Fraunhofer Institut
Experimentelles
Software Engineering

Successful Software Product Line Development in a Small Organization

A Case Study

Authors:
Cristina Gacek
Peter Knauber
Klaus Schmid
Paul Clements

IESE-Report No. 013.01/E
Version 1.0
March 15, 2001

A publication by Fraunhofer IESE

Fraunhofer IESE is an institute of the Fraunhofer Gesellschaft.
The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competetive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach
Sauerwiesen 6
D-67661 Kaiserslautern

Executive Summary

This report details the experience of a small company, Market Maker Software AG of Kaiserslautern, Germany, as they adopted and used the paradigm of software product lines to help their business grow successfully into a new market area. While software product lines are often associated with large, traditional software organizations, the experience of Market Maker and other small companies like it shows that software product lines represent an ideal development concept for companies of all sizes. This case study reports on the history of the company, how it came to adopt product lines as its prominent development strategy, the role of the key individuals involved, and how they met and overcame various technical challenges. The report concludes with a section analyzing the Market Maker approach.

Table of Contents

1 Introduction

"Software product lines may be fine for large organizations, but not for small ones. Small companies can't afford the start-up cost. Further, the burdensome processes, the strict organizational roles, and the tedious planning all run completely counter to a small organization's lean, nimble, leap-out-of-the-starting-gate culture."

So goes the conventional wisdom in many parts of the software engineering community. And it is not without a certain understandability. Many of the well-known and most publicized software product lines come with pedigrees written in large script: Nokia, Motorola, Hewlett Packard, CelsiusTech, Philips, and others. These organizations boast hundreds of developers, with budgets more than ample enough to cover an experiment or two with a new and untried paradigm. And if it fails – well, there's always a bit of a risk with an R&D project, isn't there? It's not as though the company will be in peril.

For small companies, R&D is an unaffordable luxury. And experiments are for laboratories, not lean need-it-now production shops where every product has to strike market gold for the company to survive. No wonder the conventional wisdom views software product lines as a game only for the heavyweights.

For the record, the conventional wisdom is utterly wrong.

In fact, software product lines offer many small companies their last best hope for success. Time to market and economy of production are two of a product line's best assets, and both are many times more critical for a small company. Of course, even the largest software house would like to cut costs and get products to market sooner, but the large companies enjoy reputations that help them entice customers to wait for an announced product. After release, large volume helps pay for large development staffs. Small companies for the most part have neither a global reputation nor the expectation of global-sized volume to provide a cushion. Their road to success lies through the tricky territories of turning out products that are impossibly customized, in an impossibly short time, using an impossibly small staff.

We know many small companies that have used software product lines to increase (or even establish) their viability, and in more than one case the results have spelled the difference between life and death. This report is the study of one such company, MARKET MAKER Software AG of Kaiserslautern, Germany. In the late 1980s, a university student who liked to program in his spare time noticed that there was no software available to help private individuals track the

stock market. With a father who was a bank chairman, the student recognized a business opportunity when he saw one. In 1989, the student founded MARKET MAKER, and soon thereafter released the first product, a program that ran under the DOS operating system. The program could maintain a user's watch list, and display various charts and data about selected stocks. The company sent diskettes weekly to its subscribers, full of data typed in manually from stock quotations in the newspapers' business pages.

From a single programmer in 1989 to almost 60 in early 2001, MARKET MAKER has enjoyed steady growth. Its data subscription service provides a dependable revenue stream, for its customers recognize it as higher quality and more reliable than the data available from other sources, even the markets themselves. And other products have replaced the old DOS program. A Windows version of the MARKET MAKER product for individual users was released in 1998. Along the way, the product was picked up and marketed by a large German media company under its own publishing house brand, resulting in the sale of more than 150,000 copies, making it Europe's most popular stock market software. And in 1999, MARKET MAKER decided to launch an internet-based version of its product.

The idea behind the internet version was that they could use the fundamental 1998 product functionality as the engine to power other companies' financial Websites. But that functionality would be but one part of the total product. The total system would, depending on the customer, have to integrate with other databases, other content-producing software, run on who-knows-what kind of computing platforms and servers, satisfy human-user performance requirements, and be tailored to show exactly the kind of data, in exactly the kind of charts, in exactly the kind of form required by each particular customer's Website.

That is to say, the product had to be flexible, widely tailorable, deliverable in a very short amount of time, and producible by a very small development staff. In other words, it had to be built as a software product line.

And so it was. The result is a 520 KSLOC system that meets all of those requirements and more. Six people (two of which were part-time) worked for about a year to produce the core system, from which instantiated products are turned out. Each version of the system–that is, every product–must be built to the client's specifications and installed and tested on the client's own platform. Visit http://www.diba.de/wpinfo/fs_info.html, and you'll see a member of MARKET MAKER's internet software product line running on the Website of one of Germany's leading direct brokers. And how long does it take MARKET MAKER to put up a system like this? In as little as three days.

So much for conventional wisdom.

The rest of this technical report presents the MARKET MAKER story in more detail. Section 2 describes the company, its background, and the history that brought it to the software product line approach. Section 3 picks up the story at the start of the software product line and explores its beginnings and origins in greater depth. Section 4 describes how MARKET MAKER approached many of the issues associated with software product lines. These include the software architecture for the product line, implementation and testing/maintenance issues, measurement, the organizational structure of the company, understanding the market, technology forecasting, and launching and institutionalizing the product line. Section 5 contains our analysis of the MARKET MAKER experience, placed in the context of other software product lines we have encountered and written about. Finally, Section 6 offers a glimpse at MARKET MAKER's outlook and gives a few concluding remarks.

2 Company Background and Context

In the mid 1980s, a young university student by the name of Axel Sellemerten noticed that there was no product available for private individuals to track stock market data. Recognizing a business opportunity when he saw one, he decided to fill the void. The only options at the time were systems produced by such giants as Reuters and Bloomberg. They were designed strictly for the large professional market houses, were difficult to use, and prohibitively expensive (about US$10,000 or more per year). Recognizing a business opportunity, Axel decided to fill the void. Gradually, programming supplanted Axel Sellemerten's university studies more and more, until in 1989 he founded his own small company, for legal reasons named after him. In 1990, he released a DOS version of a program that could track stock prices, show historical charts of the stocks, and compute some technical indicators. Sales the first year were about DM 300,000[1]. In 1992 Axel renamed the company after that first product: MARKET MAKER Software GmbH. Soon, a second computer science student, Christian Hank, joined in to help in the programming tasks. Except for a short break, Christian was to stay with the company for quite some time: Today he is its CEO.

Software that tracks stock market data is no good without data to track, and so hand-in-hand with the software came a data service that provided the latest stock quotes. In the beginning, the data went out by mail on floppy diskettes, on a weekly basis. The data was typed in from the financial pages of various newspapers. One of the first jobs the new company offered was data enterer, as the availability of data was so crucial to the company concept.

The data service is an important part of MARKET MAKER's business for two reasons. First, it provides a dependable revenue stream, for people came to recognize that MARKET MAKER's data was more dependable and accurate than that from other sources. Just as important, it instilled a sense of identity within MARKET MAKER: They were not just a provider of shrink-wrapped software systems, but they were a *service provider*, with a vital, direct, and ongoing link to each and every one of their customers.

Soon Market-Maker started to acquire data from other data sources and to merge and repackage it. While the diskette service stayed around for quite some time, on-line download of the data files became the dominant method for bringing data daily to the customer. Customers could subscribe to different data

1 Roughly $135,000 at January 2001 exchange rates.

packages (such as for German markets, European markets, Asian markets, various indices, etc.). Data grew to include bond markets and tracking of other securities. Soon, data provision became MARKET MAKER's major source of revenue.

