



IT University
of Copenhagen

Customization and Upgrading of ERP Systems

An Empirical Perspective

**Yvonne Dittrich
Sebastien Vaucouleur**

**Copyright © 2008, Yvonne Dittrich
Sebastien Vaucouleur**

**IT University of Copenhagen
All rights reserved.**

**Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.**

ISSN 1600-6100

ISBN 978-87-7949-164-9

Copies may be obtained by contacting:

**IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
Denmark**

Telephone: +45 72 18 50 00

Telefax: +45 72 18 50 01

Web www.itu.dk

Customization and Upgrading of ERP Systems An Empirical Perspective

Yvonne Dittrich, Sebastien Vaucouleur

Abstract

An increasing number of software systems are developed by customizing a standard product that provides the major part of the functionality. The customizations of Enterprise Resource Planning systems are examples of such a practice. Nonetheless, little empirical research on the specific characteristic of this kind of software development is available. Do the recommendations for “normal” software development also apply in this case? We present an empirical study on ERP customization practices based on video recordings, interviews and a survey. The observed and reported practices challenge some of the principles of software engineering acknowledged as good practices. Based on the analysis, we discuss essential challenges and identify directions to take when addressing specific difficulties. Besides bearing the potential to influence the development of future generations of enterprise systems, the presented research provides insights in software development practices changing and amending a software product “from within” rather than developing the central functionality from scratch or re-using components “from without”.

1 Introduction

In this technical report, we present an empirical study investigating customization and upgrade practices around Enterprise Resource Planning (ERP) systems. ERP systems aim at supporting most of the administrative and management processes of an organization, e.g. finances, human resource, supply chain management, manufacturing, customer relations. With customization, we mean changes to the functionality of the base program itself that become necessary when the flexibility provided for with configuration facilities is not sufficient. As customizations can be quite quite substantial, it seems justified to talk about development building on an existing software product. Our reality check leads us to the recognition that some of the presumptions software engineering tools and techniques built on do not apply for this development practice.

A substantial part of today’s software development builds on standard systems like ERP systems. Such software products provide functionality that would be expensive to develop from scratch. The vendors can spend more resources on domain analysis and interaction design than a single company could afford. New releases allow to keep the infrastructure up-to-date. Often, these software products have to be customized to fit with the organization they should support. Customizations are considered problematic in the related Information Systems literature, especially as they require additional effort when they should be ported to a new release [1]. However, we have found little research on ERP customization practices or research investigating tool support for practitioners when customizing software products and upgrading the customizations.

To better understand and support this kind of software engineering, empirical research is needed. What kind of customizations are developed? How do developers proceed? What tools and convention do they use to keep track of their changes? What are relevant skills to develop add-ons, fitting into a program providing the major part of the functionality? How can this kind of software development be supported? Are there ways to better support upgrades of customizations technically?

In the section “Topics for Discussion”, we highlight the challenges our observations provide for software engineering tools and techniques.

The remainder of the article is structured as follows: The next section introduces the ERP systems which are subject to the observed and reported practices. Thereafter, we discuss the methods we used for the empirical research. Section 4 contains the analysis of our research material. First, we present how a developer solves a typical customization problem, thereafter, we present the analysis of our field material with respect to a number of dimensions reaching from project organization over testing and quality assurance to cooperation practices. The above mentioned section “Topics for Discussion” is followed by a short conclusion.

2 The ERP Systems

The empirical study concerns the ERP systems Dynamics AX and Dynamics Nav whose development organization now belongs to Microsoft Dynamics [4, 9]. Both are developed for small and medium size companies. As all other ERP systems they have to be configured to fit with the structure of the specific organization. This is done through a configuration interface inserting base data, e.g. defining roles and rights for every single user. For the case that adaptation is required that is not supported configuration possibilities, an integrated development environment is provided. The design philosophy behind both systems is to open up for customizations rather than to provide an overhead of unused functionality.

2.1 Dynamics AX

Dynamics AX, earlier Axapta, addresses medium size companies. Of the two ERP systems discussed here, it provides a more flexible model and more substantial customization possibilities. Customizations are done using a proprietary object oriented language, called X++, similar to C# or Java.

On top of the normal object oriented functionality of classes and inheritance, the construct of layers is provided. These layers are meant to support customizations. An object (e.g. a class, a table or a form) in an upper layer shadows objects with the same name in any of the lower layers.

The development environment that is part of the product gives access to the whole application code. Though developers are not prohibited from editing code in the lower layers distributed by the provider, copying the code to one of the customizations layers before adapting it is regarded as good practice. The original version of the business object can then be called from within the customization code.

An important point in the development experience is that changes are immediately effective once they are saved and the code is compiled. The application does not need to be restarted, and the developer can try the modification without leaving the development platform. Even the development platform can be extended. As we will see below the development of custom tools by development organizations is not unusual.

2.2 Dynamics NAV

Dynamics NAV, earlier Navision, is targeted for small companies. Customizations are done by first defining business objects, for example classes, table and forms, in an interactive environment. The behavior of these business objects is then customized by editing procedures implementing e.g. methods of a class defined in this manner using a proprietary procedural language. The business objects and their relation to other objects are stored as meta data. Only the procedures are compiled.

The Navision development environment also allows for incremental development and iteration between editing and running the application.

