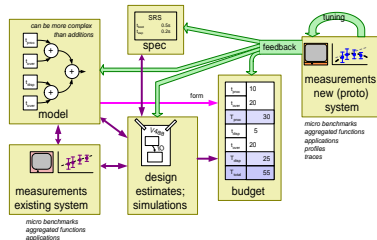


Modeling and Analysis: Budgeting

-



Gerrit Muller

Embedded Systems Institute

Den Dolech 2 (Laplace Building 0.10) P.O. Box 513, 5600 MB Eindhoven The Netherlands

gerrit.muller@embeddedsystems.nl

Abstract

This presentation addresses the fundamentals of budgeting: What is a budget, how to create and use a budget, what types of budgets are there. What is the relation with modeling and measuring.

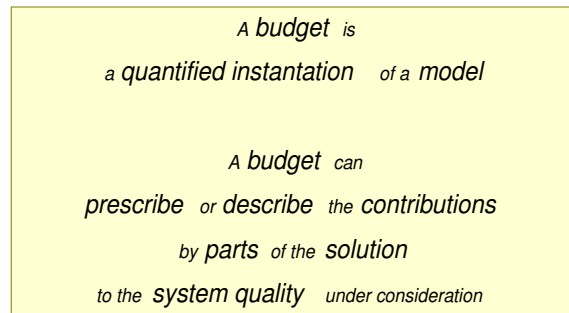
Distribution

This article or presentation is written as part of the Gaudí project. The Gaudí project philosophy is to improve by obtaining frequent feedback. Frequent feedback is pursued by an open creation process. This document is published as intermediate or nearly mature version to get feedback. Further distribution is allowed as long as the document remains complete and unchanged.

All Gaudí documents are available at:
<http://www.gaudisite.nl/>

1 Introduction

Budgets are well known from the financial world as a means to balance expenditures and income. The same mechanism can be used in the technical world to balance for instance resource use and system performance.



*A budget is
a quantified instantiation of a model*

*A budget can
prescribe or describe the contributions
by parts of the solution
to the system quality under consideration*

Figure 1: Definition of a budget in the technical world

Budgets are more than an arbitrary collection of numbers. The relationship between the numbers is guided by an underlying model. Figure 1 shows *what* a budget is. Technical budgets can be used to provide guidance by prescribing allowed contributions per function or subsystem. Another use of budgets is as a means for understanding, where the budget describes these contributions.

We will provide and illustrate a budget method with the following attributes:

- a goal
- a decomposition in smaller steps
- possible orders of taking these steps
- visualization(s) or representation(s)
- guidelines

2 Budget-Based Design method

In this section we illustrate a *budget-based design* method applied at waferstepper, health care, and document handling systems, where it has been applied on different resources: overlay, memory, and power.

2.1 Goal of the method

The goal of the budget-based design method is to guide the implementation of a technical system in the use of the most important resource constraints, such as

- to make the design explicit
- to provide a baseline to take decisions
- to specify the requirements for the detailed designs
- to have guidance during integration
- to provide a baseline for verification
- to manage the design margins explicitly

Figure 2: Goals of budget based design

memory size, response time, or positioning accuracy. The budget serves multiple purposes, as shown in Figure 2.

2.2 Decomposition into smaller steps

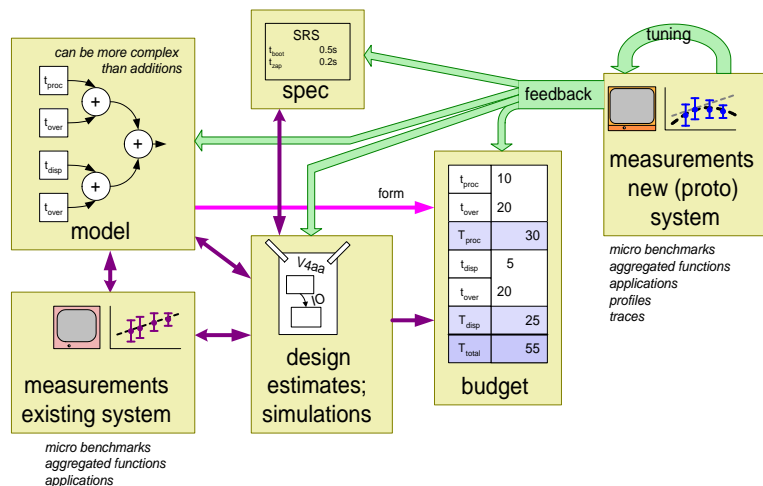


Figure 3: Visualization of Budget-Based Design Flow. This example shows a response time budget.

