Experience with Global Analysis: A Practical Method for Analyzing Factors that Influence Software Architectures

Robert L. Nord[1], Dilip Soni
Siemens Corporate Research
755 College Road East
Princeton, New Jersey 08540 USA
rn@sei.cmu.edu, dilip@scr.siemens.com

Abstract

A practical method for analyzing the factors that influence software architectures is presented. Factors include organizational context and constraints, available technologies, and product requirements. Analyzing the factors uncovers a small number of issues that drive the design of the architecture. These issues arise from the factors that have little flexibility, a high degree of changeability, and a global impact on the system. The result of the analysis is a set of global strategies that guide the architecture design.

A two-phase approach for analyzing factors and developing architecture design strategies is given. Experience has been gained with this approach in three ways: (1) developing the approach during the design of an imaging system; (2) using the approach to analyze four systems in retrospect; (3) using the approach in new software development projects.

Introducing global analysis into the software development process resulted in a new global analysis specification document that helped bridge the gap between requirements and architecture design and provided a place to explicitly record design rationale.

1. Introduction

Global analysis analyzes factors that globally influence the architecture design of a system. Factors include organizational context and constraints, available technologies, and product requirements. This analysis focuses on key issues that transcend boundaries between development activities, subsystems, and architecture views. The result of the analysis is a set of global strategies that guide the architecture design and improve its adaptability with respect to changes in the factors.

Successful projects prepare for change by noting the flexibility of influencing factors and their likelihood of change, characterizing interactions among the factors and their impact, and selecting cost-effective design strategies to reduce the expected impact of the changes [8].

Three categories of influencing factors are considered during global analysis: organizational, technological, and product.

Organizational factors arise from the business organization. Organizational factors constrain the design choices while the product is being designed and built. They are external to the product, but influence it. Their influence is important because if they are ignored, the architecture may not be buildable.

External technology solutions are embedded or embodied in the product. These factors are primarily hardware and software technologies and standards. These technological factors are external to the product being designed. Unlike the organizational factors, however, they can affect the product throughout its lifetime. Further, they can change over time, so the architecture should be designed with this changeability in mind.

Product factors are used to describe the product's requirements for functionality, the features seen by the user, and nonfunctional properties. The product factors are also subject to change over time, so the architecture should be designed to support such changes.

In this paper, we present the concept of global analysis, a practical method for analyzing factors that influence software architectures. We demonstrate its role in software architecture design and discuss its relationship to other software development activities. We present our experience with developing the method and its use by others in new software development projects. We conclude with lessons learned about the method's value and where further improvement is needed.

[1] Current Address: Software Engineering Institute, 4500 Fifth Avenue, Pittsburgh, Pennsylvania 15213USA.

2. Related Software Development Activities

Figure 1 shows the relationship of global analysis to software architecture design and project planning activities.

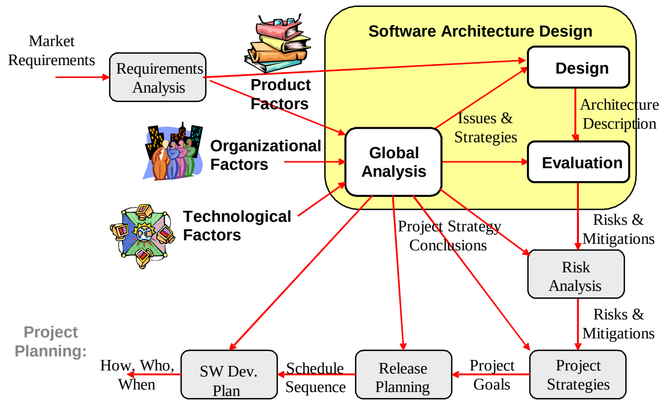

Figure 1: Software Architecture Design and Project Planning Activities

Global analysis complements requirements analysis tasks. Global analysis helps focus on the important architecture requirements; these are the quality attribute requirements. But global analysis goes further than just examining requirements; it includes organizational and technological factors that are not typically included in the requirements document.

The method helps bridge the gap between requirements and architecture design by analyzing the impact of requirements on important technical and business issues that affect design. Global analysis records rationale and provides traceability as requirements are linked to strategies that guide design.

