Fraunhofer Institut
Experimentelles
Software Engineering

Integrated Cost- and Investmentmodels for Product Family Development —

Integrierte Kosten- und Investitionsmodelle zur Produktfamilienentwicklung



Authors:
Klaus Schmid

supported by
EUREKA Σ! 2023 Programm, ITEA ip00004

IESE-Report No. 067.03/E
Version 1.0
July 21, 2003

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

Product family development focusses on systematically developing reusable assets over time that are used in later projects to achieve benefits in terms of development effort, time-to-market reduction, and quality improvement. Consequently, the costs from the initial development for reuse are more than compensated by the derived benefits. The initial costs of development for reuse thus take the role of an investment in later product line benefits [CN01].

Therefore it is key to successful product family engineering to identify costs and benefits that are associated with product family development. This provides the basis for the subsequent optimization of the investment. In this deliverable we focus on the relationship between costs and benefits and show the implications and constraints on the associated investment decisions. We present a basic economic model, which provides the necessary background for understanding the impact of investments in a product family. This model enables a systematic analysis of the product family situation and any consequences of product line investments.

A key part of any investment decisions is to define the cost implications of these decisions. Unfortunately, there is no fixed model of product family development and consequently also no fixed model of product family development costs. Thus, we provide a reference model for product family costs in this deliverable together with a collection of the most important cost drivers that we could identify through expert interviews. This will support the organizations in systematically deriving product family cost models that are particularly adapted to their situation.

In order to consistently and successfully evolve the product family infrastructure a strategy driven from a market point of view is required [KLD02]. This strategy is needed in order to decide on the appropriateness of an investment and to do so consistently over time. Thus product family strategy should be the basis for any product family investment. In this deliverable we show approaches to derive such a product family strategy and combine it with the most advanced techniques for evaluating the potential benefits of the product family even when facing considerable uncertainties. As a result, this deliverable provides a basis for analyzing costs and benefits of product family development and enables to connect these aspects through an appropriate strategy that is optimally aligned with the existing product family and the company objectives.

Copyright © Fraunhofer IESE 2003

Copyright © Fraunhofer IESE 2003

Table of Contents

1	**Introduction**	1
2	**An Economic Model for Product Lines**	3
2.1	Types of product line business	3
2.2	An Economic Framework for Product Lines	6
2.2.1	Product Line Economics Model	6
2.2.2	Applying The Model	17
2.2.3	An Illustrative Example	19
2.2.4	Relation to other Models	20
3	**Defining Investment Strategies**	24
3.1	Classifications of Strategies	24
3.2	Overview of Approaches to Strategy Definition	28
3.3	Qualitative Approaches to Strategy Definition	29
3.3.1	Portfolio Approaches	29
3.3.2	Quality Function Deployment	33
3.4	Quantitative Approaches to Strategy Definition	33
3.4.1	Defining the Key Dimensions of A Product Line Strategy	33
3.4.2	Optimizing the Return On Investment from a Product Line Portfolio	35
4	**Refinement and Control of Investment Strategies**	37
4.1	Measuring Strategic Success	37
4.2	Controlling Product Line Results	38
4.2.1	Decomposing Qualities	40
4.2.2	Product Line Development Situations	40
4.2.3	Cost Modelling	45
4.3	A Framework for Strategy Control and Implementation	45
4.3.1	Refinement of the Product Line Strategy	46
4.3.2	Operationalization of the Strategy	49
5	**Evaluating the Future**	52
5.1	Overview of Approaches	52
5.2	Decision Tree Analysis	53
5.2.1	The Concepts of Decision Tree Analysis	53
5.2.2	Applying Decision Tree Analysis to Product Line Strategies	54
5.3	Options	56
5.3.1	A Brief Overview of Options	56
5.3.2	Options in a Product Line Situation	57
5.3.3	The Option Formula	59

Copyright © Fraunhofer IESE 2003

5.3.4 Combining Options and Decision Trees 61
5.3.5 Product Line Options from a Market Perspective 63
5.4 Consequences of Option Creation for Strategy Implementation 63

6 Conclusions 66

A Bibliography 67

B Identification of Quality Drivers 73
B.1 General Cost Factors 73
B.2 Adaptive Reuse Cost Drivers 74
B.3 Cost Drivers for Reusing 78
B.4 Cost Drivers for Development for Reuse 78

Copyright © Fraunhofer IESE 2003

1 Introduction

Product family development has been established in recent years as a key approach to generate considerable benefits in terms of savings in development effort, shortened time-to-market, and improved quality. These benefits, however, come at a price: they are preceded by an initial investment in the development of the overall infrastructure [CN01]. Moreover, in order to continuously evolve the infrastructure and adapt it to the ever-changing requirements of the product family, considerable investments are required. Consequently, already in the context of earlier forms of reuse efforts, the understanding was developed that reuse requires continuous investments in order to keep a reuse effort going [Lim98, Rei97]. This holds true as well for product family reuse: developing assets for reuse leads to costs. However, as considerable benefits can arise from these costs we can rightfully regard them as investments in the future of the company, if they are appropriately targeted.

In this deliverable we will focus on costs- and investments from a product family perspective. The basis for any investments in a product family context should be a product family strategy, as otherwise no consistent decision making is possible. Forming such an optimal product line investment strategy, requires the following capabilities:

- Understanding the relationship of properties of the product family (and its development) like product quality, required effort, etc., and the resulting economic situation of the product line.
- Determining the costs that will occur in a product family situation, their origin and what will influence their size.
- Understanding product family development strategies and the characteristics of product family that are required for pursuing it.
- Identifying the benefits that can be expected from pursuing a specific product family strategy.

Additionally, we will provide an approach to deriving measures for characterizing whether an organization is still on track with its product family strategy.

This report is structured as follows. First we will discuss the basics of product family development economics, i.e., which are the benefits that can be accrued from a product family and what is the relationship to specific characteristics of the product family. We will discuss this in Section 2. In Section 3, we will discuss product family development strategies. This will enable us to provide a framework for understanding the typical motivations and constraints for investments

Copyright © Fraunhofer IESE 2003

in product families. In this section we will also discuss some approaches for tailoring a product family strategies to an organization (both from a qualitative and a quantitative point of view).

In Section 4, we will focus on the control of the investments that are required for the product family. A key part of this section is the discussing of a reference framework that allows to identify the main influencing factors on costs. Additionally, Appendix B provides a list of the main influences on product family qualities, in particular, cost.

Finally, in Section 5 we will discuss a mathematically rather complex problem: introducing future uncertainty in our estimates of product family development benefits. We need to determine and quantify our strategies in the context of a high uncertainty. This will influence the benefits that we can actually expect. In this section we will discuss some of the major approaches for constructively address this problem.

Copyright © Fraunhofer IESE 2003

2 An Economic Model for Product Lines

While product development and company strategy are traditionally seen as two sequential steps, this is no longer true in a product line context. Where in a single-system development environment the decision is made which products shall be built and then they are built in a systematic manner, in an adequately managed product line context the assets built for the product line strongly influence the further company strategy while not unduly limiting it. This entails that strategy definition, product line planning and scoping, and the actual development are interrelated to a much higher degree than this is the case in a single-system context [Cle01, KLD02].

Thus, in order to understand the constraints on product line investments we need to understand product line strategy. This will provide the basis for the investment concepts we describe in this report.

2.1 Types of product line business

An organization is not free in the strategies it may choose. Rather it needs to choose a strategy which is aligned to its specific situation. Thus, in order to characterize and understand strategies, we first need to characterize and understand the different types of product line business and starting situations.

A major distinction is whether a product line organization is predominantly ***market-driven*** or ***customer-driven***. In a market-driven situation the systems are developed by the organization without any contracts. Usually the intention is to sell the same product to many different customers without further customization. On the other hand customer-driven development means that the specific products are developed for individual customers, i.e., whether a specific product is developed at all as well as its specific requirements are largely customer specific. One should note that this does not imply that the product line development itself is started due to a customer, it might be a strategic move to start a product line development for a market, without having a customer, but still the product development is customer-driven. Further, both types of product lines may co-exist, as exemplified by MARKET MAKER Software AG, which hosts both types of product lines [CGKS01].

The specific market entails certain consequences for the product line, covering many different aspects ranging from how and when to enter a market to the number of products that can be expected:

Copyright © Fraunhofer IESE 2003

- In a market-driven situation usually fewer products will be built. This is because each single product will already attract a large number of customers, thus market-coverage will be reached with fewer products.
- In a market-driven situation a first core product should be fielded soon, as it is important to gain a significant market-share early on. Thus, this type of product line lends itself more easily to an evolutionary approach to product line development [MS02]. For a customer-driven product line the situation is different as additional customers need additional variants. Thus, once it becomes visible that the organization provides these products, additional customers will be attracted that should not be disappointed as the necessary number of variants cannot yet be built. This is particularly important for smaller organization that cannot compensate this with man-power. This is also a point where the risk of competitive entry by a larger company is particularly dangerous.
- The required flexibility and the foreseeability of the product line also differ in the two situations. In a customer-driven situation, the risk of "strange" requirements is particularly high and this also hints in favor of a more evolutionary approach and has implications on the openness of the required reference architecture.

One should note, however, that the distinction between the two types of strategies is not as absolute as it may seem. According to [KB01, Pin93] there is currently a strong tendency towards *mass-customization*. This implies that customers more and more expect that products that are offered to the general markets are adaptable to their needs. This will lead in the future to a much stronger urge for customizable products.

Besides the basic type of market (customer-driven vs. market-driven) the already existing situation in the organization has implications on the set-up of the product line. In [SV02] we distinguished the basic situations for product line set up:

independent — there is no pre-existing product development hindering or influencing product line development in any way

project-integrating — projects that conceptually are a part of the product line do already exist and need to be integrated into the envisioned product line

reengineering-driven — legacy systems in the domain of the product line do already exist, but the existing assets cannot be directly used as a basis for the product line

leveraged — a product line does already exist but only in a partially overlapping domain. Consequently the reuse infrastructure needs either to be significantly extended or a new infrastructure needs to be developed on top of the existing one.

Copyright © Fraunhofer IESE 2003

These different situations have a significant impact on the specific product line adoption strategy as discussed in [SV02]. However, as we focus in this report on continuous product line management, we will not discuss transitioning here. For this reason, we will also not discuss in detail any aspects specific to product development that happen during the transition to a product line approach.

In addition to the these basic categorizations, we can also introduce the characterization of the evolution of the product line over time. If we take this into account, we see that the amount of change that happens over time and the variety of systems that are at a given time brought to the market can vary independently of each other. Thus, Sanderson and Uzumery [SU97] introduced the following characterization (cf. Figure 1):

Commodity — A commodity exhibits only little change over time and exhibits also only very little variety.

Variety-Intensive — In a variety-intensive situation many different models are introduced and remain in place over time.

Change-Intensive — Many changes occur over time, however only few variants are in use in parallel.

Dynamic — A combination of both a variety-intensive and a change-intensive product line. Many variants are build which are released in many versions over time.

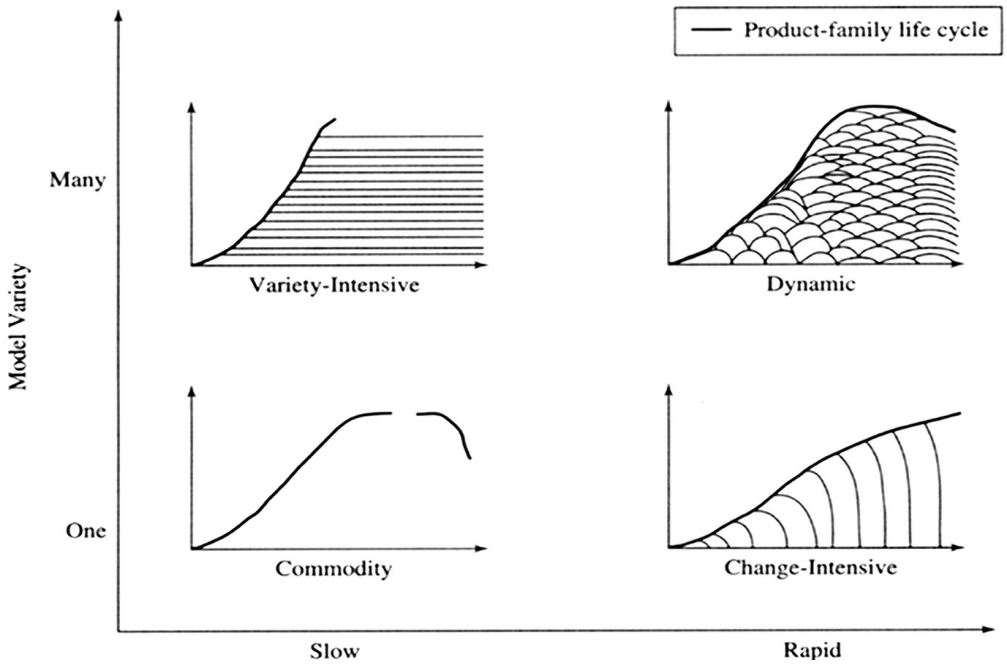


Figure 1: Different types of product family life cycles [SU97]

Copyright © Fraunhofer IESE 2003

Moreover, combinations of the above patterns can occur in a product line. For example, the pattern can change over time or some products may exhibit patterns than others. This is then called a **turbulent** pattern.