Figure 3 visualizes the budget-based design flow. This visualization makes it clear that although the budget plays a central role in this design flow, cooperation with other methods is essential. In this figure other cooperating methods are performance modeling, micro-benchmarking, measurement of aggregated functions, measurements at system level, design estimates, simulations, and requirements specification.

Measurements of all kinds are needed to provide substance to the budget.

Micro-benchmarks are measurements of elementary component characteristics. The measured values of the micro-benchmarks can be used for a bottom-up budget. Measurements at functional level provide information at a higher aggregated level; many components have to cooperate actively to perform a function. The outcome of these function measurements can be used to verify a bottom-up budget or can be used as input for the system level budget. Measurements in the early phases of the system integration are required to obtain feedback once the budget has been made. This feedback will result in design changes and could even result in specification changes. The use of budgets can help to set up an integration plan. The measurement of budget contributions should be done as early as possible, because the measurements often trigger design changes.

2.3 Possible order of steps

| step | example |
|--|--|
| 1A measure old systems | micro-benchmarks, aggregated functions, applications |
| 1B model the performance starting with old systems | flow model and analytical model |
| 1C determine requirements for new system | response time or throughput |
| 2 make a design for the new system | explore design space, estimate and simulate |
| 3 make a budget for the new system: | models provide the structure measurements and estimates provide initial numbers specification provides bottom line |
| 4 measure prototypes and new system | micro-benchmarks, aggregated functions, applications profiles, traces |
| 5 iterate steps 1B to 4 | |

Figure 4: Budget-based design steps

Figure 4 shows a budget-based design flow (the *order* of the method). The starting point of a budget is a model of the system, from the conceptual view. An existing system is used to get a first guidance to fill the budget. In general the budget of a new system is equal to the budget of the old system, with a number of explicit improvements. The improvements must be substantiated with design estimates and simulations of the new design. Of course the new budget must fulfill the specification of the new system; sufficient improvements must be designed to achieve the required improvement.

2.4 Visualization

In the following three examples different actually used *visualizations* are shown. These three examples show that a multi-domain method does not have to provide a single solution, often several useful options exist. The method description should provide some guidance in choosing a visualization.

2.5 Guidelines

A *decomposition* is the foundation of a budget. No universal recipe exists for the decomposition direction. The construction decomposition and the functional decomposition are frequently used for this purpose. Budgets are often used as part of the design specification. From project management viewpoint a decomposition is preferred that maps easily on the organization.

The architect must ensure the *manageability* of the budgets. A good budget has tens of quantities described. The danger of having a more detailed budget is loss of overview.

The simplification of the design into budgets introduces design constraints. Simple budgets are entirely static. If such a simplification is too constraining or too costly then a dynamic budget can be made. A dynamic budget uses situationally determined data to describe the budget in that situation. For instance, the amount of memory used in the system may vary widely depending on the function or the mode of the system. The budget in such a case can be made mode-dependent.

2.6 Example of overlay budget for wafersteppers

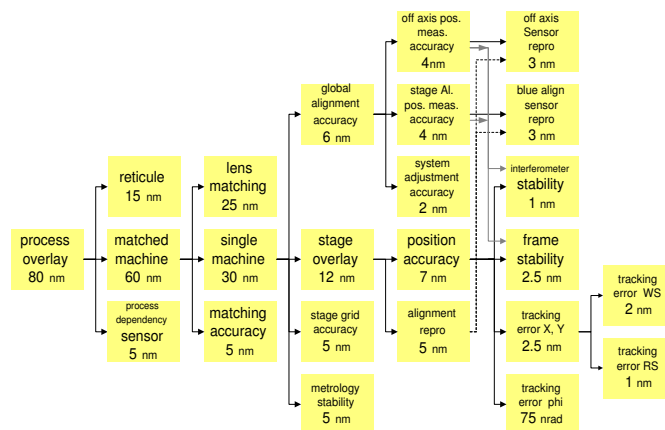


Figure 5: Example of a quantified understanding of overlay in a waferstepper

Figure 5 shows a graphical example of an “overlay” budget for a waferstepper. This figure is taken from the *System Design Specification* of the ASML TwinScan

system, although for confidentiality reasons some minor modifications have been applied.

