds the maximum capacity?

These ASR are related to the run-time interval of the system lifecycle therefore the review team needs to analyze only the execution component domain. Models of the system's execution structure should represent threads, shared resources, and schedulers of shared resources. Of specific interest may be the probability distributions for incoming requests (events), the sizes of allocated queues, message sequences associated with each event, estimated time of processing of messages by different threads, etc. In this case, the relevant ASD would include queue and thread allocation and release policies and specific load monitoring and overload control patterns employed in the system.

In some cases the review team may find documented decisions that have definite architectural implications but are too broad in scope to be easily analyzed. For example, the selection of specific technology such as CORBA or EJB has profound implications on the architecture and on the ASR. In order to properly analyze the effect of these decisions on the ASR the review team must determine their implication in different component domains by considering them separately. Eventually, the review team should synthesize the results of this analyses and make a conclusion regarding the value of the source decision.

4.5 Software Architecture Review Reference Model

A software architecture review is an activity to develop an assessment of an architecture (see Figure 4-3). The assessment is made against one or more review objectives. The result of a review is an assessment (report) and other outcomes. Since an architecture is itself an abstract entity, the review is conducted based on concrete architecture artifacts representing the architecture, possibly including an explicitly prepared architectural description document. The review may use other relevant input such as: business cases, stakeholders' concerns, standards, requirements, etc. In the course of a review, the review participants may execute one or more methods or techniques.

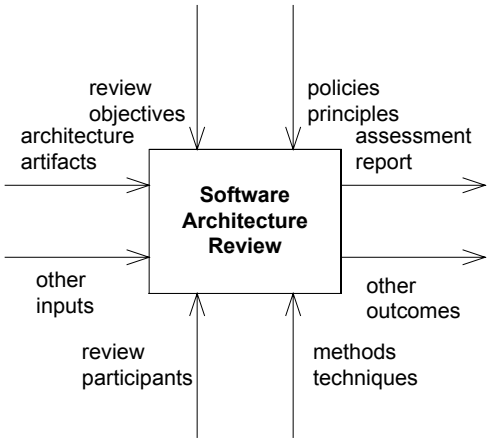

Figure 4-3. Software Architecture Review (Activity View in IDEF0**).**

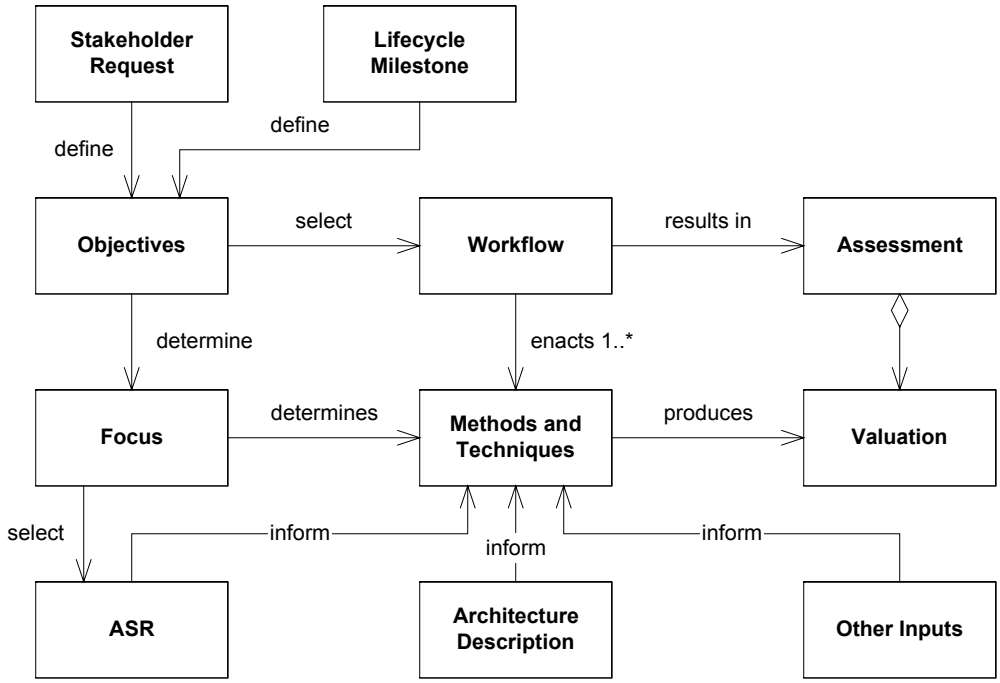

Figure 4-4. Software Architecture Review (workflow context diagram).

The architecture review objectives are the reasons for doing a specific review. The objectives of a review are driven by architectural concerns of the stakeholders. These reasons fall into several categories:

- certifying the conformance to some standard,
- assessing the quality of the architecture,
- identifying opportunities for improvement, and
- improving communication between stakeholders.

The review objectives may also be defined by the lifecycle stage of the project. Figure 4-4 shows that review objectives are determined either by the stakeholders-specified objectives or by "typical" lifecycle objectives.

Review objectives focus on specific aspects of an architecture. These aspects can include:

- the fit of the architecture to the problem or mission statement,
- the partitioning of system responsibilities to subsystems and components,
- the specific qualities (i.e., scalability, performance, etc.) to be architecturally controlled,
- the partitioning of the architectural design responsibilities,
- the identification of skills to implement the system,
- the verification of scenarios representing the critical functionality of the system, and
- the overall feasibility and specific risks of the architecture.

The review's goal is to select a subset of ASR to be considered. The review focus determines which methods and techniques may be applied to discover the relevant ASD based on available architecture descriptions and other input and to evaluate the ASD against the ASR. As a result, valuations of ASD are produced that are collected in the assessment.

The artifacts may be categorized as problem-oriented and solution-oriented. Problem-oriented artifacts will capture items such as business goals, standards, constraints, vision, and priorities. The review team uses these inputs to identify and structure the ASR.

The solution-oriented artifacts will contain items such as: principles and styles used, technologies selected, platform, infrastructure, tradeoffs considered and made, sensitivities, deferrals, models, and analyses. The review team uses these inputs to identify and structure the ASD.

The workflow enacts one or more methods/techniques to address the review objectives. Section 7 of this document describes various methods and techniques for software architecture review (note that different methods are appropriate for meeting different objectives).

A method or technique establishes a set of criteria that define a concrete means of judging whether the artifacts, and thus the architecture, meet a particular objective. Selection of these criteria follows from the refinement of the review objectives, relative to the particular type of assessment, and associated stakeholders. A method provides a way of analyzing particular artifacts with respect to certain criteria and leads to results for these criteria (valuations). The review team may aggregate or otherwise incorporate these results from the method into the architecture assessment (report).

Table 4-1 summarizes how architecture review focus differs according to the particular lifecycle phase. Table 4-2 relates different classes of stakeholders to review focus typical to the class.

Note that the tables could be merged. The method and the concerns are of course the same. In the actual review, both lifecycle and stakeholders are relevant.

Lifecycle Activity	Architecture Review Objectives
Defining Scope	– High level requirements for domain – Technological feasibility within domain
Inception	– Detailed requirements for class of systems in the domain – Identification of architectural drivers
Elaboration	– Architectural satisfaction of requirements
Construction	– Design – Design conformance to architecture
Deployment	– Running system
Evolution	– All of the above

Table 4-1. Lifecycle-initiated reviews.

Stakeholder Types	Architecture Review Focus
End User	Usability, functionality, customizability, performance, reliability
Customer (Client, Sponsor, Owner)	Price, support and maintenance cost, features, schedule, on-time delivery, stability and maintainability, cost of ownership, etc
Marketing and Product manager	Price, time-to-market, availability, extensibility, competitive features, support for long-term company strategy
Developer (Designer, tester)	Understandability, clearly stated requirements, traceability, testability, etc
Component Vendors (Supplier, contractor)	Interface and integration rules
Sales and field Personnel	Price, time-to-market, competitive features, ease of installation, ease of integration, ease of diagnosing
Project Manager	Work partitioning, localization of complexity, schedule, budget, resources, contracting
Maintainer	System structure, quality of documentation, consistent use of patterns and frameworks
System Administrator (operator)	Maintainability, operational concepts and procedures
Architect	Consistency, clarity of concepts

Table 4-2. Stakeholder-initiated review concerns.

5 Review Inputs

5.1 Introduction

This section focuses on the identification of information needed as input for an architecture review / assessment. This section is structured according to logical aspects and not according to a typical structure of related documentation. The following areas are discussed (also see section 1.2 and Figures 4-1-2):

- review objectives,
- review scope,
- architectural artifacts, and
- other relevant input.

Based on current practice, the initial input most likely will be incomplete at the start of the review and, therefore, will have to be completed during the review process. The most important inputs are the architecturally significant requirements (ASR) and the architectural descriptions.

5.2 Review Objectives

In general, there are several triggers or reasons for initiating an architecture review (e.g., poor confidence by the management in the technical concepts the architecture is based on). These triggers serve as the basis for formulating the review objectives.

In general there are two sources for review objective:

1. concerns of the system's stakeholders (that got mapped to the resulting architecture) and
2. standard process milestones (i.e., a review has to be conducted during a certain stage of the project in its lifecycle (See lifecycle-triggered reviews below)).

Review objectives explicitly describe the architecture review goals. In addition, the objectives help reviewers to identify the assessment criteria and the appropriate review method.

Review objectives fall into the following categories:

- **Certifying conformance to some standard**:
  - Does the architecture fulfill the constraints and requirements of the relevant standards?
- **Assessing the quality of the architecture:**
  - Does the architecture fit to the problem or mission statement?
  - Is the architecture a suitable basis for fulfilling the present and future requirements?
  - Can specific qualities (e.g., scalability, performance. etc.) be architecturally controlled?
  - Are there open issues that have to be clarified?
  - Can the architecture be implemented using existing skills?
  - Which new skills are needed to implement the system?
  - Does the architecture support the realization of scenarios representing the critical functionality of the system?
  - Is the realization of the architecture and its underlying technical concepts feasible?
  - What are specific technical risks of the architecture?
  - How are the responsibilities of the architecture partitioned?
- **Identifying of opportunities for improvement:**
  - Which design decisions should be revised in order to improve the architecture?
  - Which improvement measures should be taken?
- **Improving communication between stakeholders.**

5.3 Review Scope

The review scope represents the subject of review (e.g., what exactly will be reviewed) and clearly should be defined and agreed with the principal stakeholders before the review starts. Relevant questions that should be answered in defining the scope include:

- Does the review involve the whole system, part of the system, only the software of the system, only part of the software of the system, etc?
- Does the review involve all stakeholders or only a subset of the stakeholders?
- Does the review determine the satisfaction of all ASRs or only a subset (like the most critical)?
- Does the review involve all architectural views or only a subset?
- Does the review involve all architectural decisions or only a subset (like the most critical)?
- Will the architecture description also be subjected to the review?

5.4 Architectural Artifacts

5.4.1 System Description

The system description provides a high-level overview of the system and its context. It provides the review team members with an understanding of the system to be developed. It is typically architecture independent and takes the form of a problem statement (e.g., What is the system expected to provide?) not a solution statement (e.g., How does the system realize the needed functionality?).

