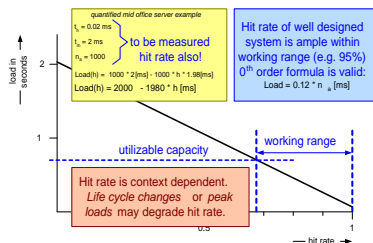


Modeling and Analysis: System Model



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 uses a web shop service as example system to construct a system model. The caching of pictures of the products in the shop is modeled to analyze performance, robustness, scalability and reliability of the system.

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

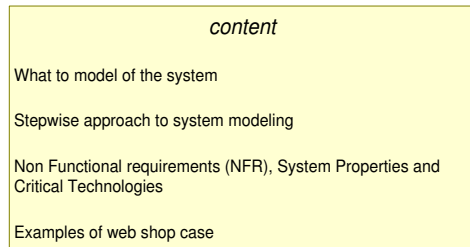


Figure 1: Overview of the content of this paper

Figure 1 shows an overview of this paper. We will discuss what to model of the system of interest, see also Figure 2. We will provide a stepwise approach to system modeling, based on the relations between Non Functional Requirements (NFR), system design properties and critical technologies. Several examples will be shown using the web shop case.

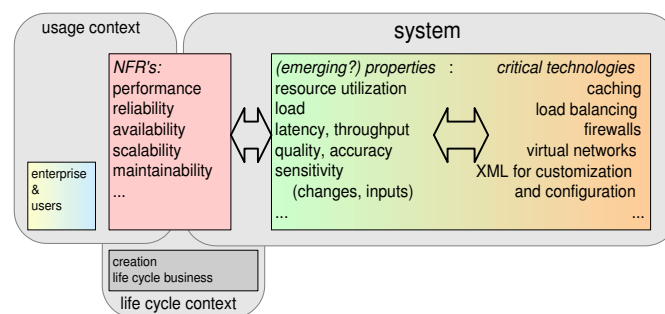


Figure 2: What to Model in System Context?

In our modeling we will focus on the NFR's, such as performance, reliability, availability, scalability, or maintainability. We assume that the functional requirements and system decomposition are being created in the system architecting and design activity. In practice many NFR's emerge, due to lack of time and attention. We recommend to reduce the risks by modeling and analysis of relevant NFR's where some higher risk is perceived. Figure 2 shows that the external visible system characteristics depend on the design of system properties, such as resource utilization, load, latency, throughput, quality, accuracy, or sensitivity for changes or varying inputs. Note that these properties also often emerge, due to lack of time or attention. Not only the design of these properties determine the external visible system characteristics, but also the chosen technologies has a big impact. Therefore we also have to look at critical technologies, for example caching, load balancing, firewalls, virtual networks, or XML for customization and configuration.

2 Stepwise approach to system modeling

We recommend an approach where first the system is explored: what is relevant, what is critical? Then the most critical issues are modeled. Figure 3 shows a stepwise approach to model a system.

1. determine relevant Non Functional Requirements (NFR's)
2. determine relevant system design properties
3. determine critical technologies
4. relate NFR's to properties to critical technologies
5. rank the relations in relevancy and criticality
6. model relations with a high score

Figure 3: Approach to System Modeling

- 1. Determine relevant Non Functional Requirements (NFR's)** where the relevance is often determined by the context: the usage context or the life cycle context.
- 2. Determine relevant system design properties** by looking either at the NFR's or at the design itself: what are the biggest design concerns?
- 3. Determine critical technologies** criticality can have many reasons, such as working close to the working range limit, new components, complex functions with unknown characteristics, sensitivity for changes or environmental conditions, et cetera.
- 4. relate NFR's to properties to critical technologies** by making a graph of relations. Such a graph often has many-to-many relations.
- 5. Rank the relations in relevancy and criticality** to find potential modeling candidates.
- 6. Model relations with a high ranking score** a time-boxed activity to build up system understanding.