In addition we can analyze how the markets evolve over time as a whole. According to Porter [Por99] this can be differentiated into:

- growing markets
- stagnating (or shrinking) markets
- and fragmented markets.

2.2 An Economic Framework for Product Lines

In order to adequately understand and quantify the impact of a product line approach we need to analyze the various factors that are relevant to value generation in a product line setting. Thus we will describe a model that we will use as a reference framework of product line economics throughout the remainder of this report. We will also provide an illustrative example of the potential applications of this model. In this example we will already provide a glimpse on the potential for optimization of the economics of a product line.

2.2.1 Product Line Economics Model

We present our model in three steps, where each higher level model (approximation) can be seen as a refinement and extension of the next lower level model. This model provides the framework for all further considerations of product line investments. This ordering is based on the complexity of the fundamental concepts from financial theory that are introduced at each level (first basic cash-flow, then discounted cash-flow analysis (DCF), finally options). This structure also has the advantage that it can be easily used as a basis for comparison with existing models from reuse economics.

We will start with a *first order model* that enables us to make explicit the tradeoffs that are involved in analyzing the economics of a certain product line situation and thus in determining an optimal scope for a reuse infrastructure. In the *second order model* we will augment this model with some implications of the aspect of time and monetary theory in the context of reuse economics. In addition, in the *third order model* we discuss the implications of accounting for risks and opportunities in the context of product line economics. We will also discuss why only a third order model will truly provide a sound basis for product portfolio scoping [Sch00b], i.e., determining an optimal combination of products. In the remainder of this section we will describe the three levels of our model. In addition we present in Section 2.2.1.4 an overview of the model.

Copyright © Fraunhofer IESE 2003

2.2.1.1 First Order Model

The basic approach of this model relies on three different, but interrelated concepts:

- Development Constraints
- Software Attributes
- Market Attributes

We will discuss the different aspects in this order, as this mostly corresponds to the direction of influence, i.e., the development constraints influence the achievable software attributes, and the software attributes influence the market attributes.

Thus, as opposed to other models of product line economics [BS96, MCGZ00, Pou97b, Wil99a], we are taking a perspective of including the environment of the company, i.e., the market. The reason is that otherwise an objective valuation of the benefits of product line engineering will not be possible. The extension from a pure cost to an investment focus is intimately related to this extension to a market focus.

Development Constraints

In principle, constraints on software development may come in many different forms. For simplicity of this analysis, we will only differentiate between *resource constraints* (which resources are available at which point in time) and *process constraints* (how are the resources used by the software development process and how does this map onto characteristics of the products and process, like effort needed for the development or resulting software quality). In this section we will focus on resource constraints; process constraints will be discussed below in the context of *development scenarios*.

In the context of modern software development, resource constraints are mostly *personnel constraints*. While other constraints like the availability of computing resources formerly played a role this is nowadays usually not the key problem. Thus, we can also equate financial resources mostly with personnel resources. Personnel resources can in turn be characterized along two key dimensions: their knowledge and their number. In case the knowledge has a key impact on the process performance, we can distinguish different knowledge types (roles) accordingly and break up the number of people based on the roles.

For determining the software attributes of a project it is of core importance to understand the availability of the personnel resources as well as their timing. Thus, it is rather surprising that in current economic models the resource aspect is usually not taken into account (e.g., [Fav96, Lim96, Pou97b, Wil99a]). In general resource constraints will always depend on time (just like resource needs),

Copyright © Fraunhofer IESE 2003

but in particular they can be due to competing projects. This can result in a resource squeeze as depicted in Figure 2 (III). This is particularly important in a product line environment, as the competing project may be part of the same product line.

Software Attributes

Software attributes are those attributes of the software or the software development process that are relevant to determining the value of a product and thus of the product line as a whole. The attributes that are discussed most often in this context are: costs, schedule, and software quality.

In the context of product line development the impact of reuse (both development for reuse and with reuse) on these aspects is of core importance. In scoping we aim at optimizing reuse in a manner that allows to maximize the overall economic benefit for the organization [Sch00b]. Thus, we need to understand the impact the scoping decision for a single scope element (e.g., feature) has on the economics of the product line as a whole.Therefore we will use the perspective of an individual scoping element in the following discussion. Where necessary, we will make the impact on the product (line) explicit.

We will now discuss each of the following software attributes in turn:

- Cost / Effort,
- Schedule,
- Software Quality, *and*
- Development Risk (any negative deviation of the aforementioned attributes)

Effort-related costs are the aspect most models of software economics focus upon. Thus, we will also provide the most detailed discussion of this aspect here. If we look at traditional presentations of economics for software development [Boe81, DeM82], we often see explicit representations of effort (staffing) over time. Figure 2 (I) (development without reuse) shows a qualitative representation of this. In the context of our discussion we will use this distribution qualitatively. The details of a quantitative model will actually depend on many different factors: the implementation technologies used (e.g., encapsulation of the to be reused assets), the way the reusable asset is developed and integrated into the reuse infrastructure structure, the details of the process, etc.

It is important to emphasize that in our discussion we focus on the cost of developing individual elements of the scope in a reusable vs. a non-reusable manner. In practice project cost is usually not fully attributable to individual scope elements, i.e., features, as activities like project-management, coarse-grained requirements-/domain analysis, integration or system test are performed on a larger level. Thus, if a detailed modeling of the overall effort and the impact of

Copyright © Fraunhofer IESE 2003

reuse is required, a role-specific modeling of the efforts is needed. However, especially if the scope elements are rather coarse-grained, this is a rather accurate approximation [Boe81, BCH+95]. This description of project costs as a composition of feature/component costs is rather common in effort modeling. Thus, in the remaining discussion we will make the approximation that all effort can be attributed to individual components, while this may be somewhat inaccurate. Also for small features that can be developed by a single developer, the effort distribution will simply be a constant for the time required for implementing the feature.

Independent of how exactly an effort distribution looks like, two attributes are of prime importance: *the area under the curve*, which represents the total effort (costs) and the point where *the curve reaches zero*, which corresponds to the time when the feature is actually finished (schedule).

When focusing on the impact of reuse on the *schedule*, we see that even if we assume that development for reuse[1] leads to a cost overhead of 30%, this does not necessarily relate to a longer development time of 30%. Rather, the actual delay incurred by development for reuse depends on the available resources. This is shown in Figure 2 (I)–(III). While the effort is always 30% larger (larger area), the development time varies between an additional 10% and 50%. Thus, we see that schedule (and thus time-to-market) aspects cannot be treated as a function of effort as is usually done by economic models. Taking into account resource constraints actually introduces a separation of effort from schedule aspects. However, if our scope element has the size of a single-person work allocation, people are not shared among feature-developments, and a constant, low number of people is available, we get the advantage that cost and schedule overheads are directly proportional again. Actually, the approach of assigning features across multiple products to individual developers or a small group of developers seems to be a preferable approach in the context of product line development for other reasons as well [Sch01].

The dependency on constraints is the main reason why we do actually discuss costs as cost distributions over time as opposed to mere cost numbers as is the case in other economic models [BS96, MCGZ00, Pou97b, Wil99a].[2] While the distributions in Figure 2 show only effort, other issues are important for costs as well, like hard- and software, training, etc. This could be included in the distributions in Figure 2.

1 Note, that the function of effort with reuse is slightly differently shaped because we assumed that in the context of reuse the effort distribution is more front-loaded due to the need of more up-front analysis.

2 While often (e.g., for discounted cash-flow approaches) the functions are discretized in years, this is not used as a basis for a discussion like the one we provide here.

Copyright © Fraunhofer IESE 2003

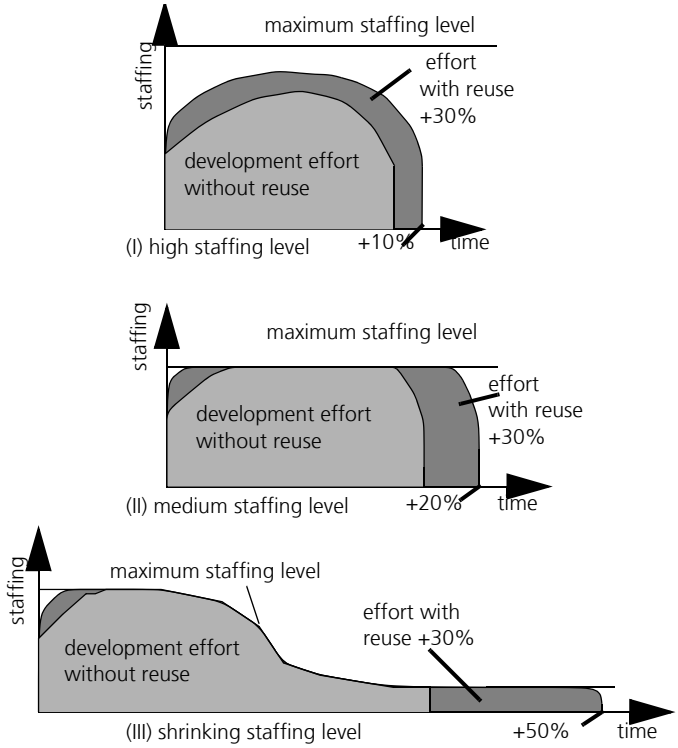
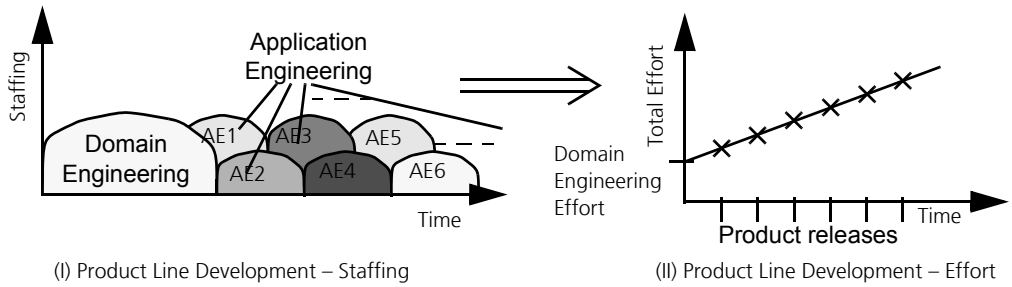


Figure 2: The Relation of Effort and Time-to-market

The discussion we provided so far focussed on the effects related to the development of a single product/feature. However, we are predominantly interested in the economics of a complete product line. For the purpose of this analysis, we need to determine the effort distribution for reusing a feature (respectively all the features in a product). Here, we make the approximation that the effort distribution for development with reuse is similar to the effort for development without reuse, albeit smaller. Based on this assumption Figure 3 (I) gives an effort distribution with unconstrained resources for developing a product line. Figure 3 (II) maps this to an aggregated representation of the effort for developing a product line [WL99].



(I) Product Line Development – Staffing (II) Product Line Development – Effort

Figure 3: Product Line Development — Effort and Staffing

Copyright © Fraunhofer IESE 2003

Actually various different approaches for integrating reuse in a product line are possible (e.g., incremental incorporation, up-front investment, and so on). Each of the different integration paths has a different impact on the economics of product line development. In Section 2.2.2 we will discuss the influence of the different development scenarios further.