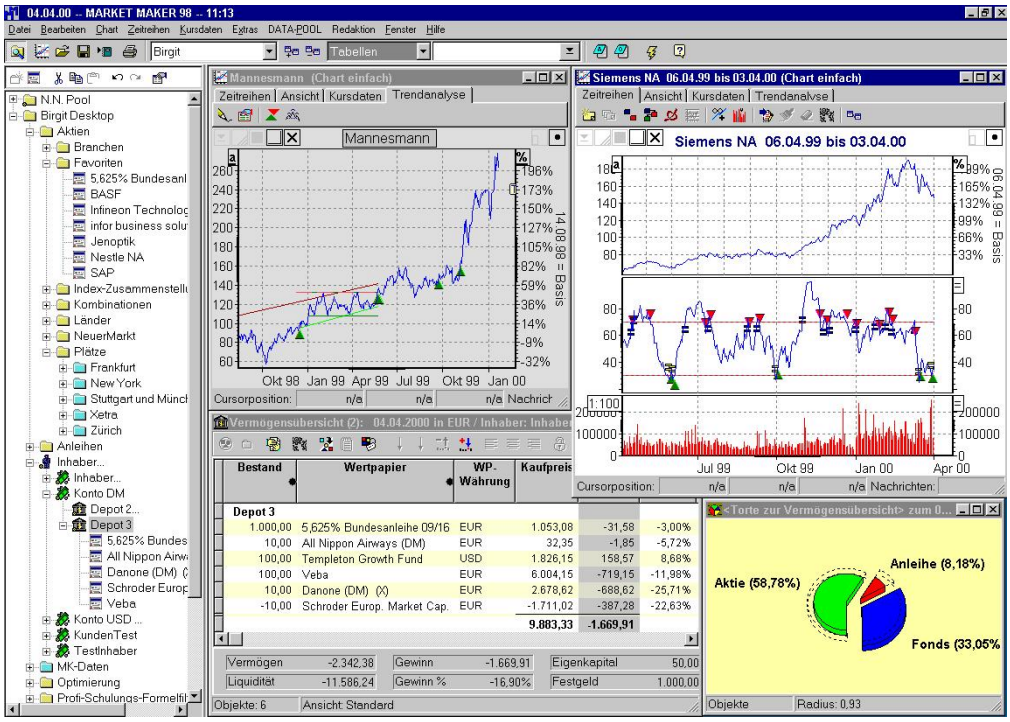
The DOS-based MARKET MAKER software needed to be able to handle all these different subscription possibilities transparently. A second, more important source of variability was introduced in 1990. Already at this early stage an additional package called "technical indicators" was developed, which was a separately chargeable package of functionality, but was at the same time strongly integrated into the core functionality of the system. Thus, early on the company had to face the challenge of developing and maintaining their software in different variants.

This, then, was MARKET MAKER's Phase 1: Their DOS product was quickly accepted by private consumers, but to the delight of the fledgling company, they also found that small asset managers and investment advisors liked it. It was highly capable, and carefully engineered to show useful information quickly. And it was affordable, the price was between DM 500 and DM 4000[1], depending on the precise version.

As the reputation of the company grew, it was also able to sell this software to banks, who equipped their investment advisors with it. But this introduced another source of variability into the software development as the system needed to be able to also work with bank-specific data. Over time more and more packages of functionality were added to the MARKET MAKER DOS version, like depot management[2], trend-analysis, and option-strategies. This led to increasing complexity of the product and a strongly growing set of variabilities the software development had to learn to deal with. This second phase of the company, informally defined, lasted until about 1997.

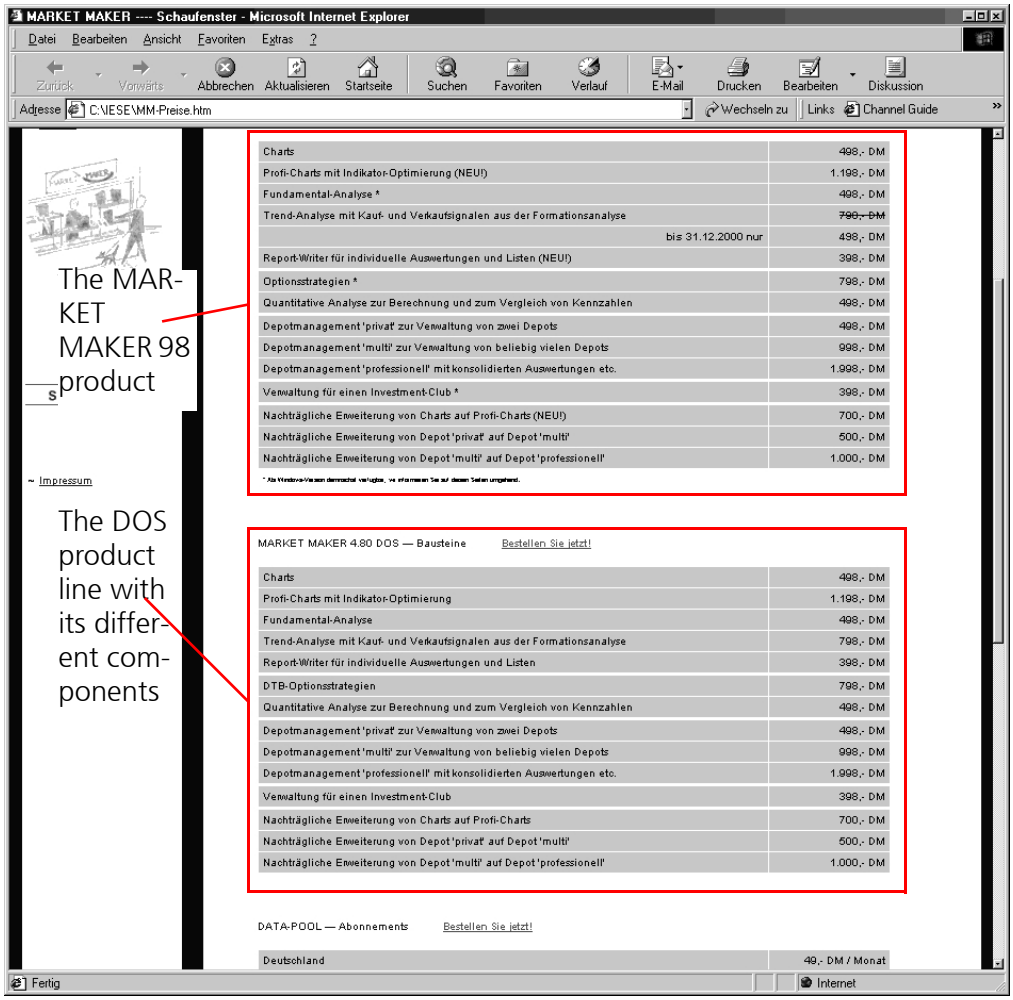
1 Ranging between $220 and $1,000 at January 2001 exchange rates.
2 A depot account is analogous to a portfolio account in America.

Figure 1:
MARKET MAKER 98



During this second phase the company's developer ranks swelled to five. These developers started in 1995 to work on a new version of the software, later to be named MARKET MAKER 98 (see Figure 1). This project aimed at a complete re-implementation of the software that would support all the previous functionality modules as well as future variability and would take full advantage of the potential of visual metaphors that the new Windows environment provided.

