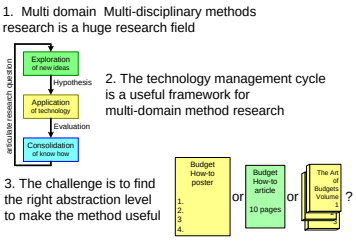
Do Useful Multi-Domain Methods Exist?

-



Gerrit Muller

Embedded Systems Institute

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

gaudisite@gmail.com

Abstract

The creation of embedded systems requires multi-disciplinary methods. The class of embedded systems is a quite heterogeneous class of systems, ranging from small high volume integrated circuits to expensive one-of-a-kind systems, such as electron microscopes or air-traffic controllers. The Embedded Systems Institute has been founded on the assumption that multi-disciplinary methods to create embedded systems can be applied in multiple doamins, despite the wide variation in embedded systems over the domains. In this article we discuss this assumption and we give a budget method as an example of a multi-disciplinary multi-domain method. Multi-disciplinary methods are used widely in the industry, but these methods are poorly consolidated and founded. We discuss the required research steps to advance from *implicit* methods to *explicit* and *founded* methods.

This work has been carried out as part of the Boderc project under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs under the Senter TS program. This work is also part of the Gaudí project.

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

version: 0.8 status: concept September 1, 2020

1 Introduction

The mission of the Embedded Systems Institute (ESI) is to establish methods to create embedded systems. The research agenda of ESI elaborates this notion. One of the purposes of the research agenda is to narrow the working field, such that the research area matches to the planned size of the institute. The fundamental assumption of the research agenda is that creation methods of embedded systems exist that can be used in different domains. Figure 1 shows the high-level assumptions that are the basis of the research agenda.

1. *Methods* that fulfil *multiple objectives* exist to *create embedded systems*

2. These methods help to *speed up* the *creation* process, *reduce* the *risks*, and *increase* the *product quality*

3. These *methods* are *generic* for multiple *market/business domains*, *application domains* and *functional domains*

4. These *methods build upon* the *software* and *electronics technologies*, and to a lesser degree these methods build upon the more *conventional technologies*, such as *mechatronics* and *physics*.

5. These *methods* need an *intelligent adaptation* to the *specific domain*

Figure 1: Assumptions of the ESI Research Agenda

The assumptions in Figure 1 are the starting point for this article. First of all we assume that methods exist to create embedded systems (1). These methods not only exist but are beneficial, by speeding up development, by reducing risks, and by increasing product quality (2). The essential assumption in creating an *Embedded Systems* Institute is that these methods can be generalized to make them useful over multiple domains.

In this article we look into the meaning of the word *domain*. We show that the industrial world uses many multi-domain methods implicitly, by providing a list of actually used methods. We discuss the status quo of the practiced methods and identify problems in today's use. We elaborate a budget method as an example of a *multi-domain method*. We finish by proposing a research approach based on a technology management cycle.

This discussion is based on experience of the author in health care equipment, wafersteppers, semiconductors, and electronics infrastructure projects. Many visits to other companies in other domains, such as telecom, consumer electronics, information technology, computer, electronic OEM suppliers, and defense have been useful to benchmark the observations in a broader perspective.

2 What is a Domain?

The meaning of the word *domain* turns out to be rather context-sensitive. In discussions with scientists, researchers from other institutes, and industrial practitioners the notion *domain* is used by all of them. However, the meaning of domain is different for all of them. Figure 2 shows how several types of domains can be mapped on the CAFCR-model. In [3] the CAFCR model is explained in much more detail. The idea behind this model is that a product to be created is described from two customer views:

- *Customer Objectives* view, **what** does the customer want to achieve
- *Application* view, **how** does the customer achieve these objectives

and from three product views:

- *Functional*, **what** does the product do
- *Conceptual*, **how** does the product work, described in the higher level concepts
- *Realization*, **how** is the product realized

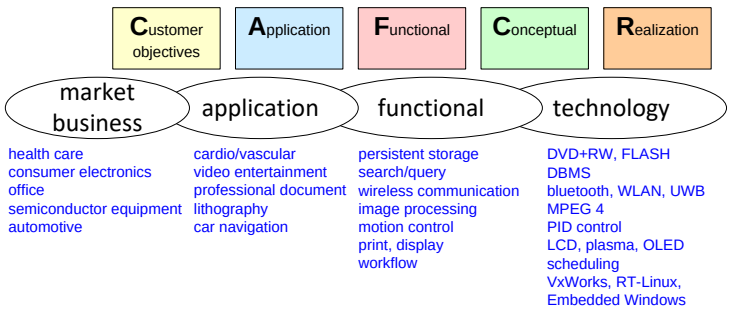


Figure 2: Domains Mapped on CAFCR

People working on highly specialized technologies use a domain to position their work in a somewhat broader technology area. This is shown in Figure 2 as technology domain. It maps on the conceptual and realization views of CAFCR. The classification focus here is often mostly on the *how*.