The description of requirements is often textual, but more rigorous requirements analysis methods may employ some combination of feature modeling [6], use case modeling, or object modeling [5]. If such an approach is used, then the artifacts will provide useful input to the global analysis method. Features will be put in global analysis factor tables for further analysis. Use cases show a specific interaction between a stakeholder and the system and provide a means to evaluate the impact of the design decisions in providing a solution to the design issue. Objects encapsulate system responsibilities and will inform the choice of conceptual components in the global analysis strategies that guide the design.

Global analysis generates issues and strategies that guide architecture design and provide input to architecture evaluation. Global analysis begins as the architecture is defined and continues as the design decisions are made. Figure 2 shows the iterative nature between global analysis and the design tasks for any given architectural view. Global analysis guides design decisions. As design decisions are made, additional constraints may arise that are in turn analyzed and in turn guide additional design decisions.

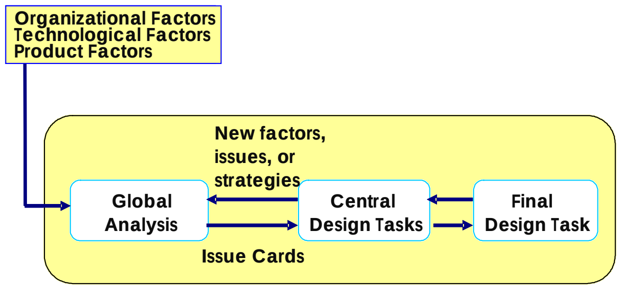

Figure 2: Architecture Design

Global analysis complements architecture evaluation tasks, such as the Architecture Tradeoff Analysis Method (ATAM) [3]. Often, much time is spent at the beginning of the evaluation capturing information about relevant business drivers, quality attribute requirements, and architectural approaches. Rather than record these after the fact, the best time to capture them is as they are made during the design activity. Global analysis captures this information and provides design strategies and their rationale that can be reviewed during the ATAM. ATAM will uncover risks for which additional strategies may need to be developed.

Global analysis provides input to project planning and management activities. It is used to generate project strategy conclusions that help define project goals [10].

3. Global Analysis Activities

The global analysis method consists of two phases: Analyze the factors and Develop issues and strategies.

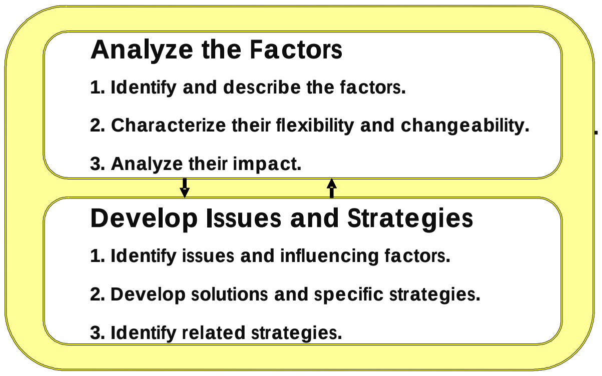
Analyze the Factors
1. Identify and describe the factors.
2. Characterize their flexibility and changeability.
3. Analyze their impact.
Develop Issues and Strategies
1. Identify issues and influencing factors.
2. Develop solutions and specific strategies.
3. Identify related strategies.

Figure 3: Global Analysis Activities

The process is iterative and may start with either phase.

Phase 1: Analyze the Factors: The first phase analyzes the factors using three steps: (1) Identify and describe the factors; (2) Characterize the flexibility or the changeability of the factors; and (3) Analyze the impact of the factors.

Identify and describe the factors: Consider factors that have a significant global influence, those that could change during development, those that are difficult to satisfy, and those with which you have little experience. Can the factor's influence be localized to one component in the design, or must it be distributed across several components? During which stages of development is the factor important? Does the factor require new expertise?

Characterize the flexibility of the factors: Describe what is negotiable about the factor. Is it possible to influence or change the factors so that it makes your task of architecture development easier? Use this information when factors conflict or for some other reason become impossible to fulfill.