3 The research method

Empirical research on ERP system customization provides a number of challenges. First, real world customizations touch on the intellectual property rights of three independent organizations: the ERP vendor, the consultancy selling, implementing and customizing the ERP system and the customer paying for implementation and customization of the system. Second, it is unsure whether – and when – the learnings from the empirical research will result in new features in a future version of the ERP system. So the practitioner’s contribution in form of time might not pay back at all. Third, to understand the complexity of the tasks and to be able to formulate relevant questions, the structure of the ERP systems under discussion has to be understood to some extent.

We addressed this challenge by combining different data collection methods, applying a flexible approach [8] that allows adjusting the research instruments to a developing understanding of the domain: First, we reviewed existing empirical material (video recordings) and discussed the issues around customization and upgrade with our industrial partner, Microsoft. Based on the initial knowledge acquired, we devised an online survey targeted towards ERP practitioners. Finally, we conducted face-to-face interviews with lead developers and managers working in the ERP customization business.

With this triangulation, we aimed at improving the confidence in the results of an empirical study. We used multiple sources of data, multiple observers, multiple methods to complement the respective short comings of each method and to counter a possibly biased interpretation.

3.1 Video recordings

Existing video recordings and empirical reports came from empirical studies previously conducted within Microsoft. The video recordings were part of a user experience study addressing the functionality and interaction design for the development environment. They consisted of several hours of screen captures of customization and about six hours discussion around previous recordings between practitioners, interaction designers and developers. The agreement of the subjects was acquired prior to analysis. We logged the discussion videos as they provided both walk-throughs through specific customizations and discussion of the functionality of the development environment.

The analysis of the video recordings provided the basis for the survey and for the interview guidelines. The accounts of development practices from the interviews could be supported by parts of the video material. The example introducing the analysis part is based on one of the videos. We complement the analysis with findings from the video analysis when suitable.

3.2 Survey

The survey was done online using a dedicated survey tool. We asked practitioners through online forums to kindly help us by answering the questions according to their experience in the field. The questions addressed mainly topics around upgrade practices, difficulties and the use of tools for upgrades. 42 persons answered the survey. We asked the participant as an optional question to leave their contact details, and approximately 2/3 of them did so. Using the specialized forums (where developers discuss bugs, workarounds, best practices, etc.) seems a-posteriori a good choice, the persons who answered the survey are daily involved in the intricacies of ERP systems.

Figure 1 gives an overview what tasks the people who answered the survey worked with. The question allowed multiple answers: Especially in small companies, employees often have more than one responsibility.

Table 1 indicates the experience of the practitioners answering the survey. The sample is well versed in the Dynamics NAV product, but seems less experienced in the Dynamics AX product.

A statistical analysis of the answers is not possible. As the survey was performed in an early stage of the study, the questions are quite exploratory. From a statistical point of









| Tasks in their respective companies | | |
|---|----|---------------------------------|
|  | 35 | Customization of ERP systems |
|  | 31 | Upgrade of ERP systems |
|  | 29 | Customer site implementation |
|  | 23 | Requirements and specifications |
|  | 23 | Managing a development team |
|  | 19 | Customer support |
|  | 8 | Sales |
|  | 3 | Marketing |

Figure 1: Tasks

| Years of Experience | 0 | < 0.5 | 0.5 – 2 | 2 – 4 | > 4 |
|---------------------|----|-------|---------|-------|-----|
| Dynamics AX | 26 | 3 | 5 | 2 | 7 |
| Dynamics NAV | 10 | 1 | 6 | 2 | 24 |

Table 1: Experience with the ERP systems considered

view, the number of participants is rather low. We use the survey answers where suitable to support our analysis based on the interviews.

3.3 Semi-structured interviews

Three interviews of approximately an hour and a half were performed on practitioners (excluding the test interview). All of them were managers and lead developers in consultancies focusing on customization around Microsoft Dynamics ERPs. They all had extensive technical experience with the ERP systems. We developed an interview guideline based on the analysis of the video recordings. We addressed technical issues as well as cooperation and knowledge sharing. The guideline was tested by performing a test interview with a senior Microsoft engineer. The test interview confirmed that the interview guideline was appropriate for our objectives: the questions had the right level of technicality and the interview lasted for a bit more than an hour, which was the time frame we aimed at. The interviews were recorded and transcribed. The transcripts were coded in a grounded theory open coding manner [8] independently by both authors. The coding provided the basis for the structure of the analysis part of the article.

3.4 How valid are our findings?

Both, from a quantitative and a qualitative perspective, the data is rather limited. The number of respondents of the survey does not allow for a statistical analysis. Three interviews are not enough to be sure to catch relevant aspects of a practice. More interviews and also observations would be needed. Given these limitations, we also see indicators that the findings are generalizable beyond the specific organizations we studied. Though the three interview partners reported different levels of formality in their approaches and there is a high level of agreement regarding the challenges of customizations and upgrades, the concrete practices, and conventions. Central aspects are confirmed by the data from the video recordings and the survey as well. Though we focused on two specific products, similar problems and practices can be expected when studying the customization of other ERP systems or similar domain specific integrated enterprise systems.

Whether the findings apply to the customization of systems like contents management systems, simulation systems, or games is an open question. Besides a very different kind of software, different pragmatic contexts for development would have to be considered. Can an on-line game player modifying his favorite game be compared to a developer customizing ERP systems? Probably not. Here more research is needed.