Note that this system modeling approach fits in the broader approach of modeling and analysis. The broader approach is discussed in *Modeling and Analysis: Reasoning*.

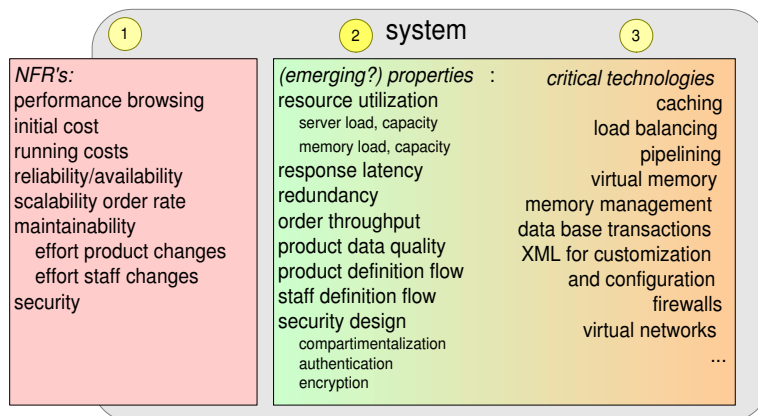


Figure 4: Web Shop: NFR's, Properties and Critical Technologies

3 Example system modeling of web shop

Figure 4 shows the results of step 1, 2, and 3 of the approach.

1. **Determine relevant Non Functional Requirements (NFR's)** For the web shop the following requirements are crucial: performance browsing, initial cost, running costs, reliability/availability, scalability order rate, maintainability (effort to enter product changes and effort to enter staff changes) and security.
2. **Determine relevant system design properties** based on experience and NFR's the following properties were identified as relevant: resource utilization (server load, server capacity, memory load, and memory capacity), response latency, redundancy, order throughput, product data quality, product definition flow, staff definition flow, security design (which can be refined further in compartmentalization, authentication, and encryption). Note that we mention here design issues such as *product definition flow* and *staff definition flow*, which have an direct equivalent in the usage context captured in some of the customer's processes.
3. **Determine critical technologies** Based on experience and risk assessment the following technologies pop up as potentially being critical: caching, load balancing, pipelining, virtual memory, memory management, data base transactions, XML for customization and configuration, firewalls, and virtual networks.

Figure 5 shows for a small subset of the identified requirements, properties and technologies the relations. The performance of browsing is related to the resource management design and the concurrency design to meet the response latency. The resource management design relates to several resource specific technologies from

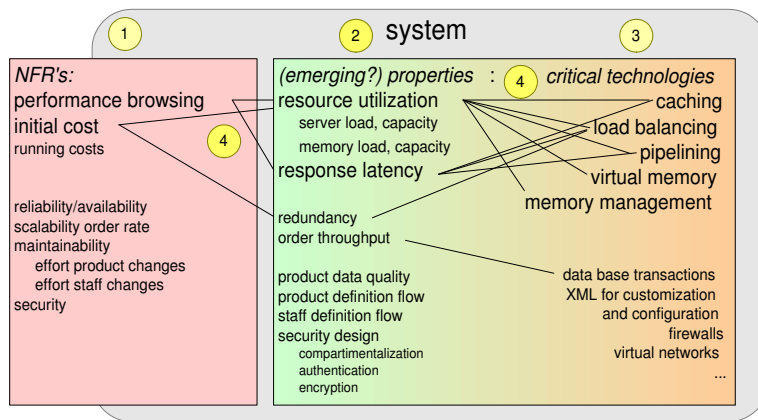


Figure 5: 4. Determine Relations

caching to memory management. The cost requirements also relate to the resource utilization and to the cost of redundancy measures. The dimensioning of the system depends also on the design of the order throughput. Crucial technology for the order throughput is the data base transaction mechanism and the related performance.

