Modularity – Often Desired, but Rarely Achieved

Jens Knodel, Matthias Naab, Balthasar Weitzel
Fraunhofer IESE
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
{jens.knodel, matthias.naab, balthasar.weitzel}@iese.fraunhofer.de

***Abstract*—"Everything should be modular" is an exalted goal stated by almost every architect – but is it really possible to achieve this goal? In this experience paper, we share our lessons learned across a number of restructuring projects that went modular. We discuss typical business motivations, restructuring efforts starting with good intentions, and reconstruction reality striking back. In retrospective, we analyze typical pitfalls to be circumvented. Examples illustrate our findings and support a truism too often ignored by architects: everything has its price, and more often than not, the price for modularity is a lot higher than initially estimated.**

Keywords—architecture, modularity, reconstruction, reverse engineering, experience report

I. INTRODUCTION

Modularity is a design goal that is often desired by various stakeholders in development organization, especially if systems have historically grown and problems in maintaining them emerge. Modularity is considered to be the silver bullet in such cases: restructuring the system towards building blocks of manageable size with defined functionality and adequate quality. To achieve modularity sounds straight-forward, but in practice it is not. We have anecdotal evidence from basically any of our industrial partners starting modularization initiatives ending in vain. The projects start with a collection of high-level modularization goals, come up with an idealistic architecture of the system and fail when reality strikes back – restructuring the system reveals a lot of technical constraints not being thought of in advance. However, there are as well success stories: the risk of failure can be significantly reduced by avoiding common pitfalls and applying best practices.

In this paper we present consolidated experiences from a number of past modularization projects. We introduce an overview of typical stakeholder goals to be achieved by modularity. We then present a way for redesigning the system in order to achieve these goals, followed by the actual reconstruction. We discuss typical findings and complement them with best practices.

II. WHY MODULARITY

The requirements of the different product stakeholders collected at the beginning of a modularization project can be divided into modularity goals and product requirements that should be supported by the modularized product. In terms of product requirements our main experience was that the system basically should support the same processes with slight changes. Modularization projects were often also used to get rid of unused features or realize adaptations that had lower priority for a longer time. Since the details vary a lot in the different projects, we want to focus on the actual modularity or decoupling requirements of the different stakeholders:

Product manager
- React on new requirements faster
- Shorter release cycles, also for parts of the product
- Keep investments of the past
- Avoid changes in the UI to not confuse customers

Development manager
- Increase internal productivity by parallelization
- Use external teams
- Use new developers with different or less skills

Sales representative
- Sell parts of existing product separately
- More combination options, also external products
- Additional value as a compensation if changes to current installations at customer's site are required

Architect
- React on new requirements easier
- Local changes in manageable parts
- Clear assignment of responsibilities to modules
- Flexible usage of new technologies

Developers and tester
- Less coordination effort by working more local
- More guidance from the systems structure
- Keep proven technologies and complex algorithms

Customizer
- Separate customizations clearly from product code
- Reuse parts of the system
- Keep existing customizations

Support staff
- Reconstruct problem situations faster
- Locate problems easier

III. APPROACHING MODULARITY

In our past projects a combination of two approaches for identifying potential components was applied [1]. A top-down strategy identifies ideal modules and relationships based on communication needs of the supported business processes, considering both product and modularization requirements. In a second step the existing implementation is taken as a starting point, trying to identify current modules, their boundaries and relationships. Both approaches are iteratively combined in order to come up with a potential to-be architecture that can be realized within the project constraints and is still able to fulfill the decoupling and product requirements. It might be necessary to negotiate some of the product requirements

This paper has been partially sponsored by the German Ministry of Education and Research, grant number 01IS12053

during redesign if it turns out that the cost-benefit is not appropriate. In parallel a migration plan is created, capable of moving the system from the current state towards the intended to-be architecture. In the final reconstruction step this migration plan is conducted and the system transformed into a more modular structure.

IV. Discussion

Based on our experience with our industry partners we collected several pitfalls often found in modularization projects.

Goal overloading: Modularity gets considered as a silver bullet for a number of problems that have been observed around the system. Even if the modularization project is a success, not all of the issues are likely to be solved, so disappointment is inevitable. Distinguishing between product and decoupling requirements is a step in the right direction to reduce this risk.