Figure 2:
Some products from MARKET MAKER's current palette



The third phase of MARKET MAKER's history can be characterized as broad exposure to the market. The release of MARKET MAKER 98 got the attention of a German television show named WISO, which focuses on financial issues (such as taxes, investments, etc.) and is presented by the ZDF TV network. The name of this TV show had already been used successfully to market specialized software (e.g., tax solutions). Now the idea was to use this label for branding stock market analysis software. A candidate for this was soon found: MARKET MAKER 98. However, the software was actually too powerful for what the marketers had in mind. Thus, a new offspring of the MARKET MAKER 98 family took form under the name of *WISO-Börse*. Soon, with the help of this branding strategy more than 150,000 copies were sold, making it Europe's most popular stock market software. As a result of this wide exposure, MARKET MAKER now enjoys a solid reputation for producing a quality product that produces customer satisfaction. The success of WISO-Börse led to more offspring products WISO Fondsmanager and Telebörse[1] and Telebörse Trader. Figure 2 is a screen shot of MAR-

KET MAKER's Web page giving order information for MARKET MAKER 98 as well as the original DOS version of MARKET MAKER. Both show the rich variety of functionality that can be ordered by a customer and provided modularly by the company.

You can visit the current product list at: http://www.market-maker.de/Schaufenster/Preise/.

All of this software was based on the concept of daily stock data, with only six data points per day. But in 1996 the company introduced a new product: MMLive!. This product was developed independently of the MARKET MAKER 98 product line and addressed the market of real-time data. However, the product never was a success comparable to the MARKET MAKER 98 products, nor could it form the basis of a product line of its own. Nevertheless, the existence of this product was still instrumental in the development of the product line which is the subject of this paper (see Figure 3).

Around that time (mid-1997) a first cooperation between MARKET MAKER and the Fraunhofer Institute for Experimental Software Engineering (IESE), a German institute for applied research, was established. Together with five other small-to-medium enterprises, MARKET MAKER joined a project run by IESE called "Software-Variantenbildung"[1] centered around the idea to adapt product line concepts to the specific needs of small organizations and to transfer them in pilot projects [3], [2]. Based on that experience, a strong cooperation was established within the project ESAPS[2] and some smaller consulting projects.

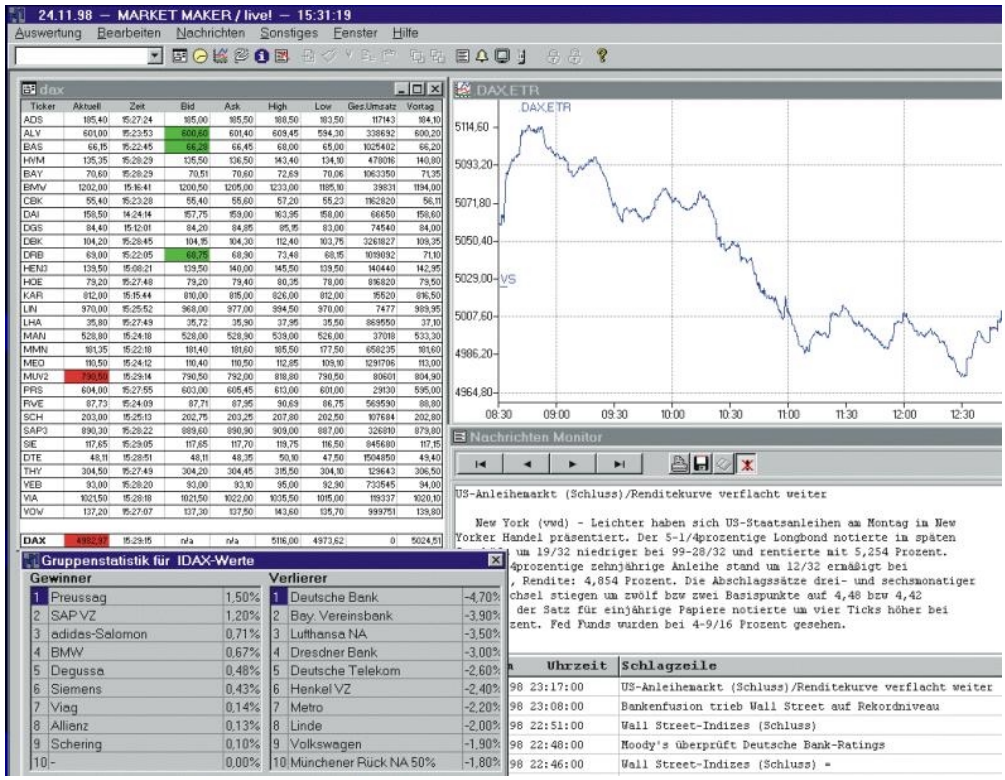
In 1998 an important business decision was made: the company ownership structure was now based on stocks, a structure which in Germany is not as common as for example in the U.S. In 1999 the publisher of the major daily financial newspaper in Germany, the Verlagsgruppe Handelsblatt GmbH, bought 35% of MARKET MAKER's shares. This investment provided the company with a potential for future growth, as sufficient financial backing for major deeds was now available.

1 Telebörse is a daily TV news magazine focussing on the private stock investor.

1 This project was partially funded by the Ministry of Economic Affairs, Transportation, Agriculture, and Viniculture of the state Rhineland-Palatinate, Germany.

2 ESAPS (Engineering Software Architectures, Processes and Platforms for System Families) is a large European research project (Eureka Σ! 2023 Programme, ITEA project 99005).

Figure 3:
MMLive!



Over time, especially recently, the company had grown steadily and strongly. In 1999, the company had about two dozen people; their average age was about 30. Flush with success, MARKET MAKER knew the time had come to orient its products more towards big businesses. The banking climate of the time was (and still is) tilted towards more centralized computing services with thin low-cost clients running identical software, mostly browsers. Branches with high-powered PCs doing specialized computing at each location was viewed as too expensive. Therefore, network solutions were and still are preferred.

Moreover, the company had to realize that at a time when everybody was talking about the internet, the company did not have anything like an internet strategy. There, a major contributor to the network-based push was the investment market. Although private German investors have lagged behind their American counterparts in rushing to private stock market investment, German banks know the wave will soon be upon them. In 1997, German banks opened the first internet-based brokerage houses, and are now investing to try to prepare their branch offices for high-volume asset account management. Widely accessible information is now the watchword. Websites have sprung up offering a multitude of different information and content, in which financial markets appeared next to daily news, sports, and weather, for example.

It was in this climate that the strategic insight was born that all those new markets could be addressed by means of an internet-/intranet-based line of products that would leverage on existing technology of the MARKET MAKER 98 line of products. It is this internet-/intranet line of products which is the subject of this software product line case study.

3 The MERGER Software Product Line

In early 1997, Martin Verlage and Christian Hank of Kaiserslautern, Germany, met. Martin was at the time a researcher and project manager at the Fraunhofer Institute for Experimental Software Engineering. His work focused mainly in the software process area, about which he also wrote his Ph.D. dissertation at the University of Kaiserslautern that same year. Christian happened to be the CEO of a small Kaiserslautern company called MARKET MAKER Software AG.

Discovering their mutual interest in technology, Martin and Christian began discussing ideas for new products. Martin was interested in Christian's domain, and Christian was interested in Martin's knowledge of state-of-the-art software system technology and ideas about how to apply it. For the next year and a half, these two held informal discussions. Christian offered Martin a job in July of 1999, and Martin accepted.

Martin was brought onboard because of his technology expertise, but he did not wish to simply run studies and provide internal consulting. He wanted to contribute to the company's cash flow. At the time, on-line products were forming (in concept) a whole new business area for the company. Martin recognized the business drivers for the new product family right away, although they were formulated informally and imprecisely. The need to bring out a tailored product "as early as possible" topped the list. Martin adopted the year 2000 as the goal for the new product. He also knew that MARKET MAKER 98 was going to serve as the engine for the new product, and that re-implementing its 400,000 lines of code was out of the question. Yet it was written in Delphi, which is a Pascal-based environment for Windows user applications and not a viable choice for the rest of the system. Martin's first job was a technology investigation to solve this mismatch and other problems. The solutions would find their way into the architecture for the new product.