4 Practices and tools for customization and upgrade of ERP systems

Before starting with the analysis of our field material, we present an example of customization practices. Then, we start with a general characterization of the companies implementing ERP systems in their customers' organization and customizing them if necessary and the division of labour between business oriented persons and the actual developers. The subsection on how customization and upgrade projects are organized prepares the presentation of the practices. Thereafter we focus on tools and conventions and on learning and knowledge sharing between practitioners.

We anonymized our interview partners using fake names for the persons and symbolic names for the companies, X (Jack), Y (Albert), and Z (Herbert). The developer, we observed on the video recordings, is called Finn below.

4.1 An example: Logging all actions related to customers

In one of the video recordings, Finn tells about an Dynamics AX customization task that turned out to be a little tricky. The task was to provide an overview for the users over all transactions in which one of his customers was involved.

For this functionality, a new customer history table has to be created containing information from and references to different tables that held data related to customer related transactions. A form has to be created to view and navigate from that table and a respective class is needed to collect and access the information. The new table should be populated whenever data in the other tables is updated. That means that for every relevant transaction a method call has to be inserted that in turn creates a new entry in the customer history table.

To solve this problem, Finn discusses with some of his more advanced colleagues. The first hint he gets is to look at a similar module logging changes to one specific table. However, this does not solve the problem. Another colleague hints at a system function that collects data for a central archive - a functionality needed for public organizations in the Scandinavian countries. Finn explores the related objects and uses the module as a template to implement the customer history functionality.

Finn goes about it in an iterative way. First the table and the respective class and form are created. The method creating the table entry is added and a call is inserted in one of the relevant business object. This is tested by creating a new customer and manipulating the system so that the respective data entry should be inserted in the customer history table. During this process the developer switches between a program editor, a table viewer and the user interface of the system. The user interface is also used to trigger transactions that then become visible as entries in the table and in the newly developed form.

This process is continued until method calls in all relevant places are inserted and the creation of the respective entries in the customer history table are checked.

4.2 The companies and the people

The companies we have been interviewing are small and medium size consultancies. Two of the three companies specialized in specific business segments, a practice which provides a head start when bidding on tenders. "When we started, we took every kind of company, but now we are taking production plants. We have a special vertical, QA [Quality Assurance] which we use to [get] into the companies." [Albert] "We have four segments we work with. One is called "Consultancy business". That is normally other software companies, other consultant companies, engineers, architects and so on. That is our segment 1. Then we have a segment 2, which is the 'Public Sector' but with a speciality regarding project oriented organization within the public sector. [...] Then we have a segment called 'Membership', which is membership organized organizations. [...] And the last segment is called 'Trading Service'." [Herbert 00:04:24] More established consultancies focus on a specific market segment,

developing so-called “verticals”, add-on modules complementing the basic functionality of the software, again providing competitive advantages. All the consultancies we interacted with aim at keeping customers over longer periods. Especially when developing custom solution – like for example a module for administrating flights for a small carrier or a module administrating royalties for a film distributor –, changes and developments on the business side result in new contracts for customization of the base software.

All interviewees referred to two kinds of expertises needed to implement ERP Systems in customer organizations; the ability to understand the customer’s business and configure the software and the ability to customize the system and develop additional functionality. In two of the companies two roles were distinguished. “In the old days, we had XAL and a small product. The employees were [doing] both things. But our experience with Axapta is, it is so complex that if you are going to be good at something, you cannot be good at all [aspects]. You have to either focus on developing or on the application. But of course you have to know some of the other part.” [Albert] As we were focusing on customization and upgrades of customizations we focused on the development side of this cooperation. But even in the technical area, few employees hold a MSc degree in computer science. Employees often come with an engineering degree, a business degree or a professional education comparable to a bachelor in computer science.

Hiring and educating developers was mentioned as one of the main challenges: General programming skills are just the basis; on the one hand, developers have to learn to understand an existing rather complex application. On the other hand, they must have enough understanding about business administration to understand the rationale behind the base application and understand the business impact of changes. New developers start by taking courses offered by the ERP provider and work with documentation and minor developing tasks under guidance of a more experienced developer. “[They start with doing] the simple things, reports and forms, and after half a year they are going to do more complex business logic.” [Albert] Experienced developers often have worked with different ERP systems, but most consultancies specialize in one of them.

4.3 Project organization and documents

The long time relationship between the consultancies and their customers results in three different ways of working. Our interviewees talked about projects for new customers, evaluation and maintenance tasks with existing customers and upgrade projects.

New projects A project starts with the consultants or “application experts” of the company discussing with the customer what is needed. Part of this initial phase is to align the expectations of the customer with the functionality the system provides. The goal is to use as much as possible of the existing functionality without customizations. Though the different companies follow different methodologies for analysis and design, some document – called “bill of functionality”, “requirement specification” or “analysis report” – is the base for the agreement with the customer, the starting point of the development process and the base for the acceptance test by the customer.

In company X, this is the only document produced. They otherwise rely on the documentation of the source code. Company Z can be seen as the other end of the scale regarding documentation. Two different processes are used depending on the complexity of a customization: larger customization projects are using flowchart-based models of the work processes in the area under discussion. Based on that description an example application is set up, and sample parts of the customization prototyped. An analysis report is developed which is signed off by the customer. Simpler customizations only use a “bill of functionality” as a starting point. Based on the analysis report or the bill of functionality, a design document is developed, the customizations are implemented, sometimes internal test cases are specified and – based on the analysis report or the bill of functionality – the acceptance test is defined. The rigour of the tests depends on the quality requirements of the customer. Company Z uses a project management tool developed by another consultancy from the same consortium.

