QUALITY OF SOFTWARE ARCHITECTURE DESCRIPTION

BY PANKAJ KAMTHAN

1. INTRODUCTION

The **quality of software artifacts**, including conceptual **models** [Nelson, Monarchi, 2007] and **documents** [Hargis, Carey, Hernandez, Hughes, 2004], produced at any stage of software development, for any purpose, is important.

The software architecture description "**inherits the criticality** of the architecture itself" [Nord, Clements, Emery, Hilliard, 2009]. Therefore, the quality of the software architecture description is a concern.

In this document, a few models for the quality of software architecture description are considered.

2. QUALITY OF SOFTWARE ARCHITECTURE DESCRIPTION: MODEL I

In the model given by [Rozanski, Woods, 2005, Page 176], an effective software architecture description needs to balance **six desirable properties**, as shown in Figure 1.

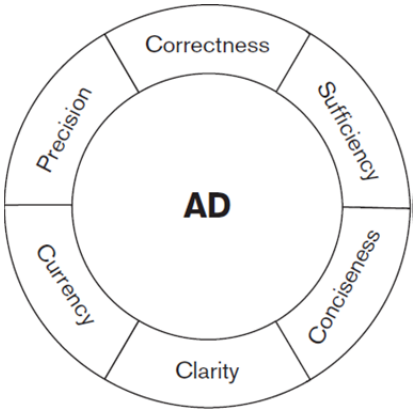

Figure 1. The desirable properties of software architecture description. (Source: [Rozanski, Woods, 2005].)

The desirable properties are:

- Correctness
- Sufficiency
- Conciseness
- Clarity
- Currency
- Precision

CORRECTNESS

Elaboration

The software architecture description must **correctly define** an architecture that will meet the **needs and concerns** of the stakeholders.

The software architecture description must **correctly represent** the **needs and concerns** of the stakeholders.

For example, (1) putting efforts in the direction of a concern that is **not** relevant to **any** stakeholder, or (2) **ignoring a legitimate concern** by a stakeholder class, make software architecture description **incorrect**.

Means

There are a number possible means.

For example, correctness of **some** aspects could be **assured** using **formal architecture description languages (ADLs)** for software architecture description. (The correctness of **all** aspects can not be assured using formal ADLs. This is because it is not possible to formalize (that is, mathematically express) **all** aspects of software architecture. Therefore, concerns related to such aspects can not be expressed in a formal ADL.)

For example, correctness could be **attained** by relying on established **architectural principles or architectural patterns**.

For example, correctness could be **verified** by **presentation of the software architecture description to stakeholders** and **feedback from stakeholders**.

SUFFICIENCY

Elaboration

The software architecture description must contain **enough detail** to answer the important questions about the architecture.

For example, the software architecture description must **not** omit a **architecturally significant decision** simply because it is "well-known by all" or postpone a significant architectural decision until later (say, implementation phase) in the process.

For example, deciding upon a **pattern** is a **architecturally significant** decision; however, deciding **variable names** is not.

The point is that such decisions will be made anyway, but the people who make them will not have the right information, and the decisions (or their impact) will not become apparent until it is too late to do anything about them.

In general, there is a **gap** if one or more of the following occurs: (1) **non-functional aspects** are ignored, (2) a **stakeholder need** is **neither explicitly nor implicitly reflected** in some element of the architecture, or (3) a **stakeholder need** has **not been documented** in the software architecture description.

Means

It is possible to attain sufficiency in many cases **if views and viewpoints are appropriately elicited**, and **architectural decisions**, especially those that are **contentious or have several alternatives**, are **rationalized**.

CONCISENESS

Elaboration

The software architecture description must only contain information that has a **high likelihood of being read** by most, if not all, stakeholders.

In general, a relatively lengthy software architecture description is also relatively **harder to maintain and to keep current**.

Means

The **factors** that affect the length of the software architecture description include the following: (1) the capabilities and experience of the stakeholders, the difficulty or novelty of the problem that is being solved, (2) the extent to which new or unfamiliar technology is being described, and (3) resources (including time and budget) available to produce and gain acceptance for the software architecture description.

CLARITY

Elaboration

In theory, the software architecture description must be **understood by all classes of stakeholders**. This is an ideal goal, and may not happen in practice.

