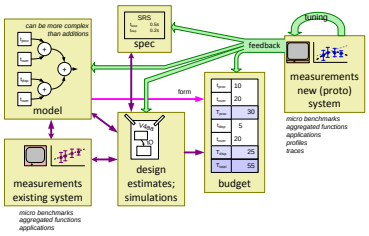
An incremental execution architecture design approach

-



Gerrit Muller
Buskerud University College
Frogs vei 41 P.O. Box 235, NO-3603 Kongsberg Norway
gaudisite@gmail.com

Abstract

An incremental design approach for the execution architecture is described. The method is based on identification of the most critical requirement from both user as well as technical point of view. The implementation itself is based on quantified budgets. The creation, modification and verification of the budget is discussed.

version: 1.0 status: draft October 20, 2017

1 Introduction

The execution architecture itself is the design step from the conceptual view to the realization view. The justification for design decisions has its roots in the customer objectives view and the application view, based on often ill articulated needs, concerns and expectations of the customer. A good understanding of mostly performance and timing related needs and expectations is needed and used to get a specific and measurable product definition with respect to performance and timing requirements. This definition is not a pure top down approach, a priori know how of the possible solutions is used to coverge more quickly on relevant specification issues.

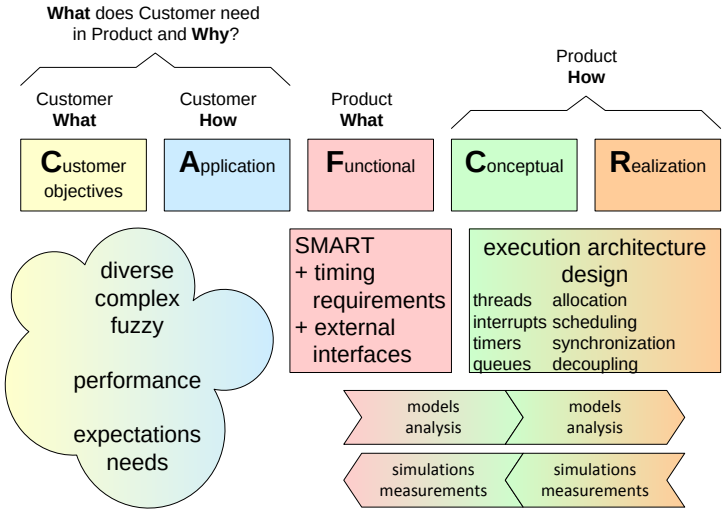


Figure 1: Positioning in CAFCR

Figure 1 visualizes these relations in the CAFCR model. The top down and bottom up iteration is shown as modeling and analyzing top down and simulating and measuring bottom up.

An incremental approach is strongly recommended. The problem and solution domain is often so complex that no human being can understand and oversee it entirely. The understanding and overview is build up in steps or passes, where all aspects are touched in one pass. The next pass deepens and enriches the insights. The reason that incremental approaches work is that it enables the humans to learn, based on the short feedback cycles. Typical cycle times are days or weeks, not months.

Figure 2 shows the spiral approach. First the **what** (requirements) and **how** (design) are studied, than the implementation, verification and evaluation is done, which closes the feedback cycle.

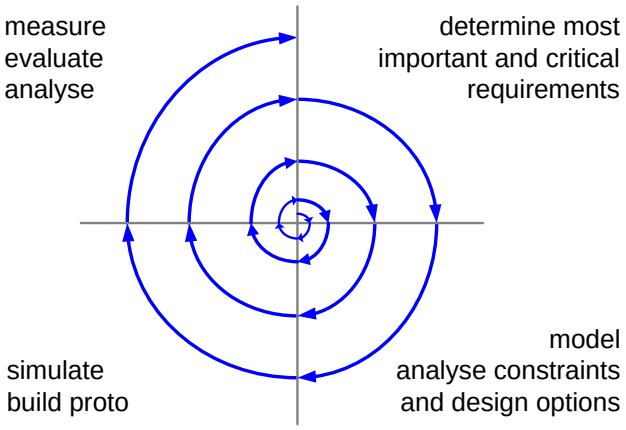


Figure 2: Incremental approach

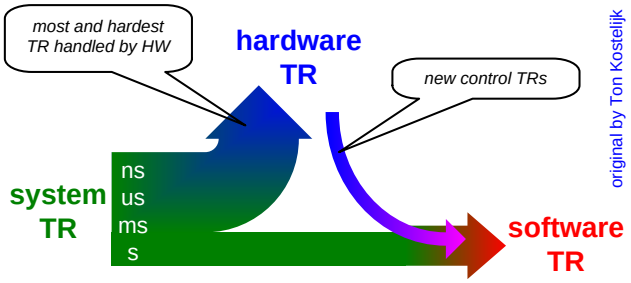


Figure 3: Decomposition of system TR in HW and SW

Most timing requirements are handled by the hardware, especially the very short response times are implemented by means of dedicated hardware. However this dedicated hardware itself needs some control, with more relaxed timing constraints. The hardware design imposes also timing requirements on the software design. Figure 3 visualizes this transformation of severe system timing requirements in somewhat more relaxed software timing requirements.