About the same time, MARKET MAKER received the opportunity to perform an in-depth analysis of their competition, such as it was. The Handelsblatt publisher group, which had purchased 35% of the stock of MARKET MAKER Software AG, asked them to perform a comparison of internet-based financial tools. They asked MARKET MAKER because of its expertise in financial tools; the part about being internet-based was purely serendipitous. Christian helped a great deal in this exercise. As the two of them visited, evaluated, and critiqued financial Web sites, Christian pointed out to Martin places where the software had obviously been (as he put it) "created by technicians" and not someone fluent in the nuances of the financial industry.

As the vision for the new product took form, MARKET MAKER management made a conscious decision to hire new people to build it. They reasoned that their existing workforce, with their experience and background, were sorely needed to keep maintaining and updating MARKET MAKER 98. This worked both ways; the new developers would not have to have their attention "polluted" by a stream of phone calls and trouble reports on the existing system. Also, the internet product was going to require skills and technological expertise not readily available in the company. And so, hiring began, but not at what one would call a frenetic pace. The first developer was hired in October of 1999, and in November of that year the first lines of new code were laid down for the product. The second and third new developers came in January and April of 2000; the latter was part-time. Another joined in April. And two more (one of whom was part-time) rounded out the new team on August, 2000.

The first order for the new product was signed in February of 2000. Throughout late 1999 and early 2000, the vision for the product line, which was christened MERGER, crystallized. It had to reuse MARKET MAKER 98, but the architects thought more in terms of reusing the vast domain knowledge embodied in that package rather than simple reuse of code. Time to market for family members was absolutely critical. Therefore, the instances had to be easy to produce and eminently flexible in pre-planned ways. The system had to be open, in that it would need to work with commercial and third-party products of unknown variety, and run on a wide range of systems, servers, and platforms. Finally, low maintenance effort was a requirement, and this flowed directly from a business philosophy: MARKET MAKER wanted to use its small cadre of personnel to expand their customer base, and not have them spend time on maintaining what they already had. Figure 4 summarizes the business drivers for the MERGER project. Even a non-speaker of German will recognize "Schnelles Time-to-Market" and "Flexibilität" as critical drivers.

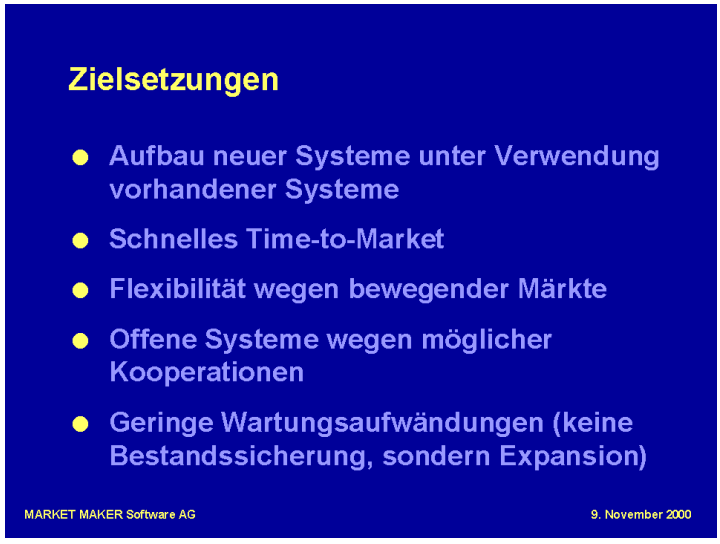
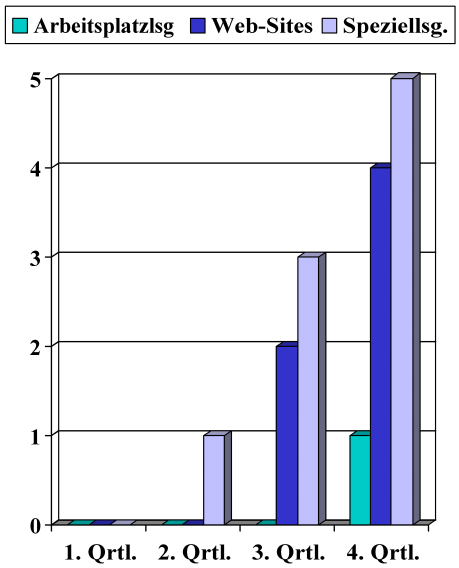


Figure 4:
A MARKET MAKER viewgraph spelling out the quality goals for the MERGER product line

After 12 calendar months (and roughly 36 person-months[1]) of development activity, the first product was deployed. As of the writing of this report (late November 2000), ten members of the product family are in service (see Figure 5), and many more are planned.

Figure 5:
Releases of MERGER

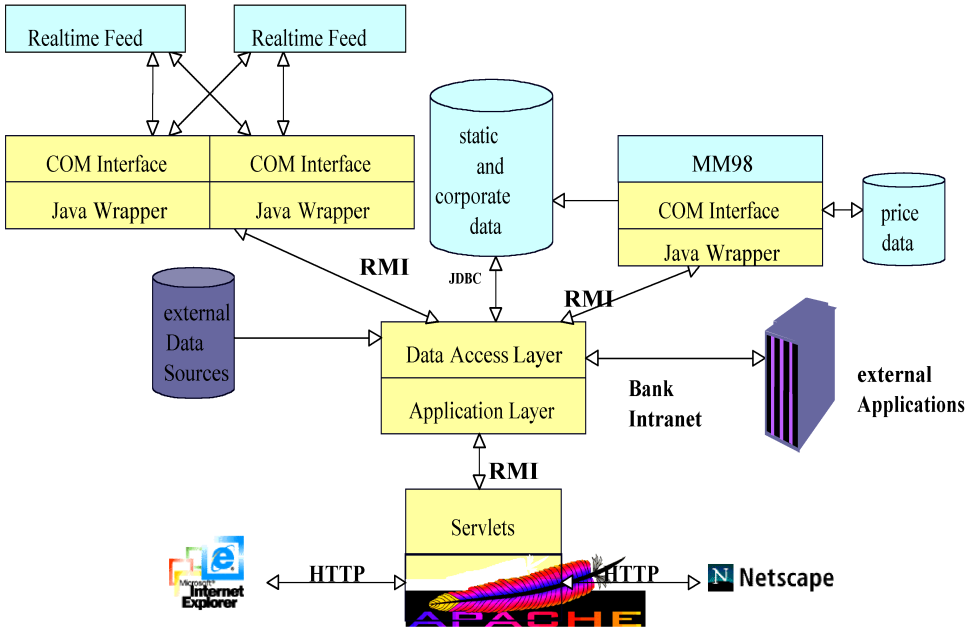


1 This number assumes eight-hour work days, although the norm was closer to twelve.

4 MARKET MAKER Software Product Line Practices

4.1 Architecture Definition

Figure 6: Architecture Overview of MERGER



The design of the architecture of the MERGER product line started in October 1999 when Martin went through in-depth discussions with the first developer hired for MERGER development. Based on their different ideas and experience they discussed possible ways to build the new product line and eventually defined the architecture as it is sketched in Figure 6. The current size of the MERGER product line is about 120,000 lines of Java code in addition to the 400,000 lines of wrapped Delphi code used from MARKET MAKER 98.