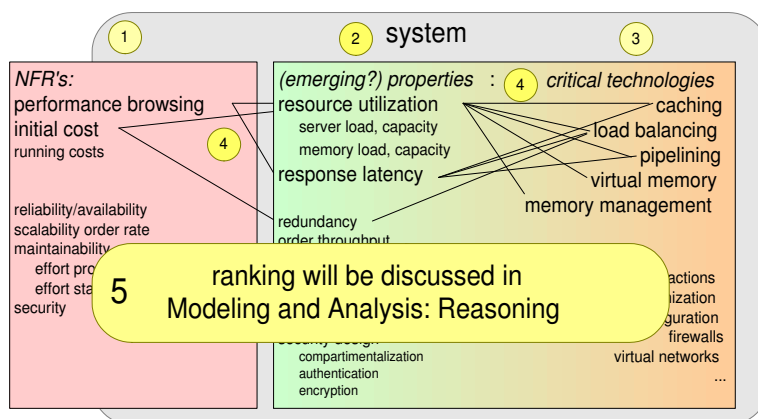


Figure 6: 5. Rank Relations

Ranking, Figure 6 will be discussed in the *Modeling and Analysis: Reasoning* paper. For this example we will mostly focus on the relations shown in Figure 5.

Figure 7 shows the picture cache as a specific example of the use of caching technology. The purpose of picture cache is to realize the required performance of product browsing at the client layer. At the web server layer and at the data base layer the picture cache should realize a limited server load of the product exhibition function.

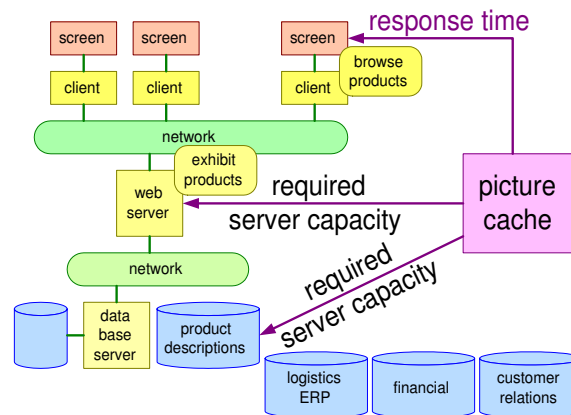


Figure 7: Purpose of Picture Cache Model in Web Shop Context

zero order web server load model

$$\text{Load} = n_a * t_a$$

n_a = total requests
 t_a = cost per request

Figure 8: Zero Order Load Model

The most simple model we can make for the server load as function of the number of requests is shown in Figure 8. This is a so called zero order model, where only the direct parameters are included in the model. It is based on the very simple assumption that the load is proportional with the number of requests.

When we introduce a cache based design, then the server load depends on the effectiveness of the cache. Requests that can be served from the cache will have a much smaller server load than requests that have to be fetched from the data base. Figure 9 shows a simple first order formula, where the contributions of requests from cache are separated of the requests that need data base access. We introduce an additional parameter h , the hit-rate of the cache. This helps us to create a simple formula where the server load is expressed as a function of the hit-rate.

The simple mathematical formula starts to make sense when we instantiate the formula with actual values. An example of such an instantiation is given in Figure 10. In this example we use values for request handling of $t_h = 20\mu s$ and $t_m = 2ms$. For the number of requests we have used $n_a = 1000$, based on the assumption that we are serving the product exhibition function with millions of customers browsing our extensive catalogue. The figure shows the server load as function of the hit-rate. If the available server capacity is known, then we can deduce the minimal required hit-rate to stay within the server capacity. The allowed

first order web server load model

$$\text{Load} = n_{a,h} * t_h + n_{a,m} * t_m$$

$n_{a,h}$ = accesses with cache hit
 $n_{a,m}$ = accesses with cache miss
 t_h = cost of cache hit
 t_m = cost of cache miss

$$n_{a,h} = n_a * h$$

$$n_{a,m} = n_a * (1-h)$$

n_a = total accesses
 h = hit rate

$$\text{Load}(h) = n_a * h * t_h + n_a * (1-h) * t_m = n_a * t_m - n_a * h * (t_m - t_h)$$

Figure 9: First Order Load Model

range of hit-rate values is called the working range.