Characterize the changeability of the factors: Describe what could change about the factor, both in the near and more distant future. In what way could the factor change? How likely is it to change during or after development? How often will it change? Will the factor be affected by changes in other factors?

Analyze the impact of the factors: If the factor will change, which of the following would be affected and how: other factors, components, modes of operation of the system, other design decisions.

Phase 2: Develop Issues and Strategies: The second phase develops strategies for the architecture design using three steps: (1) Identify issues; (2) Develop solutions and specific strategies; and (3) Identify related strategies.

Identify issues: An issue may arise from factors in many ways:

- limitations or constraints
 (e.g., Aggressive Schedule)
- reducing the impact of changeability
 (e.g., Changes in Software Technology)
- difficult-to-satisfy product factors
 (e.g., Easy Addition and Removal of Features)
- common solution to global requirements
 (e.g., Implementation of Diagnostics)

Develop solutions and specific strategies: Discuss a general solution to the issue, followed by a list of associated strategies. The solution description records analysis-based rationale that illustrates that the strategies satisfy the issue. Strategies should address the issue and one or more of the following goals:

- reduce or localize the factors' influence
 (e.g., Buy rather than build)
- reduce the impact of the factors' changeability
 (e.g., Use a pipeline for image processing)
- localize required areas of expertise (e.g., Map independent threads of control to processes)
- reduce overall time and effort
 (e.g., Use incremental development)

Identify related strategies: When a strategy belongs to more than one issue, describe it in one place and reference it as a related strategy in the other issues where it applies.

4. Experience with Developing the Method

We developed the approach informally while designing the architecture of an image acquisition and processing system. After the conclusion of the project, we developed a more rigorous description of the method and provided an example of its use in terms of a fictional system we call IS2000, inspired by this and other systems we studied [4]. The IS2000 system consists of a probe that takes sensor readings that are processed according to the type of acquisition procedure selected by the user. The results of the first phase are documented in a factor table. We illustrate the factor table with an excerpt from IS2000.

Factor	*Flexibility/ Changeability*	*Impact*
O4.2 Schedule Feature Delivery		
Features are prioritized	Negotiable	Moderate impact on the schedule
T2.1 Domain-specific Hardware Probe Hardware		
Hardware to detect and process signals	Upgraded every three years as technology improves	Large impact on image acquisition and processing components
P1.1 Features Acquisition Types		
Acquire raw signal data and convert into images	New types of acquisitions may be added every three years	Affects UI, acquisition performance, and image processing

The organizational factor (O4.2) shows there is flexibility in delivering features according to their priority. For other systems these kinds of factors may not affect the architecture, but in this system they will have a significant impact. The technological feature (T2.1) shows that change to the probe hardware is likely and will have a large impact on the imaging components. The product factor (P1.1) shows new types of acquisition algorithms may be added during the lifetime of the system.

The results of the second phase are documented in an issue card. We illustrate an issue card from IS2000.

Issue: Easy Addition and Removal of Acquisition Procedures

There are many acquisition procedures. Implementation of each feature is quite complex and time consuming. There is a need to reduce complexity and effort in implementing such features.

Influencing Factors

O4.1: Time to market is short
O4.2: Delivery of features is negotiable

P1.1: New acquisition procedures can be added every three years.
P1.2: New image-processing algorithms can be added on a regular basis.
…

Solution
Define domain-specific abstractions to facilitate the task of implementing acquisition and processing applications.
Strategy: Use a flexible pipeline model for image processing.
Develop a flexible pipeline model for implementing image processing. Use processing components as stages in the pipeline. This allows the ability to introduce new acquisition procedures quickly by constructing pipelines using both old and new components.
Strategy: Introduce components for acquisition and image processing.
…
Strategy: Encapsulate domain-specific data.
…

Related Strategies
See also **Encapsulate domain-specific hardware**.

We performed a retrospective analysis on four systems with the aid of the architects who designed the systems [4][9]. We interviewed the architects to understand the process they used to go from requirements to design. We solicited feedback on the approach to ensure that the artifacts captured the design rationale of their systems.

These systems come from domains such as instrumentation and control, signal processing, central monitoring, and communication. They vary in size, complexity, and have different system characteristics that influenced the architecture design such as fault tolerance, multiprocessing, safety critical, real-time performance, interoperability, distribution, heterogeneity.