Typically, it includes the following aspects:

- **Business opportunity and problem statement:**
  - What are the benefits of the system with regard to the developing organization and with regard to its customers?
  - What is the related product strategy?
- **Users:**
  - Which users will interact with the software system?
  - What is their background?
  - What are their skills?
  - What are the key user needs?
- **Overview of system capabilities:**
  - On a very high level, what are the product's capabilities?
  - With which other applications should it cooperate?
  - What are the external interfaces?
  - What are the main use cases to be supported by the system?

5.4.2 Architecture Descriptions

Architecture descriptions are the key input to an architecture review. They are the basis for identifying:

- key technical concepts underlying the architecture and
- major design decisions.

If an architecture description is unavailable or insufficient, then the necessary information has to be recovered and collected during the review itself. This may have to be done iteratively.

There are many different ways to document architectures and related architectural decisions. This can be done in the form of diagrams, text, etc. See [IBM 99], [HP 00], [Kruchten 95] and [Hofmeister 99] for examples of architecture descriptions or appropriate notations.

One approach for specifying architectures is to use the concept of viewpoints to separate various architectural aspects or concerns (see [RM-ODP 1995]). A viewpoint is a system representation with explicit emphasis on a specific concern or set of concerns. Viewpoints help to structure the architecture design process by separating architectural concerns (see Section 4.2 – Views).

The review team can use viewpoints during the review process to discuss and evaluate a specific concern. A viewpoint helps to organize the information needed and is typically composed from many different types of architectural artifacts.

For example, to address a performance concern, a performance viewpoint would be taken. It could be composed from several artifacts such as a hardware depiction, the processes and threads that run on the hardware, the priorities assigned to the processes, the arrival rates of messages or events to those processes, and timing requirements such as deadlines and execution times. Based on this information, one could perform analysis regarding scheduling, utilization, throughput, etc.

5.4.3 Architectural Decisions

Architectural descriptions cover the current state of the software architecture. They cannot communicate the decision process that created the architecture. A collection of these decisions is required as input and is part of what is reviewed.

5.4.4 Reused Solutions

Due to the importance of and interest in software architectures, a number of sources describe "typical" architectural solutions that have become well established in the software engineering community. Examples include:

- patterns [Buschman 96],
- styles [Shaw 96], and
- reference architectures [RM-ODP 1995], [TINA 1995].

Reuse on the architectural level means the usage of these well-known concepts that cover technical software architectures best practices. They help practitioners understand the architecture because they provide standardized and well-known concepts and use standardized terminology. They therefore support the review process (e.g., the typical pro's and con's of patterns are well-known and along with these patterns there may be a set of standard analyses that can be reused [Klein, Kazman 1999]).

5.4.5 Guidelines and Rules

In addition to the aspects covered in the architecture descriptions, the architecture design process defines guidelines and rules (e.g., common usage rules, principles, concepts, and conventions that must be followed during the design and implementation of the software system). Typically, the rules and guidelines are not related to one specific viewpoint, aspect or concern. Possible areas concerned include:

- Architecture-relevant design and coding rules, e.g.:
  - naming conventions for components, objects, interfaces, etc.,
  - calling conventions for interfaces,
  - formats of messages and persistent data,
  - debugging information,
  - exception handling, and
  - access rights for components and objects.
- Guidelines for using mechanisms and services, e.g.:
  - system-wide mechanisms for communication to be followed (e.g., mailbox, shared memory).
- Database layout and storage layout
- Test environment, conventions and rules in order to ensure testability of the system

- Standards (international, application-specific, in-house, etc.)
- Documentation types and formats
- Architecture evolution steps
- Proactive modification guidelines
- Architecture governance process guidelines

5.4.6 Architecture Supporting Evidence

During the architecture design process the review team will develop other work products to support the architecture review (i.e., to form an architects "notebook"). These work products may include:

- **feasibility studies**
  - Technical documents discussing whether some requirements can be realized (under which constraints) or whether specific architectural concepts or technical approaches can be used to realize the requirements
- **prototypes**
  - Preliminary implementations used to check whether a specific architectural concept or technical approach would be able to fulfill given requirements (regarding, e.g., performance, robustness etc.)
- **minutes, notes, white papers**
  - Partly, these documents record the design decision process and cover the design rationale (i.e., technical alternatives for the actual solution chosen in the architecture, criteria which have been used to make decisions, Pro's and Con's of the current solution)
- **measurements**

5.5 Other Relevant Inputs

5.5.1 Architecturally Significant Requirements (ASRs)

Stakeholders have concerns beyond the software. All areas (e.g., product strategy, requirements, standards and constraints, Q/A policies, etc.) form the basis for the evaluation criteria. Because not all these concerns are relevant on an architectural level, we need to extract the issues that have a significant bearing on architectural decisions. As introduced in Section 4.3, these requirements are called Architecturally Significant Requirements (ASRs) and they are derived from the existing concerns.

The following two criteria help to identify ASRs:

1. If the realization of a requirement involves one or more of the other components or the system as a whole, then it is an ASR
2. If a requirement concerns only a single component, then it is not an ASR.

Typically, ASRs are not stated explicitly in the requirement documentation (assuming that the requirements are documented at all) and the review process will need to make them explicit. ASRs are used as the criteria for the assessment and review process.

5.5.2 Product Strategy and Product Planning

A product strategy clarifies for a certain product or product family what role or position it is intended to have in the market and how it compares in the long-term to competitors products. This includes aspects such as:

- **location of competition**: which market does the product address (e.g., niche vs. core market),
- **rules of competition**: according to which rules are competitors being approached (e.g., change vs. adaptation), and
- **market direction**: what is the focus of the competition (e.g., cost leadership vs. performance leadership).

A product strategy is developed on the basis of a market analysis. This includes an understanding of:

- customers and their needs,
- main competitors, their products, and the strength and weaknesses of their products, and
- ones own current position in the market compared to the competition.

Product planning is done on the basis of the product strategy or as part of it. It includes:

- **roadmap development** (e.g., planning of products, releases / versions and variants) and
- **economic issues** (e.g., cost/benefit analysis, pricing, number of units produced, production costs, cost of R&D).

Product strategy and planning result in long term architectural requirements. They are important inputs for the architecture review and assessment process since they describe the circumstances under which the product is developed. They also form a basis for identifying the evaluation assessment criteria.

5.5.3 Requirements

A requirement is (see [IEEE Std 610.12.1990])

- "***A condition or capability needed by a user to solve a problem or achieve an objective.***"
- "***A condition or capability that must be met or possessed by a system or a system component to satisfy a contract, standard, specification, or other formally imposed documents.***"

[IEEE Std 610.12.1990] distinguishes the following types of requirements:

- **Functional Requirement:** It "***specifies a function that a system or system component must be able to perform, without taking physical constraints into consideration***." A functional requirement specifies the services a system provides and the input and output behavior of a system. It includes feature sets, capabilities, etc.,
- **Design Requirement:** It "***specifies or constrains the design of a system or system component.***"
- **Interface Requirement:** It "***specifies an external item with which a system or system component must interact, or that sets forth constrains on formats, timing, or other factors caused by such an interaction.***"
- **Implementation Requirement:** It "***specifies or constrains the coding or construction of a system or system component***."
- **Performance Requirement:** It "***imposes conditions on a functional requirement; for example, a requirement that specifies the speed, accuracy, or memory usage with which a given function must be performed***." It is a typical example of a nonfunctional requirement.
- **Physical Requirement:** It "***specifies a physical characteristic that a system or system component must possess; for example, material, shape, weight***."

Other types of requirements include:

- **Usability Requirement:** e.g., human factors, aesthetics, user interface, manuals, training, and
- **Reliability Requirement:** e.g., frequency and severity of failure

Requirements are a basic input for defining evaluation criteria. The main objective of an architecture is to define the technical concepts regarding how the system implementation fulfills these requirements.

5.5.4 Standards and Constraints

As a separate subclass of requirements, the architecture may have to support or be compliant with standards. Examples include:

- national rules, e.g., formulated by the FDA (Food and Drug Administration), TÜV (Technischer ÜberwachungsVerein in Germany),
- company standards,
- legal constraints, and
- IEEE, ISO, standards.

The architecture review team may also have to consider other pragmatic during architectural design (even though the stakeholders do not always explicitly state them). These include:

- compliance and compatibility with legacy systems,
- asset protection (e.g., the system under development and its architecture must not force the customer or development team to replace all of the existing infrastructure),
- existing core competencies of the organization and know-how of its people (e.g., the architecture should not revolutionize development and product technology in too many areas),
- political issues (e.g., compliance to ecological standards, they are not always obligatory, but they might serve as marketing arguments), and
- multi-site development (e.g., geographically distributed development teams).

5.5.5 Quality Assurance Policies

In some cases an architecture needs to take into account an organization's Quality Assurance policies (e.g., development naming conventions, design and implementation rules, and commitments for assuring quality. Some examples include:

- integration test strategies that an architecture has to support (e.g., by providing monitoring features) and
- quality metrics and goals the architecture has to conform with.

5.5.6 Risk Assessment Artifacts

When a risk management process exists, then during the risk elicitation and evaluation phase, the architect will identify the risks associated with the architecture or the product technology. Typical risks include:

- **technological risks** (e.g., maturity of the technologies to be used) and
- **open issues** (e.g. ,areas where the consequences of a certain architectural concept cannot be overseen at the current status of development).

The architect should use this information, in addition to other measurements, in order to minimize the risks during the review and to complement the review results (e.g., the findings and recommendations).

6 Review Outcomes

6.1 Introduction

This section focuses on the identification of information that forms the outcome / output of an architecture review / assessment. There is tangible and intangible output. These outputs result from the review discussion that has mapped the ASRs to architecturally significant decisions (ASDs).

The primary outcome of the review and assessment process is the documentation of:

- architecture recommendation and findings,
- decisions taken during the review, and
- suggestions to improve the quality of review plan, methodology, etc.

In addition, the review produces intangible outcomes such as:

- improved communication between stakeholders,
- better understanding of the stakeholder concerns, and
    - better understanding of the limitations of the architecture.

All relevant points from the review need to be collected in an appropriate way. The review team should organize the resulting documentation in a way that supports the project workflow efficiently. All scheduled activities require unambiguous goals and information to base decisions on. In general, the outcome helps to improve the quality of the architecture in all of its dimensions and ensures the seamless integration of architecture and the relevant project and business goals. The main outcomes and results of the review are:

- the assessment report and
- additional review outcomes (e.g., executive summary, organizational issues, strengths and weaknesses).

The following sections elaborate on the content of these documents and work products.

6.2 Assessment Report

This is the main outcome of the review. An example of an assessment report is shown in the appendix. The subsections that follow provide and synopsis of the steps the review team should following in a “typical” review and assessment effort.

6.2.1 Objectives

Before the review begins, the team must agree on the review objectives and collect the necessary input material (see Section 5.2). If review objectives haven't been met entirely, the team must document a decision on how the ‘leftovers’ will be handled.

6.2.2 Scope

The review team will document the scope of the review (see Section 5.3).

6.2.3 Methodology

The review team will document the methodology that will be used to conduct the review (see Section 8).

6.2.4 Evaluation Criteria for Architecture

The review team will define and document a set of criteria based on the ASRs (see Section 5.5.1). Note that sometimes it is helpful for the review team to state what (well known) criteria have not been put on the list and for what reason. When opportunity exists, the review team should reuse existing, concise domain-specific sets of criteria that have proven valid in the past.

