Does this remind you of your IT system?

"HISTORICALLY GROWN ..."

Successful software often has a longer lifespan than originally anticipated. Continuous extensions lead to a state that practitioners like to call "historically grown". This state is characterized by inconsistent user experience, quality that suffers, rising maintenance costs, and a lack of innovative ability. Hence, sooner or later every software company has to ask itself how to successfully renovate its own software and how profound this intervention must or may be in order to continue achieving the business goals in the future. In order to set the right course for renovation, the actual condition of the software must be analyzed thoroughly, particularly because experience has shown that this condition deviates significantly from previous plans and documents. Renovating a software system successfully requires taking an integrated look at the target state and at the migration path, careful familiarization of developers and users with new concepts, and continual management of risks and goals. Software renovation thus concerns all those who want to set standards with their software in the future as well as in the present.

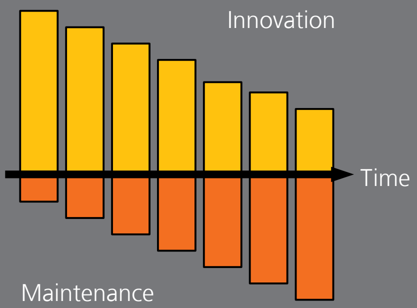
INNOVATION INSTEAD OF MAINTENANCE

Software is THE means by which innovation is created today. Unfortunately, during the lifecycle of a software system it becomes increasingly difficult to remain innovative and to respond to new requriements and conditions. Software evolves continually and new features are added, even if they do not fit into the original design. So software ages noticeably and entails numerous problems. For the users, the user experience is no longer uniform on the one hand and no longer up to date on the other hand. For the developers, the software becomes ever harder to understand and changes become more error-prone, resulting in more and more time having to be spent on maintenance activities. Consequently, less and less time is available for the development of innovative features, and cutting-edge innovation, in particular, becomes almost impossible.

Business software often lasts for several decades. The evolution of technologies and the increasing permeation of our professional and private lives by IT are enormous during this time. Even today, a very large number of applications in companies are still running on mainframes and causing huge maintenance and operation costs. Not surprisingly, there is a clear tendency to port software to more cost-efficient and up-to-date standard hardware, especially since the number of developers for old technologies keeps decreasing. Sooner or later, all software companies are faced with the issue of having to renovate "historically grown" software. The challenges then range from unsuitable architecture and code quality via outdated and extinct technologies to requirements that are impossible to realize. The goal is always to remain innovative and competitive.

"SOFTWARE DOES AGE!"

There is a rumor that doggedly persists: that software cannot age. This may be true at the most for the code as such. As soon as technologies evolve and expectations rise in the envrionment of a software system, it feels as though software is indeed aging. It is mainly continuous evolution with many compromises that mostly leads to a situation at some point in the software lifecycle where more and more effort must be spent on maintenance than on innovation.



▶▶

Renovation is always full of surprises.

“Successful software products become drivers of the ›digital transformation‹ of companies. But they will only remain successful if they allow continuous innovation.”

Werner Weiss
CEO, Insiders Technologies

“Software renovation is like renovating a building that continues to be used, where the residents must not be affected.”

Timo Rihtnieni
Manager Product Architecture
Tekla

BUILD, RENOVATE, OR JUST RE-PAINT?

The renovation of a software system can be done very differently depending on the inital condition and the objectives. Many companies first consider doing refactoring, which is relatively cost-efficient and can be done locally. Unfortunately, however, the improvement effects are rather limited because global challenges cannot be solved in this way. Therefore, the question often arises whether a system should be renovated, including a re-alignment of its architecture, or whether it even makes sense to develop a completely new system. In practice, a complete new development is often not an option either, however, since the development team cannot maintain the old system and develop a new system all at the same time.

Software renovation is often unavaidable and does entail risks. Many decisions must be made about the future product and the development path to be taken. These decisions range from features and the interaction design via the future architecture to the type of quality assurance. A comprehensive analysis of the history and good planning of the renovation are therefore indispensible prerequisites.

"SOFTWARE IS NOT SOFT!"