As defined in ISO 9126 [ISO98] there are six main types of *software quality*. Each of these qualities is in turn refined into several sub-qualities. It is generally assumed that product line development (reuse) has an impact in particular on reliability and maintainability. For reliability it is an open research problem to determine the exact impact [Fre00] as most approaches to determining reliability are based on evaluating the end product and do not take a process perspective. Only few approaches, e.g., [Chu99, RCL99, ZP00] aim at determining this impact up-front. However, none of these has been applied to the situation of product lines so far. The situation is similar with regard to maintainability. While first quantitative analysis of the effect of product line development on maintenance effort exist [SM99], we could not yet find any predictive models that would be applicable to predict the impact of product line development on this attribute.

Another important software attribute which is influenced by product line development is the reduction of *development risk* (projects become better controllable). While this has been the driver for some organizations to adopt product line development [BC96] altogether, it has to our knowledge not been studied independently in the context of reuse economics. While several forms of risk can be differentiated, the predominant form in this context is the risk of effort/schedule overruns. In case the effort models also describe the uncertainty of the estimate, monte-carlo simulations can be used for determining the probability of cost/effort deviations [BEB98] and the connection to market attributes that is established by our model allows to value them in an objective manner.

In Figure 5, the box *software attributes* gives an overview of the interactions among the different software attributes.

Market Attributes

As opposed to traditional reuse economics models [Lim96, Wil99a, MCGZ00] we include the market perspective in our model. Actually, we will differentiate two main forms of markets: *customer-driven*, i.e., systems are produced for specific customers and *market-driven*, i.e., systems are produced for the mass market.

The reason for introducing the market perspective is that cost is only one out of many aspects that influence the economic value of a product line. The various software attributes we mentioned may impact the economic value and thus the benefit of an investment both positively and negatively. In order to integrate

Copyright © Fraunhofer IESE 2003

these different influences we go back to the market to find an answer to the valuation problem. The approach we use is to value the attributes through their impact on revenues. This allows to bring the whole scientific knowledge of marketing science and economics to bear and is a major refinement of other approaches like attribute weighting in QFD [Coh95] or utility functions [Wym93] as these approaches rely on purely subjective evaluations.

However, the software attributes do not directly influence the market. They are mitigated and augmented by *market attributes*. A market attribute is any aspect of the product line in the market that has an impact on the value of the product line. We will now discuss some market attributes and their impact on revenues (in braces we give the software attribute with the strongest impact):

- product entry timing (schedule)
- perceived product quality (software quality)
- product functionality
- pricing (limited cost)
- product competition (internal and external)

These attributes are always related to software attributes, but there is always a gap that is determined by the company strategy.[1] For example, the start of the development plus the schedule gives a lower bound on the possible market entry, but it may be further delayed due to a strategic decision [WN89]. Similarly, the price is influenced by the cost, but not fully determined [DK88], this is especially true in the context of market-driven software development as here we have the possibility of recovering the prive over a large number of copies. For the sake of simplicity we will neglect this distinction for a moment and will discuss the influence of attributes on the revenues directly. Note, however, that we will not discuss cost, as the pricing is a highly strategical decision in *market-driven* development, were either a strategy of high prices combined with low volumes or of low prices combined with high volumes can be used (or different strategies for different members of the product line). In the case of *customer-driven* development the price usually corresponds more closely to the costs.

In practice it has been recognized that *time-to-market* issues can actually dominate cost issues, i.e., a slightly shorter time-to-market may have a much higher impact on the overall cash-flow of the company than a cost reduction. Some typical revenue functions are shown in Figure 4 (cf. [SU97]). Depending on whether the market is mature or not the graphs will look very different. Consequently, the benefits of a product line will vary as well. A special case is shown in Figure 4 (III), which depicts the case for a product that relies on another event (e.g., fielding a new version for a new operating systems version).

1 For a discussion of strategic aspects see Section 3.

Copyright © Fraunhofer IESE 2003

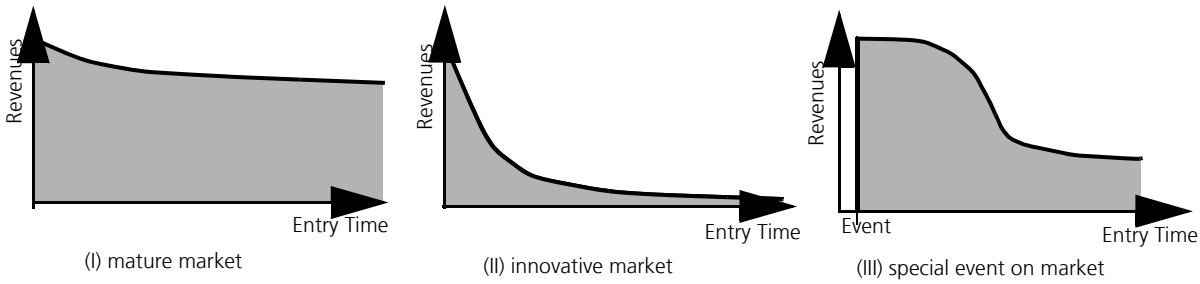


Figure 4: The Relation of Revenues and Market-Entry

Strongly related to the time-to-market issue is the aspect of *development risk*. If we equate this risk with the possibility of schedule overruns, this will result in a strong reduction of revenues in an innovative market (less so in a mature market). The value of development risk reduction can thus be equated with the expected loss in revenues[1]. Similarly, for customer-driven development we can equate the value of the risk with any corresponding liabilities.

The relation between the (perceived) product quality and the revenues will depend on the actual market situation for the products. For example, the quality impact will be different for an Office-product than for an embedded, real-time system. In the context of products like Office, the value of quality is directly described by the difference in revenues, while in high-reliability markets a direct valuation through the value of any liabilities might also be possible. If the value is expressed by the difference in revenues, it is important to take into account that we explicitly say here that this depends on the **perceived** product quality, which may differ substantially from the real quality.

Also, the overall revenues (typically through a pricing relationship) will be influenced by the product functionality that is available. The impact of product functionality on desirability of products has been extensively studied in marketing science [GK85, MZ88].

Obviously, the absolute revenues that can be achieved will also depend on the demand and the competition. While demand and external competition are outside of our control, the internal competition, i.e., the competition from other products in the same product line is within our control. This will be discussed further in the Section "Third Order Model".

While all the market attributes discussed above like functionality, quality, competition, timing, but also the software attributes like costs will influence the actual pricing, this aspect is especially important for market-oriented software

1 Development risk is actually a probability distribution, thus it is meaningful to define the *expected value* of revenue loss.

Copyright © Fraunhofer IESE 2003

development driven by strategy (do we expect to sell high volumes with low prices or do we expect to sell low volumes with high prices) [Ste91]. Thus, there is no direct connection among these attributes. However, they provide constraints; e.g., costs determine the amount of money we need to recover through sales, the other factors determine the price a customer might be willing to pay [DK88, KVC96]

The key point is that the specific mapping of software attributes to market attributes is of core importance to the economic evaluation of the product line and must be performed in a context-specific manner. In particular, we must not equate software and market attributes. This is particularly well illustrated by the aspect of software quality: the perceived and the objective software quality may strongly differ (e.g., as a result of good advertisement and branding). However, ultimately the real quality imposes constraints on the perceived quality. Often the mapping will vary strongly depending on the specific product, i.e., it will be different for different products in the product line (e.g., pricing). Marketing science developed approaches to determine objective valuations of product attributes [Dea].

We can summarize our product line economics model as follows (this holds in principle for the first order as well as for high order models): in order to determine the worth of a product line, we need to understand the resulting revenues and costs associated with it, as well as the coupling among the two. Costs are determined mostly by the software attribute effort, the other software attribute are mostly relevant as factors that influence certain market attributes (e.g., software quality impacts perceived software quality). The market attributes in turn are mostly relevant due to the fact that they influence the ultimate revenues that can be gained (e.g., product entry timing impacts the overall revenues). Thus for valuing a certain product line development scenario we need to construct the dependency graph of these valuations for the specific situation in a bottom-up manner.

The interrelation of the different software and market attributes in the first order model is described in Figure 5, Box first order model.

2.2.1.2 Second Order Model

While in the 1st order model we needed the aspect of time only to represent the impact of the attributes of schedule (and development risk), respectively market entry timing and possible late entry, it is on this level required for all attributes, as we introduce fiscal ideas in the model.

So far we did take the numbers (e.g., revenues) at face value. In the second order model we introduce an aspect called time-value of money into the analysis: money earned (or spent) now is worth more than money earned (or spent)

Copyright © Fraunhofer IESE 2003

later. This leads us to discounted cash-flow analysis [Fav96, Lim96, Pou97b,MCGZ00, Wil99a], i.e., we need to discount all cash flows by a certain percentage depending on the point in time when they are realized. The specific percentage depends on the organization, especially on the interest rate the organization needs to pay on the capital market.

In our approach we can accommodate this rather easily by discounting all functions related to cash flow accordingly. These are only the cost and the revenue functions. All other attributes are valued relative to these (e.g., quality impact on revenues) and are thus valued appropriately. In this way the result is automatically adapted and leads to a discounted cash-flow analysis.

2.2.1.3 Third Order Model

While on the previous two levels we presented the economics as a straightforward evaluation of the possible situations (making a feature reusable vs. making it system-specific), we can also include the evaluation of alternatives that are provided to us through reusability itself and the effects this has on the value of the product line approach. This is extremely important in the context of investment decisions like product portfolio scoping [Sch00b], as there we aim at identifying the optimal product/feature-mix so that the benefit to the organization is maximized. This has been addressed from the marketing perspective as the product line selection problem [GK85]. However, we go a step further then these approaches and include the interaction with the software development situation.

Related to this is also the evaluation of possibilities like changing certain development plans due to evolving risks in the process, which is exemplified by the approach used by Withey [Wit96] or accounting for options [AK99, FFF98]. A particularly interesting example of such an option is the case that additional products may be introduced into the product line that would not be economically producable without the reuse infrastructure. However, on the other hand fielding more products may lead to a cannibalization effect in the product line thus changing the revenue curve [MM94]. We will illustrate this aspect in our example in Section 2.2.3.

At this level feedback cycles can also occur that lead to modifications in the software development plan. For example, adding another product or discontinuing a certain development path changes the resource availability and thus the constraints. However, in the context of this analysis the feedback cycles related to the creation of options are of key importance (cf. Section 5).