In Figure 11 we zoom in on the hit-rate model. First of all we should realize that we have used assumed values for t_h , t_m and n_a . These assumptions were based on experience, since we know as experienced designers that transactions cost approximately 1 ms, and that the cache roughly improves the request with a factor 100. However, the credibility of the model increases significantly by measuring these quantities. Common design practice is to design a system well within the working range, for example with a hit-rate of 95% or higher. If the system operates at these high hit-rates, then we can use the zero-order formula for the system load again. Using the same numbers for performance we should then use $t_a \approx 0.95 * t_h + 0.05 * t_m \approx 0.12ms$. Another assumption we have made is that the hit-rate is constant and independent of the circumstances. In practice this is not true. We will, for instance, have varying request rates, perhaps with some high peak values. If the peak values coincide with lower hit rates, then we might expect some nasty performance problems. The hit-rate might also be impacted by future life cycle changes. For example new and different browser functionality might decrease the hit-rate dramatically.

Another system property that was characterized as relevant was the response time design. Response time depends on the degree of concurrency, the synchronization design and the time required for individual operations. Figure 12 shows a timing model for the response time, visualizing the above mentioned aspects for the retrieval of a picture in case of a cache miss.

Yet another system design property is the use of resources, such as memory. Figure 13 shows the flow of pictures throughout the system, as a first step to address

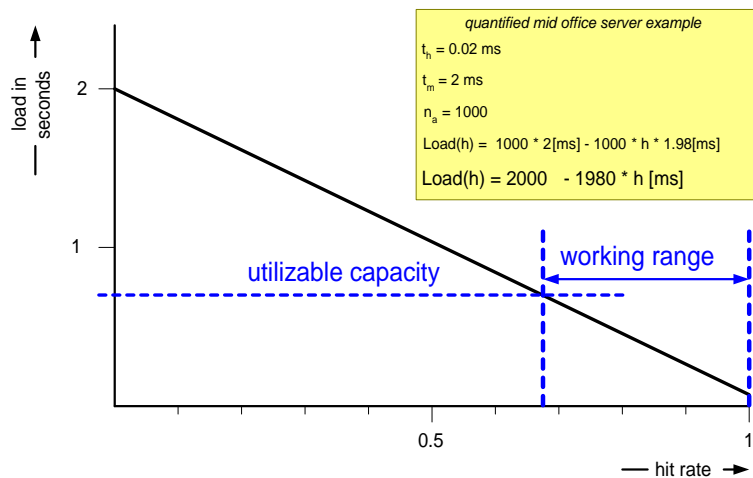


Figure 10: Quantification: From Formulas to Insight

the question how much memory is needed for picture transfers.

In Figure 14 we zoom in on the web server to have a look at the memory used for picture transfers. This figure shows a number of alternative design options, ranging from a minimal set of copies in the memory to the more realistic situation where every thread contains multiple copies of every picture, while at the same time multiple threads serve concurrent customers.

These alternative design options are transformed in a simple mathematical model in Figure 15. This formula is parametrized for the different specification and design parameters:

n = number of data base access threads a design parameter dimensioning the amount of concurrency towards the data base layer.

m = number of picture cache threads a design parameter dimensioning the amount of concurrency of the picture cache itself.

k = number of web server threads a design parameter dimensioning the amount of concurrency of client access to the web server.

s = picture size in bytes an application and design dependent parameter, the average size in bytes of the pictures.

c = in memory cache capacity in number of pictures a design parameter determining the size of a picture cache in number of pictures per picture cache thread.