The following table lists typical categories of influencing factors based on our observations. Within each category there will be a number of factors. For example, the schedule (O4) will record the time to market and how features are to be delivered; performance (P3) will record latency and bandwidth considerations.

Organizational	*Technological*	*Product*
O1: Management	T1: General-purpose Hardware	P1: Features
O2: Staff Skills	T2: Domain-specific Hardware	P2: User Interface
O3: Development Environment	T3: Software Technology	P3: Performance
O4: Schedule	T4: Architecture Technology	P4: Recovery
O5: Budget	T5: Standards	P5: Diagnostics

The following table gives an indication of the kinds of strategies we found in the systems we examined.

Organizational	*Technological*	*Product*
Reuse existing components	Encapsulate hardware	Use feature-based components
Build rather than buy	Separate processing, control, and data	Separate the user interaction model
Make it easy to add or remove features	Use vendor-independent interfaces	Separate time-critical components

5. Experience with Using the Method

We have taught the global analysis method in courses and have observed its use as it has been applied to four additional systems as part of a forward-engineering software development process.

	A	*B*	*C*	*D*
Application	data mgt.	image mgt.	business mgt.	automation mgt.
Factors				
Org.	14	9	28	28
Tech.	8	7	22	14
Product	7	11	28	25
Issues	11	3	19	23
Strategies	24	21	100	64

System A is representative of the way global analysis was applied. System A is a software system for acquiring and processing meter data from electrical, gas, and water meters [10]. System A performs calculations on the meter data and the results are sent to a utility's billing system. A global analysis specification was produced.

Factor tables were adopted as is. They are recorded in tables in a global analysis specification document. Columns record the factor name, description, flexibility and changeability, and impact.

Experience with System A provided evidence of the generality of the original collection of factors and categories. The author of the global analysis document was able to cut and paste many of the factors from the IS2000 system and make minor modifications to adapt the analysis to his situation. An example of such a technological factor was the database system. Although marketing specified Oracle 8 be used it was known that it would change over time. New database versions would become available and some customers would prefer databases from other vendors. The strategy for dealing with this factor was to design a layer in the architecture to isolate and encapsulate the database so that the effect of changes could be localized and accommodated in the future.

Experience with System A reinforced the importance of considering organizational factors in addition to traditional requirements and enhanced the collection of project management strategies. An example of such an organizational factor was that company management wanted to get the product to market as quickly as possible. Since the market was changing rapidly, it was important to provide users with a subset of features so that they can provide feedback. The strategy employed to address this factor was to develop products incrementally so that scheduled release dates could be met.

Experience with System A suggested improved support for additional topics such as product lines. An example of such a product factor was to support a product line in the market place. The graphical user interface must accommodate many types of users for different applications. A web-based GUI was employed so that additional flexibility could be achieved as new applications are added and location independence achieved for the various user populations. The performance of the system must scale for higher-end applications so a scalable distributed platform was necessary to meet these more stringent calculation time requirements.

A summary of issues and strategies was documented. The summary provided a listing of the issue name with a short description, factor cross-reference by number, and strategy name. Issue cards were not documented.

The strategies have implications for the project management. Strategies were analyzed and consolidated to develop project strategy conclusions about how the system should be designed and developed. This short list of major project strategies served as guiding principles for all the development team members. These project strategies helped define the project goals and risks that must be mitigated for success.

System B is similar in scope to System A and yielded similar conclusions. Systems C and D continued to expand our repertoire of factors and strategies; but the large number of factors and strategies that needed to be considered challenged us to think about new ways of managing and ordering this information. We address this in the following section where we discuss lessons learned.

6. Lessons Learned

What value did global analysis add that wasn't present before global analysis was used?

Introducing global analysis into the software development process of new projects resulted in a global analysis specification document that helped bridge the gap between requirements and architecture design and provided a place to explicitly record design rationale. The process of global analysis also can be used to build stakeholder consensus. In one case, a global analysis workshop was held to elicit feedback from stakeholders, discuss conflicting stakeholder requests and possible tradeoffs, and prioritize the factors.