Copyright © Fraunhofer IESE 2003

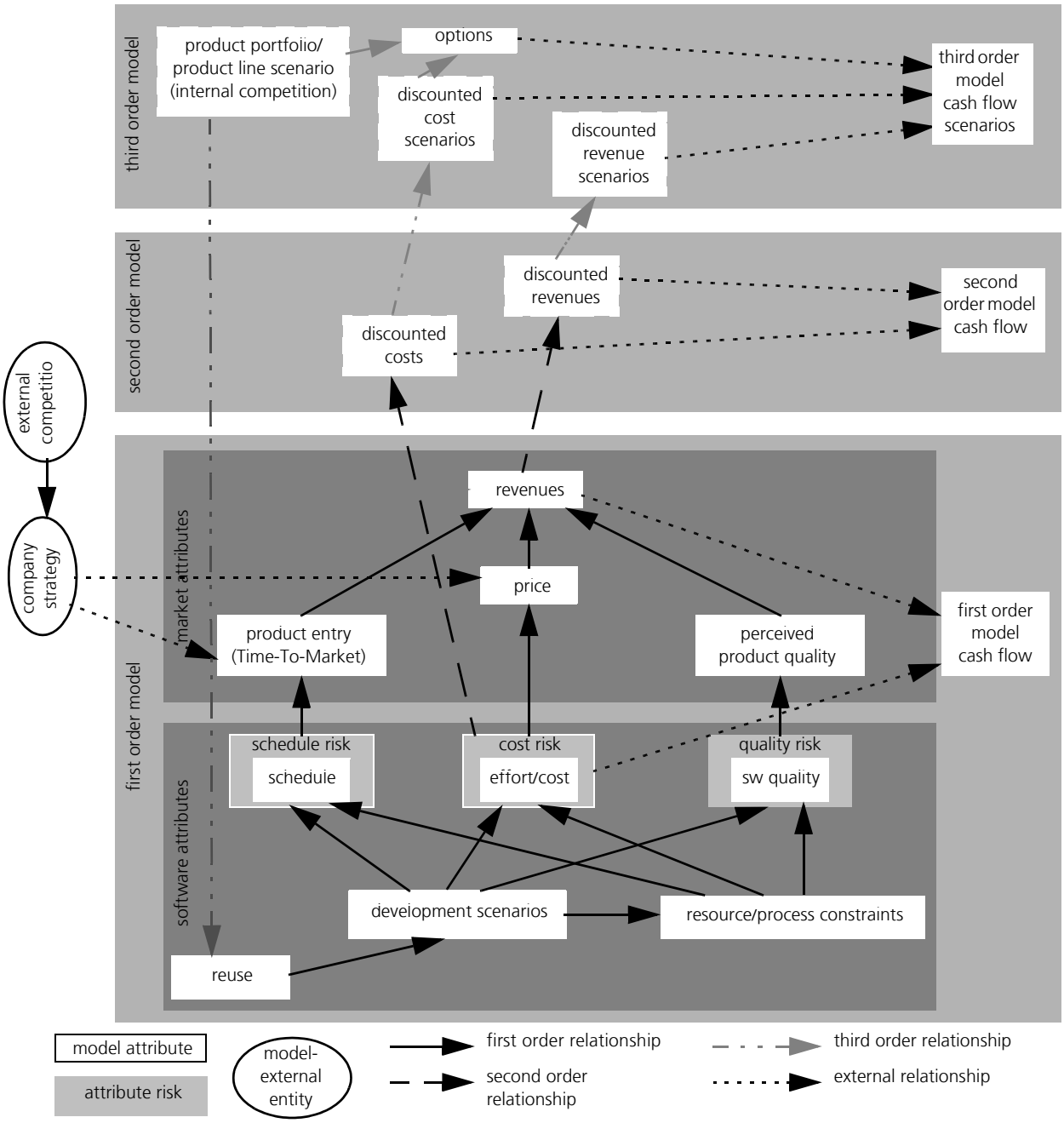


Figure 5: Interrelationships of different aspects in the product line economics model

2.2.1.4 Overview of the Model

In the preceding sections we gave a step-wise introduction of the model of product line economics we proposed. An overview of the model is given in

Copyright © Fraunhofer IESE 2003

Figure 5. In this figure the second and third order model seem to be much simpler, however, one should not forget that they rely on the aggregation of the various attributes in terms of the main attributes costs and revenues. Thus, these higher order models implicitly refer to all the market and software attributes.

In this diagram certain attributes (e.g., product portfolio) are understood as constants in the lower order models. For this reason they are placed at a higher level and the relationships are marked as higher order. Further, we did put the company strategy and the external competition outside of the model. These are seen as independent input to the model and beyond its explanatory power.

In the third order model we do also make explicit the notion of an option. While the scenario-based approach directly supports the evaluation of alternatives, options make explicit the value of flexibility wrt. the uncertainty of the products that will be developed. (Note, that this is different from the internal uncertainty as given by the notion of development risk.)

The way, the model is given also shows the bottom-up evaluation approach for which it provides the basis (cf. Section 2.2.2). Basically, any evaluation starts with the product portfolio and the reuse scope derived from this (plus external entities). This is then propagated to the market level and then upwards until the third order cash flow scenarios can be derived.

2.2.2 Applying The Model

We will now discuss in more detail how our model of product line economics can be applied in practice and will illustrate this and some of the ramifications of this model using a concrete example.

2.2.2.1 Scenario-Based Evaluation

So far, our discussion focused on an overview of the factors influencing the value of product line development, respectively a certain asset scope. However, what does it mean to determine an optimal scope and how can this be done according to this model? For the sake of this discussion we introduce the notion of *development* and *product line scenarios*:

- A *product line scenario* is characterized by a product development plan that describes which products with which features should be fielded at which point in time. In particular, a product line scenario determines a product portfolio.
- A *development scenario* is given by a partial ordering (which may include parallelism) of development activities, that will lead to the development of a product line. These activities can in principle be any kind of software develop-

Copyright © Fraunhofer IESE 2003

ment activities, in particular, adaptations of software (e.g., re-engineering), inspections, etc.)

Thus for each product line we want to develop we may want to analyze different product line scenarios, possibly connected through options (cf. Section 2.2.1.3). For each product line scenario several development scenarios may exist. In this interpretation optimal scoping means selecting product line and development scenarios simultaneously in such a way as to maximize the resulting cash-flow scenario.

The activities in the development scenario can be specified both with respect to their phase (e.g., analysis, implementation, etc.) as well as the sub-functionality (e.g., feature) they are referring to. Thus, we can define scenarios like:

1 First develop features f_1, .., f_n as reusable assets, then develop f'_{n+1}, .., f'_{m1} especially for product p_1 and f''_{n+1}, .. ,f''_{m2} especially for product p_2.

2 Develop the products independently, i.e., develop specific implementations of f'_1, .., f'_{m1} and f''_1, .., f''_{m2}.

If the different functions for software and market attributes are known, as well as the constraints, it is in principle possible to derive an objective measure of the economic value of a scoping decision to develop reusable assets encapsulating features f_1, .., f_n. However, the approach goes even beyond merely determining the asset scope. It can also be used in order to address the question of how to introduce a certain asset into a reuse infrastructure. If we look at a single feature that is supposed to become reusable, we have at least the following possibilities:

1 We can develop it as a reusable feature and then reuse it in both products.

2 We can develop it specifically in the context of the first product, then extract it, make it more generic, and use this as the basis for later products as well as for future versions of the first product.

Obviously, the second version will usually be more expensive. However, as it will often be more effective in terms of time-to-market for the initial product, it is often chosen in practice as a form of incremental product line introduction [Coh01]. Thus, at this more fine-grained level of analysis, we have to take into account the process constraints as well. Using a simulation approach based on the described concepts could provide a basis for analyzing these trade-offs in detail. Effectively performing such a set of simulations would provide us with a portfolio of development scenarios as well as the economic benefit that can be expected from them. Simulation approaches have already been used for analyzing a wealth of questions about software development [KMR99]. However, they have to our knowledge not yet been used as a basis for identifying the optimal product line scope. Further, such approaches have only rarely been linked to explicit models of the market. Rather they were restricted to models of the software attributes. In particular, in such a case we would need to perform the sim-

Copyright © Fraunhofer IESE 2003

ulations on a rather fine-grained level and would need to analyze a rather large number of scenarios. However, this approach would provide the most objective basis for asset scoping, and would allow in an integrated manner the derivation of an accompanying product line development plan and business case.

2.2.3 An Illustrative Example

Let us now illustrate the concepts we discussed so far in terms of a small example. In particular, this will describe how third order feedback loops add a level of understanding of product line economics that cannot be achieved without taking them into account. Moreover, the example illustrates how the feedback loops can alter the overall result considerably. Consider the revenues and investments as given in Table 1. This table illustrates the application of our product line economics model to the definition of a business case (this is sufficient to discuss the principles, the application to scoping would require additionally the comparison of different reuse scopes). Note, that we introduced different revenue columns depending on whether we will build three or four products. This distinction is already a tribute to a third order model. While the overall revenues can be expected to be higher, the per product revenues are expected to be lower due to some cannibalization of revenues among the products that can be expected to occur. The table also gives per product the cost of development with and without reuse and the investment necessary for reuse. As can be seen the reuse savings are independent of the number of products. We will now discuss the question of whether or not to develop the product line (and which products should be part of the product line) using our model:

Our first product line scenario is to develop only the first three products. The first corresponding development scenario is to develop the three products without reuse. In this case we can expect a return of $1.1 Million (revenues – development cost without reuse for first three products). The second development scenario is to develop the first three products with reuse. This leads to a return of $0.6 millions.

Turning to the product line scenario of developing all four products, we immediately see that developing all four products without reuse does not make sense economically, as for the fourth product alone development costs without reuse are higher than expected revenues. Thus reuse is not the adequate strategy here.[1] So we turn to the development scenario of developing the four products with reuse. Depending on the kind of model we are using, two different evaluations will be made. Using the first order model, we would still use the revenues in the "three product column", as this does not

1 However, what we skip here is the influence of reuse itself on the software attributes, that may lead to differences in the market attributes and thus may influence revenues. For example, earlier product delivery or higher quality may lead to higher revenues in the context of reuse.

Copyright © Fraunhofer IESE 2003

account for feedback cycles of the developed products and thus for product line cannibalization [MM94]. In this case we arrive at overall benefits of $1.3 Million, making this the preferred scenario. However, if we use the corrected revenues in the four product column, we arrive only at a benefit of about $0.7 Million. Thus, developing only three products without reuse is actually the most beneficial scenario. This kind of insight can only be achieved using a third order model which integrates this kind of marketing knowledge. (However, in our example, we did not account for options, which may also be part of a third order model.)

This shows that including third order aspects, like feedbacks from the possibility of developing different products, may lead to profoundly different decisions. Similar scenarios can be constructed e.g., for moving from a first order model that focuses merely on software attributes (as typical reuse economics models do; cf. Section 1) to a first order model that includes market attributes as well. In the latter case, in particular the introduction of time-to-market can significantly alter the decision making situation.

Product	Revenues [Million $] (3 products)	Revenues [Million $] (4 products)	Reuse investment [Million $]	Development cost with reuse [Million $]	Development cost w/o reuse [Million $]	Reuse savings in product dev. [Million $]
Reuse Inv.	—	—	3.5	—	—	—
Product 1	2.0	1.7	—	1.0	1.8	0.8
Product 2	2.4	2.2	—	1.2	2.0	0.8
Product 3	3.5	3.4	—	1.6	3.0	1.4
Product 4	(1.5)	1.5	—	0.8	1.8	1.0

Table 1: Applying the Third Order Model to the Optimization of Product Portfolio

2.2.4 Relation to other Models

Economic analysis of reuse is not a new topic. Rather, already a plethora of reuse economic models exist and also a large number of surveys and categorizations of these models were developed [Fav96, Lim96, MCGZ00, Pou97b, Wil99a]. Our model is not meant like these surveys as a basis for comparing different models of reuse economics. Rather, we will stay at a more abstract level and simply compare different areas of research that are relevant to product line economics with our model, reuse economics just being one of them.

The different fields we will be looking at are:

- reuse economics
- marketing science
- product line planning approaches from classical engineering

Copyright © Fraunhofer IESE 2003

2.2.4.1 Reuse Economics

Most approaches to characterizing the economics of reuse come from a component point of view and build on that to finally reach a corporate/product line viewpoint [MCGZ00, BS96]. This is opposed to our approach which has been developed purely from a product line level perspective.

If we look at the aspects that existing reuse economics models cover [MCGZ00, BS96, Fav96, Wil99a, Lim96, BCH+95, Pou97b, Pou97a, EMT98], we see from Figure 6 that actually a rather large area seems to be covered. However, a closer look reveals that most models focus only on the cost aspect and are either on level 1 or 2. Examples for a level 1 model would be the payback-model as presented in [Fav96]. Level 2 models are actually rather common: examples here are net-present-value (NPV), incremental-rate-of-return (IRR), profitability index (PI), etc. Using Figure 6 as a reference, we can say for example that NPV as described in [Lim96] is located in the cost column on level 2. Similarly, using the more fine-grained Figure 5 as a reference, we can say it covers: reuse scope, development scenario, cost, discounted costs, and the second order cash flow out of the complete model.