The *goal* of the overlay budget is:

- to provide requirements for subsystems and components.
- to enable measurements of the actual contributions to the overlay during the design and integration process, on functional models or prototypes.
- to get early feedback of the overlay design by measurements.

The *steps* taken in the creation, use and validation of the budget follow the description of Figure 4. This budget is based on a model of the overlay functionality in the waferstepper (step 1B). The system engineers made an explicit model of the overlay. This explicit model captures the way in which the contributions accumulate: quadratic summation for purely stochastic, linear addition for systematic effects and some weighted addition for mixed effects. The waferstepper budget is created by measuring the contributions in an existing system (step 1A). At the same time a top-down budget is made, because the new generation of machines needs a much better overlay specification than the old generation (step 1C). In discussions with the subsystem engineers, design alternatives are discussed to achieve the required improvements (step 2 and 3). The system engineers also strive for measurable contributions. The measurability of contributions influences the subsystem specifications. If needed the budget or the design is changed on the basis of this feedback (step 4).

Two *visualizations* were used for the overlay budget: tables and graphs, as shown in Figure 5.

The overlay budget plays a crucial role in the development of wafersteppers. The interaction between the system and the customer environment is taken into account in the budget. However, many open issues remain at this interface level, because the customer environment is outside the scope of control and a lot of customer information is highly confidential. The translation of this system level budget into mono-disciplinary design decisions is still a completely human activity with lots of interaction between system engineers and mono-disciplinary engineers.

2.7 Example of memory budget for Medical Imaging Workstation

The *goal* of the memory budget for the medical imaging workstation is to obtain predictable and acceptable system performance within the resource constraints dictated by the cost requirements. The *steps* taken to create the budget follow the order as described in Figure 4. The *visualization* was table based.

The rationale behind the budget can be used to derive *guidelines* for the creation of memory budgets. Figure 6 shows an example of an actual memory budget for a

| <i>memory budget in Mbytes</i> | code | obj data | bulk data | total |
|--------------------------------|------|----------|-----------|-------|
| shared code | 11.0 | | | 11.0 |
| User Interface process | 0.3 | 3.0 | 12.0 | 15.3 |
| database server | 0.3 | 3.2 | 3.0 | 6.5 |
| print server | 0.3 | 1.2 | 9.0 | 10.5 |
| optical storage server | 0.3 | 2.0 | 1.0 | 3.3 |
| communication server | 0.3 | 2.0 | 4.0 | 6.3 |
| UNIX commands | 0.3 | 0.2 | 0 | 0.5 |
| compute server | 0.3 | 0.5 | 6.0 | 6.8 |
| system monitor | 0.3 | 0.5 | 0 | 0.8 |
| application SW total | 13.4 | 12.6 | 35.0 | 61.0 |
| UNIX Solaris 2.x | | | | 10.0 |
| file cache | | | | 3.0 |
| total | | | | 74.0 |

Figure 6: Example of a memory budget

medical imaging workstation from Philips Medical Systems. This budget decomposes the memory into three different types of memory use: code (“read only” memory with the program), object data (all small data allocations for control and bookkeeping purposes) and bulk data (large data sets, such as images, which is explicitly managed to fit the allocated amount and to prevent memory fragmentation). The difference in behavior of these three memory types is an important reason to separate into different budget entries. The operating system and the system infrastructure, at the other hand, provide means to measure these three types at any moment, which helps for the initial definition, for the integration, and for the verification.

The second decomposition direction is the *process*. The number of processes is manageable, since processes are related to specific development teams. Also in this case the operating system and system infrastructure support measurement at process level.

The memory budget played a crucial role in the development of this workstation. The translation of this system level budget into mono-disciplinary design decisions was, as in the case of overlay in wafersteppers, a purely human activity. The software discipline likes to abstract away from physical constraints, such as memory consumption and time. A lot of room for improvement exists at this interface between system level design and mono-disciplinary design.

2.8 Example of power budget visualizations in document handling

Visualizations of a budget can help to share the design issues with a large multi-disciplinary team. The tables and graphs, as shown in the previous subsections, and as used in actual practice, contain all the information about the resource use.