6.2.5 Employed Architectural Foundation and Approach

This section focuses on information gathering based on the methodology (Section 6.2.4) the review team has decided to use to map evaluation criteria to architectural artifacts and concepts (see Sections 5.4 and 5.5).

6.2.5.1 Identification of Architectural Concerns and Concepts

A software architecture is based on common principles and expert knowledge that stems both from the problem domain or is domain independent. The following areas should be supported by proven concepts when they are applicable to the architecture under review:

- system topology and scalability,
- basic system structure paradigms, e.g., n-tier architecture or components,
- communication and addressing issues,
- database issues,
- fault tolerance and redundancy,
- security issues, e.g., authentication, authorization, encryption,
- GUI issues, e.g., common look an feel, web technology, internationalization,
- real-time requirements,
- administration and monitoring of the system,
- support for versioning,
- support for configuration and extensibility,
- support for interoperability,
- support for portability and multi- programming language support, and
- integration of hardware and hardware dependencies.

6.2.5.2 Identification of Key Architecture Use Cases and Scenarios

Domain modeling captures use cases and scenarios. The available documentation should reflect all necessary use cases in order to:

- meet the requirements,
- satisfy stakeholders,
- support reuse of existing artifacts,
- support software analysis and design, and
- support test activities.

6.2.6 Architecture Analysis, Findings and Recommendations

This section forms the core part of the assessment report. According to the methodology selected by the review team, they will map the set of criteria (ASRs) to the set of architectural concepts and decisions that they identified. This results in a systematic evaluation of the current architecture and helps to set up an action plan for future improvement. The outcomes of this process are described in the subsections that follow

6.2.6.1 Strength and Issues

6.2.6.1.1 Strengths

The review team detects all employed best-practice issues. These issues are such that they

- form an architectural invariant the project wants and/or has to rely on,
- form a broader base for future architectural decisions,

- extend current guidelines, and
- communicate best practices to project team members.

6.2.6.1.2 Issues

The review team identifies issues that are subject to further investigation. They might cover enhancements in the following areas:

- business plan,
- process issues,
- requirements analysis,
- applied standards and technologies,
- design and implementation,
- test and monitoring, or
- human resource issues.

6.2.6.2 Other Findings

The review team may have discovered issues that haven't been evaluated because:

- documentation / information was missing,
- required experts were not available, or
- the issue was not to be taken into account at this early stage.

6.2.6.3 Recommendations

During team discussion, recommendations need to be collected that might help the project at a later date, or are optional in the given context.

6.2.6.4 Action Plan

For all findings, the review team needs to schedule actions and assign them to working groups. The results of the groups form new inputs to future review / assessment meetings and may result in a revision to the action plan. For recommendations, the relevant stakeholders must take appropriate actions conditional on other stakeholder's actions.

6.3 Additional Review Outcomes

There might be different additional outcomes according to the needs of supplier or customer. Typical examples are listed in the sections that follow.

6.3.1 Executive Summary

The review team can use the executive summary of the review as basis for creating a presentation of the most important results to the various stakeholders. Additional detailed information is found in the assessment report. The subsections that follow contain suggested elements for inclusion in the executive summary.

6.3.1.1 Organizational Issues

This portion of the executive summary could contain the following:

- a description of the project that was reviewed,
- some background information on the customer who owns the project,
- a list of the review participants, and
- the time period and schedule overview.

6.3.1.2 Objectives of the Analysis

This portion of the executive summary would contain a selection of the identified objectives.

6.3.1.3 Overview of the Investigated Architecture

This portion of the executive summary would show an overview of the product architecture..

6.3.1.4 Summary of the Result

This portion of the executive summary would highlight the findings.

6.3.1.5 Methodology of the Analysis

This portion of the executive summary would explain, in a condensed way, the review and assessment methodology the team used to conduct this review. It may also include a set of selected criteria and architectural concepts.

6.3.1.6 Detailed Results

This portion of the executive summary would present the condensed version of the analysis results.

6.3.1.7 Appendices

Assuming that the review has resulted in creating several more detailed documents, these documents can form appendices to the summary. Examples of such material could include:

- the set of evaluation criteria,
- the set of concepts that were analyzed, or
- the evaluation metrics and matrices.

6.3.2 Architecture Review Plan Update and Lessons Learned

This final output of the review process is strategically important for the success of future review and assessment efforts. During the review process, the participants will gather experience on which parts of the process, or what types of artifacts are or are not effective. As a result, the team should recommend changes and updates to the review plan. The detailed description of the review plan can be found in the workflow section (Section 7).

Improvements might be possible in the following areas:

- definition of the review approach,
- evaluation of the applied methodology,
- identification of the review work breakdown structure (agenda),
- establishment of the review staffing plan (team and customer),
- identification of review objectives,
- identification of key stakeholders (system and review),
- definition of the review cost estimates,
- establishment of the review logistic requirements, and
- determination of information need requests.

Other review plan improvements might be added at this point in time. Also, the stakeholders might want to list additional information that was specific for that review, like:

- organization data,
- production environment, and
- development process.

7 Architecture Review Workflow

This section describes a typical workflow for an architecture review. We assume that a review is a relatively short effort (three days or less) with the following underlying principles:

- the key objective of the review is to demonstrate that the architecture addresses (or does not address) a set of architecturally significant requirements (ASRs),
- the ASRs must be understood and explicitly stated for the review to succeed, and
- the ASRs are compared with (or mapped into) an architectural description of the system that is sufficient for reasoning about the expected system's behavior and properties.

We do not assume that the ASR list and the architecture description are complete before starting the comparison (analyzing them against each other). They may be created or discovered concurrently and iteratively as the review progresses. This, however, does not change the principle assumption, that we are comparing explicitly stated ASRs against an explicit architecture description.

The length of the review will depend on its type (lifecycle or stakeholder-initiated), the size of the system, and if it is the first review or a follow-on review. However, we assume that a review should take no more than a few days.

We have divided the review activities into three groups (or phases):

1. review inception,
2. review, and
3. post review.

The first phase results in the agreement on the review scope, cost, duration, participants, etc. The second phase is the above-mentioned, iterative process of discovering, capturing and comparing ASRs and the architecture description. The last phase concentrates on summarizing and communicating the finding, as well as improving the review techniques and methods.

We use a simple template to describe the review activities. The template captures:

1. activity name,
2. activity objectives,
3. input to the activity,
4. applicable methods and techniques,
5. outcome of the activity, and
6. roles/workers involved in executing the activity.

Inputs and outcomes to/of a review are described in Sections 5 and 6 respectively. Activity templates list examples of inputs and outcomes, but are not exhaustive. Each activity template also suggests methods and techniques that can be used to perform the activity. General description of review methods and techniques is provided in the following section (Section 8). Finally, each activity template suggests the roles/workers involved in the activity. We recognize the following roles/workers:

- architecture review team (ART) lead,
- architecture owner,
- ART member,
- project owner,
- project team lead/manager,
- review requester,
- lead architect, and
- product manager.

7.1 Inception Activities

The following subsections describe the activities the ART must engage in in order to generate a review plan and secure management resource commitment to execute the review.

7.1.1 Identify Type of Review and Its Business Objectives

Activity Name	**Identify type of review and its business objectives**
Objectives	Identify what type of review has been requested and/or is required. Identify the review goals and issues, if any that need be addressed? Try to qualify the review in terms of business objectives and benefits.

Input	Development schedule, process documentation, Quality Assurance policies, stakeholders' requests, review type selection guide, business plan, business case, road map, project/system issues and risk assessment artifacts
Methods/Techniques	Interviews, moderated discussions, business documentation analysis, or use of some pre-defined questionnaire
Outputs	Prioritized list of business objectives
Roles/Workers	Line management, project owner, product manager, architecture owner, ART lead, and review requester

7.1.2 Identify Key Stakeholders and Review Scope

Activity Name	**Identify key stakeholders and review scope**
Objectives	Understand who the key project stakeholders are. Understand what part(s) of the system needs be reviewed. Understand the types of work-products and deliverables that we have to be reviewed/analyzed. Define the set of qualities to be reviewed (e.g., performance, cost, understandability, etc.).
Input	Project/product line structure overview, organization charts with skills and competencies, stakeholders' proposal (proposed statement for work), review history, schedules, budgets, type of review, etc.
Methods/Techniques	Questionnaires, interviews, checklists
Outputs	Names of subsystem(s) that will be the subject of the review, key stakeholders list
Roles/Workers	ART lead, project team lead, lead architect

7.1.3 Identify (Initial) Set of Detailed Review Objectives

Activity Name	**Identify (initial) set of detailed review objectives**
Objectives	Create a specific list of review objectives (hypotheses) and information needs that can be handed to the review team for investigation.
Input	Review business objectives, review type, review scope, system requirements and constraints, risks analysis
Methods/Techniques	Interviews, risk prioritization techniques, impact analysis techniques, issue-based analysis techniques, check-lists, …
Outputs	Initial list of architecture review objectives and information needs, initial list of deliverables
Roles/Workers	ART lead

7.1.4 Prepare Review Plan and Obtain Approval

Activity Name	**Prepare review plan and obtain approval**

Objectives	Propose and get approval for a review plan. Make sure that roles and responsibilities are assigned and committed to. Validate logistics (e.g., facilities, individual schedules, access to infrastructure like phones, printers, etc.).
Input	Deliverables list, schedules, scope of the review (in terms of subsystems to be reviewed), org charts with responsibilities, skill set descriptions, staffing plans, etc.
Methods/Techniques	Planning techniques and tools (e.g., MS Project), templates, interviews, etc.
Outputs	Cost estimate, review plan, presentation for management, agreement document (different from statement of work?), ART members list, what/when stakeholders are needed
Roles/Workers	ART lead, project owner, review initiating stakeholder(s)

7.2 Review Activities

Architecture review is a learning process. The ART identifies and captures ASRs, develops and refines the architecture description. and analyzes the description against the ASRs to prove that they can be met, which may lead to discovery of new ASRs, ASR reprioritization and architecture description refinements.

One would like to perform the activities described below in a sequence. However, the nature of the review described above indicates, that in most cases these activities will be performed iteratively.

7.2.1 Identify, Describe and Prioritize ASRs

Activity Name	**Identify, describe and prioritize ASRs**
Objectives	Create a prioritized list of ASRs that have been approved by key stakeholders
Input	Initial list of ASRs
Methods/Techniques	Requirements discovery methods. Requirement classification methods. Requirement prioritization methods
Outputs	Prioritized ASR table
Roles/Workers	ART Members, Project Owner, review initiating stakeholder(s)

7.2.2 Identify/develop architecture description

Activity Name	**Identify/develop architecture description**
Objectives	Identify and/or develop the architectural descriptions of the system that can be analyzed against the ASRs
Input	Review objectives and information needs, ASR list, architecture description(s), system implementation and prototypes, analytical models, etc.
Methods/Techniques	System demonstrations, interviews, code analysis, architecture documentation analysis, design recovery techniques
Outputs	Architecture description, refined architecture review objectives list, refined ASR list
Roles/Workers	ART members

7.2.3 Analyze Architecture Description Against ASRs

Activity Name	Analyze architecture description against ASRs
Objectives	For each review ASR find out how it has been addressed in the design. Evaluate architecture decisions and collect and organize preliminary findings. Refine architecture review objectives and ASRs if needed.
Input	Refined review objectives list, architecture descriptions, the system
Methods/Techniques	Issue driven analysis, scenario analysis, model-driven analysis, check list of detailed review questions, traceability analysis, etc.
Outputs	Preliminary findings: risks, issues, tradeoffs, sensitivity points
Roles/Workers	ART members

The above activity is the core of the review and it is exploratory and iterative in nature. The ART is analyzing the architecture "against" the stated ASRs. For example, if one of the objectives is to verify that the system will scale up to a certain level of throughput, the ART will be looking for design decisions, empirical evidence, or other ways of verifying that the system will indeed scale up.