This formula is instantiated in a quantified table in Figure 16, for different values of the design parameters. Note that depending on the chosen design param-

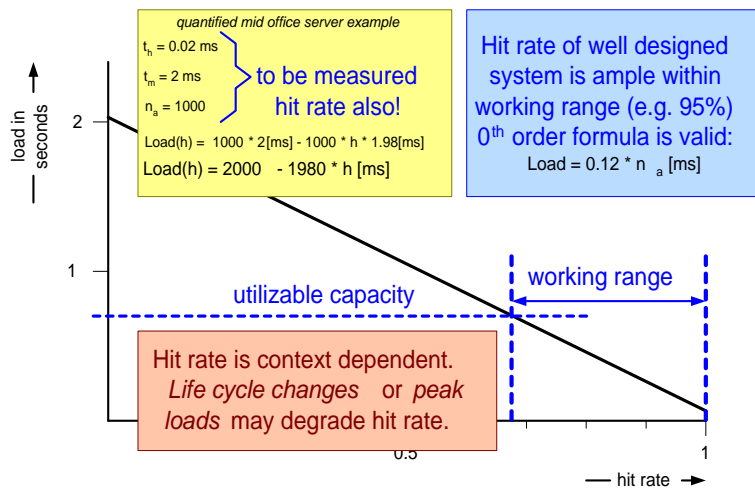


Figure 11: Hit Rate Considerations

eters the picture cache maps on completely different storage technologies, with the related different performance characteristics.

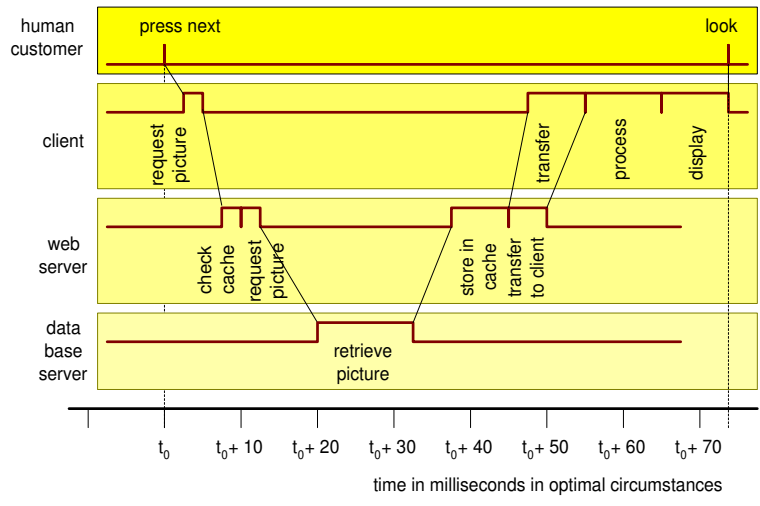


Figure 12: Response Time

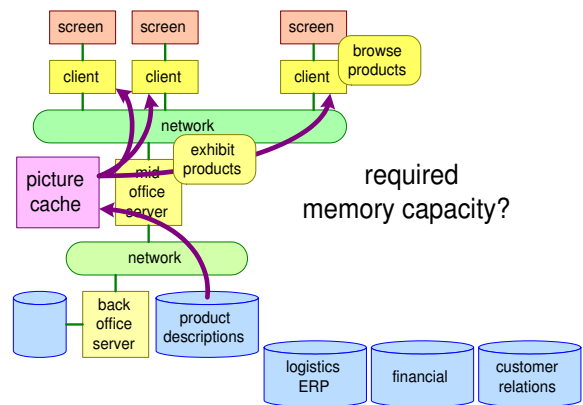


Figure 13: What Memory Capacity is Required for Picture Transfers?