2 Quantification

The architect is continuously trying to improve his understanding of problem and solution[2]. This understanding is based on many different interacting insights, such as functionality, behavior, relationships et cetera. An important factor in understanding is the **quantification**. Quantification helps to get grip on the many vague aspects of problem and solution. Many aspects can be quantified, much more than most designers are willing to quantify.

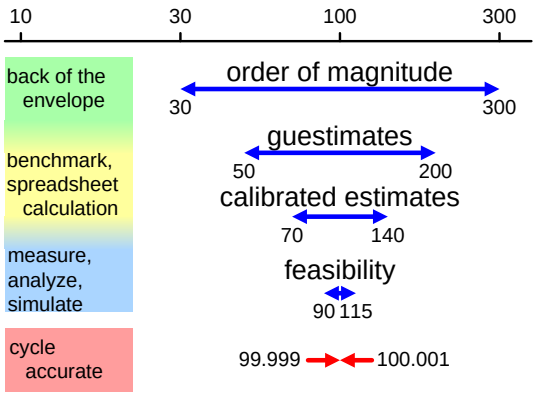


Figure 4: Successive quantification refined

The precision of the quantification increases during the project. Figure 4 shows the stepwise refinement of the quantification. In first instance it is important to get a feeling for the problem by quantifying orders of magnitude. For example:

- How fast should the system respond, for instance zap?
- What is the affordable cost, how much is the customer willing and able to spend?
- How many pictures/movies do they want to watch, transfer, store concurrently?
- How much storage and bandwidth is needed?

The order of magnitude numbers can be refined by making back of the envelop calculations, making simple models and making assumptions and estimates. From this work it becomes clear where the major uncertainties are and which measurements ot other data acquisitions will help to refine the numbers further.

At the bottom of figure 4 the other extreme of the spectrum of quantification is shown, in this example cycle accurate simulation of video frame processing results in very accurate numbers. It is a challenge for an architect to bridge these worlds.

3 Budget based approach

A central means to realize timing requirements is a (design) budget, a decomposition of the required timing into timing requirements for individual components or functions. Figure 5 shows an overview of a budget based approach to execution archietcture.

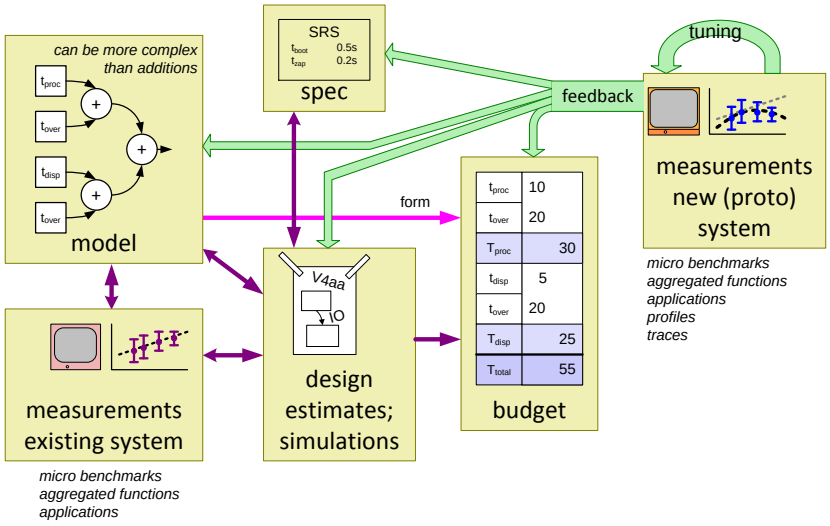


Figure 5: Budget based design

The basis of such a budget is a model of the timing: which functions or components play a significant role and how do they relate? The model articulates the understanding of the design and enables discussion, communication, analysis, simulation et cetera. Existing systems can be used to verify and calibrate the model, by measuring at different aggregation levels, from very fundamental functions (eg context switch time, transfer speed) to aggregated functions (image retrieval or throughput) to application level (zap response time).

Design discussions with component designers are used to combine requirements, model and data from existing systems into a design budget for the new system.

The project integration plan should strive for early feedback on this budget, by means of prototypes, partial systems and other creative means. The measurements must continue after the first integration feedback. Experience shows that performance characteristics degrade during developments, by addition of more features, more overhead and insufficient attention. By performing regular regression measurements the design is monitored. The results of those measurements are used to improve the model, design and budget, while sometimes even specification changes are needed.

Note that also resource budgets are useful as part of the design. The same approach is valid for resource budgets: modelling, measuring, designing et cetera.

4 Acknowledgements

This course is a joint effort of Ton Kostelijk and myself. Ton creates some of the course modules. He proved to be an inspiring sparring partner. Reinder Bril gave feedback which was used to improve the sheets.

References

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

[2] Gerrit Muller. Architectural reasoning explained. `http://www.gaudisite.nl/ArchitecturalReasoningBook.pdf`, 2002.

History

Version: 1.0, date: December 4, 2002 changed by: Gerrit Muller

- Created the article version
- minor improvements to the figures
- changed the order of the sheets

Version: 0, date: August 28, 2002 changed by: Gerrit Muller

- Created, no changelog yet