However the *hot spots* are not emphasized. The visualization does not help to see the contributions in perspective. Some mental activity by the reader of the table or figure is needed to identify the design issues.

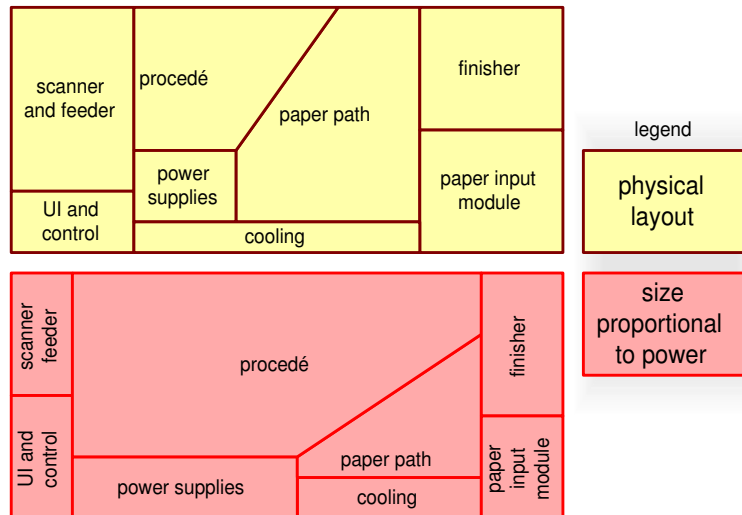


Figure 7: Power Budget Visualization for Document Handler

Figure 7 shows a visualization where at the top the physical layout is shown and at the bottom the same layout is used, however the size of all units is scaled with the allocated power contribution. The bottom visualization shows the *power foot print* of the document handler units.

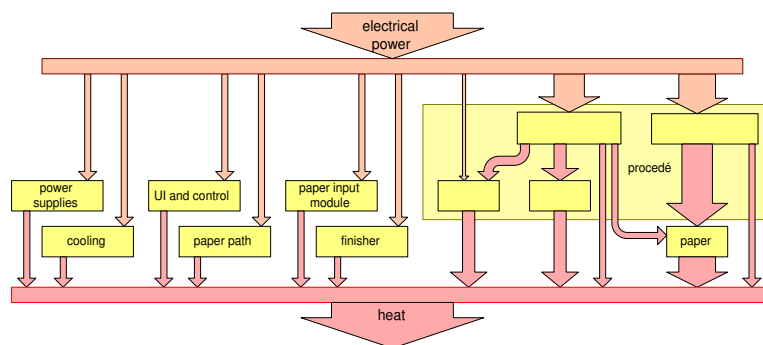


Figure 8: Alternative Power Visualization

Figure 8 shows an alternative power visualization. In this visualization the energy transformation is shown: incoming electrical power is in different ways transformed into heat. The width of the arrows is proportional to the amount of energy. This visualization shows two aspects at the same time: required electrical

power and required heat disposition capacity, two sides of the same coin.

2.9 Evolution of budget over time

| | |
|--------------|------------|
| static | dynamic |
| typical case | worst case |
| global | detailed |
| approximate | accurate |

is the budget based on
wish, empirical data, extrapolation,
educated guess, or expectation?

Figure 9: What kind of budget is required?

Figure 9 shows a classification for budget types. It will be clear that already with four different attributes the amount of different types of budgets is large. Every type of budget might have its own peculiarities that have to be covered by the method. For instance, worst case budgets need some kind of over-kill prevention. Add to these different types the potential different purposes of the budget (design space exploration, design guidance, design verification, or quality assurance) and the amount of method variations explodes even more.

We recommend to start with a budget as simple as possible:

- coarse guesstimate values
- typical case
- static, steady state system conditions
- derived from existing systems

This is also shown in Figure 10. This figure adds the later evolutionary increments, such as increased accuracy, more attention for boundary conditions and dynamic behavior.

However, some fact finding has to take place before making the budget, where lots of details can not be avoided. Facts can be detailed technical data (memory access speed, context switch time) or at customer requirement level (response time for specific functions). The challenge is to mold these facts into information at the *appropriate* abstraction level. Too much detail causes lack of overview and understanding, too little detail may render the budget unusable.