Company Z also works actively with their method and the format of the documents produced. For example, the interview partner tells that the format for the analysis report and the bill of functionality changed from developer centric notation to a format that the customer can understand. To compensate for the lacking information, the main developer associated with the project takes part in the workshops and the meetings with the customer to get a first-hand idea of the requirements. Some of the companies use custom tools – sometimes internal customizations of the ERP system’s development environment – to keep track of all communication and documentation around the customizations for a specific customer such tools are further described in subsection 4.5. All companies use standards for code comments, both to identify own code and to document the implemented customizations.

Acceptance tests normally result in a list of issues that are to be resolved before the project is signed off by the customer. All interviewees problematize testing. How much and how rigourously the customizations are tested depends on the willingness of the customer to pay for the additional effort. Verticals and parts with high reliability requirements are tested more rigorously than simple reports. Often small changes are done at the customer site, for example, to adjust the reports to a local printer set-up and to add keyboard short-cuts for often used functionality. Customer contact continues throughout the project. Normally that implies that both the business expert and the developers are involved.

Evolution and Maintenance As previously mentioned, the consultancies also take care of evolution of the installations and customizations for their customers. Sometimes several developers are working full time for one central customer. Consultancy Y, for example, has two developers working fulltime with evolution and maintenance of the implementation for the air carrier mentioned above. Often a developer is responsible for maintenance and evolution for a number of customers and takes care of a new project. In such long term cooperation the developers often develop a more direct contact to the customer. At company Z, each customer site installation is taken care of by one consultant and one developer in order to insure continuity for the customer, and keep access to knowledge about relevant historical discussions and rationales for the implementation which is not always documented. “. . . [N]ormally, we have one developer, and also, we prefer to have one consultant at least who owns the customer, who knows what is going on. Because we have a long history and the amount of documentation is also too low. So we need to know the customer’s business.” [Herbert]

Upgrade Processes According to our interviewees, upgrades are handled similar to a new implementation though the existing implementation makes the process easier. The consultant goes through the existing customizations together with the customer and discusses one by one whether to keep the customization or whether to “go standard”. Our interview partners report two different strategies for upgrade projects: Two of them strictly confine upgrades to porting necessary customizations to a new version. The existing implementation provides a specification for this porting. New customizations that are caused by the new functionality not being sufficient any longer or by the need of additional adaptations are postponed to a dedicated maintenance project thereafter. The third interviewee emphasized that, especially when the existing implementation is outdated due to, for example, substantial growth of the company, the consultants recommend a total new design.

If the upgrade refers to own customizations, the documentations developed in the original design process is referred to. The resulting implementation is then tested against the old installation.

The upgrade process is often complicated by the fact that some customers have had several consultancies customizing their installation. None of the interviewees appreciated to work with or upgrade customizations not developed within their own company. However, this is regarded as an unavoidable problem.

If upgrades are cumbersome and expensive who then initiates an upgrade and for what reasons? We asked these questions in the survey. Figure 2 reveals interestingly that the partners have a strong influence on the decision to upgrade. Table 2 shows the survey answers to the question regarding the motivation to start an upgrade project. Interestingly




| Who initiates an upgrade? | | |
|---|----|--|
|  | 24 | The customers, on their own initiative |
|  | 34 | The partners (vendors) suggest the upgrade |
|  | 4 | Other |

Figure 2: Who

| | To benefit from a new technology | To ensure support from vendors | To get bug fixes | To make it easier to upgrade later | To get compatibility with external software | Customize and upgrade at the same time |
|---------------------------------------|----------------------------------|--------------------------------|------------------|------------------------------------|---|--|
| Typically not a reason at all | 7 | 12 | 1 | 13 | 8 | 13 |
| Good to have but not essential | 21 | 21 | 21 | 22 | 24 | 15 |
| Typically the main reason | 14 | 9 | 20 | 7 | 10 | 14 |

Table 2: Reasons to upgrade

getting bug fixes for the standard system scores even higher than new technologies and features.

4.4 Customizing ERP systems

The concept of customization is not clearly defined. Practitioners distinguish between different categories of customizations according to their complexity and the difficulties connected with each of them. After discussing these categories we discuss some of the challenges the faced by the developers.

Categorization of customizations and structuring of tasks Starting with a bill of functionality or with a requirements specification, how do you start to work with the customizations, how do you split up the work in small chunks? We did not get a clear answer to this question. The reason might be that the term “customization” covers a range of different changes to the application:

- Most simple are customizations of existing reports (invoices, order confirmation, and so on), for example, by hiding one of the fields. Often, custom reports are required to provide additional information for management. Here, additional data has to be collected either by accessing the database directly or by calling specific procedures provided by the standard system. This is often a task for new developers, who have to get familiar with the system to be able to take care of more complex tasks.
- The existing functionality of the ERP system can be enhanced: a simple enhancement would be to capture additional data for some entity, requiring the change of a form, the change of some code unit, and the extension of a database table. Our introductory example falls into this category. A more complex example would be changes to the business logic of the general ledger, a central accountancy module [Herbert]. Under this category the integration with other systems is treated as well.
- Add-ons comprise a third category. So called “verticals” consist of more independent complementary functionality, like a contract management system mentioned by Herbert,

| | Adaptations poorly structured | Changes to the system layers | Missing documentation about existing customizations |
|-----------------|--------------------------------------|-------------------------------------|--|
| Minor | 2 | 11 | 3 |
| Moderate | 29 | 23 | 16 |
| Critical | 11 | 8 | 23 |

Table 3: Most important factors that complicate an upgrade

a module to administrate royalties for a film distributor [Jack] or a flight model for a small carrier [Albert]. Such add-ons are less intertwined with the standard application.