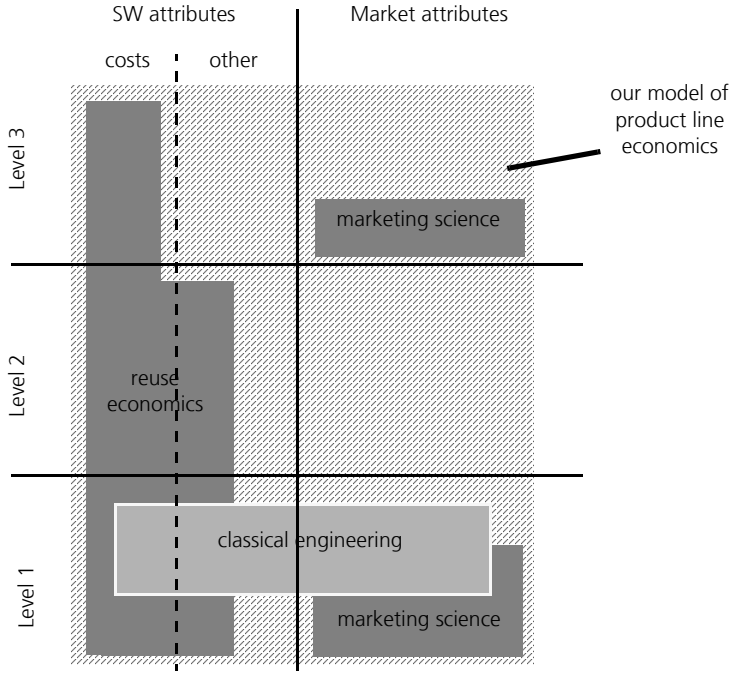


Figure 6: Coverage of our Model by other Approaches

Only few models actually try to include other aspects as well. Poulin includes in his model of product line economics [Pou97a] the notion of the value of quality,

Copyright © Fraunhofer IESE 2003

however, this is then only operationalized in terms of maintenance effort. Another example is the model described by Mili et al. [MCGZ00]. They mention their goal to include the time-to-market (TTM) aspect, but admit that they did not yet achieve this goal. However, from our point of view an adequate valuation of TTM is only possible, if the market is included. Actually, it was this kind of problems that initially lead to the expanded model of product line economics we present here.

Finally, in terms of level 3 models, there exist only very little work so far. The only examples so far are the work by Withey [Wit96] and Favaro et al. [FFF98]. These models introduce decision tree analysis, an approach for the comparison of different staged investment scenarios, and contingent claims analysis, which additionally includes the aspect of option valuation.

However, none of the models addresses the explicit feedback loop that was motivated through the market impact the introduction of a new product might have on the revenues. This is due to the fact that only software development internal aspects are taken into account by them when discussing reuse economics. This is the case despite the fact that some approaches only discuss revenues and costs in general terms without explicitly referring to the market.

2.2.4.2 Marketing Science

In this area the analysis of optimal product lines has been a quite active field for some time. This did culminate in explicit optimization approaches [GK85, MZ88]. By nature, these approaches focus on market attributes. While they include a certain amount of accounting for profitability of products they do not explicitly address the correlation of software attributes (except for cost) and are in particular not able to handle any reuse aspects. On the other hand these approaches usually address explicitly the interaction among products and the change of the product line portfolio. Thus, they must be classified as a combination of level 1 and level 3 models.

2.2.4.3 Classical Engineering Approaches

Another area that can contribute approaches to understanding product line economics are the planning approaches from classical engineering. However, even in this area there were few approaches that we could find. First of all, there is quality function deployment (QFD). While mostly applied for single system development, it can also be applied for product line planning [Coh95]. QFD is inherently a purely qualitative approach, which aims at bringing together the different stakeholder views (from customers and developers) and bases product planning and development on their consensus.

Copyright © Fraunhofer IESE 2003

Quantitative approaches are described by Robertson and Ulrich [RU98] and Esqueda [Esq98]. In particular the approach by Robertson and Ulrich is interesting as it combines the viewpoint of revenues with the cost side. It also addresses to some extent the desirability of features from a customer point of view. However, the approach is highly informal, relies on stakeholder experience, and does not take into account level 2 and 3 aspects.

2.2.4.4 Summary of Relation to other Models

To summarize one can say that different parts of our model were already addressed by approaches in different fields of science, however, these areas are mostly non-overlapping and actually sometimes do not have a common denominator. Further even if seen in their completeness, they leave a large area uncovered so far.

It is one of the contributions of our model to interrelate these existing items of knowledge and to clarify how it is possible to benefit from different existing fields of research. Additionally, the model highlights many interactions that have so far not been intensively covered and that need further work in order to fully understand them. Especially, if we want to introduce appropriate aspects like time-to-market into a quantitative evaluation of a product line business case.

However, we do not want to imply that all evaluations of scope or a business case need to be made with approaches that fully cover our model — this will often not be possible. Actually, an analysis that is based on a NPV-analysis of software production costs may in many cases be just right. But at this point our model can be consulted to get a better understanding of the aspects that are included in the product line valuation and what aspects are excluded and what kind of deviations from a full analysis this may result in.

Copyright © Fraunhofer IESE 2003

3 Defining Investment Strategies

In the preceding sections we discussed the basics of software product line development. This characterization basically holds independently of whether the specific company we are looking at is a pure software company or a systems development company in which the software is only part of the overall development. In the later case the external constraints are simply mitigated through system development and only in cases where the software part is not a critical element in the system development will the situation be slightly different. But as in these cases the software product line strategy will not have key importance, we can safely exclude this situation from further consideration. This is especially the case as software becomes more and more *the* key part of embedded systems development in practice [ESR2002].

The existence of an adequate strategy is extremely important even in situations where products are developed one at a time and form a product line only from a marketing point of view [Por99, KB01]. It is even more important in case a software product line is set up or already under development, as software product line development is particularly powerful in case this is decided right from the start. Thus, integrating software product line development into the overall corporate strategy is a key issue for any organization running a software product line initiative.

3.1 Classifications of Strategies

Many different meanings are often associated with the term „Strategy". So before we discuss in more detail approaches to derive such a strategy, we first need to define what exactly we mean with a strategy in the context of this report. As we are analyzing strategies from the point of view of software product line development, we are focusing on these aspects and types of strategies that are relevant to actual software development.

Growth and Innovation Strategies

A key distinction that can be made is between a ***growth strategy*** and an ***innovation strategy***:

- Growth strategies are obviously the most common. Their bottom line is that they focus on „getting more of the same". Most traditional companies operate on this assumption. Actually, any form of continuous development always goes back to this approach.

Copyright © Fraunhofer IESE 2003

- Innovation strategies are much rarer. In its extremest form some venture capital companies operate on this basis, however, they rely on external innovation. The key to innovation strategies is that they focus on introducing new concepts (unknown at the point of strategy formulation) into the product line over time. If such a strategy is pursued by an organization, it is usually within the bounds of a growth strategy (e.g., car companies seek for innovations *for their new cars* to provide them with unique selling propositions).

Both types of strategies are in principal relevant to software product line development. The classical concept of a growth strategy maps particularly easy to product line development. Thus it is typically associated with a product line. However, the concept of leveraged product line adoption [Cle01, SV02] can often match to an innovation strategy as first a new type of product that exceeds the current limits of the product line must be developed. However, guidance on innovation strategies is scarce, it usually boils down to so called „research management" operating under the assumption that having a research unit will lead to innovation. This current lack of innovation support leads us to focus on growth strategies.

Business Models and Their Influence on the Strategy

The way a company is doing its business also has an impact on its product line strategy. A key distinction here is the question whether the products are aimed at individual customers *(customer-driven)* or whether they are focused at broad markets *(market-driven)*. This does strongly influence certain characteristics of the product line itself and thus also the strategy used for its development. A market-driven product line usually has a much lower number of variants that are simultaneously active. Here ten variants is often already a high number. On the other hand this is actually a small number for a customer-oriented product line as it would imply that the organization only has a total of ten different customers. On the other hand we found cases with customer-oriented product lines where even more than thousand variants had to be produced.

This distinction also impacts the type of variation that occurs most frequently. In a market-driven product line most differentiation originates from the sales strategy, thus we often find the typical low-range, mid-range, and high-range products. This is a typical situation dominated by optional variation. On the other hand, the situation in a customer-specific product line is different insofar as here we often find that features are treated as alternative.

Other influences on the strategy originate from different sources [SV02, BHJ+]:

- Overall size and complexity of the products
- Are these software-only products? If not, are they embedded? Are they bundled with consulting?

Copyright © Fraunhofer IESE 2003

- Entry points: whether the product line was built from scratch; whether it is based on previously unrelated products; or whether it is built on reengineered software? However, this is mainly important for the transition strategy of a product line, not so much for the strategy of evolving the product line.

However, these points have less of an impact on the strategy, thus we will not discuss them in detail here.

Types of Strategies

According to Porter [Por99] it is possible to differentiate three main types of strategy:

- *Cost leadership* — the company aims at providing the product at the lowest possible cost
- *Differentiation* — through a certain feature (e.g., service, brand name, etc.) the company sets itself apart from the competitors
- *Focussing* — the company focuses on a specific niche

Surprisingly enough, while product lines focus on the support of variation, according to this categorization they mainly contribute to cost leadership. However, product line development can also be employed to contribute to the other two strategies. For example for differentiation it can be used to set the company apart from other competitors by offering a wider portfolio of products. It can also be used to provide a coherent look-and-feel throughout a whole range of products by regarding these products as a set of products. For the focussing strategy product line even offers the possibility that the company can offer focussed products even for a number of niches. However, care needs to be taken not to misinterpret the strategy in this context, as still the organization with all its functions needs to be able to adequately address these niches.

A slightly more detailed differentiation of strategies is given in [KB01]. The authors describe the following types of strategies, the first three can be seen as refinements of the differentiation strategy:

- *Improving* — The company improves in certain aspects (from the point of view of the customer) over the offerings of its competitors. This may include cost improvement, quality improvement, accompanying services, breadth of possible products and so forth.
- *Newer* — The ideas made available by the producer are new to the market. This implies that the company is strong on innovations, i.e., that it has good innovation strategies.
- *Faster* — New possibilities are made fast available to the customers. As opposed to the previous strategy, this does not necessarily mean the company is the inventor of these concepts.

Copyright © Fraunhofer IESE 2003

- *Cheaper* — This strategy corresponds to cost leadership as defined by Porter [Por99].

According to this description of strategies, we see again a strong contribution of product line development to the strategy *"cheaper"*. But product lines contribute to the faster and the improving strategy (in terms of large set of product offerings and improved quality).

In principle, we can always attribute a number of product line development goals to a certain strategy. However, a company will always pursue a mixture of the above-mentioned strategies, but one of these strategies will always have the lead. Consequently, also the different development goals will have different weights. We give here a possible mapping of strategy types and product line development goals:

- Cost-Leadership/Cheaper: of course cost and in particular cost over the future products is a key goal here.
- Differentiation: depending on the differentiation criterion different goals can be particularly important:
  - *Sub-goal Faster*:
    - Time-To-Market: being the first on the market with new features/products can be an approach to be regarded as technology-leader
  - Sub-goal Improving:
    - Quality: product line development can be used to improve the overall product quality through reuse. Thus used appropriately, it can be employed to differentiate through high quality
    - Look and Feel: product line development can be used to differentiate either through having highly customizable and differently looking products. This is especially the case in customer-driven product lines. Product line technology can also be used as a basis for generating a cohesive look-and-feel. This is relevant in the context of product suites like Microsoft Office or HP Openview [BHJ+].
    - Mass-Customization: product lines enable an organization to more specifically target the individual customer [Por99, KB01].
- Focussing: here the ability of product lines to support the development of multiple alternative products, i.e., the flexibility is key. Especially, the ability to maintain a large number of variants with only little extra effort.

This provides a short overview of the different types of strategies and the contributions product line development can make. An investment in product lines should be evaluated agains the subset of those goals that are part of the company strategy.

Copyright © Fraunhofer IESE 2003

3.2 Overview of Approaches to Strategy Definition

Many different descriptions of approaches to defining product strategies exist. However, some common principles provide a basis for them. A key starting point for all of them is the identification of a basic point of view from which to address the market.

Overall a product line strategy can be characterized based on the following three aspects (based on [Bey01, SU97]):

- *Market segmentation* — which market segments are identified and which of them are actually targeted by the organization.
- *Product evolution* — which product variants are produced, when are they introduced into the market, when are they retracted from the market, etc.
- *Pricing policy* — which prices are associated with the products.

The first of these aspects is *market segmentation* [Por99, KB01]. Many different criteria for such a segmentation exist. Examples are technical criteria, personal traits of customers, size of the company, ethnical characteristics, and so forth. An optimal segmentation can be identified by analyzing the potential customer base and applying statistical techniques [KB01]. Selecting a specific segment and addressing it is actually more difficult. This choice is at the core of any strategy. It is difficult to provide a specific process for this step, as this step needs to be dealt with in integral manner together with the other two aspects. However, various approaches (both qualitative and quantitative) assist in this step (cf. Section 3.3 and 3.4).

The *product evolution* is strongly related to the first issue. This brings about a refinement from the general market segments to the specific products that are used to address these market segments. Moreover, specific entry and exit times for these market segments must be determined.