Everything should be modular: Achieving full modularity is impossible, so a general modularity goal for the overall system is not sufficient [2]. Some modules have to be connected to others, if there are dependencies in the supported business processes. Thus it is the main challenge to anticipate areas of potentially changing requirements and explicitly make those parts of the system modular that are impacted by these changes. It requires finding the right level of modularity for every part of the system, keeping the balance between cost for creation and maintenance of the modularity and the savings gained by that modularity.

Modularity as an end in itself: Modularity itself does not automatically guarantee the fulfillment of the implicitly desired flexibility goals. These goals need to be elicited thoroughly and the adequacy of the to-be architecture has to be checked with respect to these change scenarios.

Simplification of functional dependencies: If business processes supported by the system are not analyzed enough or over-simplified dependencies that are inevitable tend to get forgotten. This results in inadequate module definition and compromising of the to-be architecture during reconstruction.

Disregarding as-is architecture: Another common pitfall is to base decisions about the to-be architecture only on the ideal architecture without analyzing the current as-is architecture in detail. This ultimately leads to unexpected issues during reconstruction when reality strikes back and the intended module interfaces are not sufficient.

Getting lost in details: Trying to recover the complete as-is architecture to a high level of detail consumes significant effort and does not reduce the complexity. A more efficient approach is to let experienced developers do a tool based request driven analysis where identified dependencies are rated and only important ones are refined.

Not using existing knowledge: Not involving developers that created the system makes it hard to get the rationales of past design decisions, which is especially valuable if these decisions should be revised.

Perception over analysis: There are often assumptions about the system that have been said so often so that they are considered as reality. More often than not it turns out that they are not exactly true. Basing decisions on these "ensured assumptions" is critical, checking them by analyzing the code, for example, reduces risk significantly.

Neglecting iterative nature of redesigning: Another pitfall is not being prepared for performing several iterations in the redesign process, thus wasting effort by creating detailed documentations of intermediate to-be architectures.

New technology considered as savior: New technology tends to look "shiny", having many advantages over the current one. Disadvantages of the current technology can be easily observed, its advantages compared to the new one seem to be small. The source for this conception is the missing knowledge of the actual behavior of the new technology in the actual environment. A comparison based on sound prototypes often reveals that the differences are not as evident as expected. A cost-benefit comparison that includes the replacement effort is often debunking the new technology.

Changing fundamental architectural decisions: Trying to change fundamental architectural decisions that have been identified as being suboptimal during modularization is tempting. In such cases the actual impact of such a change is often underestimated. Starting with a more reachable intermediate goal is a much more risk-aware approach.

Unmanaged reconstruction: The to-be architecture is only valuable if it is also realized as it was intended. To achieve that, a continuous monitoring of the reconstruction work in terms of a fact-based tracking is required. Regular architecture compliance assessments are an efficient way of achieving that.

Wrong expectation management: Admitting that some expectations of the project are not realistic is hard. Procrastinating to tell the truth results in disappointment and significant loss of credibility.

Modularization as a disguise: In most systems there are some technical modernization tasks that are considered as important by architects or developers, but not by those who decide about budget allocation. Hiding such tasks in a modularization project is a risky approach since it will result in a significant loss of credibility if this disguise gets public.

No clear project goal: As in any project it is necessary to clearly define its goals upfront, so that an evaluation of its success is possible. In case of modularization this best practice gets often ignored, especially if no clear requirements have been elicited at the beginning of the project.

Gold plating: A similar pitfall like on project level can also happen on task level. Especially clean-up tasks that are common in such reconstructing projects need a measurable stop criteria. Otherwise developers tend to make good things even better and not seeing other, more pressing tasks.

V. Conclusion

We gave an overview of typical modularization goals and common pitfalls when aiming at them. A positive conclusion is that for every one of them strategies are available to avoid them, in most cases just by making them explicit.

References

[1] Naab, M.; Weitzel B.; et. al: *Isolation modularer Technologie-komponenten aus smart FIX*; IESE-Report; Kaiserslautern; 2013.

[2] Naab, M.: *Enhancing architecture design methods for improved flexibility in long-living information systems*; PhD Theses; Fraunhofer-Verlag; Kaiserslautern; 2011.