Customizations are done according to the bill of functionality, the requirements specification of the feature list. Additional design is done and documented only for more complex customizations. For a complex feature, some developers start up with implementing the necessary changes to the database scheme; then they add adjustments of the data model; and finally the respective forms in order to be able to view and add data are implemented. Based on that frame, the business logic is developed iteratively. Finally the reports are adjusted.

Challenges of customizations. All our interview partners confirmed that the main challenge when customizing standard systems is the understanding of the ERP system. According to our interview partners, even very experienced developers do not understand the whole system. Experts for specific modules are consulted if the customizations in this area are of the more complex kind.

One of our interviewees reported how he goes about to develop an understanding of an unknown part of the system: He explored the code and in parallel ran trials through the interface in order to see what tables are affected and in which way. A proper documentation of the code base is lacking. Often changes in procedures impact distant parts of the system. In some cases the order of procedure calls influences the result. In accordance with that, experience with a specific ERP system is reported as the main skill needed for the customization of ERP systems.

4.5 Upgrading customizations.

As mentioned above, one strategy for upgrading customizations in many cases is to get rid of them, to “go standard”. If that is not possible the customizations have to be ported to the new version of the standard system. Below, we first discuss various reasons that make upgrades of customizations cumbersome. Thereafter we report different strategies for an upgrade.

Difficulties when upgrading customizations Based on the interviews and the answers to two open questions, we distinguished three different groups of difficulties:

Intrinsic difficulties

- Many custom features require related modifications of a number of objects or code units. The related modifications have to be found, their interaction needs to be understood. Depending on the changes in the new standard version, the whole set of modifications may have to be redesigned.
- Some objects and code units are not only modified to implement one feature but tend to be subject to a number of modifications related to different custom features. E.g. the postings to the general ledger are often modified to mirror specific business rules. Here each modification has to be identified, analysed and re-applied or redesigned.

Problems due to non-expert code One of the survey questions addressed this problem. Table 3 summarizes the answers. The issue was brought up in the interviews as well.

- The most critical issue is undocumented customizations. Our interview partners mostly complained about customizations made by other consultancies. As the documentation is not available, they sometimes have to perform a serious re-engineering job.
- Poorly structured adaptations are often due to the lack of experience of junior developers. Our interview partners mentioned reports where the database is accessed and complex algorithms are implemented instead of calling the standard function that implements that functionality. Another typical “faux pas” is to code directly into the standard functionality instead of developing an independent object and only calling the additional functionality.
- Changes to the core/system layer of Dynamics AX are considered dangerous and risky when upgrading.

Different kind of changes to the standard system are of course the reason for the upgrade. However, our interview partners distinguished between different levels of difficulties depending on what kind of changes conflicted with their upgrades.

- Changes to the data model, a custom feature relies on, require a total redesign of the feature.
- When business logic has changed, the new business logic has to be inspected. Either it replaces the customization and can be used as it is, or the old functionality has to be re-implemented, as part of the customization.
- The layout of forms and reports is often tailored to company specific standards. If the standard of the forms and reports module changes, this code has to be re-adjusted, often at the customer’s site.

Upgrade practices One of our interview partners explained his strategy when upgrading customizations for Dynamics AX. He takes a combined strategy:

He begins with going through the logical layers of the system: First, he ports changes to the database scheme. Then, he is doing the same for the business logic, the classes or business objects. And finally, he upgrades the customizations to reports and forms.

When he gets to a modification belonging to a more complex customization, he switches to another mode: “And, of course, sometimes you have to stop with a feature, if you can’t make the upgrade the easy way [. . .] Then you have to understand what the purpose of the feature is. Then you have to focus on: this feature links to this class and this class, [you need] to have a whole overview.” [Albert] When available, the documentation of the old customization is accessed.

Knowledge about major changes in the new version informs the upgrade process: “For instance, if you are upgrading from 3.0 to 5.0. There are a lot of changes in the project module, and if you have some customization there, you nearly have to start over with customization. Because all the data model is changed, the class hierarchy is changed. Then you have to understand what was the meaning of the old customization. Then you have to, in some way, make it anew.” [Albert] All interview partners talked about the difficulties of understanding old complex modifications.

The “verticals” were not mentioned in the discussion of the upgrade difficulties. They seem not to provide major problems for the upgrade. Probably this is because normally the language does not change. Of course the parts interfacing with the standard system have to be upgraded.

4.6 Quality and tools

All of our interview partners emphasized the importance of documentation and conventions for code and documentation of the code. We specifically asked about testing and about what tools beyond the development environment they used for both customizations and upgrade.