Figure 6 shows the different possible data sources for the MERGER product line: the Data Access Layer can process data from realtime feed (reusing parts of the MMLive! product), from external (i.e., customer-specific) data sources, and from MARKET MAKER 98. All these data sources are accessed using the Java RMI (Remote Method Invocation) technique. Only for external (i.e., customer-specific) applications specific interfaces have to be built, these applications are then accessed directly. The Data Access Layer passed the data to the Application Layer to perform computations on them. The resulting data objects are passed on to servlets (using RMI again) that generate respective HTML pages. These pages are made available by an Apache Web server[1] and can be accessed using standard Web browers like Netscape or the Internet Explorer.

The two most important requirements for the systems to be built were performance and 24/7 reliability. The approach taken was to abundant redundancy to the architecture via a varying number of servers, and to provide extensive caching for achieving good performance. Other important aspects to be considered were the following:

- Although functionality of MARKET MAKER 98 had to be reused, MARKET MAKER 98 could not predetermine the architecture of the new product line. To achieve this, the 400,000 lines of Delphi code of MARKET MAKER 98 were encapsulated as COM components and for access in black box fashion.
- In a similar way, the MMLive! product has been integrated. The reuse of this otherwise peripheral product (compared to the success of MARKET MAKER 98) provides MERGER with the ability to process tick-by-tick data (while MARKET MAKER 98 works with historical data only).
- The MERGER products must be open, in that it they need to work with commercial and third-party products and run on a wide range of systems, servers, and platforms. This requirement led to the use of standardized approaches wherever possible: standard (not enhanced) Java virtual machines, standard (not extended) SQL, and other established standards like RMI, COM, HTTP for communication, and JDBC/ODBC (Java Database Connectivity/Open Database Connectivity) for accessing external data bases are used.
- The MERGER products have to be integrated in different customer environments, implying that they have to be able to access external data sources present in the target environment. This was achieved by separating data handling in a layer different from the application layer. The data access layer manages the handling of different key sets used in the applications and databases, and caches data. It may interact with existing external data sources and applications to retrieve and deliver data.
- The application layer is responsible for representing/visualizing the data. Here, Servlets are used, allowing for separation of data visualization from internal data representation. This technique enables easy customer-specific design of GUIs, making each customer's site look unique and different from those of competitors, while being based on the same product line.
- The application layer holds complex objects (e.g., dynamic tables) and performs computations on the data. The objects managed by the data layer are passed to servlets that generate HTML pages for presentation to the user. The servlets also manage sessions to overcome the deficiencies of the stateless HTTP (Hypertext Transfer Protocol).
- No code changes are permissible for adapting to local environments. All instances must be fully derivable from the code base.

1 The Apache Project is a collaborative software development effort aimed at creating a robust, commercial-grade, featureful, and freely-available source code implementation of an HTTP (Web) server.

The concept to build a system from modules is well-known and well-liked at MARKET MAKER: the first DOS version of MARKET MAKER was constructed from components in order to be able to develop one component at a time (bowing to resource constraints) and to be able to sell partial functionality to customers (achieving a marketing strategy). That strategy has been applied even more to the MERGER products by making them customizable internally. Each box in Figure 6 is subject to tailoring (but not re-coding) when delivered to a customer.

Customization of the system is (mainly) done by changing property files. These files control the system behavior in various ways while avoiding the need to change any code (as far as the adaptations have been planned and realized). Examples for possible customizations are the services available at a customer site, the external data sources used, the external applications to communicate with, or the platform the system runs on. "My vision is to have one common code for all our customers and to have all information necessary for the installation of a customer-specific version on one floppy disk." Martin Verlage said. "That means, the code should not be changed in order to be adapted to a local environment."

To realize configuration by property files the Java reflection mechanism is used: on demand, the (versions of) classes needed are instantiated and used in the particular system. Sophisticated Makefiles are no longer needed. Currently, eight to twelve property files have to be adapted per customer. "An installation manager that would generate these property files would help to make the installation even easier," according to Martin.

4.2 Component Development

The development of the software was from the beginning driven by the vision of a product line. This is visible in the implementation as well as the architecture. Throughout development a product line vision has always been a major driver for decision making. Even if delivery pressures necessitated taking shortcuts, MARKET MAKER was always careful to analyze implementations to look for (and extract) new opportunities for commonality and variation. Each product in the product line is not a bulk piece of code, but consists of individual services that are actually started at initialization time from a master. This requires that each of the service implementations is actually quite self-contained.

During analysis and implementation a small number of services was identified which are present in nearly every system but are subject to specific tailoring, for example, caching. These services were implemented in a very generic manner using Java abstract interfaces. For each service a single class implements all interfaces (e.g., access to cached objects, refreshing objects, removing objects, controlling the service, monitoring). The class is instantiated in a particular instantiation by using Java's reflection mechanism. A string specifying the class is read

from a configuration file and used as parameter for **java.lang.Class.forName(<string>)**. The result is input to **java.lang.Class.newInstance(<class>)** method call. While the current implementation looks quite simple, a considerable amount of effort had to be expended to come up with such an elegant (and small) solution. This illustrates a major fact: While reusability does not come for free, it can be aided significantly through the use of modern programming mechanisms.

While in the cases above carefully hand-crafted implementations were used, in some areas of functionality generator-based approaches were used. A particular example for this is the design of Web pages. Here, a third-party generator package is used to describe the to-be-generated Web pages in an abstract manner: the page layout can be described in the form of a template, instead of programming the format explicitly. A similar approach is also used for generating reports. The introduction of this implementation approach has even organizational ramifications. In this case, it is now possible to delegate the design of the Web pages to a designer, who does not need to have a true programming background. The pages he is designing are later turned into templates.

The strongest usage of generation is in the area of chart generation, which can even serve as a template for the future development of the implementation approaches. While charts can also be generated using MARKET MAKER 98, it was decided to re-implement this functionality and to make this re-implementation strongly configurable, anticipating the needs of the customers who want strong diversification in the look-and-feel of their Web sites. Now it is possible to configure the look-and-feel of the charts using an additional customization tool, to see how the charts will look, and generate the corresponding configuration file with a mouse click. This reduces the customization time to minutes.

One other piece of replicated functionality is portfolio management. The intention here was to offer a less sophisticated version of the MARKET MAKER 98 features to whet the appetites of MERGER users. High-end functionality of MARKET MAKER 98 would be available for a higher price, but with no extra development effort.

4.3 Structuring (and Staffing) the Organization

Martin Verlage was hired to create a new line of products. In fact, he started with the in-depth analysis of competitors in the market (the first project for Handelsblatt) and in another project where he did some technology evaluation. Both topics helped him to understand the domain and the opportunities that new technology concepts like Web development with Java and its abstraction mechanisms offer.

For the development, new people were hired. Hiring external people proved to be an advantage because they were not involved in existing MARKET MAKER products, that is, they could concentrate on their new tasks and learn about new technology without being distracted by pressure in the development of old products. Moreover, all of them were hired for developing in a product line context, which helped to instill a common understanding of the products, the development style, and the philosophy.

The team is now moving towards two organizational parts: the first group will deal with development, the second one with installation. The development group comprises the more strategic planning of the product line as a whole and the development of generic (customizable) components. The second group (currently only one person) is concerned with adaptation of these generic components to the needs and wishes of specific customers including the installation at customer's site. These two tasks require (at least partially) different skill sets but also offer different opportunities for the people performing them.

Recently, two people with experience in the banking domain were hired, mostly for testing purposes from a domain (i.e., not component) point of view.

One thing Martin emphasizes as being important with respect to company organization is to work closely with new people to integrate them from the beginning and to communicate the product line concepts to them.