Global analysis strategies advocated the adoption of an architectural pattern or style, provided design guidelines (encapsulation, separation of concerns), placed constraints on elements of the systems, or introduced additional structure. In essence, the strategies yielded a set of constraints on the architecture design in terms of prescribing a collection of component types and their patterns of interaction. These building blocks were developed from software engineering principles and the experience of building previous products. Component types, their relationships, properties, and constraints define an architectural pattern or style. As experience grows these patterns may be codified and the architect could select common patterns from a repository. The patterns embody a set of predefined design decisions. Constraints that emerge during global analysis could be used to select the appropriate ones.

Another benefit is improved documentation of the system. Design decisions between and within views of the architecture and the supporting rationale are recorded. The strategies are linked backward to requirements and forward to design decisions to provide traceability and validation [2].

In addition to guiding architecture design, it was not surprising to see the outputs of global analysis used by project management, since architecture plays a central role in software development activities. Issues and strategies provide input for project strategies that are used in release planning and scheduling in the software development plan. Issues also capture risks that the project manager is interested in tracking. Global analysis helps identify project and technical risks and suggest strategies for mitigating them.

What should be changed as a result of using global analysis in practice?

Many of the systems we examined had characteristics of product lines. Global analysis takes on an even more prominent role in product line design. The architect must characterize how the influencing factors vary among the products within a product line. The architect develops and selects strategies in response to these factors to make global decisions about the architecture that allows the developers of the products to make uniform decisions locally. Guiding the developers in this way ensures the integrity of the architecture. This is an iterative process. During the design, certain decisions feed back into the global analysis, resulting in new strategies.

Since product lines focus on variations among products, it would be advantageous to have separate columns for flexibility, changeability, and variation so that more guidance can be offered and the characterization and its type of impact can be more precisely captured.

Strategies suggest solutions for addressing a problem highlighted by an issue. As the architect selects a strategy, it is being evaluated in a continuous activity that we call global evaluation. Later on, these decisions could be evaluated during an architecture evaluation exercise. It would be beneficial while the issue is being articulated to also link it to an evaluation technique such as scenarios that would provide criteria for successfully meeting the requirement. It makes sense to do so as the issue is being formed and input gathered from the architect and relevant stakeholders rather than being captured after the fact during an evaluation exercise.

What wasn't used from global analysis and needs better elaboration?

Issue cards were not explicitly documented. The information they were meant to capture is therefore missing: text describing the problem and explaining tradeoffs and the degree of difficulty, text describing the factors in relation to the problem, and the solution statement.

Instead of the issue cards, a summary of issues and strategies table was used. This could be because the first time global analysis was used the document was written by the project manager. This experience showed the need for two views of the global analysis information. Using the summary of strategies served the project management view well, but trying to use it for the architecture view in lieu of the issue cards resulted in a number of problems. This was seen in a subsequent project where an architect used the global analysis document of the first as a template.

A problem with not using issue cards is that the summary table is not easy to read, especially the factor numbers. Instead of using numbers, it would be more readable to include the factor name with a link to the factor description and analysis. Issue cards help cross-reference information among the factors relevant to particular issues. Without their use, the factor table is used to pick up the slack. But because it was not designed for this purpose, the global tradeoffs and issues are more difficult to discern. For example, factor tables are used to address tradeoffs, such as schedule vs. quality and function. The impact column is used to address analysis and the solution. Issues tend to get grouped into factor categories instead of being cross-cutting across factors.

What needs further study for improving the global analysis method?

Issue cards were inspired by design patterns [7]. Further study and codification of the artifacts is needed to see them effectively adopted in practice.

A catalog of common factors, issues, and strategies is emerging. The original list of factors and categories was not meant to be exhaustive but illustrative. These factors were inspired by standards such as ISO/IEC 9126, the SEI taxonomy on software development risks, and our experience with numerous case study systems. Some of the additional factors we have seen include: legacy systems, global development, project engineering (for product lines), internet architecture technology (e.g., middleware, clients, and servers), scalability, and usability.

Similarly the list of issues and strategies were meant to be illustrative. Strategies are drawn from software engineering principles (loose coupling and high cohesion, separation of concerns, encapsulation), heuristics, patterns, and styles. As experience grows these strategies may be codified [1].