Coding and documentation conventions. All our interview partners agreed that it is part of good practice to separate one's own code as much as possible from the code of the standard system. That means for Dynamics AX that additional functionality is implemented in additional classes. The interface to the standard system is done via call of standard functionality or through small insertions, where the new code has to be called from the standard functionality.

All companies used conventions for documenting code: normally the name of the developer, a date, some reference to the relevant requirement or feature in the analysis document. For the development within the Dynamics AX environment, the disciplined usage of the layer system helps to separate customizations into layers. That way also the self developed "verticals" can be customize as they are often sold to more than one customer. Non-adherence to the good practice and the coding conventions becomes especially visible during maintenance and upgrade. All interview partners mention badly documented and bad code as one of the reason why upgrades can be difficult.

Two of the companies we interviewed have conventions for the overall documentation: They require that the requirements specification or the analysis report is annotated with the parts (tables, business objects and classes, code units, forms or reports) that are used to implement the respective feature.

Tools The development environment was often enhanced with custom tools developed by the company itself or purchased. In one of the video sessions, a developer describes such an internally developed documentation system: Each requirement is split up in a number of "programs"; for each of such "programs" additional text and figures describing the design can be added; the customized and the new developed "application objects" – e.g. classes, tables, forms – are linked in; a number of test cases are defined (both program driven and manual ones).

One of our interview partners reported about a similar add-on: For a given customer, the whole communication history around the customization is accessible for the consultants and developers as well as – through a web interface – for the customer.

We asked specifically for upgrade tools. To our surprise a tool highlighting differences between a customized version and an standard version was mentioned as the main tool for upgrades. The documentation with the links to changed objects was only used when more complex features required additional analysis.

One of our interview partners asked for a tool that connected the different changes belonging to one feature: "That is one of the problems, because if we had some kind of tool saying that for doing this task, I make this and this and this modification. Then I could more or less isolate what has been done previously. Because one modification could take some tables, it could take some forms, it could take a code unit, and some data ports. And then you do not know how it is linked together. [Of course] we have the version tool, but it is not done properly." [Herbert]

Testing Rigor testing depends on the willingness of the customer to pay for the extra effort. Testing is not always prioritized by the customers. The standard system is taken as tested and correct with the exception of known bugs. Another difficulty mentioned in the interviews is that it is often difficult to test changes to the standard system in isolation.

Tests are normally done on a system level. That means, the feature is tested through the respective forms, and the tester eventually checks the database tables through a dedicated viewer to see whether the data is saved in the right way in the right place. The test cases are often part of the documentation.

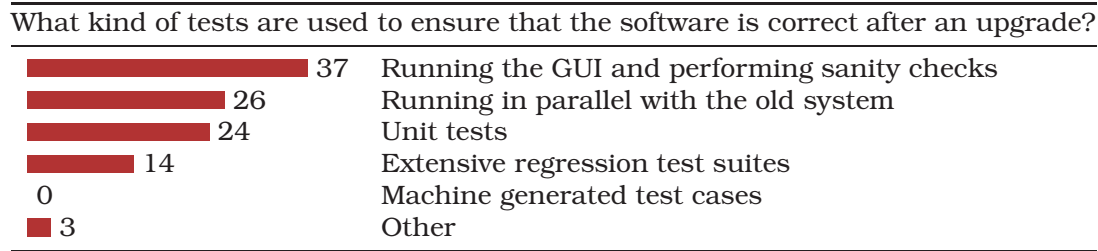


Figure 3: Testing

Customers check off the features that are subject to the contract in an acceptance test. Normally, errors surfacing during the first few weeks of use are corrected by the consultancies without extra fees.

Figure 3 shows the distribution of answers to the question regarding the testing of upgrades. The results confirm the analysis based on the interviews.

4.7 Peer learning and knowledge sharing

Developing software based on an existing large software product is often referred to by practitioners as a very different experience from writing traditional software. One must build on top of a large code base – so large that one cannot hope to comprehend it fully. The new developed features have to be integrated with the existing code base. So how do new-comers acquire the necessary skills and the comprehensive knowledge about the software? Our interview partners reported a number of measures used by their companies to help both to develop and maintain the expertise of their employees.

How does expertise show Expertise shows in the overview a developer has of the base system. Such an overview allows him for example to make use of functionality provided by the standard system instead of accessing the database to collect the data to be presented in various reports. It allows him to fit customizations smoothly into the standard application. You have to know “Where to do [something] and the consequences exactly of what you are doing.” [Herbert] The encapsulation of as much as possible of the implementation in own classes or objects is another indication of experience. The more the business logic can be isolated the easier it is to debug and also to upgrade customizations. As expertise and with it the peer recognition depends on the knowledge of the base system, we expected new releases to be considered problematic by the developers: part of their knowledge would become obsolete. However, this seems not to be the case. A new version is welcome as a new challenge, providing new possibilities, new smart ways to do things. One of our interviewees compared trying a new version of Dynamics NAV with trying out a newly released car of a brand you know and like.

Apprenticeship A more comprehensive project requires customizations of different levels of difficulty. Such a project offers possibilities for educating new developers “on the job”. The lead developer on a project supervises new developers, starting with documenting customizations and doing simple changes, e.g. developing customized reports.

As visible in the introduction of this section, this peer education continues even when the developer is able to take on customization tasks independently. Knowledge about how to solve more intricate problems is acquired by asking colleagues and applying their hints in a trial and error fashion.