7.3 Post-Review Activities

The following is a set of activities that conclude a review. The main objective of these activities is to summarize and present the findings. If possible, the ART should look back at the review to see if the review guidelines (like this workflow) could be improved.

7.3.1 Summarize Findings and Review Them with Architecture Owners

Activity Name	Summarize findings and review them with architecture owners.
Objectives	Create a preliminary summary of the review findings. Discuss the findings with the architecture owners before producing the review report.
Input	Preliminary findings
Methods/Techniques	
Outputs	Summary of the findings (strengths/weakness, issues, risks, tradeoffs), preliminary version of recommendations, outline of the action plan
Roles/Workers	ART members, lead architect, architecture owner

7.3.2 Present Review Report and Recommendations

Activity Name	**Present review report and recommendations.**
Objectives	Finalize the outcome of the review. Present the outcome to the architecture owners and stakeholders who requested the review. Review recommendations with the stakeholders.
Input	Summary of the findings (strengths/weakness, issues, risks, tradeoffs), preliminary version of recommendations, outline of the action plan
Methods/Techniques	Templates of reports and presentations
Outputs	Review report, review presentation, suggested corrective actions
Roles/Workers	ART lead, review requester, architecture owner, lead architect

7.3.3 Refine Review Methods

Activity Name	**Refine review methods.**
Objectives	Conduct a retrospective analysis of the review to see if there are opportunities to improve the applied process and techniques.
Input	ART observations and notes
Methods/Techniques	
Outputs	Improved review guidelines, techniques and methods
Roles/Workers	ART lead

8 Methods and Techniques

8.1 Introduction

To efficiently and completely address each step of an architecture review may require a specialized method. Each method is defined by: a set of steps (a process), an associated analytic technique, a notation, a set of outputs (work products), and a set of roles for the participants. These are ideally associated with estimates of cost and time. This section addresses the specific techniques needed to produce the desired outputs from each of the steps of the review process.

This section collects the techniques that support the activities described in the Workflow section (Section 7) and are based on the actual experiences of the authors. The basic idea is that for every ASR a particular method or technique can be identified to assess whether the architecture under review satisfies that ASR or not.

8.2 Template for Describing a Method or Technique

All architecture review methods and techniques described in this section use the template show in Table 8-1. In addition to the techniques that follows, the ART should consider using any applicable techniques for holding meetings or interviews, e.g., techniques to handle difficult people, to elicit input, to get closure on an issue, etc. These techniques can be found in [Freedman, Weinberg 1990] and [Gilb 1993]. The individual relevant methods and techniques are described in section 7.3.

Name	A succinct name given to the method or technique
Context	In which circumstances would you invoke (execute, use) the technique; which activities of the architecture review workflow does it support? What problem does it help solve?
Purpose	What does the technique achieve? What additional insight does it provide? What intermediary artifact does it produce?
Input	What are the artifacts that the technique uses?
Output	What are the results of applying the technique? What artifact does it produces or updates, and how do you interpret theses results?
Steps	What are the a series of steps or workflow for this method (if it is a complex technique)?
Roles	Who are the participants?
Estimates	What is the estimated effort to apply the technique?
Reference	Where has this technique been published/described?
Tools	What tools support this technique?
Alternative	What other technique could used for similar purpose?

Table 8-1: Template for Method and Techniques for Architecture Analysis

There are review process techniques, related to the preparation, conduct and conclusion of the review, and techniques related to the evaluation of issues related to certain classes of ASRs, which can be organized following the rules and guidelines found in Section 4.3: design-time, build-time, run-time, start-up and shutdown time, upgrade time, configuration-time. We have identified the following techniques:

Review process methods and techniques:

- individual interviews,
- critical scenario,
- change case,

- check list, and
- probing about alternatives.

ASR-related methods and techniques:

- Rate Monotonic Analysis for schedulability (run-time),
- module structure analysis (design time, build time),
- queuing system evaluation

The following table lists the architectural concerns for which we are seeking evaluation methods and techniques.

CATEGORY	ARCHITECUTURAL CONCERN	DESCRIPTION	I	M	H
Creatability	Ease of creation	Degree of effort required to create the system according to the stated requirements			
	Outsourceability	Degree to which the implementation of parts of the system can be outsourced			
	Buy-in	Degree to which existing components can be applied in the system			
	Conformance	Adherence to technology or industrial standards for product (process)			
	Manufacturability	To be manufactured with low cost, high throughput, low drop out, etc.			
	Environment impact	Regarding manufacturing, life time, recycling costs, disposable consumption, power consumption			
Functionality	Suitability	Providing an appropriate set of functions for specified tasks and user objectives			
	Interoperability	Interaction with one or more specified systems			
	Security	Detection and prevention of unauthorized access (accidental or deliberate) to programs or data			
	Compliance	Adherence to application related standards or conventions or regulations to laws			
	Integrability	Degree to which components of the system can be easily integrated			
	Configurability	Adaptation the system to different needs			
	Compatibility	Of the system with earlier or future systems			
Reliability	Correctness	Degree to which the system conforms to the stated requirements			
	Accuracy	Providing the right or agreed results or effects (including data with needed degree of precision)			
	Availability	Degree to which the system is available to the user on the time it is needed			
	Fault tolerance	Maintaining a specified level of performance in case of system failures or infringement of its interface			
	Recoverability	Re-establishing the level of performance and recover data directly affected in the case of a failure			
	Safety	Absence of unsafe system conditions that could lead to loss of life or liability, or damage to property			

Usability	Understandability	Enabling the user to understand whether the system is suitable, and how it can be used for particular tasks			
	Learnability	Enabling the user to learn the system's application			
	Operability	Enabling the user to operate and control the system (required effort)			
	Explicitness	Clarity of the system with regard to its status			
	Responsiveness	Of the system regarding reaction according to user expectations (feedback during processing)			
	Customizability	Enabling the system to be customized by the user to reduce effort needed for use and increase satisfaction			
	Clarity	Clarity of making the user aware of the functions the system can perform			
	Helpfulness	Availability of instructions for the user on how to interact with the system			
	Attractiveness	To be liked by the user			
Efficiency	Time behavior	Providing appropriate response/processing times and throughput rates (no degeneration over time)			
	Resource utilization	Using appropriate resources in an appropriate time when performing the functions (memory, comm.)			
Maintainability	Analyzability	To be diagnosed for deficiencies or causes of failures, or for the parts to be modified to be identified			
	Correctability	Enabling an identified fault to be removed			
	Expandability	Increasing the system's functionality or performance to meet new needs			
	Stability	Minimizing unexpected effects from modifications of the system			
	Testability	Enabling the developed or modified system to be validated			
	Scalability	Supporting modifications that strongly increase the system's internal capacity (same functionality)			
	Serviceability	Servicing the system in its operating environment (ease, effort)			
Portability	Adaptability	To be modified for different specified environments (including hardware/software independence)			
	Co-existence	Co-existing with other independent software in a common environment sharing common resources			
	Installability	Of the system to be initially installed, set up, calibrated, etc. in a specified environment			
	Upgradability	To be upgraded (with new functions, releases, etc.) in the system's operating environment			
	Replaceability	To be used in the place of specified other system (parts) in the environment of that system			
	Reusability	To be complete or partially reused in another system			

8.3 Inventory of Techniques

8.3.1 Individual Interviews

Name	**Individual interviews**
Context	Use this techniques to gather information in a politically or emotionally difficult context, such as potential issues.
Purpose	Information gathering without open conflict or blockage
Input	Depends on the information being gathered
Output	Depends on the information being gathered
Steps	1. Define the scope of the interview. 2. Build a short initial questionnaire. 3. Identify key individuals to interview and make appointments. 4. Conduct a first round of interviews. 5. Using pertinent information gathered with the first few interviewees, such as perceived issues, refine the questionnaire to focus on potential problem area to confirm; go to Step 3. 6. Consolidate results in a list of confirmed issues.
Roles	The reviewers, and selected interviewees (depending on nature of information gathered)
Estimates	Time consuming, indeed, but more effective than open war
Reference	

8.3.2 Critical Scenario

Name	**Critical scenario**
Context	1. Recovery of architecture 2. Evaluation of operational characteristics: performance, interfaces etc.
Purpose	To get an end-to-end view of the system, its integrity, and how the critical function are implemented, and get some insight on potential performance issues (e.g., response time and load).
Input	
Output	
Steps	1. Identify 2-3 scenarios that are critical to the raison d'être of the system. 2. Describe the black box view of the scenario: starting condition, events at the boundary, ending condition. 3. …
Roles	
Estimates	
Reference	

8.3.3 Change Case

Name	**Change case**
Context	Evaluation
Purpose	Evaluate the robustness, modifiability, extensibility of an architecture
Input	Architecture description
Output	Identification of hard to change architectural decision; or their cost

Steps	1. Brainstorm a list of potential changes in the systems environment (platform, new requirements, new functions, new scale, technology change, etc.) 2. Sort them out. 3. What would be the damage to the architecture by doing the corresponding change, what interface or components are affected, which ones are **not** affected? 4. Estimate the cost of implementing the change.
Roles	
Estimates	
Reference	SAAM [|Kazman 1994]

8.3.4 Check List

Name	**Check list**
Context	Use throughout the architecture review, from organizing it to concluding it.
Purpose	Reuse the know-how from previous reviews: typical problems, problem areas, overlooked issues. Coverage of the review.
Input	Check list (in review plan or procedure)
Output	Issues list
Steps	
Roles	
Estimates	
Reference	

8.3.5 Rate Monotonic Analysis

Name	**Rate Monotonic Analysis**
Context	When concerns with system behavior and timing, and evaluating execution architecture
Purpose	To determine schedulability of a set of processes
Input	Description of process structure (tasks, priorities, execution times, resources, periodicity,…)
Output	
Steps	
Roles	
Estimates	
Reference	Lui Sha et al.. circa 1990 [Sha 1991]

8.3.6 Module Structure Analysis

Name	**Module structure analysis**
Context	When evaluating
Purpose	To gain insight as to the quality of the module structure: understandability, portability, localization of complexity, cohesion and coupling
Input	Proposed module structure
Output	Specific properties: cohesion, layering, circularity, coupling, etc
Steps	
Roles	
Estimates	

Reference	

8.3.7 Probing About Alternatives

Name	**Probing about alternatives**
Context	When the rationale of the design decision is unknown; when the rationale is not fully supported by evidence or obvious.
Purpose	To confirm that it is the best choice
Input	A decision, its context and the ASR it addresses
Output	Confidence in a choice, or maybe identification of an issue or a better alternative to explore
Steps	
Roles	
Estimates	
Reference	

8.3.8 Prototyping

Name	**Prototyping**
Context	When inspection of artifact or reasoning cannot reach a conclusion
Purpose	Get a realistic range of value for an architectural attribute/property
Input	A design and an expected range
Output	A value within the expected range (hopefully)
Steps	Build an operational prototype.
Roles	Architect, developer
Estimates	Rather costly, depending on level of expected accuracy
Reference	Many…
Tools	Many…
Alternative	Take your chance

8.3.9 Queuing Model

Name	**Queuing model**
Context	When inspection of artifact cannot get to a conclusion; for certain classes of performance and capacity issues
Purpose	Get a realistic range of value for an architectural attribute/property related to response-time, throughput or capacity
Input	A design and an expected range; hypothesis about usage patterns and workload model
Output	A set of values
Steps	
Roles	Specialist in queuing theory; architect
Estimates	Inexpensive (excluding tool acquisition)
Reference	
Tools	Several commercial tools: QNAP et.
Alternative	

8.3.10 Quality Function Deployment

Name	**QFD Quality Function Deployment**
Context	Transition from Requirements to architecture
Purpose	Balance and match multiple concerns
Input	Stakeholders' needs
Output	The "house of quality"; prioritized requirements
Steps	
Roles	
Estimates	
Reference	
Tools	
Alternative	Impact matrices

9 Pragmatics and People Issues

In addition to technical issues, to have a successful architecture review requires an understanding of the social, psychological, and managerial context and implications of the review. These non-technical factors arise from the environment in which the review is situated: when in the lifecycle it occurs, who initiates the review, who the stakeholders are, and what is at stake in the review.