A somewhat broader use of the word domain is to indicate the functionality to be achieved by the technology, for example motion control. The term function is heavily overloaded in the engineering world, ranging from a low-level input-output transformation to a high-level user feature. The classification focus is more on the *what* than on the *how* as in the technology domain. Note that the functional and technology domains have overlap; for instance scheduling is shown as technology, but it can also be seen as a function.

The next broader use of the word domain is the *application domain*: where and how do we apply the system with its functionality. The examples shown as application domains are related to very specific human and organizational activities. In some hospitals cardio and vascular problems are treated by the same physicians, specialized in this medical field. The cardio vascular domain is the domain that covers all the functionality to treat patients with cardio vascular problems: the cathlab, the viewing rooms, portable monitors, supporting information systems et cetera.

From business perspective many application domains are related. A business classification is mostly based on marketing considerations. The cardio vascular application domain is part of the much broader health care market. Large providers operate in the health care market. These large providers own and operate significant parts of the health care infrastructure, such as hospitals and diagnostic centers. These providers purchase their equipment in large deals. From business perspective many application domains belong to a single business domain.

In this paper we use the domains covered by the customer perspective as *domain*: i.e. we look from the *Customer Objectives*, the *Application* and the *Functional* views. A multi-domain method in this paper is useful if it can be applied across multiple markets, businesses, applications or functions.

Many mono-disciplinary methods are multi-domain by nature. For instance, *rate monotonic analysis* or *garbage collection* methods are applicable in all markets, businesses, applications and functions. We focus on the multi-disciplinary methods.

3 What is a Method?

Designers use tools, notations, templates and other means to create designs. They follow a recipe to determine when and how to use these means. Many recipes have a close resemblance. The approach shared by many recipes can be captured by generalizing the recipes into a method. Figure 3 shows this abstraction from right to left. It shows principles or heuristics as an even higher level of abstraction. Examples of principles are: decomposition, recursive refinement, and quantification. Principles are very powerful, but at the same time very abstract, insights. These insights are useful input when recipes are generalized into methods.

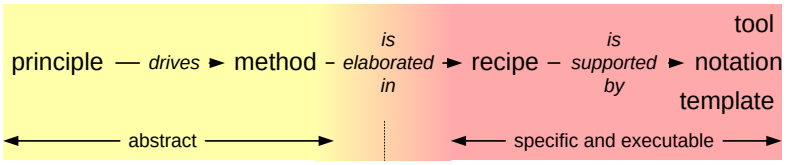


Figure 3: A method is a generalized description of a solution approach.

In general all methods have the following ingredients:

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

Researching methods implies studying (implicit) recipes, and more domain specific tools, representations and templates. The industrial partners of this type of research formulate their needs in the more specific executable form of recipes and tools. The rationale behind recipes and tools is captured in the method. Transfer of know-how requires transfer of explicit methods, illustrated by examples of recipes, tools, and representations.

4 Examples of Methods Applied in Multiple Domains

In the day-to-day industrial practice many multi-disciplinary methods are used implicitly or explicitly. Figure 4 shows a list of multi-disciplinary methods applied in multiple domains, taken from [3]. The methods at the left-hand side have been applied in multiple domains shown at the right-hand side of the figure.