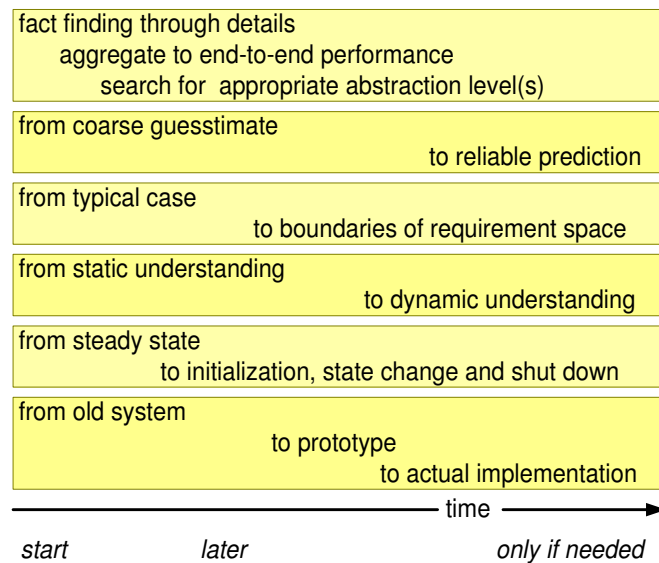


Figure 10: Evolution of Budget over Time

2.10 Potential applications of budget method

For instance the following list shows potential applications, but this list can be extended much more. At the same time the question arises whether *budget-based design* is really the right submethod for these applications.

- resource use (CPU, memory, disk, bus, network)
- timing (response time, latency, start up, shutdown)
- productivity (throughput, reliability)
- image quality (contrast, signal to noise ratio, deformation, overlay, depth-of-focus)
- cost, space, time, effort (for instance expressed in lines of code)

3 Summary

4 Acknowledgements

The Boderc project contributed to the budget based design method. Figure 12 shows the main contributors.

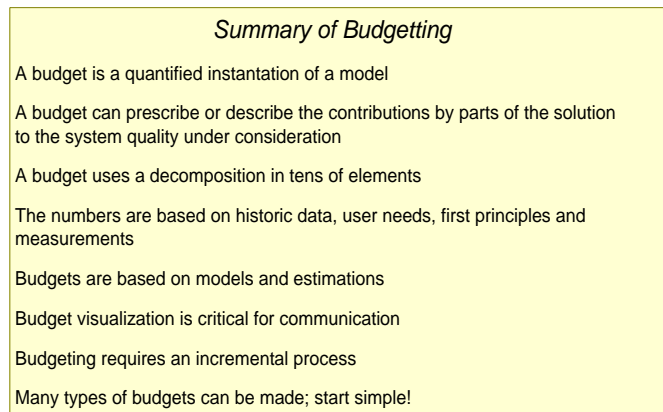


Figure 11: Summary of budget based design

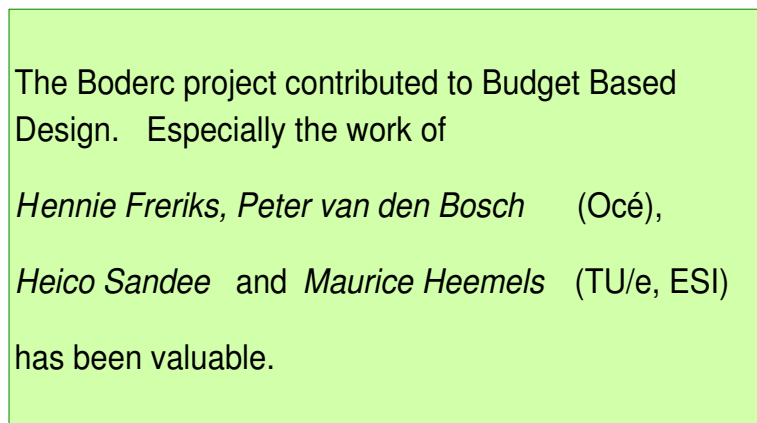


Figure 12: Colophon

References

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

History

Version: 1.0, date: 28 November 2006 changed by: Gerrit Muller

- Added text version
- added colofon

Version: 0.1, date: 17 November 2006 changed by: Gerrit Muller

- Added summary slide
- changed status to preliminary draft

Version: 0, date: 16 November 2006 changed by: Gerrit Muller

- Created, no changelog yet