You should read this section if you will be organizing a review and want to be aware of the potential sources of non-technical conflicts that may arise. This section discusses the different types of reviews and how you can deal with the non-technical problems that will inevitably arise.

Highlights

- Keep your review participants as homogeneous as possible.
- Focus on technical issues. In particular, do not get into a discussion on management decisions that may be reflected in the architecture.
- Agree on rules for the review beforehand.
- Make developers feel confident, not defensive.
- Find win-win conditions of different stakeholders.
- Remember post-review work.

9.1 People Issues

We discuss people issues using two dimensions, reflecting the homogeneity of the people that are about to participate in the review, and predictability of the review. The following table illustrates some of the key attributes related to these dimensions. Below the table, a discussion is given for all the terms used in the table, structured in accordance with different categories of reviews.

	Scheduled	**Stakeholder Triggered**
Homogeneous	Egoless Civilized Helpful	Egoless Less Civilized Cooperative/Duty driven
Non-homogeneous	Egofull Less Civilized Cooperative/Duty driven	Egofull Potentially Uncivilized Hostile/Blaming

Table 9-1: Characteristics of Reviews

Obviously, while the table presents strict categories, practical implementations usually have a mixed set of participants. For instance, the reviewer may come from a consulting organization, and still the audience could remain somewhat homogeneous. Similarly, a stakeholder-triggered review could be such the development personnel have agreed to it, resulting in a review that is egoless, helpful, and civilized. However, reviewing is very much about teamwork, and even a single misbehaving person can ruin the entire review. So it is important to consider the potential advantages and disadvantages of each review type.

9.1.1 Scheduled and Homogeneous

In a prescheduled architecture review with homogeneous participants, people are not trying to highlight their own achievements. Rather, the focus tends to be on the technical aspects of the architecture. To further strengthen the cooperative and technical nature of the event, all the participants have more or less similar background, and they more or less share the concerns over the architecture. A review of this type is one indicated in the project plan for instance, where minimal personal feelings are expressed.

Main characteristics of this type of a review are listed as follows:

- *Egoless*. Participants are not trying to highlight their own achievements or degrade other people's work. Instead, the focus tends to be on the technical decisions, and the designers responsible for making these decisions have minimal interest in preserving such decisions for their own sakes. Reviews have very positive spirit, as there is no feeling of being judged by the past actions in the project. Rather, there is a common goal and consensus to study the system as thoroughly as possible with the available resources for maximal output.
- *Civilized*. The focus of the meeting remains centered on technical issues. No attacks on individuals are made, and the spirit of the meeting should be comradely. The developers are prepared to openly discuss the adequacy of their designs. Similarly, other participants are ready to accept, comment, or argue over the technical reasoning behind the design decisions. No strong feelings arise during these discussions, and nobody is offended by the opinions raised during the review.
- *Helpful*. Participants are eager to help each other. Everybody is open to new suggestions, and nobody has strong feelings or jealousy over their own designs. As a downside, sometimes a helpful attitude results in a review where the focus shifts from the identification of the problem to brain storming for solution. While this may ruin the focus of the review, it is sometimes necessary in order to ensure the smooth proceeding of the meeting. However, if you allow side conversations, then you soon end up in a chaotic situation where there is no focus on the actual review work. For details on how to handle pragmatic issues like the above problem, the reader is referred to section 9.2.2. For this particular situation, refer to item *'What to do when the rules of the game are violated.'*

9.1.2 Stakeholder Triggered and Homogeneous

Whenever an architecture review is stakeholder initiated, there is a potential threat that some of the participants find the review personally offending or threatening, i.e., that the review is organized to detect personal failures. Further, a stakeholder-triggered review can be a sign of an alarm from higher management, which introduces even more potential for a hostile review. Notice that there is a range of stakeholder initiated reviews, some of which need not be hostile. For instance, if the chief architect initiates review, it can be handled more or less like a scheduled review. Main characteristics of a stakeholder triggered homogeneous review are the following:

- *Egoless*. As in 9.1.1.
- *Less Civilized*. The focus of the review is in still technical issues. Most of the aspects in 9.1.1 hold.
- *Cooperative/Duty driven*. When participants of a review form a less homogeneous group, or there is an important concern raised by some of the stakeholders, the review becomes cooperative and duty driven. Interest in personal aspects starts to play a bigger role, resulting in less willingness to help each other. Rather, the review is carried out as a duty. Cooperation aims at the satisfaction of other stakeholders' expectations, not from a genuine desire to help each other. For obvious reasons, such reviews may fail to discover some problems that should have been detected. Further, people may start to feel protective of their own work and be less willing to admit potential design flaws. Obviously, this results in degraded productivity of the review, and potentially introduces bigger threats to the goals of the review if not dealt with adequately.

9.1.3 Scheduled and Non-homogeneous

In a non-homogeneous review, participants have different backgrounds and/or different commitment to the system whose architecture is being reviewed. For instance, high-level line managers may not have any idea about the technical aspects of the system, whereas chief designers tend to be interested only in the technical portions. Therefore, when the participants' backgrounds diverge, ranging from a general-issue designer to a project manager to a line manager, there is a threat that no technical contribution is made in the review. Notice, however, that although the presence of outsiders introduce non-homogeneity, it may be liberating for the developers to have outside reviewers with whom to talk about the technical aspects of the system. This obviously increases the productivity of the review, as the developers are not taking on fixed roles determined by the development organization or other fixed structures. Also there can be a varying range of homogeneity.

The main characteristics of this type of a review are.

- *Egoful*. The developers are not willing to reveal technical issues, or openly discuss their decisions. There is jealousy over designs, and technical arguments are overridden by personal opinions and contributions.

- *Less Civilized*. As in 9.1.2.
- *Cooperative/Duty driven*. As in 9.1.2.

9.1.4 Stakeholder Triggered and Non-homogeneous

This is the most difficult type of architecture review. People are not open, and they are threatened by the presence of outsiders. Further, there is a strong feeling of pressure, as stakeholder triggered architecture reviews can be taken as a sign of a severe project crisis. Technical staff may enter a "garrison mentality", where they try to minimize the effects of the review to their part of the development, and blame others for wrong decisions. Then, instead of focusing on the technology, time will be spent on blaming and on attacks towards different stakeholders. Because stakeholders can have contradicting concerns, like the support for a portable implementation and maximal performance in some prefixed platform, the non-homogeneous participants of an architecture review can turn the review into a very hostile event.

- *Egoful*. As in 9.1.3.
- *Uncivilized*. The review is intended to compensate a potential risk to the development, so it is likely to be conducted in a crisis situation. As a result of such external audit, it may be impossible to focus on the necessary issues without a certain amount of personal pride involved. This pride can harm openness, and result in stiff defense or opposition out of personal likes and dislikes rather than judging the design on purely technical aspects.
- *Hostile/Blaming*. The designers/developers often feel threatened by such a review. In practice, it may be that line management participates in the review, for instance, which can be interpreted as an act of espionage on technology achievement by the developers. Technical issues lose the focus, and the review may become a series of quarrels over issues that people have strong feelings about. No willingness for open discussion can be found. Discovered issues are quickly blamed on an individual designer, who feels even more threatened by the review.

Clearly from the above discussion we encourage reviews to be scheduled and to be as homogenous as possible. While the latter is not always possible (or always desirable) the former is certainly possible; architecture review activities should be built into the project plan from the start.

9.2 Pragmatic Issues

In the following, we list aspects that typically need attention in an architecture review. The discussion is divided into three parts. Firstly, we introduce necessary inception stage work to be done before conducting an architecture review. Secondly, we discuss issues to be taken into account during a review, and finally, we give a discussion on issues that should not be overlooked after conducting the review. As the structure of these issues consists of various bits and pieces, each different issue is separately addressed. The viewpoint is that of a review team leader, as that usually is the role where pragmatics is most applicable. However, also auxiliary roles can benefit from understanding the main pragmatic principles.

The discussion assumes that the reference workflow described in Section 4 is followed, and that a well-defined method like ATAM (discussed in Section 8) can be used where applicable [Kazman 2000]. However, many of the pragmatic issues can be adequately taken into account even if the above assumption is not valid, as they more or less describe the typical human behavior in reviews.

9.2.1 Inception Activities

The following discussion addresses pragmatics in the inception stage of a project. The focus is on aspects that must be clear for all the participants to ensure the smooth proceeding of the review. This includes both commitment to the goals of the review as well as acknowledgement of the review process and practices.

Setting expectations. It is important to appropriately set the expectations of all participants in the review. This includes explaining (and perhaps inscribing in a contract) what the evaluators do and what they don't do; who gets told what and when; what form the outputs are in and when they arrive; and who gets a chance, if any, to respond to the recommendations before they are made "public." This is a process of negotiating the win-win conditions for all the stakeholders.

Committing to the process. An architecture review should follow a process model. One of the benefits of such a model is that it explicitly lists the inputs and outputs of each step. In addition, when we explain the process to the stakeholders we can provide examples of all artifacts: architectural documentation, scenarios, risks, sensitivities, and tradeoffs. In this way the stakeholders know what to expect and can come appropriately prepared to provide the kind of input that will expedite the review. It is important to nominate an architecture review team lead who can force the review to move on if it stalls on an unimportant issue or starts to become chaotic. For more details, refer to item '*What to do when the rules of the game are violated*' in the following subsection.

Deciding who must be there for what stages. Any review, no matter how crucial, will be perceived, at least by some, as an intrusion on their precious time. Hence, it is important to identify and forewarn the stakeholders of the review with plenty of lead-time, and to ensure that each of them knows which steps of the process require their presence. In fact, it is often useful to strictly limit attendance in many of the steps to the *minimal* set of stakeholders needed, since this typically makes for a more efficient and productive use of the time.
Putting up gates. As explained above, there are many possible contexts for a review. Reviews are frequently perceived as undesirable by the team being reviewed. This isn't surprising—nobody likes being tested and a review may be imposed upon the contractor or development organization. As a result, the development team being reviewed may drag its feet in producing the necessary documentation, or in responding to requests for information from the review team. However, it should be made clear from the outset that the review will not take place unless the required information is provided up front. So the development team must be told, well in advance, what things they have to do before proceeding, including responding to requests for information and providing adequate documentation up front. Note that this may require *educating* the development team about the information that is being elicited. When considering how you will put up gates, you should insist on having a single point of contact on the customer team--the project team leader/manager--and have this person responsible for coordinating the activities and outputs of the stakeholders. You should also think carefully about what information is really needed and digestible *during* the review, as compared with what can be provided in advance. Similarly, consider what information must be presented during the review as opposed to what can be delivered afterwards.