For the various products specific prices must be determined. This leads us to the *pricing policy*. For each of the products prices compatible with the general product strategy needs to be defined. Pricing and the identification of costs and potential revenues are the basis for any quantitative strategy definition and evaluation (cf. Section 3.4).

Thus, quantitative strategy definition can only be performed as a refinement of qualitative strategy definition. This gives an interleaving of the three different activities and the specific ways of performing either in a quantitative or a qualitative way. This is depicted in Figure 7.

As we can see in Figure 7 there is a certain logical flow among the main activities and the relation between quantitative and qualitative strategy definition.

Copyright © Fraunhofer IESE 2003

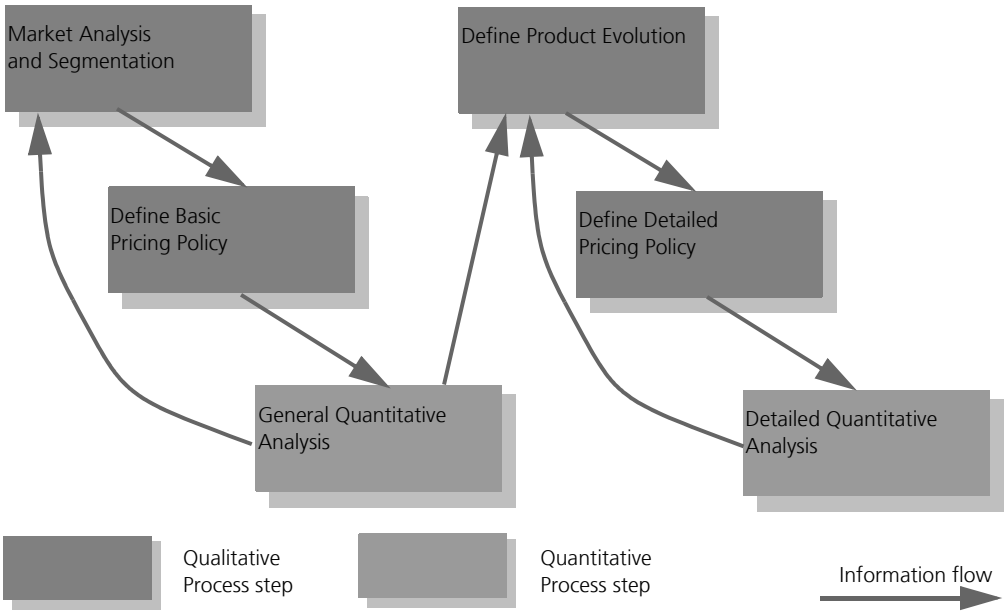


Figure 7: Overview of the general product strategy process

3.3 Qualitative Approaches to Strategy Definition

There is no clear boundary between qualitative and quantitative approaches to strategy definition. For the sake of this report we draw the borderline between approaches focussing on a more coarse grained characterization like portfolio analysis and more fine-grained approaches like metrics-based approaches. Most qualitative approaches to strategy definition resort to the definition of a portfolio [Gab97, Spr00]. A second approach, which we will discuss in this section is Quality Function Deployment [Coh95].

3.3.1 Portfolio Approaches

The basic idea of portfolio definition (and management) is to capture a picture of the various market opportunities as a whole. This picture can then be captured in a two-dimensional matrix, characterizing the state of the various products. At the same time, for each of the various fields certain recommended actions are associated.

An appropriate application of a portfolio technique consists of the following parts [Gab97]:

- definition of the portfolio dimensions — typically portfolios are two-dimensional (some exceptions exist like technology portfolios), the specific dimen-

Copyright © Fraunhofer IESE 2003

sions also vary (e.g., like market growth vs. market share, competitive position vs. life cycle phase)

- detailed characterization of these dimensions — this is usually done with the help of questionnaires and is necessary to avoid an arbitrary classification of the products; typically each axis is subdivided into two to five categories
- definition of actions — for each quadrant an appropriate measure is defined that will be taken for any product that falls into this category (this avoids the need for reinterpretation)

Different approaches for portfolio management are defined. The simplest approach is the one taken by the Boston Consulting Group. Here, a two-dimensional matrix with two subdivisions is defined. The corresponding diagram is given in Figure 8.

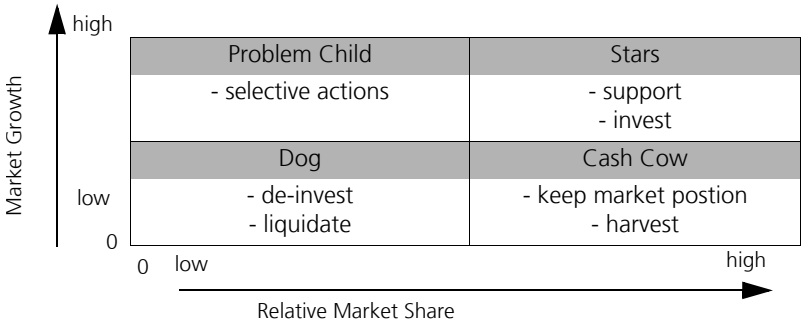


Figure 8: Portfolio Matrix — Boston Consulting Group

A rather complex example is the matrix defined in Figure 9. In this schema the different products are rated based on their life-cycle phase and their relative positioning in the market. Again, based on the categorization of the products recommendations for the further evolution of the products are given.

The approach of product portfolio definition can be applied on two levels to product lines. First, the product line as a whole can be placed in one of the fields and the resulting judgement can be applied. Of course, for an organization simultaneously investing in several product lines this is an important application.

However, in the context of this analysis, the more interesting application is to place the products into the individual cells. We aim at having the different products in different cells, thus covering a whole range of life-cycle stages and market strategies at the same time. Of course, in the context of customer-driven development, this approach can only be applied on the level of product groups. If these product groups correspond to customer groups, then we can also apply the approach to identify whether we are serving a customer group insufficiently.

Copyright © Fraunhofer IESE 2003

Lifecycle Phase

Competitive Position

	Birth	Growth	Maturity	Decline
strong dominant	capture further market share (or at least stabilize)	keep position / market share	keep position; grow with market	keep position
fortunate	invest for improving position, capture market share	• invest for improving position • capture market share	keep position; grow with market	keep position or harvest
weak supportable	selective improvement of competitive position	• try improiving position • selective capture market share	• minimal investment for keeping position • retreat to a niche	harvest or stepwise reduce commitment
	selective improvement of competitive position	• migrate to niche and keep it	• migrate to niche or stepwise reduce commitment	stepwise reduction of commitment or liquidation
	strong improvement or stop	• strong improvement or • liquidation	stepwise reduction of commitment	liquidation

Figure 9: Portfolio Matrix — Market / Product Lifecycle

If we use the market-/product lifecycle as a basis, a healthy product line provides products that are in all stages of the product life cycle [SU97] and at least some of the products should be in strong market position area or higher. If no product has a strong market position, then we need to conclude that the product line as a whole is too weak. Moreover, if no product is in the birth stage, we need to identify whether our product line infrastructure as a whole has come to age and may no longer be strong enough for new products (requiring some reengineering of the platform).

Similarly, if we use the portfolio of Figure 8 as a basis, we would expect that some products have a relatively high market share and some should be positioned in areas with strong market growth.

A portfolio-based approach to analyzing reuse opportunities (the so-called MECP-framework) has been proposed by Favaro and Favaro [FF99]. In this approach, the portfolio has two dimensions:

- **Market Economics** — this can be either attractive or unattractive and is further defined through the following forces (the first two ones are direct forces, the latter four are limiting forces):
  - intensity of direct competition
  - customer pressures

Copyright © Fraunhofer IESE 2003

  - intensity of indirect competition (pressures from replacement products)
  - threat of competitor entry
  - supplier pressures
  - regulatory pressures
- **Competitive Position** — the competitive position can be either advantaged or disadvantaged. This is in turn characterized by cost & asset position and the differentiation aspect:
  - the cost and asset position describes the cost advantage over the competition that can be drawn from the reusable assets
  - the differentiation dimension describes the diversity of products that can be produced with the asset base

While the MECP-framework at least addresses the issue of analyzing the strategic situation of a reuse development, it does not support the analysis of strategic product developments in a product line context: how do identify further products that should be developed in the context of a product line. For characterizing the product portfolio of a product line we propose the following dimensions, which can be split into two different portfolio descriptions:

The *External Portfolio*:

- *market coverage* — how well are different market segments covered based on the product line?
- *market growth* — how strong are the underlying market(s) growing?

These two aspects together establish the quality of the underlying market and the advantage we take from the opportunity of providing differentiation through a product line. On the other hand we need to characterize the actual benefits and the evolution stages of our product line

The *Internal Portfolio*:

- *Profitability* — how good are the margins, we derive from the products? Some time after the setting up of the product line these margins should at least be higher than for products that are developed in a single system manner.
- *Product-life-cycle* — this describes how far in the product life-cycle our products are evolved

Both portfolio definitions can be applied on a product line level or on a per product level. If applied on the product line level, this tells us what are future evolution opportunities for the product line as a whole; when applied for the individual products, it informs us where these improvement possibilities could come from.

Copyright © Fraunhofer IESE 2003

3.3.2 Quality Function Deployment

The portfolio approaches can be used to identify gaps in the strategy that need to be filled with new products. The characteristics of the products can be derived on a very high level from the portfolio approach (e.g., if we identify an insufficient market coverage we can conclude the need to develop products that cover additional market niches. In addition the strategy goals chosen according to the categorizations given in Section 3.1 provide us with input on the qualities of the to be defined products. Of course, in order to develop adequate products a further refinement of these requirements through customer investigation is necessary.

A systematic process, that enables to introduce such forces in a product design process is quality function deployment. In this approach customer preferences are systematically identified and quantified. In order to derive market relevant products this approach relies on two principles:

- Rating of functions based on customer importance [Coh95, HSM00]. This can be done using many different approaches like rating importance, measuring absolute importance, and so forth.
- In addition characteristics of the products can be qualified as satisfiers, dissatisfiers, and delighters. This allows us to easily characterize what are the key requirements for a product that fills the role identified by the strategy.

3.4 Quantitative Approaches to Strategy Definition

The qualitative approaches to strategy definition are a major first stepping stone to understand to possible directions for further evolution of a product line strategy. However, a qualitatively defined strategy is incomplete without the definition of quantitative goals. Defining such quantitative goals is the key focus of this section. We will focus here on two kinds of approaches:

- The *balanced scorecard* approach aimed at identifying key measures for describing a product line strategy (and thus capturing quantifying its key focus)
- Techniques from *marketing science* that aim at optimizing the specific features offered by a product line

3.4.1 Defining the Key Dimensions of A Product Line Strategy

The *Balanced Scorecard* approach is a general approach to the definition of quantitative goals for a company strategy [KN96]. Fundamentally, this is a goal-oriented approach which approaches the definition of measures mostly top-down, similar to the GQM approach in software engineering [BCR94]. While the

Copyright © Fraunhofer IESE 2003

approach focusses mostly on the specific metrics that should be fulfilled by a strategy and not so much on how this may happen, it is still more adequately defined as an approach to strategy definition then as an approach for controlling it. The reason for this is that the measures are mostly defined to give detail and depth to the high level strategic concept, i.e., to determine a way to operationalize the basic strategic concept.

Balanced scorecard explicitly aims to provide a wide, company-encompassing view. For this reason it does not solely focus on the Return-On-Investment (ROI), but addresses the following four dimensions:

Financial: the core measures of a company, typically they relate to profitability and are measured by ROI are similar metrics

Customer: these measures take a customer focus and analyze to which degree the customer is addressed by the work of the company. Typical examples are: customer satisfaction, customer retention, or market share in targeted market segments

Internal processes: this focuses on the internal processes of the organization, trying to capture how good the organization is at doing what it is supposed to do

Learning and growth: this takes on long-term view and addresses how good the organization is at setting up an environment for long-term learning and growth. Investments in people, organizational memory and procedures can contribute to this

For each of these categories objectives, measures, targets, and initiatives should be defined [Bug]. This results in the aforementioned step-wise refinement approach. There is no general rule on how to determine good measures for a balanced scorecard. Rather the process itself is used to provide the necessary focus.