4.4 Testing / Maintenance

For testing MARKET MAKER follows a two-fold strategy:

- First, single components are tested by the developers themselves. All components are enhanced with special code which enables also later testing at runtime, independent from the (runtime) environment. After successful component tests, integration testing in the MARKET MAKER environment is done. Then property files[1] are set up to adapt the system to the customer's (runtime) environment and the system is installed there.
- Second, the system can be tested at runtime in the target environment using the special test code in each component. If the components (that are instantiated using the property files) still adhere to their specification the whole system is running properly. Otherwise, the source of the error, that is, the misbehaving component, can be identified directly.

1 These property files control the system behavior in various ways while avoiding the need to change any code, cf. Section 4.1.

Testing the system from a domain perspective becomes more and more important over testing single components. For that purpose, sometimes people from the MARKET MAKER 98 group with experience in the domain of funds management have been engaged. Recently, two people with experience in the banking domain have been hired to support the domain-oriented testing.

Components are not tested in their full flexibility, that is, with respect to all possibly instantiations in the family, but only with respect to the next product to be instantiated.

According to Martin Verlage, the documentation of the product line architecture is kept too generic to be evaluated directly. Instead, MARKET MAKER always talks about example systems – product line instances. Thus, the architecture is instantiated at least partially before it is evaluated using scenarios. That is, testing of the architecture as well as for the code is done for (partial) instances only.

4.5 Data Collection and Metrics

Until recently, MARKET MAKER did not perform any detailed measurement, relying instead on the experience of their developers to predict effort required. However, they are searching for a more disciplined and repeatable way. Thus, according to Martin Verlage, they would like to pay more attention to measurement in the future to be able to tell precisely when investment in the product line pays off.

Thus, recently, MARKET MAKER started a rather ambitious measurement program, aiming not only at a detailed analysis of their costs, but beyond trying to identify the factors influencing their costs and the benefits they achieve through reuse. For measurement, MARKET MAKER distinguishes development *for* reuse, development *with* reuse and *system-specific* development:

- Development for reuse comprises the creation of new components or integration of new features in existing components.
- Development with reuse is basically the instantiation of existing generic components for new systems to serve the needs of certain customers.
- System-specific development aims at the development of customer specifics.

However, at this point it is not yet possible to precisely describe the costs for each of these three forms of development activities and determine the respective influential factors. By the time they are successful with this effort, they will be able to thoroughly plan where spending extra effort for reuse will truly pay off.

4.6 Launching and Institutionalizing the Product Line

Based on the success of the previous MARKET MAKER products it was possible to carefully plan and construct the MERGER product line without being forced to make money from the beginning by developing for a particular customer. This way, the MERGER platform was not built as an extension of a "first MERGER product" but it was possible to consider more than one single product for planning from the beginning. Here, the comparison of different existing systems that was ordered by the Handelsblatt publisher group (cf. Section 3) came in handy. In addition technology feasibility studies were performed.

Of core importance for defining a successful product line is to identify the right requirements and market segments that are to be addressed. This is only possible if one has already an adequate understanding of the market (cf. Section 4.7). After initial development had been performed, the MERGER concepts were shown around at trade fairs as a kind of a customer study to get ideas, comments, and other feedback from potential future customers for evaluating the initial concepts.

After delivery of the first product, the visibility of MERGER suddenly became quite high. This created a lot of market pressure, forcing MARKET MAKER to release new MERGER products very fast. At this point in time, it was impossible to do careful planning and generic development for potential future requirements, instead many products had to be derived from the existing platform in a very short period of time. Sometimes this resulted in later needs for the re-consolidation of the platforms. But the important part is: these were actually performed. This development path of the product line is actually one expression of the vision of all the MERGER systems together as *one product*.

Martin Verlage believes that having some kind of a quiet planning and development phase before the actual delivery of a product was the only way to plan and create MERGER as a sound platform for the development of customer-specific products later on. This avoided the strong influence of a "reference customer" from the beginning.

4.7 Understanding the Market

"Good knowledge about the market is a prerequisite for success." The people at MARKET MAKER view their systems as providing higher-quality information and unique and more valuable services than that of their competitors. To achieve this, they work very hard very consciously to establish and maintain close contact with their customers, to know what they want, how they use the information, and what their jobs entail. Information about their domain flows in by various means:

- The company makes it easy for customers to contact them. For example, MARKET MAKER 98 customers can call a free hotline to ask questions, voice complaints, report problems, or ask for new features. Some customers call over 150 times a year. Once, a customer called to say that the software was computing a technical stock indicator incorrectly. The indicator was one that was not that well-defined to begin with in the economic community, and it would have been easy to side-step the problem. The company responded, however, by mobilizing a small research force. People went to libraries to pore over economics books, others scoured the software to make sure they understood the nuances of what it was computing, and all worked with the customer to make sure they understood the problem. In the end, the software was modified, the company was more knowledgeable, and the customer was delighted.
- The company holds seminars for its users. Classes for beginner investors, for portfolio analysis, for chart analysis, and others are held regularly. Asset managers are invited to gather to talk about their desires for future versions of the company's products. "You know your customers well or you're out of the market."
- Knowledge of competing systems is part of the job description of the company's chief marketer.

Over the years MARKET MAKER came to know their customers well using these and other means. And the comparably small size of the company strongly helped to make the customer awareness a pervasive feature throughout the organization. This knowledge was an important prerequisite to define the product line right and define the right product line. A simple example may illustrate this: over the years MARKET MAKER came to know their customer group of independent investment advisors (typically independent two- to three-person agencies offering investment advice) very well. They knew that the customers of these advisors would like to check on the status of their current investments over the Web. On the other hand, the investment advisors would certainly not like to set up and run their Web servers. Thus, it was a requirement for the MERGER products addressing this market segment to support Web-hosting, so that the customers of the investment advisor could get their investment account information from the MARKET MAKER Web-Site, where the information was always automatically updated by the MERGER copy running at the investment advisor's office. These ideas were present from the start of product definition. Thus, there was no need to retrofit these key distribution requirements into the MERGER architecture at a later point in time, perhaps compromising the architectural integrity of the system.

4.8 Technology Forecasting

Besides knowing all this information about your market you also need to know what you don't know. MARKET MAKER knows that they can plan for at most the next three to five years. Beyond this uncertainties become to big. Thus, they are committed to use this time frame well.

"As a small company, we can't afford the resources to run our own research department. Instead, we use Fraunhofer IESE as kind of an external research department." Based on the experience from past cooperations (cf. Section 2), there exists enough trust within MARKET MAKER to rely on IESE expertise when questions about, for example, product lines, security, or documentation standards are discussed.

Most domain-related technology know-how necessary was acquired during the first project for Handelsblatt (cf. Section 2). In this project, competitor products present on the Web already were analyzed with special emphasis on their way to present information and their support for interaction with the user.

5 A Few Observations

This section shares lessons learned from a few of the key facets of the MARKET MAKER experience

5.1 Effects of Company Culture

When trying to extract lessons from the MARKET MAKER experience, one cannot help but be impressed by two aspects of the company's culture that have conspired to breed success. The first is an abiding, almost cherished, relationship with their customers and customer community. And the second is a long tradition of, and faith in, the idea of a few products tailored for many customers.

MARKET MAKER saw itself early on as a customer service company; its high-quality data distribution service cemented that vision from the earliest days. We have already seen how MARKET MAKER works hard to cultivate customer relationships. But the watchword at MARKET MAKER is expansion of the customer base, not just maintaining the status quo. Deep insight into customer needs allows them to do this. During Martin and Christian's first informal discussions, they often talked about new capabilities. "Suppose our client wanted to work while sitting in his garden on a Saturday morning, sipping tea," they might say. "He wouldn't want to drag his laptop out with him. And he certainly wouldn't want to scan the newspaper to try to extract the blend of information he wanted. Wouldn't it be nice if we could fax him the information he needed? He could just pick it up on his way out of the house." Detailed and somewhat offbeat scenarios like this reflect both a deep understanding of who their customers are, and also a deep commitment to provide an outstanding level of customer service. The latter is not possible without the former.