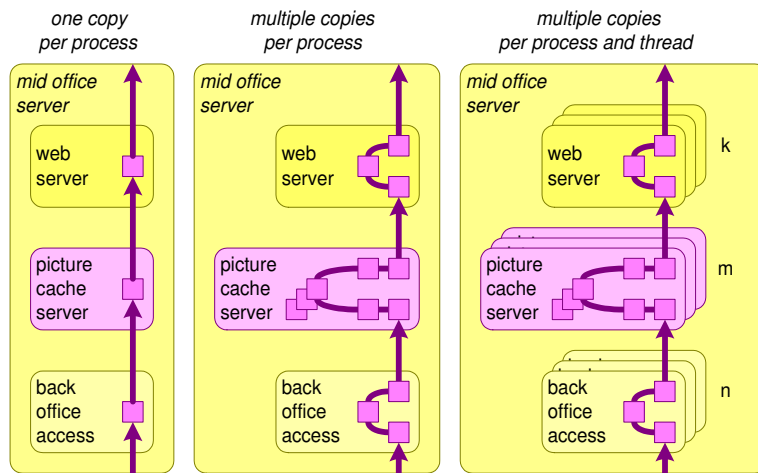


Figure 14: Process view of picture flow in web server

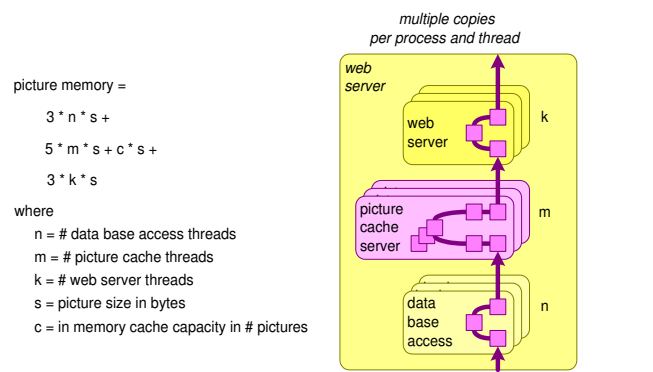


Figure 15: Formula memory Use Web Server

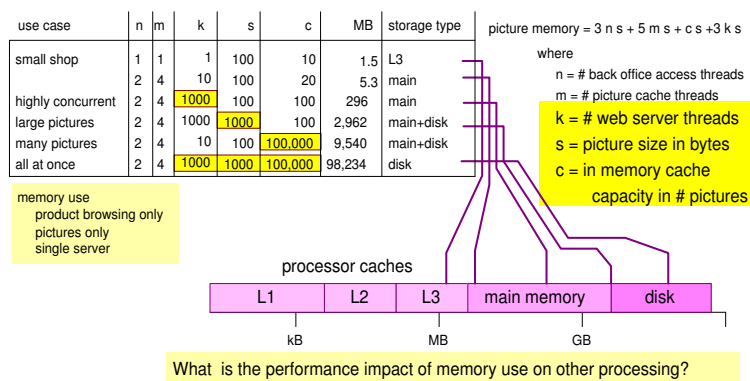


Figure 16: Web server memory capacity

4 Discussion

In the previous section we have modeled a few parts of the system. Figure 17 shows that we have covered so far a small part of the system space. We can describe the system space by several dimensions: functions, data and aspects. Our coverage so far has been limited to the browse and exhibit products function, looking at the pictures as data, looking at the aspects of server memory use, response time and server load.

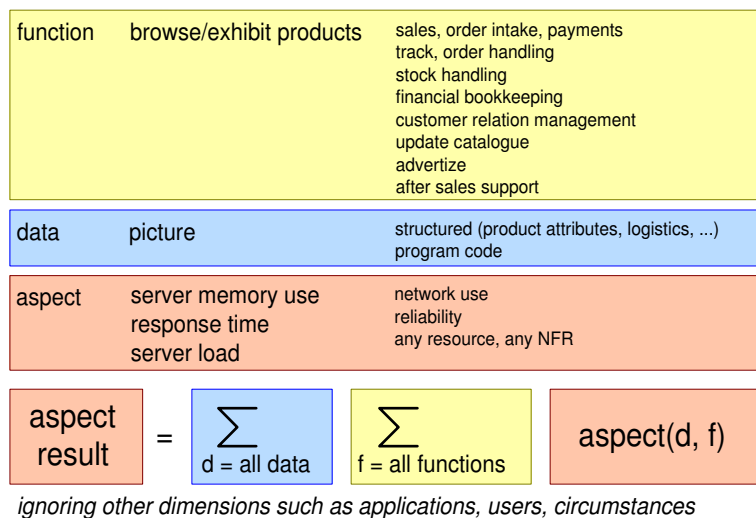


Figure 17: Only a small part of the system has been modeled so far

This figure shows many more functions, types of data and aspects present in systems. To answer one of the NFR like questions we have to combine the aspect results of functions and all data types. In practice the context also impacts the NFR's, we have still ignored applications, users, and circumstances.