The different dimensions of the balanced scorecard are impacted in numerous ways by product line development:

Internal Processes: In general product line development leads to a reduction of the total development effort and the time required to development new products. This can often go up to 70–90%. Moreover, from measurements at HP it is also well documented that the quality may be strongly improved [TCO00]. CelsiusTech also reported a strong decrease in project risk [BC96].

Customer: From a customer perspective product line development enables faster delivery of new products and broader selection of products, enabling the company to offer a product portfolio which is better adapted to the customer needs. Moreover, these products can be offered at lower prices. The improved

Copyright © Fraunhofer IESE 2003

quality will also positively impact the customer satisfaction and thus customer retention.

Internal Processes: Product line development leads to building a common reuse infrastructure. This reuse infrastructure actually contains the competence of the organization in product development in this area. This is a typical example of organizational learning.

Financial: From a financial point of view product line development enables the organization to develop products at lower cost, thus enabling the organization to either sell them with higher margins or at lower prices in order to attain higher volumes. In both cases a strong improvement of the return on investment can be expected.

Of course, the specific benefits that an organization seeks must be identified from its general strategy as discussed above. This must then be matched to the specific benefits that can be attained along the various dimensions.

3.4.2 Optimizing the Return On Investment from a Product Line Portfolio

There is a surprisingly large amount of literature on defining a product portfolio for a product line. However, these approaches mostly address the non-software area. They typically stem from the area of marketing science. These approaches rely on mathematical optimization based on a detailed characterization of the potential products. The different techniques take various different approaches at improving the structure of a product portfolio.

Most approaches are based on the maximization of an utility function (which is based on an estimate of the customer's preference). This problem comes in two varieties (cf. [GK85]):

- *Buyer's welfare* — The seller selects the product line, so that it maximizes (within the given constraints) the utility of the buyer.
- *Seller's welfare* — Maximize the value to the seller based on known preferences on the side of the buyer.

Using one of these approaches, it is now possible to optimize the structure of the product portfolio by selecting products (and their features) appropriately [GK85, MZ88]. Some methods use an indirect approach, by introducing another form of criterion or by introducing additional parameters for optimizing the product portfolio. Examples are:

- *Cannibalization* (i.e., competition among products) [MM94] — different products in a product portfolio will compete partially for the same customers, thus, the number of sold products will not rise in the same manner as the

Copyright © Fraunhofer IESE 2003

number of offered products. Instead, a threshold has to be determined beyond which the introduction of additional products will not provide an additional benefit.

- *Pricing of the product line* [DK88] — depending on the products that are made available and the prices the companies ask for, revenues will vary. As prices may impact the customer decision of which product to pick, pricing and product line composition interact.
- *Length of the product line* [Fra92] — economies of scale and scope that can be realized during the production of the product line will influence the additional costs incurred by adding products to the line. On the other hand, products will be substitutable with each other from a customer point of view. Thus, a trade-off decision has to be made on how many products to include in the product line.
- *Selection of market segments* [Ste91] — typically the market can be envisioned as being composed of several market segments, i.e., major groups of similar products. Which market segments to address is a strategic decision that will strongly influence both the investments that need to be made as well as the revenues that are possible.

While the above-mentioned approaches rely on explicit codification of the customer preferences and a linear programming approach to the optimization of the product portfolio, some approaches use a more informal way of addressing the problem. Examples are [RU98, Esq98], which aim at balancing core product requirements and total component development and production costs. A similar approach has recently been proposed in the context of the VBSE approach [FHR00, RFH00]. This approach focuses on the buyer value as a starting point for determining and actually controlling product development. Some approaches also aim at balancing the product portfolio and the reusable assets to be produced simultaneously in a formal decision making framework. This approach is exhibited by the MoVE method (cf. [BJJ+01]).

The approaches we discussed in this section are rather complex in terms of the mathematics they involve. However, when applying them, one also needs to take into account that these approaches solely focus on the financial perspective. This can be considered a severe shortcoming considering the discussion in Section 3.4.1.

Copyright © Fraunhofer IESE 2003

4 Refinement and Control of Investment Strategies

In the preceding section we proposed a simple framework for defining a product line strategy. However, merely devising a strategy is one thing, actively controlling and enforcing something different. We will now discuss in more detail the various models that can be used for controlling and enforcing a product line strategy.

In order to actually ensure that a product line strategy is followed and delivers the expected results, we need to continuously monitor the results of product line development. This actually needs to be done on two levels:

1 We need to monitor the ultimate adherence to the strategic goals of product line development and their effectiveness

2 We need to monitor the influence of performing product line development on a detailed level as a basis for identifying any reasons for divergence from the ultimate goals.

A simple example may help to illustrate this distinction. If we have the strategic goal to take over feature-leadership, i.e., always be the company which brings a new feature to the market first, than we need to control our ability to do so, we need to control the effectiveness of this strategy, and so on. This is what corresponds to the first point and we will discuss it in more detail in Section 4.1. However, in order to understand any reasons for deviations, we need to be able to analyze product line development on a much more fine-grained level. We understand the effectiveness of our reuse. The amount of potentially reusable functionality and so forth. All this is the basis for understanding our effectiveness at employing product line development as a strategic approach. This is illustrated by the second point above and we will discuss it in more detail in Section 4.2.

4.1 Measuring Strategic Success

As we discussed above in Section 3.4, it is important to define a strategy in quantifiable terms. One of the key reasons for doing so is of course that it enables us to identify deviations from the strategy early on and thus enables us to determine focussed improvement actions.

The approach that has taken this idea the furthest is the balanced scorecard approach [KN96]. In this approach one aims at deriving objectives, measures, targets, and initiatives for the different strategic goals. While objectives, targets,

Copyright © Fraunhofer IESE 2003

and initiatives are all qualitative information and refinements of the strategy, the measures are key to really setting controllable goals and identifying whether the strategy is ultimately successful. In order to arrive at a concise and relevant set of metrics that also enables the organization to identify the reasons for deviations from the expected results, if they should occur, the measures should adhere to the following criteria [Bug]:

- *Cause-and-effect relationship:* every identified measure should be part of a cause-and-effect relationship, with the overall effect resulting in the expected strategy result
- *Performance drivers:* those factors that are expected to impact the performance (and that are thus starting points for corrections) must be included in the overall analysis in order to understand deviations
- *Relation to financial indicators:* the bottom line of strategic success is financial success. Thus, the various indicators should be clearly linked to financial measures

In the context of product line development this implies that the identified measures need to address the different parts of the value chain of product line development. These are on a high level domain engineering and application engineering. Without addressing both parts a cause-and-effect relationship cannot be established. On a more fine-grained level, the various steps in the development (like requirements, architecture, coding, testing) etc. should be addressed, as each of these steps needs to be in place and must work properly in order to achieve the overall results expected from product line development. The specific decomposition model we propose, will be discussed in Section 4.2.2. The notion of performance drivers is well known from measurement in software development [BCH+95, BCR94, Pfl96, MCGZ00]. Thus, we will not discuss it here further.

The relationship to financial indicators can act again as a reminder that the various perspectives of the balanced scorecard actually must be taken into account. We discussed this already in Section 3.3 and will thus not go into further detail on this here.

4.2 Controlling Product Line Results

Based on the discussion, we can now turn to defining specific metrics for controlling product line results. The metrics necessary to control that the product line effort achieves the appropriate results are just as diverse as the manifold different strategic goals that may underlie a product line initiative. Some of the most common strategic goals we discussed above are:

- Cost-Leadership

Copyright © Fraunhofer IESE 2003

- Feature-Leadership
- Time-To-Market
- Diversity (ability to offer a large range of products)
- Customizability (ability to offer specifically customized system variants for a customer)
- Quality-Leadership (ability to offer solutions of high quality)
- Integrated Solution (a number of solutions from the product line can be easily and seamlessly integrated to address customer problems)
- ...

While this list relies on a large number of different goals for product line development, the metrics commonly proposed for product line development focus exclusively on variations of the cost theme [Pou97b, Pou97a, Fav96, Lim96, MCGZ00, BS96]. An analysis of these models highlights that they mostly vary along three main dimensions:

- The underlying reuse model
- The level of organizational granularity
- The goodness criterion

In the specific context of our analysis, the underlying reuse model is of course fixed: it is the product line model, with its prototypical division into a domain engineering and an application engineering phase. Regarding the level of granularity of the organizational unit, we ultimately need to arrive at the strategic level (the company level). This may be by means of more detailed metrics. However, in the end the various metrics need to add up. Finally, regarding the goodness criterion we follow the argumentation of Favaro [Fav96], who based on a detailed analysis favors the criterion of the Net Present Value (NPV) as the key criterion for analyzing reuse investments.

We follow here the argumentation that we did already put forward in Section 2.2.1 that an adequate economics model must take the full breadth of influences into account. So far only few models have taken into account more than just costs. Mostly these approaches were based on the Goal Question Metric paradigm [BCR94] and argued for a situation-specific choice of metrics [Pfl96, DSF95]. While we are strongly in accordance with this view that metrics need to be adapted, we propose to provide some guidance in the process of metrics selection. We did already provide some basic help to this by proposing a framework for product line quality model development [Sch00a]. A key contribution of this model is its decomposition of product line activities, which allows to analyze the contributions to product line success in a piece-wise fashion.

Copyright © Fraunhofer IESE 2003

4.2.1 Decomposing Qualities

Given a quality Q (e.g., eff: effort or rel: reliability) and a feature f for which the to-be-expected quality has to be determined, $Q_{base}(sit(f, P))$ denotes the value of the quality attribute for this feature according to the basic quality model, where $sit(f, P)$ denotes the characterization of the feature and its overall development situation in terms of influential factors. In particular, P denotes the product (P) context. In cases, where functionality is retrieved from a different product P' this is denoted by mentioning the P'-context in addition.

On the other hand, in the case of product line development different product line development situations (i.e., modes) have to be distinguished (cf. Section 4.2.2). Consequently, we denote the corresponding quality attribute value for the product line situation as $Q_{PL}(mode, sit(f, P, P'))$ and denote the product line distortion itself as: $\partial Q_{PL}(mode, sit(f, P, P'))$. Using this notation we can describe Q_{PL} as:
$Q_{PL}(mode, sit(f, P, P')) = \partial Q_{PL}(mode, sit(f, P, P')) \text{ Þ } Q_{base}(sit(f, P, P'))$

This allows to characterize quality attributes on a per feature basis. However, there are some general (infrastructure) activities, which will usually be difficult to relate to a single feature.

Depending on the specific quality analyzed, it may be very hard to attribute the quality contribution of this activity to a certain feature. For example, gathering the high-level requirements will be very hard to relate to single features from an effort point-of-view. However, from a defect point-of-view the requirements will show up as defects in specific software features. We will discuss these infrastructure activities further in Section 4.2.2.

In general the quality impact of these activities can only be analyzed on a per-project level. Consequently, they depend on project characteristics. We will denote this by: $Q_{PL}(mode, sit(proj))$. Where $sit(proj)$ gives a project-classification in terms of situational factors.

In order to understand the role of the product line development plan, one needs to understand various different software development modes that are relevant to product line development. This will be discussed in Section 4.2.2. In the subsequent sections we will then discuss various issues related to the notion of product line development situations.

4.2.2 Product Line Development Situations

We can distinguish different situations in which product line development happens depending on whether we are doing development for reuse or with reuse. In addition, we can differentiate different starting situations, depending on

Copyright © Fraunhofer IESE 2003

whether some software already exists and if so, in what state it is. If we take the development of software for a specific system additionally into account, then we can classify the product line development situations along the following two dimensions:

- development approach
  - development for specific system (one-at-a-time development; single systems are developed)
  - development with reuse[1] (using an existing reuse base a new system is developed – this is also called *application engineering*)
  - development for reuse (reusable assets are developed – this is also called *domain engineering*)
- starting situation
  - nothing exists (development from scratch)
  - reusable asset exists
  - implemented in a different product (needs adaptation)
  - implemented in legacy system (needs reengineering)

These two dimensions define the matrix given in Table 2[2].Using this notation we can write:

$$Q_{base}(sit(f, P, P')) = Q_{PL}(spec, new, sit(f, P, P'))$$

The core reason for distinguishing between the different situations using such a matrix is that the actual processes enacted will vary depending on the situation and thus also the influence of the quality will change accordingly. As the specific activities will actually vary from company to company, we discuss here prototypical, high-level categories of activities.

	reusable asset Reuse	development from scratch New	implemented in different product Adapt	implemented in legacy product Reeng
system-specific development (**spec**)	Q_{PL}(spec,reuse,sit(f,P)	Q_{PL}(spec,new,sit(f,P))	Q_{PL}(spec,adapt,sit(f,P,P'))	Q_{PL}(spec,reeng,sit(f,P,P'))
development with reuse (**with**)	Q_{PL}(with,reuse,sit(f,P))	Q_{PL}(with,new,sit(f,P))	Q_{PL}(with,adapt,sit(f,P,P'))	Q_{PL}(with,reeng,sit(f,P,P'))
development for reuse (**for**)	Q_{PL}(for,reuse,sit(f,P))	Q_{PL}(for,new,sit(f,P))	Q_{PL}(for,adapt,sit(f,P,P'))	Q_{PL}(for,reeng,sit(f,P,P'))

Table 2: Product Line Development Situations

1 Note, that in the development with reuse specific system parts may also be developed as system specifics, i.e., they are developed from scratch, but they are developed in the context of developing a system mostly from reusable pieces.