MARKET MAKER's relentless drive to satisfy its customers led right away to the concept that different customers require different capabilities. The DOS version of the tool started out as a core product, with new functions added modularly over time to meet the needs of a growing market. MARKET MAKER 98 has highly modular functionality: Customers purchase the features what they need, and are not burdened with what they do not. In both of these cases, a software tailoring capability flowed directly from a marketing decision. But the extra idea here, the intellectual coup that leads to a software product line, is the realization that *it is possible to serve different needs with the same product.* In the case of MARKET MAKER 98, every customer receives exactly the same software, no matter what functions he has purchased. Some features are simply disabled. There is a breathtakingly elegant vision that is in place in the company, and

although not quite realized yet, the vision sets the stage for the unshakable institutionalization of the software product line paradigm. The vision is this: Every time a new version of the software is released, say four times a year, new CDs are burned. Every customer gets a CD with the precompiled binaries for his operating system (Web servers, database drivers, and installation routines), and a floppy disk containing keys that allow the functions he ordered to be accessed. At installation time, the customer loads the CD and the floppy, and the system auto-installs itself. This epitomizes a product line culture.

To reinforce this admittedly abstract notion of product line culture, we note that during our initial conversations, Martin kept referring to "the product". A product line is, of course, a set of products. When we asked for clarification, Martin told us that the MARKET MAKER culture is to produce "one product to serve many customers." All then became clear, and we knew that the product line culture was strong. In our experience, mature product line organizations refer to what they do in the singular: They build and maintain and nurture the *single* entity that is the product line. By contrast, immature or struggling product line organizations think of themselves in the business of turning out *products*, albeit tightly coupled ones.

5.2 Cost Issues

As we have seen with other software product line examples, cost is conspicuously absent from the list of driving forces behind the effort. "Reducing cost" simply doesn't give one a foothold with which to make progress. Or, as Martin put it, "You can't reduce cost by thinking about cost." But certainly cost is in the back of everyone's mind, and savings from the rapid introduction of a multitude of mass-customized products will undoubtedly satisfy the implicit goals.

MARKET MAKER expects their internet effort to pay for itself by mid-year 2001, about two years after the project's inception. If the price of the products were not artificially low to gain market entry, the pay-off would occur about six months sooner. There is certainly every reason to believe that for each product deployed subsequent to the break-even day, the payoff will be substantial. And the market outlook is optimistic: MARKET MAKER can field more flexible products at lower cost than its competitors, which gives it the freedom to price flexibly in order to execute market strategies.

5.3 The Customer Paradox

A product needs a customer – but not too quickly. For its internet product, MARKET MAKER "had the luck not to have a customer at first." In fact, in retrospect, another six months without a customer would have been welcome. Although this sounds heretical, a customer waiting anxiously for a product

induces delivery and specialization pressures that can derail long-term goals. Developing in the absence of such pressures helps assure more generic (and therefore, more broadly useful) designs.

And yet, developing in a vacuum is not a good idea, either. A customer presents you with absolutely authentic needs and requirements. Absent that, your product may be elegant but useless.

MARKET MAKER balanced these conflicting goals by gathering feedback from early presentations and showing prototypes at trade fairs, and by keeping their existing customers informed. But they resisted the strong temptation to promise delivery of those prototypes, even when faced with eager buyers.

5.4 Tool Support

Product line development in general requires not only changes in the development process but also in tooling. Proven approaches to cope with complexity might get less usable when starting to use product line ideas. Setting up an environment prior to launching was not an issue at MARKET MAKER. Time did not allow for, cost did not allow for. Although having experience in the field of instantiating products by switching on modules in MARKET MAKER 98, these instantiations were somewhat limited in the set of possible instances. MERGER is much more complicated and therefore more need exists for tools to handle the complexity. When being asked about the tools he wished he had, Martin's answers were all about instantiation:

- An Instantiation tool: this would allow to generate the property files (cf. Section Section 4.1) for adaptation of the products to the target environment.
- A tool for document management: this would allow to develop (technical and user) documentation of the products in a generic way like the software. This way, customer-specific documentation of the software could be instantiated like the software itself.
- A tool for testing: this would check the correct installation of the software in the environment target and at runtime of the system the correct behavior of the instantiated system components.

Of course, the absence of tools has consequences to quality but more to time. Instantiating a product by changing dozens of lines in configuration files is a burdensome task, error-prone and annoying. Nevertheless, without learning from these painful lessons there couldn't be drawn precise requirements for automated support of a product line process.

Martin's advice about tool support is characteristically to the point: "Don't expect perfect environments," he says. "Just do it."

5.5 Lessons Learned

Martin shared with us what he would do differently, if he could. The list is a short one, and includes these concerns that will not be a surprise to any manager:

- *Produce more and higher-quality documentation.* In the beginning there was too little staff and hardly any need to communicate information beyond the inner circle of developers of this information, later there was just no time for systematic documentation given the numerous contracts.
- *Give his team more time before the first delivery – ideally, about six more months.* Time is needed to design and develop the platform right, from the start. However, at some point you just have to get started. This is hard to balance, even in retrospect.
- *Spend more time with his team, to better understand their satisfaction drivers.* All the team members were new people, just entering the company. If you want to work successfully over a long time with a team, you have to get to know them.
- *Pay more attention to data collection, to better understand where effort is being expended.* Data analysis would help to build first models for a second product line. This would help to guide project managers through the process of defining the product line, architecture analysis, and instantiating first products.
- *Install an error-reporting system to better track problems and problem fixes.* Because that the team was working on architecting the system instead of programming, their attention was pulled away from small functional flaws, which resided afterwards in the software for some time.
- *Pay more attention to the environments in which the products are to be installed*, to better understand the needs and requirements of the operators, the installers, etc. These stakeholders are not particularly impressed that the software they're getting is a product line, but are impressed by ease of use and understandability of the configuration files.

5.6 Drawbacks

The MARKET MAKER story is one full of pleasing results, but of course all is not perfect. Martin shared these weaknesses of the software product line approach.

- There is no clearly defined archetypal product. It is therefore easy to promise a customer just about anything. For example, in the MARKET MAKER demo, the customer sees the full version, with all the features enabled, and is often unpleasantly surprised when something less is delivered. Reminding him of the contractual specifications he signed does not usually diminish the disappointment. This problem should diminish as more installations are completed, providing more products available to show.

- Some changes must be made in certain ways simply to maintain architectural integrity, causing them to take longer than they might other wise. Such changes are hard for a customer to understand. Any given customer will want his system delivered (or fixed) right away, and has no desire to wait longer for changes that will benefit the company or other future customers in what is to him a very abstract way.
- Changes to the core assets require more time. In particular, they require group thinking because the goal is to arrive at the best solution for the whole family. Sometimes the core group spends the better part of a day hours in a meeting to work out one change. Their strategy is to ask, "What is the more general problem behind this special-case change?"
- The software product line endows flexibility, but only in limited (that is, pre-defined) ways. Not every change or customer whim can be satisfied. This underscores the need for careful design and deep domain expertise, so that the pre-defined avenues of variation are the right ones.

