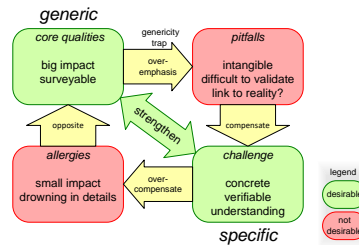# Balancing Genericity and Specificity

-

Gerrit Muller

University of South-Eastern Norway-NISE

Hasbergsvei 36 P.O. Box 235, NO-3603 Kongsberg Norway

gaudisite@gmail.com

Abstract

The balance between generic and specific architecting methods is discussed. The output of the architect must be compact and hence generic, but this output is based on many specific details which have been taken into account.

version: 1.2   status: finished   September 1, 2020

1 Introduction

The subtitle of this thesis is "Balancing Genericity and Specificity". The background of this subtitle is that nearly all development teams of complex systems seem to struggle with this balance.

Section 2 discusses this issue in a generic way. Section 3 illustrates the the role of this balance in case of the medical imaging workstation. Section 4 discusses the interaction of generic and specific elements in the context of the architecting method.

2 Core Qualities

The deliverables of the architect are mostly at a high conceptual level. The system specification defines the outline of the system and the system design provides the outline of implementation of the system. Most of the output is rather generic: defining many properties of a system with a minimum set of words and diagrams. The engineers work at all the specific details of the system, which can be millions lines of code or millions of transistors on a chip.
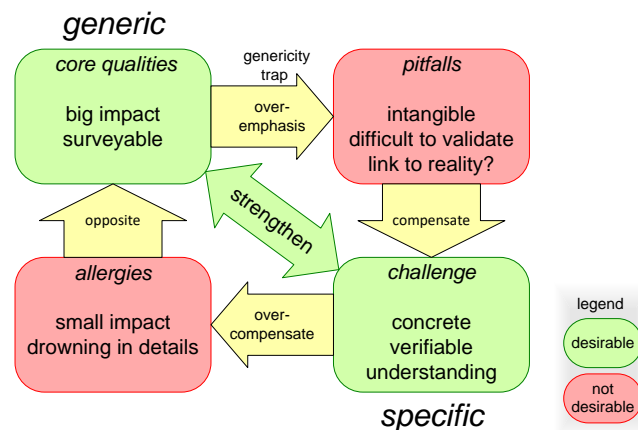
Figure 1: Core quadrant representation that shows the complementary nature of generic and specific approaches.

Figure 1 shows the two approaches, generic and specific, as complementary approaches in a core quadrant representation, as described by Daniel Offman [2]. The top left quadrant describes the core qualities of a generic systems approach. The top right quadrant describes the pitfalls that occur when the core qualities are overemphasized. The bottom right quadrant shows the challenges to prevent the pitfalls to happen. The challenge is to be sufficiently specific. The bottom left

quadrant shows the allergies: what happens if too much compensation is applied? These allergies are the opposite of the original core qualities in the top left quadrant.

Generic definitions are very powerful: one sentence may impact thousands or even millions lines of code. The compactness of the output allows all stakeholders to get a good overview of the system and its context.

Working in a too generic way is dangerous: the relation with the real world can be lost (generic motherhood statements). The generic definitions can be rather abstract, thereby making the concepts intangible for many stakeholders. Last but not least, the definitions may have become so generic that their validation is difficult. What is the value of statements that cannot be validated?

Taking a very detailed, highly specific approach has the advantage of being very understandable, due to the closeness with the real implementation and the very concrete nature. Very specific details tend to be easily verifiable.

A disadvantage of a too specific approach is that most individual statements have a very limited impact. Many details have to be specified to cover the entire design. In very specific approaches one can easily get lost and drown in a sea of details.

3   Genericity and Specificity in the Case

The entire system specification and design of the medical imaging workstation can be captured in a very limited set of diagrams and tables. Figure 2 shows the main diagrams that are needed to understand the design of the medical imaging workstation.

Note that every diagram or table fits on a single A4 or a single slide. The amount of information in every diagram is reduced to the level where the diagram can be explained to stakeholders with a reasonable amount of domain know how in a very limited amount of time. The compactness of the diagram is important to create and maintain overview. The combination of the diagrams creates an integrated understanding of the system design.

Every word, block, or number in every diagram is based on hundreds of details. A block in the construction diagram typically contains 10,000 lines of code, with hundreds of instance variables and hundreds of methods. An arrow in the software process diagram will be used for tens of different connections with hundreds of attributes. Many of these details have been touched by the architect. However, many more details will never be touched by the architect, as described in Sections **??** and **??**. Submethods such as the *Question Generator*, see Section **??**, help in finding the details to be covered.