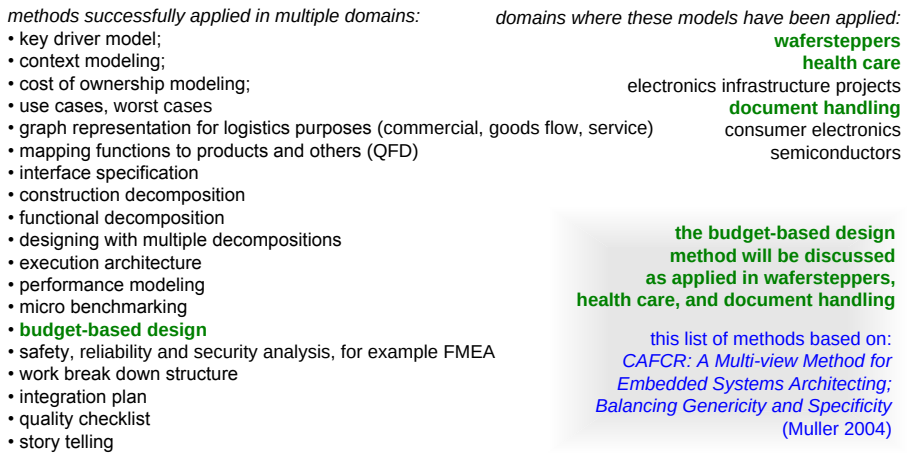


Figure 4: Examples of Methods Applied in Multiple Domains

We refer to [3] for a description of these methods. In section 6 we elaborate the *budget-based design* method to illustrate the nature of these kinds of methods, and we apply this method to the domains of wafersteppers, health care and document handling.

5 What is the Problem?

The list of Figure 4 and the five ingredients mentioned in Section 3 suggest that multi-domain methods exist. The use of these methods in the industry, however, is very implicit. The lack of explicit method description means that a lot of open issues remain. Open issues erode the value of these multi-domain methods.

We discuss the following classes of problems:

- generic nature of methods (5.1)
- lack of description (5.2)
- lack of education in this type of methods (5.3)
- lack of research (exploration and consolidation) (5.4)
- lack of relation with mono-disciplinary methods (5.5)
- lack of tools? (5.6)

5.1 Generic nature of methods

The multi-domain goal of these methods leads to more generic methods. A successful method that has been applied in one domain is transformed into a multi-domain method by abstracting from the domain. The generalization makes the method applicable in other domains. However this generalization also has other consequences:

- need for customization
- need for highly skilled designers

The domain-specific information has to be added to use the method. In other words, the method has to be customized to be used in a specific domain. Such a customization requires the skills to understand the generic method and the skills to transform domain insight into method adaptations.

5.2 Lack of description

Most methods are transferred from person to person and from project to project. Often no method description is available; the method is ingrained in a person or in an organization. To make a method better transferable we need a description of it. This description must contain the *concepts* and the *how to* information.

5.3 Lack of education in this type of methods

The ingrained nature of these methods also handicaps the education in these methods. A problem in education is the positioning of these methods in the existing educational system:

- When to learn: undergraduate, graduate, or postdoc?
- Who will teach multi-disciplinary methods?
- Do we add multi-disciplinary subjects to mono-disciplinary curriculums?
- Is the emphasis of teaching on practical use (vocational training) or on the scientific background (academic)?
- How to teach, while lacking method and case descriptions?

5.4 Lack of research (exploration and consolidation)

Even if a method is explicitly available then still many research questions remain. For instance:

- When to apply the method?
- What are the limits of the method?
- What are alternative methods?
- What are the options for (partial) solutions?

The area of multi-disciplinary methods is more or less uncultivated territory. The combination of domain customization and the more generic research questions mentioned above opens a tremendous research area.

5.5 Lack of connection with mono-disciplinary methods

Mono-disciplinary designers expected input from the system and subsystem architects. These architects use multi-disciplinary methods. The multi-disciplinary methods should ideally result in mono-disciplinary inputs. Unfortunately, the mono-disciplinary and multi-disciplinary field are not well-connected. This raises the following questions:

- How can the mono-disciplinary methods use the high level results from the multi-disciplinary methods?
- How do we make the transformation from these high level results into mono-disciplinary design inputs? For instance, how to transform a construction decomposition, with tens of components, into a class decomposition, with thousands of classes?
- How to decompose over different technologies?
- How to make trade-offs over technology boundaries?

The questions illustrate that the relation between mono-disciplinary methods and multi-disciplinary methods is bi-directional. For instance, to make trade-offs across technology boundaries the multi-disciplinary methods need input data from mono-disciplinary methods. To decompose over different technologies also mono-disciplinary data is needed as input to multi-disciplinary methods.