Knowledge sharing Peer education continues even among well-experienced developers. Often experts for specific modules are consulted when these modules have to be adjusted as part of a more complex customization. In one of the companies subject to the study, developers assigned themselves to investigate different modules of the new release in order to act as

expert support for their colleagues. Additionally, two of the companies subject to our interviews organized more formal seminars, where topics of interest are presented and discussed.

5 Topics for Discussion

In this section, we take up some aspects of the development practices observed and discussed in the interviews. The purpose of this discussion is to highlight some issues worth further discussion.

When comparing customization of ERP systems with custom software development, a number of challenges becomes visible. What on a superficial glance can be seen as idiosyncrasies can actually be related to the very characteristics of adapting an existing product.

5.1 A different kind of development

Any programming task is dependent on the implementation technique, for example in the form of a programming language. Nonetheless, many of the most influential methods and approaches like structured programming, stepwise refinement and the popular versions of the waterfall model, take for granted that the development team has full control over the central model, the architectural structure, and the code base of the core system. A clearly structured design, good interface specifications, a complete set of test cases belong from this point of view to the cornerstones of good practices. In this perspective, black box is the favored approach to re-use. The aim is to encapsulate functionality so that it only interferes with the existing design in a limited way.

These methods and principles contributed much to the ability to manage the complexity of software development. However, their underlying assumptions have to be reconsidered in the context of ERP customization: They might be valid when designing independent “verticals” where developer designs new functionality. For small and medium size customizations the developer tend to inline many small code patches to the existing code base.

Frameworks, tools and libraries on the one hand support development, on the other hand, they constrain the design. For example, the development of web based applications requires to handle and design the interplay of a number of tools, protocols, notations and frameworks. Here, the developer has to adjust his design to the interfaces of the foreign code.

The development based on standard products radicalizes this situation. Frameworks and libraries are designed to be used and configured in an anticipated way. By contrast, customizations are normally unanticipated changes. Therefore, no dedicated way of customization can be designed for in advance. The own code is sliced into the standard code, carefully avoiding jeopardizing the existing functionality. Instead of using a framework, the framework becomes the environment into which the developer has to fit his own smaller and larger add-ons.

Because the use of frameworks and libraries can be anticipated, documentation supporting the anticipated use and configuration is possible. Research on documentation of frameworks and libraries like [5] emphasizes the anticipated usage. As ERP system customizations cannot be anticipated, documentation to support the customizations cannot easily be devised. Product specific research is necessary to understand typical customization scenarios and develop relevant documentation.

The implementation of a framework or library is meant to be hidden from the developer using it. This is not possible when customizing ERP systems: The configurations are changes to the very implementation of the code. Customizations require an understanding of the existing program in order to be able to estimate the implications of a change both regarding the functionality of the program and regarding the domain the program should support. Accordingly, knowing the respective ERP system is one of the central skills of an experienced developer.

One could argue that the difference between software development “from scratch”, the use of frameworks and the customization is not a fundamental one. Our observations indicate that the degree of constrain the ERP system provides for the development results in radically

different requirements both to the skills practices of the developers and the tool support for these practices.

5.2 Implications on Testing

As customizations are not always changes to isolated parts of the ERP system testing also becomes problematic. Systematic testing of the customization would have to be based on a set of test cases for the base system. It would be a major endeavor to define such a test base: The functionality of an ERP system depends not only on the input but also on the configuration, the base data and the production data in the database. This results in some systematic challenges:

- Running such a test base, that would cover all different configurations and possible combinations of data and base data would not be possible as part of the incremental development we observed. Maybe a full regression test would require more than a night or a week end.
- To help that situation, one could select a relevant subset of test cases. But how to identify the subset so that specific regression tests can be defined. A static analysis of the source code is probably not enough. For example the configuration through base data might decide on the dependencies that have to be taken into account. regarding Dynamics NAV part of the application is defined using and stored using a proprietary meta notation. Here dependencies in the source code and dependency in the model have to be combined.
- Parallel to the customizations of the source code, the relevant test cases would have to be customized. How to navigate the test base in order to identify the test cases to be changed with a customization? It is not clear whether the support provided by the test cases would compensate for the complexity they add to the implementation task.

These difficulties are mirrored in the reports on testing and quality assurance provided in the interviews. Testing is regarded as cumbersome and expensive. Test automation, e.g. in form of unit tests, is only fully applicable for the so-called verticals. The quality of the validation thus depends on the requirements of the customer and the customer's willingness to pay for the time more systematic testing requires in this environment.

Instead of simply applying verification and validation methods developed for custom development where the software engineer has full control over the design or the software, the respective methods have to be adapted to the specific challenges of ERP customizations.

5.3 Development groups as communities of practice

The preferred way of working seems to be based on exploration and experimentation rather than on reading documentation. This way of working is enabled by the flexibility of the development environment. The knowledge the developers gather that way is shared informally. Team work is important and is taken care for in the manning of the projects.

A group of developers can be described as "communities of practice" [11]. Informal knowledge sharing and peer learning is the main way of developing skills. Throughout all our field material, this side of the development practices was emphasized. All of the companies we interviewed were actively addressing the sharing of knowledge:

- Employees took on the exploration of specific modules of new versions.
- Newbies and skilled developers teamed up in projects. That way informal apprenticeship relations were established.
- Informal knowledge sharing was accepted and encouraged.
- Skilled developers implemented and documented their customizations according to accepted standards thus supporting maintenance and upgrades as much as possible.