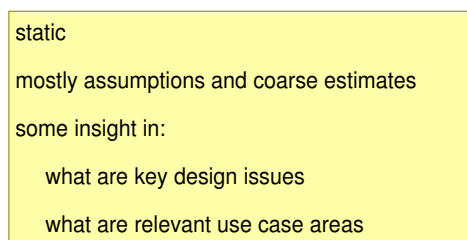


Figure 18: The modeling so far has resulted in understand some of the systems aspects

Figure 18 adds to this by reminding us that we so far have only made static

models, mostly based on assumptions and coarse estimates. Nevertheless we have obtained some insight in key design issues and relevant use cases.

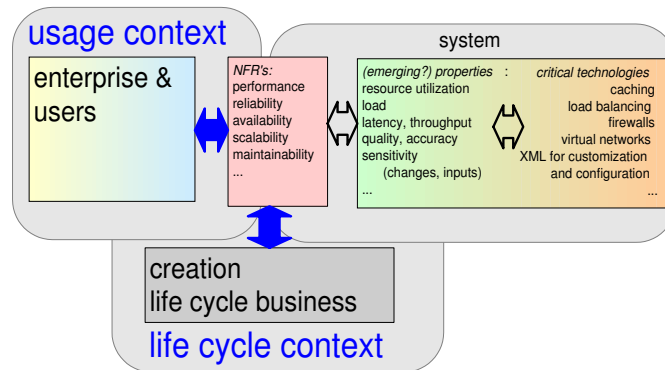


Figure 19: Refinement of the system models takes place after context modeling

We are far from finished with system modeling. However, with the results obtained so far it is important to take the next step of the broader iteration: modeling of the contexts, see Figure 19. Many of the models we have made of the system trigger questions about the system use and the life cycle. What is the expected amount of browsing, by how many customers, for what size of catalogue? What is the preferred picture quality? How relevant is the maintenance effort related to the product catalogue? et cetera.

5 Summary

<i>Conclusions</i>
Non Functional Requirements are the starting point for system modeling
Focus on highest ranking relations between NFR's and critical technologies
Make simple mathematical models
Evaluate quantified instantiations
<i>Techniques, Models, Heuristics of this module</i>
Non functional requirements
System properties
Critical technologies
Graph of relations

Figure 20: Summary of system modeling

Figure 20 shows a summary of this paper. We have shown that Non Functional Requirements are the starting point for system modeling. Our approach focuses on the highest ranking relations between NFR's and critical technologies. For these relations we make simple mathematical models that are evaluated by quantified instantiations of these models.

6 Acknowledgements

Roelof Hamberg caught several errors in the detailed models

References

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

History

Version: 0.4, date: March 6, 2009 changed by: Gerrit Muller

- added acknowledgements
- corrected MASMquantified. MASMdiscussion
- calculated t_a for zero order formula

Version: 0.3, date: February 26, 2007 changed by: Gerrit Muller

- added slide with formula for amount of memory in web server

Version: 0.2, date: February 23, 2007 changed by: Gerrit Muller

- added What to model
- added stepwise approach
- added web shop steps 1 to 5
- added follow up
- added content and summary
- added text version
- changed status to preliminary draft

Version: 0.1, date: January 5, 2007 changed by: Gerrit Muller

- choose logo
- Added Memory Capacity
- added response time
- added system space
- added conclusion

Version: 0, date: January 2, 2007 changed by: Gerrit Muller

- Created, no changelog yet