5.6 Lack of tools?

Many managers and researchers use the lack of tools as an excuse to stay away from these methods. With tools in this context we mean computer or software support. It is an open question if the lack of tools is really a problem. The methods shown in Section 4 functioned well without any specific tool support.

The availability of tools is not a prerequisite to study or apply multi-domain methods. However, during the research of these methods many tools will be used. These tools are often "borrowed" from mono-disciplinary methods: spectrometers, frequency analyzers, compilers, emulators, et cetera. Another category of tools is the set of homemade tools: hardware test rigs, and all kinds of software scripts. The homemade tools are used for measurements, simulation, analysis, documentation, and validation. The last category of tools we mention here is the set of very general-purpose tools, such as spreadsheets, configuration management, and word processors.

6 Illustration by a Budget-Based Design method

In this section we discuss a *budget-based design* method as an example of a multi-domain method. This method will be illustrated in the waferstepper, health care, and document handling domains, where it has been applied on different resources: overlay, memory, and power.

6.1 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 memory size, response time, or positioning accuracy. The budget serves multiple purposes:

- 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

6.2 Decomposition into smaller steps

Figure 5 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

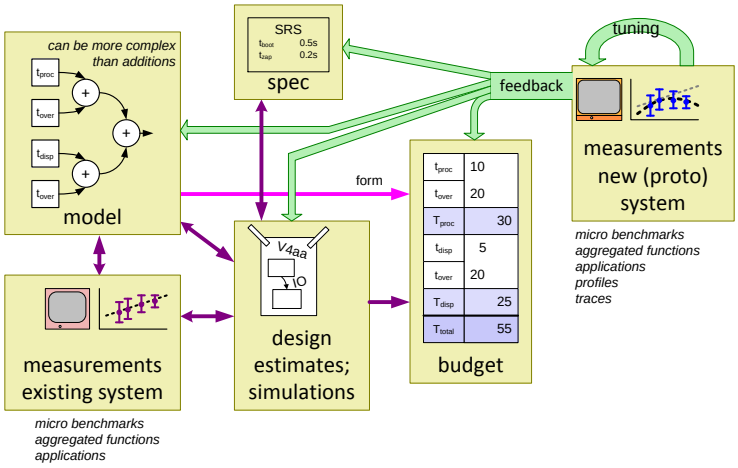


Figure 5: Visualization of Budget-Based Design Flow. This example shows a response time budget.

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.

6.3 Possible order of steps

Figure 6 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

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 6: Budget-based design steps

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.

6.4 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.

6.5 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.

6.6 Example of overlay budget for wafersteppers

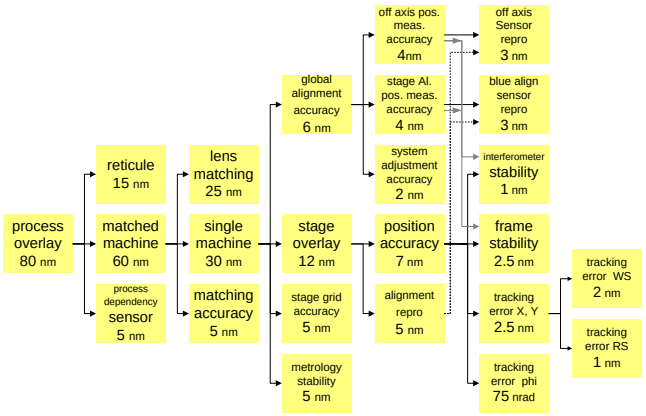


Figure 7: Example of a quantified understanding of overlay in a waferstepper

Figure 7 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 6. 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 7.

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.

6.7 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 6. The *visualization* was table based.

memory budget in Mbytes	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 8: Example of a memory budget

The rationale behind the budget can be used to derive *guidelines* for the creation of memory budgets. Figure 8 shows an example of an actual memory budget for a 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.

6.8 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.