Getting the correct requirements. An architectural review is driven by requirements. These requirements cover both functional and quality aspects of the system. It is essential to get the correct and up-to-date requirements as an input to the review. While this seems obvious and trivial, it is common to be furnished with a requirements document prior to a review, only to be told during the review: "when we said 1 second response time we didn't mean it *always* had to be 1 second" or "when we said continuous availability in the requirements we really are only concerned with availability from 0800 to 1800 hours", and so forth. Many requirements are not written down, or are part of a tacit agreement among the key stakeholders, or are in fact conflicting assumptions made by different groups of stakeholders. So getting the correct requirements as part of the inception stage work is both essential and potentially tricky. All of the important requirements should be discussed with the key stakeholders before embarking on the review.

9.2.2 Pragmatics During a Review

In the following, we point out some pragmatic issues that need to be taken into account. The list is not complete, as any pragmatic issue of people-ware could be included. *Crowd control*. It is critical to establish, at the start of the review meeting, how and when people are allowed to interact with each other. For example, it is important to ensure that there are no side conversations, for this disrupts the proceedings. It should be established whether people can come and go, and when they need to be present.

Getting buy-in from the key stakeholders. Because reviews tend to originate from different sources within an organization, it is often the case that some of the participants are unhappy with the review. This is particularly the case when the review activity is not part of the normal software process. In such a case it is often viewed as an intrusion on the precious time of the stakeholders. So it is crucially important, both before and during the review, to sell management, the architects, and the other stakeholders on the *value* of the review. This can be partially achieved by

letting the stakeholders know that the review is occurring to help improve the architecture, not to point fingers or to find blame. Any activity that involves the entire group interacting also helps to achieve buy-in.

Engaging all participants. It is not uncommon, in any kind of meeting, to have people who dominate the "air waves" and to have people who, for one reason or another, feel quite shy about participating. However, to get the most out of the participants, it is important to figure out how you can engage them. For example, some people might be reluctant to speak frankly in front of their bosses or their subordinates. For these people it is important to provide a forum for them where they are either alone or only among peers so that they can speak freely. It is also important to provide a mix of free-for-all participation and periods where each person has a dedicated time to speak. It is also important to engage all members of the review team—they are not immune to the social forces that affect the other stakeholders. For this reason the roles on the review team must be established well in advance.

Making decisions and determining priorities. As discussed above, one of the problems that you see over and over again when doing architectural reviews is that requirements, even when they are well documented, are often not well understood. Because of this it is frequently difficult to make decisions. For example, when choosing between two architectural alternatives, if the requirements are vague or incomplete or ambiguous, how would you go about making a decision? So any review, and architectural reviews in particular, need to have techniques for aiding decision making and prioritization.. Voting is an effective way of eliciting a set of priorities from a group of stakeholders. Discussion and consensus help to engage people and achieve buy-in, as discussed above, which means that the decisions that they make will be informed ones. The precise mix of these techniques varies from review to review and depends on understanding the organizational dynamics of the system being reviewed. For example, is the development organization strictly hierarchical or is it more egalitarian and team-based? Sometimes a single stakeholder makes all the important decisions and consensus is not high priority. We make no value judgments on which of these models is better, but it is critical to understand the group dynamics to facilitate the decision making and to make priorities.

What to do when the rules of the game are violated? There will be times when an evaluation gets out of control: people will have side conversations, or will try to steal the agenda, or will resist providing information. The review team needs to know who has the ultimate authority if people are being disruptive or are having catfights. The review team leader? The customer? A specific manager? The architect? It must be clear, before the review starts, who has the ultimate control when (not if) an evaluation becomes chaotic.

Controlling the pace. Any meeting with a wide group of (likely highly opinionated) stakeholders will range out of control from time to time, as stated above. But sometimes these conversations are revealing—hidden agendas, worries, future requirements, past problems, and a myriad of other issues come to light. So the facilitator must be aware of these digressions and know both when to squelch them and when to let a conversation continue. It should be noted, however that this kind of facilitation can be exhausting: assimilating huge amounts of information, looking for problems, and managing all of the political and personal issues simultaneously. Thus, we have found that it is useful to have two facilitators, and to switch between them periodically to give each other a mental break.

9.2.3 Post-review Activities

A review is not conclusive. There usually are some loose strings that need to be tied up to have the necessary feedback for all the involved personnel. This is a two-way street: the reviewer should provide information and feedback on the system that was reviewed, and the participants should provide information on how they felt about the architecture review. Further, it is also possible to plan things ahead for the next review, for instance. The following discussion summarizes the main activities that should be conducted after the review has taken place.

Getting concurrence and feedback. The activities and information generated in a review are not really directed at the review team, even though the review team is frequently the focus of the conversation and source of many of the probing questions that need to be answered. The outputs of the review are really for the stakeholders of the project being reviewed—the review team is just there to act as catalysts, experts, and facilitators. Because of this mismatch between the producers/consumers of the information and the way that the information is elicited (via the facilitation

of the review team) extra care must be paid to ensure that all stakeholders concur with whatever is recorded. There is one other interesting implication of the review team's "special" relationship to the outputs of a review: some results that surface during the review are already known to the architecture team and so the reviewers are sometimes made a vehicle for carrying messages to management. Other results are not known and truly represent value added by evaluators. It is useful for the reviewers to distinguish between these two cases (at least internally if not visibly). If the reviewers are solely the vehicles for communicating hard decisions from the architects to their management, then the validity of the evaluations will degrade over time.

Reporting out. When the review has been completed, there must be some agreement on how the outputs will be communicated back to the stakeholders, in particular to management and to the architecture team. So it is important to have determined what the form of the outputs is. The review team needs to know who gets told what and when. For example, do the architects get a chance to respond to the review report before it goes to management or to the other stakeholders? The outputs of the review are available to the architecture and management teams as assets that they can use in planning the future evolution of their system. These outputs are valuable and so their eventual form and destination must be planned.

Follow-on. An out-brief or report from a review can have many possible destinies. For example:

- to validate existing project practices,
- to change existing project practices (such as how architectures are designed and documented),
- to argue for and get additional funding from management,
- to plan for future architectural evolution.

It is important, therefore, to consider what is the eventual destiny of the information that is produced from a review activity and to plan accordingly. This is also related to item *'Putting up gates'* already discussed above.

Reassessment agreement. For the review team, it is very rewarding to have continuity in their work. Similarly, for the development team it is easier to have the same review team look at the product all over again due to personal involvement that has already taken place. This can be best served by agreeing on a practice to hold reviews for each release, for instance. This lets the reviewers become more integrated with the development team, thus ensuring more rewarding reviews in the future. Further, the input from different stakeholders need not be repeated, if there have been no changes in their requirements, as they have already been recorded. This tends to make the reassessment less laborious. There, however, is the possibility that the review team becomes a part of the development personnel. This may result in less objective assessments, which obviously imposes a potential risk to the development.

10 Case Studies, Examples

This chapter contains examples of reviews conducted by members companies of the SARA group.
1. Medical Imaging Platform (Origin, Herman Postema)
2. Public Address and Conference System (Sioux, Henk Obbink, Herman Postema)
3. Battlefield Control System (SEI, Rick Kazman)
4. Earth Observation System (EOS) (SEI, Rick Kazman)
5. Cartography (Rational, Philippe Kruchten)
6. Telecommunication (Calls, Messaging and Multimedia Support) (Siemens, Norbert Weber)

Each case study tries to address the following topics/issues:

- Brief description of the domain, the project or product, the organization and its size; what triggered the review; type of ASR addressed
- Departure from the overall SARA template in this review, its organization and workflow; unique methods used
- Cost/time actuals (duration, number of people and roles involved)
- Major outcome of the review, positive and negative (effectiveness) and follow-up activities
- Lessons learned relative to the process of reviewing an architecture

10.1 Medical Imaging Platform (Origin/Philips, Herman Postema)

10.1.1 Objective and Scope

A review was performed on a product platform architecture. The platform serves as the basis for deriving a number of product lines for complex medical diagnostic equipment. The objective of the review was to identify architectural risks at a point in time where corrective actions could still be initiated. The review was performed early in the development project when the major architectural decisions had been made. The customer (principal) of the assessment was the manager of the department responsible for the development of the product platform.

The scope of the review was the platform as a whole, consisting of both hardware and software components. However, it was decided to focus primarily on the software architecture. All architectural views were subjected to the review. The concerns of all stakeholders of the system where addressed during the review**.**

10.1.2 Approach and Organization

The architecture review was performed as a guided self-evaluation. This implies that the local architects perform the analysis by themselves, possibly in co-operation with a number of external architects. In total eight local architects were involved in the review, and four architects from outside were invited in order to bring in an objective view and additional expertise in specific architectural areas. The architecture review approach was defined by Origin Technical Automation. Two Origin consultants facilitated the review process (like preparing the plan, performing interviews with the stakeholders, guiding the architects through the analysis work, documenting and presenting the results, etc.).

The review approach comprises the following phases:

1. ***Initiation Phase*:** Define the review objective, scope, organization, activities, resources, schedule, and risks. Document and approve the review plan. Perform a kick-off meeting with all review participants.
2. ***Requirements Consensus Phase:*** Identify the non-functional requirements (ASRs) and their relative priorities by interviewing the stakeholders. Get agreement on the non-functional requirements and priorities by means of a consensus meeting with all stakeholders.
3. ***Architecture Analysis Phase***: Identify the architectural decisions made and relate these to the non-functional requirements. This is done by preparing a so-called 'impact analysis matrix' which communicates the impact of each decision on each requirement. Based on this matrix, architectural strengths, weaknesses, risks, and recommendations are defined.
4. ***Architecture Review Phase***: Document the results of the analysis in the form of presentation material (slides). Organize a review meeting where the results are discussed with the facilitators and the external architects. Adapt the results based on these discussions where needed.
5. ***Reporting Phase***: Document the review results in a Review Report. Review the report with the local architects and perform rework where required. Present the contents of the report to the principal and the stakeholders.

10.1.3 Review Conduct

During the Requirements Consensus Phase, about 22 stakeholders have been interviewed in order to identify the non-functional requirements and their priorities. Stakeholders who represented similar interests were grouped together. The requirements were identified by means of a pre-defined template of non-functional attributes. The relative priorities were determined by providing each stakeholder with a budget of points that could be allocated to 'high', 'medium', and 'low' priority. During the consensus meeting, the conflicting requirements or priorities were presented and solved. This resulted in a final set of over 100 non-functional requirements that would serve as input to the next phase.