In practice, at the very least, each stakeholder should be able to **understand those parts that are relevant** to him or her. (These parts are about **relevant views and viewpoints**.)

Means

This requires **skills in technical communication** and the **use of notations that are broadly understood** (or can be readily understood) by non-technical stakeholders.

The presence of a **glossary** can contribute towards **reducing ambiguity** and **increasing clarity**.

CURRENCY

Elaboration

The software architecture may **evolve** over time. There can be a number of sources of such a change, including issues in design, discovery of better ways of designing, or accommodation of new technologies, and so on.

These may result in **changes** to the software architecture, and these changes must be reflected in the software architecture description.

A **divergence** between **software architecture and its description** makes the software architecture description **untrustworthy** as the definitive source of information about the architecture.

Means

The software architecture description that includes too much detailed information (for example, treading into micro-architecture design) will be in a constant state of change, while the software architecture description that defines only the **architecturally significant** aspects of the system will be easier to keep current and relevant.

PRECISION

Elaboration

The software architecture description must describe the architectural aspects of the system precisely enough to allow the system to be designed and implemented.

Means

The design and implementation of such a system should be possible, by **two different teams with similar background and experience**, given the **same** software architecture description.

Therefore, the software architecture description must be **detailed enough** to be useful to technical stakeholders, and **succinct enough** to be readable by non-technical stakeholders.

REMARKS

Remark [Exhaustiveness of Properties]. There are certain properties, for example, **consistency and completeness** [Hargis, Carey, Hernandez, Hughes, 2004] that are relevant but are not considered by the model. For essential purposes, **sufficiency** could be considered **synonymous with completeness**.

3. QUALITY OF SOFTWARE ARCHITECTURE DESCRIPTION: MODEL II

In the model given by [Hämäläinen, Markkula, 2009], an effective software architecture description needs to consider **four high-level desirable categories of properties.**

These **categories of** desirable properties are:

- Stakeholder and Purpose Orientation of Software Architecture Description
- Content Quality of Software Architecture Description
- Presentation and Visualization Quality of Software Architecture Description
- Management of Software Architecture Description

These categories have been **decomposed** further into specific properties, and a list of **questions** has been associated with each property. Furthermore, each question has been signified on a **four point scale**:

1 = Important to Evaluate
2 = Useful to Evaluate
3 = Not Necessary to Evaluate
4 = Useless to Evaluate

From this scale, it can be concluded that **Stakeholder and Purpose Orientation of Software Architecture Description** is the **most significant** among the categories, followed by the others in the order presented above.

REMARKS

Remark [Basis of Categories, Properties, and Questions]. The categories, properties, and questions are based on the **authors' experience, literature review, and semi-structured focus group interviews of employees of a variety of companies** (specifically, information and communication technology organizations and service providers).

Remark [Basis for Review]. The questions can be used as a **checklist** for the evaluation (specifically, **review**) of the software architecture description.

Remark [Relationship to Other Models]. The questions considered in this model **overlap** with some of the properties in the previous model. Furthermore, these questions address **consistency and completeness**.

ACKNOWLEDGEMENT

The inclusion of images from external sources is only for non-commercial educational purposes, and their use is hereby acknowledged.

REFERENCES

[Hämäläinen, Markkula, 2009] Question Framework for Architectural Description Quality Evaluation. By N. Hämäläinen, J. Markkula. Software Quality Control. Volume 17. Issue 2. 2009. Pages 215–228.

[Hargis, Carey, Hernandez, Hughes, 2004] Developing Quality Technical Information: A Handbook for Writers and Editors. By G. Hargis, M. Carey, A. K. Hernandez, P. Hughes. Second Edition. IBM Press. 2004.

[Nelson, Monarchi, 2007] Ensuring the Quality of Conceptual Representations. By H. J. Nelson, D. E. Monarchi. Software Quality Journal. Volume 15. Number 2. 2007. Pages 213-233.

[Nord, Clements, Emery, Hilliard, 2009] A Structured Approach for Reviewing Architecture Documentation. By R. L. Nord, P. C. Clements, D. Emery, R. Hilliard. Technical Note CMU/SEI-2009-TN-030. Software Engineering Institute. Carnegie Mellon University. Pittsburgh, U.S.A. 2009.

[Rozanski, Woods, 2005] Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. By N. Rozanski, E. Woods. Addison-Wesley. 2005.