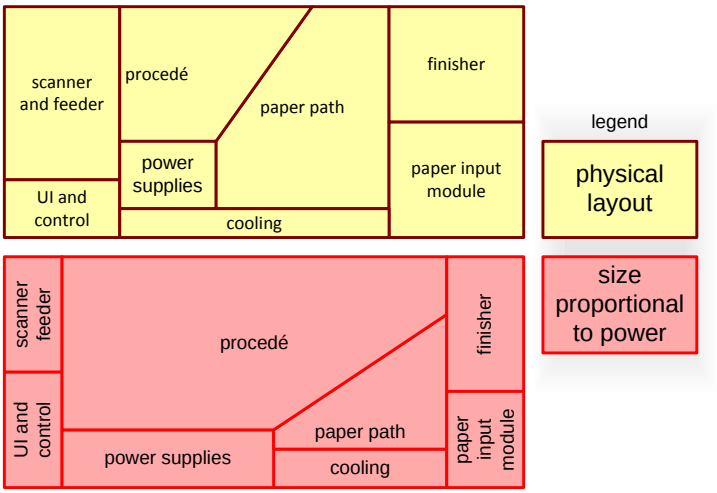


Figure 9: Power Budget Visualization for Document Handler

Figure 9 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.

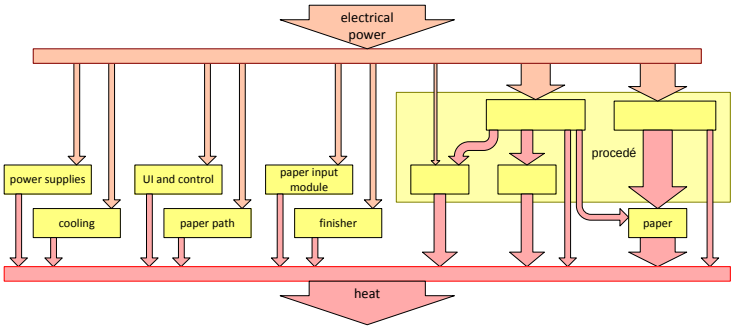


Figure 10: Alternative Power Visualization

Figure 10 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.

6.9 Research questions for budget-based design

Budgets are widely used in Systems Engineering, in many domains, and for many applications. However, the problems mentioned in Section 5 are also present for this methodology. Potential research questions for *budget-based design* research are:

- What are potential applications for budgets?
- What kind of budget is required?
- What is the decomposition to be used?
- How to manage margins?
- How to verify a budget?
- How to use and maintain a budget?
- Does it provide value when a budget is coupled to other design information?
- and many more...

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)

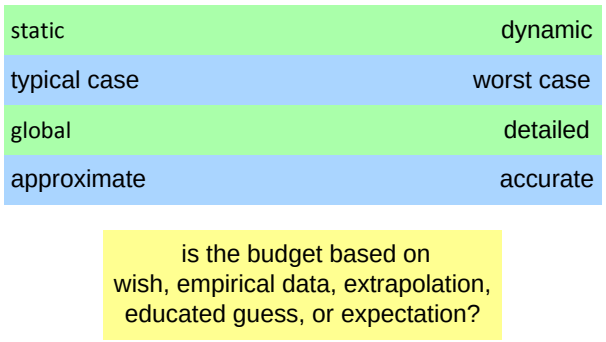


Figure 11: What kind of budget is required?

Figure 11 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.

7 How to Proceed?

As shown in the previous sections many multi-domain methods are implicitly used in industry. The implicit nature of these methods means that we don't know the limits and the validity of these methods, neither the broader opportunities for the application. A systematic approach is required to transform today's implicit use into explicit know-how. We propose to use the technology management cycle, as proposed in[1], and described in [3], to do systematic research of multi-domain methods.

The technology management cycle consists of three phases: *exploration*, *application*, and *consolidation*. If we add the activity needed at the moment of the phase transition then we get the following approach:

Consolidation of status quo Create case descriptions, more elaborated than in Section 6 of this paper. Extract an explicit description of the underlying

method, and the lessons learned. Make an inventory of open issues. As a spin-off this consolidated material must be transformed into educational material.

Articulate research question The status quo and the list of open issues are used to articulate a leveled set of research questions. A research question is a clear description of what we want to research. A stepwise refinement of the research question helps to cope with the dynamic range of the problem from very generic to highly specific; a high level research question refined in more specific subquestions.

Exploration In the exploration phase we search for:

- solutions for open issues
- promising alternative methods
- foundation and formalization of practically used methods
- alternative application possibilities of these methods.

Hypothesis At the end of the exploration phase the research objectives are captured in a hypothesis. A good hypothesis is very explicit with clear evaluation criteria. For research of multi-domain methods the hypothesis should address *who* will benefit from the method *how* these benefits are achieved in *what* context.

Application Application of methods in industrial context is needed to study the methods itself. Observation of the application in this industrial context provides the data needed for the evaluation afterwards.