During the Architecture Analysis Phase, the local architects identified the major architectural decisions that had been taken. These decisions were classified into 'key enabling technologies' and 'architectural concepts'. Next, the local architects were divided in three groups, each group dealing with a subset of the architectural decisions. The groups had the task to analyze the impact of each decision on the total set of non-functional requirements. This impact was expressed by means of so called 'impact indicators' (++, +, 0, -, --) reflecting whether and into what extend an architectural decision supports (or obstructs) the realization of a non-functional requirement. At the end, the results of the groups were integrated into an 'impact analysis matrix'. Each group prepared a slide presentation of their work (according a pre-defined template). Apart from a motivation of each impact indicator this presentation included the architectural strengths, weaknesses, risks, and recommendations.

The external architects were extensively briefed in order to be adequately prepared for their contribution during the Architecture Review Phase. Briefing items included the product context, the review approach, their specific role, and a code of conduct. Moreover, the external architects were given an opportunity to study the available system specifications and architectural descriptions. It was decided to organize for two review meetings covering the review of the decisions regarding the 'key enabling technologies', and the 'architectural concepts' separately.

During the review meetings (which took a full day each), the external architects had the task to comment on the presentations that were given by the local architects and to raise discussions where needed. As a result, the findings from the local architects were sometimes adapted as a result of these discussions.

During the Reporting Phase, the facilitators documented the outcome of the architecture review in a Review Report. This report was structured into the sections:

- Introduction
- Non-functional Requirements
- Findings – Key Enabling Technologies
- Findings - Architecture Concepts
- Conclusions

- Recommendations

The complete set of non-functional requirements was documented in an appendix. In the two 'Findings' chapters, a table was prepared for each architectural decision. This table included a motivation of the decision, possible risks, and ways to reduce these risks. Each risk was attributed to the non-functional requirements that were negatively influenced as a result of the architectural decision. By documenting which stakeholders owned these non-functional requirements, the stakeholders could use the table to determine into what extend their non-functional requirements would be satisfied (or obstructed) by the architectural decisions made.

10.1.4 Review Outcome

As a result of this architecture review it was concluded that the architecture was of an adequate level of quality. Only a few minor risks had been identified. These were translated into corresponding improvement activities. Apart from this primary outcome, a number of additional results could be identified:

- The analysis and review phase resulted in a number of findings regarding the process of architecture development. This was because the way of working was often discussed during the meetings. These process findings were finally considered even more important because severe weaknesses were detected.
- The local architects were given the opportunity to work together for an intensive period during the analysis and review phases. During their normal work they did not have many communications. As a result, the architects have learned much from each other and teambuilding was enforced.
- The architects got familiar with the concept of non-functional requirements and the relationship with their architectural decisions. They were not used to work in that way. In addition, the local architects learned from the external architects.
- The organization has experienced the value of an architecture review. It was decided that this kind of systematic review should become an integral part of the development process.

10.1.5 Lessons Learned

This architecture review was actually the first review that was performed by Origin in an operational context according their defined approach. The most important lessons learned were:

- The large number of stakeholders, local architects and external architects involved contributed to the level of detail and quality of the architecture review. However, the elapsed time of the review became very large (about three months) due to scheduling difficulties (scheduling the interviews and getting people together in meetings at the same time). In other words, quality and efficiency should be carefully balanced.
- The large number of non-functional requirements (over 100) was difficult to manage and caused a lot of effort during the analysis phase. During the course of the review, these requirements were combined into groups. Each group represented a set of requirements that were almost similar and that were influenced in a similar way by the set of architectural decisions. Another way of solving this problem is to restrict the architecture review to the most critical set of non-functional requirements.
- The involvement of external architects could have been even more effective by having these persons participate already during the Architecture Analysis Phase. During this particular review they only participated in the Architecture Review Phase.
- The 'Findings' chapters in the Review Report should be structured according to the stakeholders involved. In this way, for each stakeholder a report section is created that communicates the findings that this stakeholder is interested in (i.e. the level of satisfaction of the non-functional requirements that are important for this stakeholder).

More information on this specific case can be found in two papers that are available on the SARA website.

10.2 Assessment of the architectural approach for a telecommunications environment

10.2.1 Background Information

Application Domain:

- Telecommunication (Calls, Messaging, Multimedia support)

Scope of the architecture

- Overall architecture specifying the way of cooperation of and interaction between switches, applications, end devices (phones, PCs), servers, etc.
- Overall architecture for different products which are sold independently

The architectural concept under evaluation should form the basis for the future marketing strategy in this application area (e.g. integration of call solutions and messaging). Existing systems, products and solutions were not based on this architecture. However, the new architecture had to take into account existing products and systems already installed at the customers site (asset protection) and provide a migration strategy for moving smoothly from the current solution to the future one.

The Assessment was done during a very early phase in the development process: customer requirements were available, the architecture was specified rather on a conceptual level than on a detailed one. It primarily consisted of a high-level definition of the hardware and module structure. The definition work was done by an inter-disciplinary team consisting of experts from marketing, development, and service from locations in the US and Germany.

10.2.2 Goal of the assessment

The Assessment was done during a very early phase in the development process: customer requirements were available, the architecture was specified more on a conceptual level rather than on a detailed one.
The key question for the assessment was:

- Is the architectural concept suited to fulfill the strategy for multimedia communication products and other applications like mobility as well as the future extensions of voice features?
- Therefore, the focus of the analysis was
- to examine the design decisions already made, i.e. to analyze the strengths and weaknesses of the architectural concept
- to identify the essential risks associated with these decisions
- to identify open issues (design issues which were essential and had not been addressed so far)
- to provide an independent view of the architectural concept for the responsible senior management

10.2.3 Assessment approach

The assessment was done based on the principles and procedural model as defined by the architecture assessment method System Architecture Analysis (SAA) (see [GJW97]).
The assessment team consisted of 3 members:

- 2 SAA experts with experiences in conducting SAA based assessment (providing general architectural know-how, e.g. what are typical architectural issues to be covered / clarified by an architectural concept
- 1 telecommunication expert (providing specific application related know-how)

As SAA recommends, 2 teams had been built for the interviews / workshops:

- Marketing Team: 6 members with representatives from Marketing, Sales, Service (to ensure that the requirements are fully identified and understood)
- Architecture Team: 4 members responsible for the architectural design

It is important to separate clearly between the assessment team and the interview partners (Marketing / Architecture Team) in order to ensure that the assessment results provide a really independent view on the architecture under assessment. Furthermore, involvement of an telecommunication expert helped to reach acceptance for the assessment results with the development experts. As part of the preparation phase, it must be identified which

specific know-how is needed for the assessment; this is of course dependent on the specific architecture to be assessed (e.g. data bases, distributed systems, real time requirements)
The full assessment (preparation / planning of the assessment to final presentation) took 8 weeks. The assessment team dedicated 100 % of their time to the assessment. The interview partners had only to attend the workshops

- Marketing Team: 3 Workshops (3 hours) on Requirements and 2 Workshops (4 hours) on the detailed evaluation of design decisions
- Architecture Team: 4 Workshops (4 hours) on Architecture and 2 Workshops (4 hours) on the detailed evaluation of design decisions

In addition, a kickoff meeting (assessment goal, approach and schedules), a feedback meeting and a final presentation had been conducted

10.2.4 Assessment Information Input

The approach to gather the input necessary for the assessment was twofold:

- Documentation
- Interviews / Workshops

Documents which were available for the assessment included:

- Requirements Specification: This document was already released and provided a structured view of the requirements. Priorities were not available and had to be specified during the workshops together with the Marketing Team. Because the requirements are the basis for deriving evaluation criteria, priorities are necessary to focus the assessment on the really important criteria.
- System Design Specification: This document was still under development during the assessment. As a consequence, some design issues were still open (not clarified / specified), some information was inconsistent due to the fact that several persons were working at different chapters at the same time without having the chance of synchronizing their results.

Having to base on documentation which is not consolidated makes an analysis more difficult (additional clarifications needed). However, this seems to be a realistic scenario that an assessment cannot wait until all documentation is released: therefore, results from the assessment can be included in the documentation before it is finally released. However, criteria should be defined regarding which documentation status is deemed to be sufficient for an architectural assessment.
Documentation is not enough for gathering the information needed. Therefore, interviews / workshops had been conducted.

10.2.5 Assessment Issues

The assessment criteria used covered, in general, two aspects

- Customer and Customer Demand Aspects (in total 22 criteria, e.g. Customer Demands, Operation and Administration Requirements, Investment Protection, Quality Aspects)
- The Strategy of the Telecom business group and Business Aspects of this group (in total 8 criteria, e.g. Business Goals, Time and Cost Aspects)

During the Workshops with the Architecture Team a checklist of architectural issues had been used. This helped

- to cover all relevant aspects
- to clarify the agenda of the workshops (participants can prepare themselves on specific issues and provide additional material)

The following provides a high-level overview of architectural issues discussed during the assessment:

- General concepts (e.g. “Network Topology and Scaling”, “Distribution of Intelligence”)
- Software Structure (e.g. “Software Platform for Terminals/Servers”, “Software Structure of Switches”)
- Interface and Integration Concepts (e.g. “Interfacing to Transport Layer”)
- HW Components and Operating Systems (incl. Communication Hardware)
- Dynamic Aspects (e.g. “Control Structure”)
-

10.2.6 Results

As the market and technical context was very complex, the requirements and system concept have been specified and documented in different teams. One benefit of the analysis was that the SAA representation gave the first comprehensive view on the system concept and its capability to fulfill the requirements.
During the evaluation, several potential problems caused by some design decisions were identified. Recommendations for improvement had been identified and discussed with the Architecture Team in order to reach agreement, e.g.

- a conflict between the encapsulation principle of the software platform and the usage of middleware products which should be available at the system service level as well as at the application level; this problem was solved by revising the software platform concept
- a conflict between the hardware scalability concept and one possible communication technology approach which had been considered; this conflict motivated the definite choice of a suitable communication technology

In addition, some architectural design issues were not covered at all, e.g.

- dynamic system behavior (interaction of components, data flow, ...)
- For several system aspects, it turned out that technically advanced solutions related e.g. to distributed computing approaches got surprisingly low evaluations due to the influence of criteria like ”short initial time to market” or “easy serviceability”. These evaluations were indicators for the risks involved in choosing certain technological approaches that were best recognized by the experts during the ”localized” evaluation procedure in SAA.

11 References

The documents included in the list below are explicitly referred to within this report.

[Booch96] Grady Booch; "Conducting a Software Architecture Assessment;" *ROAD* 1996

[Buschmann 96] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M. *Pattern-Oriented Software Architecture: A System of Patterns*, Wiley & Sons, 1996

[Freedman, Weinberg 1990] Daniel P. Freedman, Gerald M. Weinberg, *Handbook of Walkthroughs, Inspections, and Technical Reviews : Evaluating Programs, Projects, and Products*, 3rd ed., Dorset House, 1990

[Gilb 1993] Tom Gilb and Dorothy Graham, *Software Inspection*, Addison-Wesley, 1993

[GJW97] M. Gloger, S. Jockusch, N. Weber. System architecture analysis - Optimizing architectures in the industrial context at Siemens. In European Software Engineering Process Group Conference, 1997.

[Hofmeister 99] Christine Hofmeister, Robert Nord, Soni Dilip, *Applied Software Architecture*, Addison Wesley Longman, 1999

[HP 00] A Template for Documenting Software and Firmware Architectures 1.3, draft, 27 Jan 2000

[IBM 99] SI/AD Architecture Description Standard: Overview V1.0 LWP 13/08/99, IBM Corporation

[IEEE 1471] ANSI/IEEE Std 1471-2000, *Recommended Practice for Architectural Description of Software Intensive System*, October 2000.