6 Conclusions: Software product Lines in Small Organizations

Why did MARKET MAKER adopt a software product line strategy for its internet product? It was, we were told, "the obvious solution." MARKET MAKER was operating with an extremely small cadre of developers, who could not hope to produce code for each customer's Website. Even had they employed a brigade of coders and testers, the time to market requirements precluded one-at-a-time product development. The software product line approach added a layer of complexity to an already (technologically) complex situation. But it removed complexity as well. As Martin put it to us, it is analogous to cars of today versus cars of 30 years ago. Today, they are vastly more complicated, but that complexity is hidden in the components, making things much simpler for the people who assemble and drive them. So it is with software product lines. The ability to generate code is adds complexity, but once accomplished the instantiation step is orders of magnitude simpler.

In addition to the expansion plans highlighted in Section 2 (including, for example, completing the electronic interface to banks), MARKET MAKER has plans to pursue other market segments. They, for example, understand the need to stay ahead of security issues. Some internet customers do not wish to maintain their own servers, nor do they wish to be exposed to the risk of hackers breaking in and stealing or corrupting resident data. MARKET MAKER builds up a new service as application service provider for financial advisors, where data is stored confidential and clients may retrieve the files via the Internet.

In our experience, there are several necessary ingredients for a software product line. The first is a champion, someone who has the vision for the software product line and has the communication skills and management attention (or power) so that others come to share the vision. Clearly the MERGER product line has a product line champion in Martin. He is articulate and likable, and his understanding of the approach and why it is supremely important for MARKET MAKER to follow it is utterly unshakable. The second is some sort of pressure to drive the organization away from traditional development paradigms. In other case studies, the software product line approach has been adopted because the company's clear alternative was annihilation; see, for example, the CelsiusTech experience [1]. Circumstances were not so dire for MARKET MAKER, and yet there was a clear sense that the company *could* not operate traditionally. It was more than a matter of product lines being a good idea that would bring about efficiencies. The third essential ingredient is, in our view, long and deep expertise in the relevant application domains. MARKET MAKER has this in spades, and has gone to great lengths to explicitly gather it and bring it to bear on their product designs. It is in large part what makes their products successful. Fourth

is an emphasis on and expertise in architecture. MARKET MAKER excelled in this area. The fifth essential is management commitment. Because MARKET MAKER is a small company, management is not distant from the working organization. In fact, it was the management that championed product lines-an ideal situation. The last essential ingredient is the ability of an organization to exercise process discipline. Here, to be candid, MARKET MAKER is not an exemplar. And yet, they do follow strong processes. When new versions of older products need to be hacked to meet, say, a Christmas marketing deadline, they always re-capture the rogue version by re-aligning it with their mainline release tree. No one in the internet group would dream of covertly cloning a piece of the core architecture to short-cut a product development task. And MARKET MAKER enjoys the advantages of being a small organization, in which oral cultures can effectively take the place of written process definitions.

Which brings us to our last point. We began this report by pointing out that small companies are not always considered the best candidate to launch a software product line. Throughout our information gathering, we listened for advantages and disadvantages that small organizations could expect to encounter when launching and fielding a software product line. Advantages of a small company over a large one include the following.

- It is much more easy to articulate a vision in a small organization and make it stick. This is true of *any* vision, but especially true of a product line vision. Product lines are typically sold to management because they hold the promise of lower cost and quicker turnaround. In a large organization, those goals are fairly abstract to the troops grinding out code. But in a small company, the developers are much more tuned in to the company's economic picture; there is a short distance from economics to developers. Nevertheless, Martin felt it was important to explain the rationale for the product line approach in terms meaningful to developers. He simply pointed out that a software product line approach would let them spend more time doing the things they wanted to – working on new features and improving existing ones – instead of spending mountains of time on repetitive tasks such as bug fixes, code patches, and installations. It worked. There was never any formal training in product lines at MARKET MAKER. The developers simply "got it". Of course, it didn't hurt that the average age of the developers was around 30. This tends to pre-empt arguments of the form "But we've never done it that way before."
- A corollary to the previous point is that it is much easier to find places where the vision needs reinforcing. An example is in forming effective development teams. As Martin told us, "I think that _____ and _____ are working very well together, and complement each other's skills. The same is not true of _____."
- A small organization can get by with lightweight product line processes, for they are used to lightweight processes anyway. For example, MARKET MAKER does not have a strong configuration management process, which is

a concern. But their small size allows them to get by for now, because word-of-mouth is an effective means of configuration management protocol, especially when (as in MARKET MAKER's case) only two people are allowed ever to touch the code. This is not by any means to be recommended, but it must be acknowledged that informal processes work in the small, and allows a company like MARKET MAKER to phase in stronger processes as they grow. Project and organizational planning are similarly less burdensome – which is helpful, because it is hard to make detailed plans involving a new approach that is employing new technology.

- A small organization's developers can more easily acquire useful domain knowledge.
- Managers in a small organization typically apply their hand at many tasks, including development, so they know firsthand where the approach is falling short.

And the disadvantages? There are a few.

- A small organization just starting out and struggling to survive may be sorely tempted to "sell its soul" to its first customer, promising anything and taking any shortcut to make it happen. And when the second customer comes along, survival pressures may make hacking the first product a tempting idea. As Martin put it to us, a small organization "always has a sense of urgency, and it is easier to induce pressure." Clearly MARKET MAKER could resist this pressure in part because it had another source of revenue – but so do many small companies these days, at least until the tidal wave of high-tech venture capital dries up. But MARKET MAKER also could resist because it knew the difference between short-term expediency and long-term profitability. What, after all, was going to happen when the third customer came along, or the fourth? The hacking, one-at-a-time approach would lead to unmanageable complexity, and in the not-too-distant future, disaster.
- A small organization has more risk exposure to personnel loss – but this is a risk not only limited to software product lines.

We began this report by repeating the conventional wisdom that small organizations are not well-suited for software product lines. We conclude by observing that, all in all, small organizations appear to be exceptionally well-poised to adopt the software product line approach. If product flexibility, short time to market, high staff productivity, and low maintenance are important business drivers, we now have existence proofs across all parts of the organizational spectrum that software product lines represent a viable approach.

7 Acknowledgments

MARKET MAKER is not the only small organization that has adopted the software product line paradigm and flourished with it; there are many others. But another observation about small organizations is relevant here. Since their survival is often tenuous, their people are reluctant to take time away from busy production schedules. They also tend to perceive the competition's hot breath on the back of their necks, making them understandably reluctant to see anything published that might give an advantage. Hence, it is exceedingly difficult to get a representative of a small software product line organization to sit still for an open-literature case study.

With this prelude, we acknowledge the extraordinary cooperation of MARKET MAKER MERGER's product manager, Martin Verlage. Martin spent an entire work day with us, answering our questions patiently and painstakingly. Of even more value, he answered important questions that we hadn't thought to ask. The authors deeply appreciate his insight, thoughtful analysis, and good humor. Without him this report would not have been possible.

8 References

Unless otherwise explained, quoted statements without citations are personal communication with Martin Verlage.

[1] Brownsword, Lisa & Clements, Paul. *A Case Study in Successful Product Line Development* (CMU/SEI-96-TR-016). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1996. Available WWW: <URL: http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.016.html>.

[2] J. Bayer, O. Flege, P. Knauber, R. Laqua, D. Muthig, K. Schmid, T. Widen, J.-M. DeBaud: *PuLSE: A Methodology to Develop Software Product Line*s, Symposium on Software Reliability (SSR'99), May 1999

[3] P. Knauber, D. Muthig, K. Schmid, T. Widen: *Applying Product Line Concepts in Small- and Medium-Sized Companies*, IEEE Software, September 2000

Document Information

Title:	Successful Software Product Line Development in a Small Organization – A Case Study
Date:	March 15, 2000
Report:	IESE-013.01/E
Status:	Final
Distribution:	Public