Evaluation The observation results are analyzed and evaluated against the hypothesis.

Repeat the cycle starting again with consolidation

The iteration as proposed in the technology management cycle should start with the consolidation of the status quo. The huge amount of implicit know-how needs to be made explicit. The consolidated status quo is an excellent starting point to formulate the research questions. Starting immediately with exploration is dangerous, because most researchers are not aware of the status quo. The danger is that a lot of research is wasted, because it addresses non existing problems. Applied research is inspired by existing problems, in contrast to fundamental research that is driven by the need for more fundamental knowledge.

8 Conclusions

Many multi-disciplinary methods are already used in multiple domains. However, the consolidation of these multi-disciplinary methods is very poor; the amount of

papers and textbooks describing multi-disciplinary methods is rather limited. The education in these methods is inadequate. Also many research questions for these methods are still open.

Multi-disciplinary methods research is a huge research field.

We need a systematic research approach to cope with this huge research field. The technology management cycle offers a research approach for this type of applied research. The use of research questions, hypothesis and evaluations are means to force the researchers to be sharp and to the point.

The technology management cycle in combination with hypothesis and evaluation provides a systematic approach for applied multi-domain method research.

The challenge is to get sufficiently *generic* methods, such that a limited description and education facilitates designers in using the methods. At the same time the methods must be sufficiently *specific* to limit the amount of customization work and to reduce the required skills for the designers. are we able to provide a single *How-to* poster, or do we overwhelm the designer with a thick stack of books? Preferably a method tutorial fits in a 10 page article!

The challenge is to find the right abstraction level of the methods: generic *to facilitate a compact description and a broad education, but* specific *to facilitate the immediate use.*

9 Acknowledgements

Feedback from Boderc project members and ESI colleagues did help significantly to shape this article. Amongst the contributors are: Erik Gaal, Jozef Hooman, Anget Mestrom, Martin Prins, Martin Rem, Hans Spitshuis, Jan-Mathijs Wijnands, Berry van der Wijst.

References

[1] Hay Management Consultants. Technology management cycle. Hay Managament Consultants showed me this model in 1997/1998, taken from an article by a Japanese author. The original title of the Japanese article is unknown.

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

[3] Gerrit Muller. CAFCR: A multi-view method for embedded systems architecting: Balancing genericity and specificity. `http://www.gaudisite.nl/ThesisBook.pdf`, 2004.

History

Version: 0.8, date: March 4, 2005 changed by: Gerrit Muller

- changed status to concept
- minor textual changes

Version: 0.7, date: October 25, 2004 changed by: Gerrit Muller

- clarified multi-domain and multi-disciplinary
- changed status to draft
- rephrased assumptions of the research agenda

Version: 0.6, date: September 22, 2004 changed by: Gerrit Muller

- added text about recipes and tools to the section What is a method?
- added more text to the subsection lack of tools
- reworked the overlay budget example to follow the structure of the method description.
- reworked the memory budget example to follow the structure of the method description.

Version: 0.5, date: September 17, 2004 changed by: Gerrit Muller

- moved domain discussion to new section
- added short discussion of ESI research agenda assumptions
- added short description of CAFCR
- factored out a section What is a Method?
- reorganized the budget section, following the ingredients of a method

Version: 0.4, date: September 15, 2004 changed by: Gerrit Muller

- updated the abstract
- added reference to Boderc
- changed status to preliminary draft
- added experience of the author as the source of the observations
- many small textual improvements
- defined in the introduction what we use as domain
- moved text about mono-disciplinary methods to the introduction
- added more text about measurements in relation to budgetting
- made the four method characteristics metioned in section 2 more explicit in section 4.
- added slide with technology management cycle

Version: 0.3, date: September 8, 2004 changed by: Gerrit Muller

- added section "How to proceed?"
- added conclusion about approach to the conclusions.
- added power budget in document handling as third budget example

Version: 0.2, date: September 2, 2004 changed by: Gerrit Muller

- added explanation of domains to introduction
- added text about the relationship between mono-disciplinary and multi-disciplinary methods

Version: 0.1, date: August 30, 2004 changed by: Gerrit Muller

- added section Problems
- added text to section Examples
- added some text to waferstepper overlay budget example

Version: 0, date: August 17, 2004 changed by: Gerrit Muller

- Created, no changelog yet