- Custom tools support the commonly accepted standards of good development.

With respect to “normal” software development, knowledge and knowledge management is emphasized as well [3]. However, in the reported cases the emphasis is on method related knowledge, specific technologies, and the maintenance of project specific knowledge. Though all of our interview partners also emphasized the importance to support the development of project specific knowledge – through the organization of the projects and through adequate documentation – the knowledge that was emphasized as most important was the knowledge about the functionality and implementation of the ERP system.

5.4 Making customizations first order inhabitants in the development environment.

The implementation of one feature or requirement often requires changes to different parts of the ERP system. Changes to the same class or code unit might belong to the implementation of very independent features. Many of the documentation guidelines and tools addressed this problem: How to keep track of a set of logically related changes to different parts of the ERP system.

The programming environments use structures defined through the programming language as the unit of work. However, the unit of work when customizing ERP systems cuts across the structures provided by the programming languages. One solution to this problem would be to allow the developer to indicate that different changes belong to the implementation of one feature and highlighting them in a unique way when one of the code change is selected. Maybe, aspect oriented techniques can be adapted to cluster related code changes and provide more control e.g. regarding type safe adaptations [6].

The development of suitable support for the developers – both regarding functionality of the development environment and the interaction with it – requires more detailed observational studies of different customization practices.

5.5 Supporting practices of Artful Integration

In the area of computer supported cooperative work, tinkering and bricolage were recognized as organizationally viable ways to deal with heterogeneous technical infrastructures [2, 7].

In 1994, Lucy Suchman proposed, based on a rich body of empirical research around the use of document handling systems, to conceptualize design as “artful integration” rather than continue “designing from nowhere”. Designers have to relate the new piece of technology to the existing technical and social context. Whereas traditional design approaches teach and develop methods that disregard existing infrastructures and work practices [10]. The very existence of the practices we describe supports Suchman’s proposal. ERP systems need to be adjusted to be useful, and that to an extent that a larger and larger number of consultancies can make a living out of it.

The necessity to develop “within” a given application requires skillful integration between new and standard functionality. At the same time shortcomings of the tools and techniques developed for “normal” software engineering become visible: The development environment focuses on programming language features as the unit of work rather than clustering and navigating modifications belonging to the same feature. Verification and validation techniques are another issue where problems with traditional approaches become visible in our study. Taking the above described practices serious might lead to a more artful integration of software development methods in software development practices.

6 Conclusions

To better understand and support the customization and upgrade of ERP systems, we implemented an empirical study. What we found were development practices far from what is taught as good practices in software engineering. However they mirror the conditions of this

kind of development. Developers have to adjust to an existing code base and design. In the discussion, we propose to take these practices as a challenge to adjust software engineering tools and techniques to support these practices. We give concrete indications how some improvements can be implemented.

Software development does not necessarily take place the way developers and teachers of techniques and tools anticipate. The findings here refer to one of many development practices that do not fit the ideal promoted. End user tailoring, end user software engineering, game modding, opportunistic software development are terms denoting related practices. These diverging development practices are becoming more and more important. They will need suitable support. Adapting and integrating software engineering tools and techniques with these practices might result in more flexible and therefore more usable support for “normal” software development as well.

Acknowledgments

Thanks to all ERP system developers that offered us their time answering the survey and our interview questions. The Danish Council for Strategic Research and Microsoft Dynamics sponsored the research in the context of a Research project on “Evolvable Software Products”. Thanks to the colleagues that helped us with their generous feedback.

References

- [1] L Brehm, A Heinzl, and M L Markus. Tailoring erp systems: a spectrum of choices and their implications. *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*, pages 9 pp.–, 3-6 Jan. 2001.
- [2] M Büscher, S Gill, P Mogensen, and D Shapiro. Landscapes of practice: Bricolage as a method for situated design. *Comput. Supported Coop. Work*, 10(1):1–28, 2001.
- [3] T Dingsøy and R Conradi. A survey of case studies of the use of knowledge management in software engineering. *International Journal of Software Engineering and Knowledge Engineering*, 2(1):391–414, 2002.
- [4] Arthur Greef, Michael Fruergaard Pontoppidan, and Lars Dragheim Olsen. *Inside Microsoft Dynamics AX 4.0*. Microsoft Press, 2006.
- [5] D Kirk, M Roper, and M Wood. Identifying and addressing problems in object-oriented framework reuse. *Empirical Software Engineering*, 12(3):243–274, 2007.
- [6] S Vaucouleur P Sestoft. Evolvable software products. technologies for customizations. *forthcoming*.
- [7] Toni Robertson. Embodied actions in time and place: the cooperative design of a multimedia, educational computer game. *Comput. Supported Coop. Work*, 5(4):341–367, 1996.
- [8] C Robson. *Real World Research*. Blackwell Publishers Ltd, Oxford, UK, 2002.
- [9] Microsoft Business Solutions. *Application Designer’s Guide*. Microsoft Business Solutions, 2006.
- [10] L Suchman. Working relations of technology production and use. *Computer Supported Cooperative Work (CSCW)*, 2(1):21–39, 1994.
- [11] E Wenger. *Communities of Practice. Learning, Meaning, and Identity*. Cambridge University Press, Cambridge, UK, 1998.