The sampling of reality by touching many of the implementation details is the balancing factor in generating the required high-level, compact output. System designs that do not use such fact finding procedures to connect to reality via such

Figure 15.5
SW processes
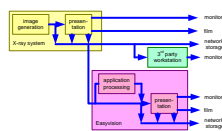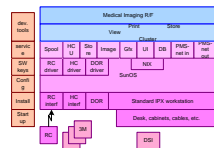
Figure 15.1
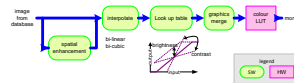image quality context

Figure 15.8
memory budget

high level, generic diagrams:
large impact, providing overview

Figure 15.7
construction decomposition

Figure 15.2
processing pipeline

every block, number of word is based on hundreds of specific
design details (loc, measurements, images, connections, etc.)

Figure 2: Five generic diagrams are shown. Every diagram is based on hundreds of specific design details.

fact finding are very risky and do not deserve to be called *system design*.

4   Genericity and Specificity in the Architecting Method

The architecting method has many submethods, which invite to be generic. The CAFCR model and the quality checklist are very generic. Every submethod is powerful, and can have a significant impact on the specification and the design. Figure 3 positions the architecting method with respect to the level of genericity or specificity, using the scale of abstraction levels from Figure **??**. The story telling complements the CAFCR submethods and qualities by addressing specific details. The threads of reasoning integrate and balance the generic and specific submethods.

Story telling is an effective means to make discussions concrete. A good story is **over**specific. On purpose a very narrow, but representative, sample of the target use is described and analyzed. This forces the architect and the designers to look into many specific details.

The threads of reasoning iterate between the generic models and objectives and these details that enable or obstruct the design. The threads of reasoning help to balance genericity and specificity.

Section **??** explains the dynamic range of description and implementation facts that needs to be covered by the submethods. The evaluation in Chapter **??** concludes
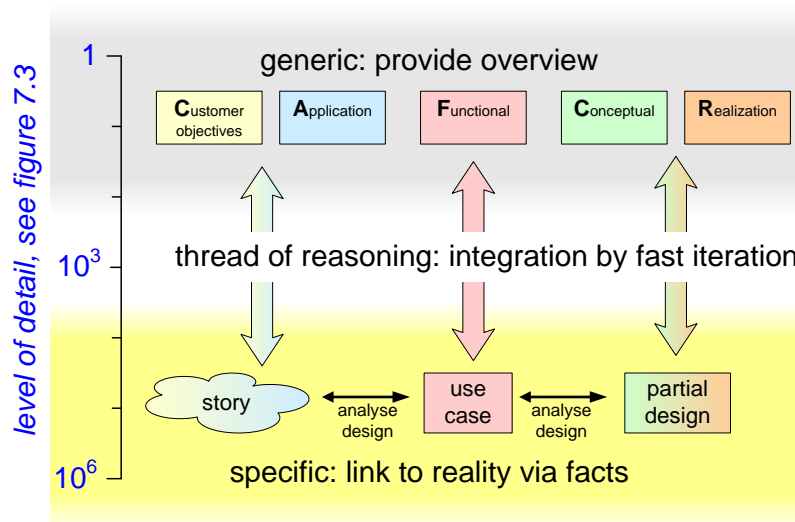
Figure 3: The CAFCR views provide generic insight; story telling enables analysis of specific facts.

that the method satisfies the needs. The only exception is that the outcome is sometimes too abstract and that the outcome is sometimes too late. The criticism of being too abstract is directly related to the generic nature of the CAFCR submethods. The step from the generic diagrams in Figure 2, with hundreds of facts, to the detailed designs of modules and components, with tens of thousands of detailed facts, spans a factor of more than one hundred! Making the generic diagrams more detailed worsens the surveyability and worsens the timely availability, which is the second concern. This tension between the need for genericity for power and understanding, and the need for specificity for the link with reality, is the core problem that architecting methods must tackle.

5   Conclusion

The overall architecting methods described in this thesis work successfully in products that span a very large dynamic range of concerns. The criticism of some of the stakeholders clearly identifies a major attention point for the architect when he is deploying the method: the balance between genericity and specificity.

The heuristics for the architect in finding the balance are:

- *Genericity*: overview diagrams fit on a single A4 (3).

- *Specificity*: the coverage of detailed facts may vary widely over different parts of the system (3).

- *Story telling* facilitates specific discussion, analysis, and design (4).

- *Threads of reasoning* integrate generic and specific viewpoints (4).

References

[1] Gerrit Muller. The system architecture homepage. `http://www.gaudisite.nl/index.html`, 1999.

[2] Daniel Offman. Bezieling en kwaliteit in organisaties.

History
Version: 1.2, date: April 7, 2004 changed by: Gerrit Muller
- changed status to finished

Version: 1.1, date: January 19, 2004 changed by: Gerrit Muller
- some small text improvements
- updated the level of details figure
- changed status to concept

Version: 1.0, date: January 6, 2004 changed by: Gerrit Muller
- many small text improvements
- updated the core quadrant figure
- changed status to draft

Version: 0.1, date: November 4, 2003 changed by: Gerrit Muller
- many small text improvements
- added some text about "dynamic range"
- added references to the figure showing generic diagrams that are based on many details
- added Section "conclusion"

Version: 0, date: July 7, 2003 changed by: Gerrit Muller
- Created, no changelog yet