[IEEE Std 610.12-1990] Glossary of Software Engineering Terminology, 1990.

[Jazayeri 2000] Mehdi Jazayeri, Alexander Ran, Frank Van Der Linden, Philip Van Der Linden, *Software Architecture for Product Families: Principles and Practice*, Addison-Wesley, 2000.

[Kazman 1994] Rick Kazman, Len Bass, Gregory Abowd, Mike Webb, "SAAM: A Method for Analyzing the Properties of Software Architectures" *Proc. ICSE 16*., 1994, pp.81-90

[Kazman 2000] Kazman, R.; Klein, M.; & Clements, P. *ATAM: Method for Architecture Evaluation,* CMU/SEI-2000-TR-004

[Klein, Kazman 1999] M. Klein, R. Kazman, *Attribute-Based Architectural Styles*, Software Engineering Institute Technical Report CMU/SEI-99-TR-22

[Kruchten 95] Philippe B. Kruchten, *The 4+1 View Model of Architecture*, IEEE Software, November 1995, pages 42-50.

[RM-ODP 1995] *Reference Model of Open Distributed Processing* Part 1 – 4; ISO/IEC DIS 10746

[Ran 2000] Ran, A. "ARES Conceptual Framework for Software Architecture" in M. Jazayeri, A. Ran, F. van der Linden (eds.), *Software Architecture for Product Families Principles and Practice*, Addison Wesley, 2000.

[Sha 1991] Sha, L, Klein, M. & Goodenough, J. "Rate Monotonic Analysis for Real-Time Systems," 129-155. *Foundations of Real-Time Computing: Scheduling and Resource Management.* Boston, MA: Kluwer Academic Publishers, 1991

[Shaw 96] Shaw, M., Garlan, D., *Software Architecture: Perspectives on an Emerging Discipline*, Prentice Hall, 1996.

[TINA 1995] *Overall concepts and principles of TINA*, TINA Consortium, February 1995, http://www.tinac.com/specifications/specifications.htm

12 Appendix: Glossary

Architect: “The person, team or organization responsible for systems architecting.” [IEEE 1471]

Architecting: “The activities of defining, documenting, maintaining, improving and certifying proper implementation of an architecture.” [IEEE 1471]

Architectural Assessment: The end result of conducting an architectural review.

Architectural Description (AD): A collection of artifacts used to document an architecture. [IEEE 1471]

Architectural Review: The process and activities performed to develop an architecture assessment.

Architecturally-Significant Decision (ASD): A concept or decision pertaining to the structures or textures of a software system, addressing one or more architecturally-significant requirements.

Architecturally-Significant Requirement (ASR): A requirement upon a software system which influences its architecture.

Architecture: “The fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution.” [IEEE 1471] The set of concepts and design decisions about the structures and texture of software that must be made prior to concurrent engineering to enable effective satisfaction of architecturally significant, explicit functional and quality requirements, and implicit requirements of the problem and the solution domains. [Ran 2000]

Conceptual Framework: A common set of terms and concepts used to elucidate a subject of interest.

Concern, Architectural: An area of interest pertaining to a “system’s development, its operation or any other aspects which are critical or otherwise important to one or more stakeholders.” [IEEE 1471] Architectural concerns include system considerations such as feasibility, functionality, performance, reliability, security, distribution, and evolvability. *See also*: architecturally-significant requirement, quality attribute.

Criterion, Architectural Review: A concrete criterion for judging whether the architectural artifacts under review, and thus the architecture under study, meet a particular objective. An architectural review method establishes a set of review criteria.

Method, Architectural Review: A documented, repeatable approach for carrying out one or more activities of an architectural review. *Also known as*: architectural review technique.

Objective, Architectural Review: A reason for conducting a specific architecture review, a question about the system to be answered by a review. The objectives of an architectural review are driven by architectural concerns of the stakeholders.

Quality Attribute: An observable aspect of a system that contributes to system quality, a desired characteristic of a system. *Also known* as: “ility”. *See also*: architectural concern.

Scenario: A pattern of use or operation of a software system. Often extended to include patterns of maintenance, modification, or other types of interaction with the system. *Also known as*: use case.

Stakeholder: “An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system.” [IEEE 1471]

Structure, Architectural: An identification and arrangement of the concrete elements of a software system to address architectural concerns.

Texture, Architectural: The repeating characteristics of an architectural structure, particularly at a fine-grained level of concern.

Valuation: The result of applying a particular architectural review criterion to a specific system, using an architectural review method.

View, Architectural: “A representation of a whole system from the perspective of a related set of concerns.” [IEEE 1471] An architectural view may capture one or more architectural structures or textures.

Viewpoint, Architectural: “A specification of the conventions for constructing and using a view. A viewpoint acts as a pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis.” [IEEE 1471]

Workflow: The definition of a process in terms of its constituent activities, its participants and the methods and techniques enacted in its execution.

13 Appendix: Logistics and Templates

Examples of agenda's. Kickoff presentations, Sample Assessment Reports, Sample plans, Sample interview guides, Scripted presentation sample, templates for statement of work, example minutes, Cost estimates, Promotion presentation, …

13.1 Agenda for an Architecture Assessment

By default a two-days period followed by a presentation session :
First day : 'Initiation' + number of 'Aspect Investigations' (second part afternoon)
Second day : number of 'Aspect Investigations' + 'Finalization' (second part afternoon)
Presentation session : 'Findings Review and Presentation'

The total number of aspects to be investigated determine the number of days required for the assessment.

TIME	INITIATION	Remarks
30 min	Problem statement Goals of the assessment Intended follow-up Scope of the assessment	Presentation
15 min	Expectations	Flip-over
15 min	Agenda for the 2 days period Assessment concepts & approach Structure and content of the agenda Remarks, adjustments, agreement	Handouts
15 min	Rules of the road Assessment roles and names Separation of process and content Good meeting behaviour Parking lot	Presentation Flip-over
15 min	Global system architecture (business view)	Presentation
15 min	Break	
1h30	Market Perspective Current market opportunities and trends Role claimed by the organization Implication of products in general Consequences for the architecture Consequences for the Product Creation Process Product Requirements Kind of products to be delivered by organization Resulting functional requirements (new/remain/disappear) Resulting non functional requirements	Presentation Discussion Presentation Discussion
1h	Lunch	
30 min	Architectural Aspects to be addressed Summary of topics identified so far Filtering/clustering of Aspects Assigning weighing factors Selection of 5-7 most important Aspects	Impact matrix Discussion
1 h	Global system architecture (technical view)	Presentation

	Major architecture issues (up front) Architecture overview How architecture issues have been resolved Architecture tradeoffs made Remaining architecture issues (if any)	Discussion
15 min	Planning for the remainder of the assessment	Handouts
15 min	Break	

Two benefits and concern reviews will be planned (15 min each)

- At the end of the first day (including discussing the plan for the second day)
- Around noon of the second day

TIME	ASPECT INVESTIGATION	REMARKS
10 min	Definition of Aspect Keyword Clarification Other topics	Discussion
30 min	Architecture coverage of Aspect Supporting architecture choices Destructing architecture choices Impact indicators for choices	Architects tell Assessors ask Discussions Impact matrix
20 min	Conclusions for Aspect Coverage of Aspect (statement) Review topics list (for additional conclusions) Possible 'hot issues' (that cannot be agreed on, postpone) Possible recommendations	Flip-over Impact matrix Discussion

TIME	FINALIZATION	REMARKS
30 min	Discuss possible 'hot issues'	Flip-over
30 min	Summary of conclusions Conclusions per aspect Overall conclusions	Impact matrix Discussion
30 min	Summary of recommendations Recommendations per aspect Overall recommendations	Discussion
15 min	Evaluation of assessment & assessment process Review expectations, benefits and concerns Evaluate assessment process	Flip-over Discussion

TIME	FINDINGS REVIEW AND PRESENTATION	REMARKS
15 min	Review problem statement Goals of the assessment Intended follow-up Scope of the assessment	Presentation
1 h	Concept findings presentation	Presentation
30 min	Remarks on concept findings presentation	Discussion

15 min	Update of follow-up (to-do) activities Outcome of 'To-do' issues Additional insights	Discussion
1 h	Adjusting the findings presentation	
1 h	Final presentation to management	

13.2 Architecture Review Agreement Template

1. Objectives

- Provide information about the organization requesting the assessment: name, address, type of business, etc.
- Describe objectives as a set of issues or questions to be addressed during the review. These should have been discussed at least with the key stakeholders: customer, development management, architect(s), and possibly with sales and filed support.

2. Scope

Provide a brief description of what information will be collected and analyzed. In particular what system or sub-systems will be the subject of review. This should eliminate late surprises such as "I thought you would have analyzed the XYZ subsystem as well?".

3. Architecture review team

Identify individuals from the customer organization and from the outside that should be on the review team. Describe their respective roles.

4. Costs and Schedule

Architecture reviews are more open-ended in terms of scope and cost than architecture capability assessments. Depending on the size and complexity of the systems they can last from 5 to 10 days. If they last more than to two weeks they are probably more than just reviews and include elements of consulting and mentoring.
In addition to variable price of a review, the customer has to accept potentially significant internal costs of a review. This is briefly discussed in the review guidelines.

The document should specify

- Standard fee structure for review based on few critical variables: size of organization, product complexity, proposed duration of the assessment and number of people involved
- Costs to the customer. This will be mainly labor, so list all customer people that will be involved and provide min/max time estimate for each of them.
- Project schedule at the milestone level: information gathering and analysis, initial findings, final report and presentation.

5. Summary

Restate the benefits, necessity of having the right people participating in the review, and a need of a buy-in from the key stakeholders to produce useful results.

13.3 Review report template

1. Executive summary
This is a one-page reiteration of the key objectives for the assessment and a summary of the most important findings and recommendations.

2. Goals and the review process
- Recap the objectives as stated in the review agreement document
- List the members of the architecture review team and other key contributors
- Briefly describe the information gathering and analysis methods applied
- Thank people for their contributions

3. Key findings
3.1 Strengths
Describe the positive findings of the assessment. Rank them from most significant to less significant. Use a simple enumeration (the MSW Numbering feature).

3.2 Issues
Describe issues identified during the evaluation. Rank them from most severe to less severe. Use the following table for consistency.

Issue	Give the issue a name
Description	Describe the nature of the issue and its consequences
Info-source	Provide references to the information sources (documents, models, interviews, etc.) that illustrate the issue.
Recommendation	Propose an approach to addressing the issue

The report should be concise. If there is an issue that needs an extended discussion, such discussion should be moved to the Appendixes section. Hence, the recommendations should be short and to the point.

3.3 Architecture Review Profile
Discuss the result of the Architecture Review Profile. To a large degree this should be a summary of the findings discussed in Section 3.

4. Supplementary findings
During a review it is common to come across issues and concerns that are not directly related to the architecture or architecting capabilities, but may still be important. This section can be used to address the top few of them.
This section may be also used to point out the need for other assessments (process, project, and infrastructure).

5. Recommendation
Summarize the recommendations.

6. Appendixes
Discuss here any/all issues that need special attention but were too complex or too large to be exhaustively discussed in the main sections.