It would be useful to identify a core set of factors, issues, and strategies applicable to all systems. They could be used to derive a global analysis checklist used in conjunction with a template that the architect would use as an integral part of design and not be viewed as an extra documentation obligation.

A better articulation of the solution field in the issue card is needed, explaining the dependencies and tradeoffs among the strategies and how they might be used separately or in conjunction with one another.

There is value in creating a global analysis document at the beginning of architecture design to support management functions. However, global analysis is not meant to be a static document but one that evolves as the architecture is designed. The architect needs better support in this iterative process.

The global analysis data needs to be presented in different ways to different stakeholders. For example, we saw examples of how strategies were grouped by issues, by project recommendations and by architecture structure that they influence.

7. Conclusions

This paper has presented our experiences with a practical approach for analyzing the factors that influence software architecture. Approaches we have observed tend to focus on the functional requirements. But it is the quality attributes and constraints from the organization and the underlying technology that most strongly shape the architecture. These organizational, technological, and product factors are analyzed in global analysis. We have presented examples of factors based on experience and see a role for a catalog of such factors.

Global analysis helps the architect make the conceptual leap from the requirements to architecture design. Global analysis identifies factors that influence the architecture and yields a set of constraints on a collection of architecture design element types and their patterns of interaction. Global analysis also helps the architect record design decisions made between and within views of the architecture and the supporting rationale.

These factors are constantly changing. We found that successful architects analyze factors that have a global influence to produce an architecture that localizes the effects of change. Global analysis aids the architect in designing for change and building flexibility into the software.

To help the architect in this process, we have provided a two-phase approach for analyzing factors and developing strategies. The process is iterative and may start with either phase. We have provided factor tables and issue cards to capture the information.

We have validated and gained experience with this approach in three ways. First we developed the approach informally while designing the architecture for an image acquisition and processing system. Second, we did a retrospective analysis of four existing systems, interviewing the architects to understand the process they used to go from requirements to design, and getting their feedback on the resulting global analysis approach and the artifacts captured for their systems. Third, global analysis is being taught in courses and used in new software development projects. The result is the production of global analysis documents that are used by the architect, project manager, and other stakeholders. The benefits they have realized include: documented factors and design strategies that guide the architecture design; inputs for developing project strategy conclusions, goals, and risks; and improved documentation of the architecture. These applications give us confidence that the approach is practical and helpful.

8. References

[1] Bass, L., M. Klein, F. Bachmann, "Quality Attribute Design Primitives and the Attribute Driven Design Method." 4th Conference on Product Family Engineering. Bilbao, Spain, 4 October 2001.

[2] Clements, P., F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, J. Stafford. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, Boston, 2002.

[3] Clements, P., R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*, Addison-Wesley, Boston, MA, 2002.

[4] Hofmeister, C., R. Nord, D. Soni, *Applied Software Architecture*, Addison-Wesley, Reading, MA, 2000.

[5] Jacobson, I., M. Griss, and P. Jonsson, *Software Reuse: Architecture Process and Organization for Business Success*, Addison Wesley Longman, New York, NY, 1997.

[6] Kang, K.C., S.G. Cohen, J.A. Hess, W.E. Novak, and A.S. Peterson, *Feature-Oriented Domain Analysis Feasibility Study* (CMU/SEI-90-TR-21), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1990.

[7] Meszaros, G., and J. Doble, *A Pattern Language for Pattern Writing*, 1997. URL: http://www.hillside.net/patterns/

[8] Nord, R.L., C. Hofmeister, D. Soni, "Preparing for Change in the Architecture Design of Large Software Systems," Position paper accepted at the *TC2 First Working IFIP Conference on Software Architecture (WICSA1)*, 1999.

[9] Nord, R.L., "Meeting the Product Line Goals for an Embedded Real-Time System," In *Proceedings of the 3rd International Workshop on the Development and Evolution of Software Architectures of Product Families*, 2000.

[10] Paulish, D.J., *Architecture-Centric Software Project Management: A Practical Guide*, Addison-Wesley, Boston, MA, 2002.