One of the greatest accomplishments of software is that it can be changed without the need for physical changes. This has led to the situation where all aspects in which changes must be made to a system are nowadays implemented as much as possible via software. This is true for software in companies as well as for software in automobiles.
Although software can basically be changed easily and in nearly every direction, in practice this is usually not possible. Many changes extend across vast areas of the system, have large and unexpected side effects, and make it very hard to get back to a state of high quality.

NO RENOVATION WITHOUT ANALYZING THE HISTORY

"After the renovation, our software must be able to do at least the same things it is doing now!" is a sentence frequently heard at the start of a renovation project. The reason for this is, on the one hand, that this requirement is very easy to formulate and that nobody can say exactly what the software really does. On the other hand, many companies find it very easy to add new features, but very hard to give up existing features. Experience has shown, however, that any renovation should always be accompanied by a consolidation of features, too.

A renovation project can almost never be built upon consistent documentation of the software. In other words, an analysis must not only investigate the current state of the software, but usually it must first reconstruct it. In this endeavor, the source code is often the only reliable truth and source. The reconstruction of implemented requirements and interaction designs is time-consuming manual work, which requires an understanding of the domain. The existing code basis is often large and hard to comprehend. With the help of reverse engineering tools, the code can be examined semi-automatically. Only by recording the identified information in a re-documented architecture can a level of abstraction be created on which the complexity of the system to be renovated can be mastered.

It is important to realize that the analysis of the existing system is an investment that is needed for the renovation to be successful. This analysis must be comprehensive in nature and needs to cover all aspects regarding usage, operation, and development of the system in a methodological manner.

▶▶

Complex renovations require engineering tools.

"A well-conceived architecture that is implemented as strictly as possible means that expenses for costly repairs can be saved. The unavoidable, natural degeneration is then tackled in the context of value-adding renovation measures."

Dr. Dirk Muthig
Head of Product and Systems Design
Lufthansa Systems

ANYONE CAN BUILD, BUT IT TAKES SPECIALISTS TO RENOVATE.

Many renovations of software systems fail although they would be necessary. Whereas new developments are often quickly excluded as an option for numerous reasons, renovation appears to be a feasible and controllable way to get a software system back on the right track. But the result is often that the priority given to such a project is too low, or that it is performed half-heartedly. Then even the analysis sometimes appears to be too great an obstacle.

Software renovation must be addressed as a strategic task and requires the use of technical and methodological specialists. Building on an analysis of the history of the software system, the new target state is constructed. While new construction can work with significantly fewer restrictions, renovation must always take the existing software system into consideration. This means constant trade-offs between the renovation costs and the new benefit being created, which are hard to quantify.

From a holistic point of view, features and quality requirements must be taken into account from the perspectives of usage, operation, and development. Renovation must regard the external design of a software system in the sense of interaction design and visual design as closely linked with the internal design in the sense of the software architecture. The decisions about future features, interfaces, and interactions affect many stakeholder groups and should not be made unilaterally (e.g., only by Sales). In particular, they should be underpinned by facts, e.g. by measuring the actual usage.

The challenge lies not only in designing a target state for the software system, but also in the matching design of a migration path. Renovation almost always takes place concurrently to the evolution of the system, and these two activities must therefore be coordinated in order to enable incremental renovation. Renovation also includes a lot of change management. There are not only changes to the software, but also impacts on the users, the developers, the operators, and the sales department. Changes are often perceived as negative, even if they constitute improvements. These stakeholder groups must therefore be involved early on and, depending on the software changes, also need to get new qualifications or at least training. Constant and goal-oriented migration is very demanding for quality assurance and especially requires automated tests for checking the impact of changes.

The renovation of a software system is always a complex and individual task, and there is no silver-bullet solution for it. This is why Fraunhofer IESE relies on experience gained from more than 100 renovation projects and provides a well-filled tool box of methods and tools.

Fraunhofer IESE has already supported many software companies in the renovation of their software systems and is continuing its research into further improving methods and tools for software renovation.

■ Marcus Trapp, Matthias Naab

"AFTER THE RENOVATION IS BEFORE THE RENOVATION!"

Even after the successful renovation of a software system, the world continues to revolve, new requirements appear, and new technologies become available. Renovation should therefore not be seen as a one-off, but rather as a continuous activity that can be designed according to the required level. However, the option of completely new development should not always be excluded categorically either, since a renovation should remain a renovation, and its objective should not be to remodel an existing system into a completely different one.