2 Note, that a mode is actually described using two parameters: $mode = (x, y)$. As a notational simplification we write $Q_{PL}(x, y, sit(f))$ instead of $Q_{PL}((x, y), sit(f))$.

Copyright © Fraunhofer IESE 2003

As discussed above, there are some process steps, which have an impact on a quality aspect, which cannot be easily attributed to a single feature. A prototypical example for this is the development of a reference architecture and the relationship of this activity to effort.

In the case of single system development (**spec**), we can identify the following main activities, if we assume that implementation happens via the successive integration of features:

- General requirements gathering
- Detailed requirements gathering *(can be attributed to individual features)*
- System architecture development
- Detailed design *(can be attributed to individual features)*
- Implementation *(can be attributed to individual features)*
- Unit test *(can be attributed to individual features)*
- Integration (test) *(can be attributed to individual features)*

Besides the basic development phases, we also want to take into account the maintenance phase. Here, we focus on pure problem correction:

- Problem correction *(can be attributed to individual features)*

The activities given above apply for the starting situation *new*. They have to be adapted for the other starting situations accordingly. However, the identified infrastructure activities stay the same in this case. When looking at the categorizations as "related to individual features" vs. "general activities" given above, one should keep in mind that depending on the specific quality under study the categorization may look somewhat different.

Similarly to the categorization given above, if we look at the product-line based development of software, we can in a matching form identify general activities and activities that can be attributed to individual features.

If we look at the standard product line process given in Figure 10, we can distinguish the following activities for the two life-cycles (here, we count scoping as part of the domain engineering life-cycle).

As two different main life cycles exist (domain engineering and application engineering), we have to distinguish the identified main activities accordingly:

Domain Engineering:

- Product line mapping
- Scoping

Copyright © Fraunhofer IESE 2003

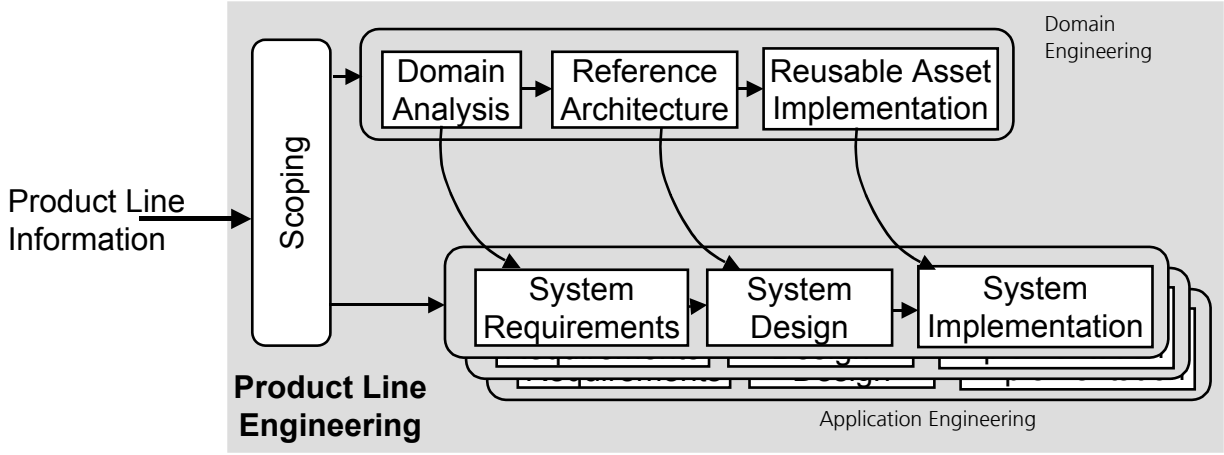


Figure 10: Product Line Process Illustration

- Detailed Domain Analysis (can be attributed to individual features)
- Development of reference architecture
- Detailed design (can be attributed to individual features)
- Implement feature reusable (can be attributed to individual features)
- Unit test (can be attributed to individual features)
- Problem correction (can be attributed to individual features)
- Product rebuilding[1] (can be attributed to individual features)

The product rebuilding step is specific to a product line development approach, as in this case if a generic asset is updated, we need to integrate the new instance of the component into all systems, so that the problem correction is integrated everywhere.

Similarly, there is the application engineering life-cycle. Here, we have again two main process variants, depending on whether a reusable asset for this feature already exists or not. In the case that no reusable asset for the system part exists, the process is rather similar to the process for single system development.[2]

1 Is performed only as a consequence of problem correction.

2 Note, that in the case that non-reusable sub-systems become as small as individual features, we have a direct mapping to the quality impact also for "generic" activities.

Copyright © Fraunhofer IESE 2003

	Parts that are reusable	**Parts that are not yet reusable**
	General requirements gathering	
(per feature)	Details for system-specific adaptation	Detailed requirements gathering
	Derive product instance architecture	
(per feature)	Retrieve feature implementation	
(per feature)	Instantiate feature implementation	Detailed design
(per feature)	Component implementation	
(per feature)	Unit test	
(per feature)	Integration (test)	
(per feature)	problem correction done in domain engineering	problem correction

Again, we have specific variants of these processes for the situations where already products exist which can be adapted or reengineered.

If we look at the steps that have now been identified as general infrastructure steps we find that these are the high level analysis (requirements, scoping) and development (architecture) steps. In addition, general activities like configuration management may play a role, but this is outside the scope of this study. Based on the discussion above, we can derive the list of activities given in Table 3:

	reusable asset **Reuse**	development from scratch **New**	implemented in different product **Adapt**	implemented in legacy product **Reeng**
system-specific development (**spec**)	not applicable	a) Detailed requirements b) Detailed design c) Implement F d) Unit test e) Integrate F in P	a) Detailed requirements^{+} b) Identify F in P' c) Recover code from P' d) Adapt Code for P e) Unit^{+} test d) Integrate^{+} F in P	a) Detailed requirements^{++} b) Identify^{+} F in P' c) Recover^{+} code from P' d) Adapt^{+} Code for P e) Unit^{++} test f) Integrate^{+} F in P
development with reuse (**with**)	a) Detailed requir. b) Instantiate feature b) Retrieve F b) Integrate F in P^{+++*}	a) Detailed requirements* b) Detailed design* c) Implement* F d) Unit* test e) Integrate* F in P	a) Detailed requirements b) Identify* F in P' c) Recover* code from P' d) Adapt* Code for P e) Unit^{+*} test f) Integrate^{+*} F in P	a) Detailed requirements b) Identify^{+*} F in P' c) Recover^{+*} code from P' d) Adapt^{+*} Code for P e) Unit^{++*} test f) Integrate^{++*} F in P
development for reuse (**for**)	not applicable	a) Detailed domain analysis b) Detailed design** c) Implement** F d) Unit test**	a) Detailed domain analysis^{+} b) Identify** F in P' c) Recover** code from P' d) Make code generically reusable e) Unit test^{+**}	a) Detailed domain analysis^{++} b) Identify^{+**} F in P' c) Recover^{+**} code from P' d) Make code generically reusable^{+} e) Unit test^{++**}

Table 3: High-level Development Activities Based on Product Line Development Situations

Copyright © Fraunhofer IESE 2003

If we want to look at the total quality for a specific feature (or product) then we will usually have to take contributions from different modes into account (e.g., feature is developed for reuse, later it is reused in different products).

In the above table many activities can happen in different modes leading to variations of how the activities will be performed and this may consequently have different impacts on those qualities. In order to distinguish between the same activity in different modes, we mark multiple occurrences with ▫ and *. We use ▫ to differentiate among instances of the same activity in different starting situations and * to differentiate among instances of the same activity in different development approaches (e.g., integrate).

4.2.3 Cost Modelling

So far, we discussed our reference framework for product line quality models in a rather generic manner. However, for some qualities (e.g., reliability) still today models are missing in a single system context that allow to derive forecasting models [Fre00]. In the context of single system development the most successful — and most investigated — thread has been the development of cost models. This is motivated by the importance particularly costs have for development decisions. Thus, we did also focus in our work on cost models for product family situations [Sch02]. From our experience we propose in a way similar to Pfleeger [Pfl96] to develop cost models in a context-specific manner. A basis for the necessary breakdown of the development activities can be found in the reference model we described in Table 3. In addition, we need to understand the various cost drivers of product line development. While this is very hard to do in general, we provide a list of the key cost drivers we could identify in Appendix B. Of course, according to the notion of GQM, we do not propose to use them as is, but to rather use them as basis for derivinng a situation-specific characterization.

4.3 A Framework for Strategy Control and Implementation

In the previous two sections we discussed the general concerns that need to be taken into account for controlling the success of a product line strategy. Now, we will discuss on this basis a specific framework (the PuLSE Strategy Control Framework), based on GQM, that supports the identification of the necessary metrics for controlling the success of a strategy.

If the implementation of a product line strategy fails, this can have two reasons: either the strategy was inappropriate for achieving the desired results (like increased revenue) or it was inappropriately implemented. Thus, we define two levels of strategy control in our approach:

Copyright © Fraunhofer IESE 2003

1 Controlling whether the strategy brings about the desired results

2 Controlling whether the strategy is adequately implemented

4.3.1 Refinement of the Product Line Strategy

The first part focusses on the question whether we can observe the intended results from strategy implementation with sufficient reliability. Key to this analysis is an adequate refinement of the high level strategic goals to an operational level. We consider this refinement actually a main and integral aspect of strategy definition. If we do not know in concrete terms how success of the strategy would look like, we have not defined the strategy in sufficiently tangible terms in the first place.

For deriving a sufficient and measurable refinement we use the approach of a goal-directed operationalization, similar to the GQM-approach [BCR94]. However, we perform a step-wise operationalization to bring in the various concerns. As an example for illustrating this approach let's consider the following strategy goal:

> A highly innovative company that operates in a fast moving market (e.g., in the telecommunication industry), that it should aim at being time-to-market leader. That is, its strategy should be to be the first on the market with new functionality. This should result in being able to get the highest margins on its products.

Thus, we can define the strategy as:

> Strive to become leader in terms of time-to-market for new features in new products.

The first step for clarifying this strategy is disambiguate it, by answering the following questions:

- What exactly is the market? (Is it mobile phones or *professional* mobile phones, does it include non-mobile phones and so forth)
- What exactly are the products? (This strongly relates to the previous question, but especially for innovation strategies this can entail a strong shift of focus, e.g., if we discuss mobile phones, are mobile PDAs part of the strategy? what about mobile Tamagotchis?)

This refers to the main distinction we made in Section 3 between growth and innovation strategies. Remember that we decided to focus on growth strategies. This simplifies our task considerably, as it enables us to more easily identify metrics that by assuming larger numbers testify that the strategy was successful.

Copyright © Fraunhofer IESE 2003

As the next step of refinement, we use the four perspectives described by the balanced score card approach (cf. Section 3.4):

- Financial
- Customer
- Internal Processes
- Learning and Growth

From each of these perspectives we can refine the goal and identify criteria that help us to identify whether the strategy was successful and whether it was successfully implemented. In principle, all four perspectives contribute to answering both questions. However, there is in general one of the four perspectives that is more closely related with one of the questions. In general we can say that:

- the customer perspective focuses more on the success of the strategy
- the perspectives internal processes and learning and growth focus more on the success in implementing the strategy
- the financial perspective combines the result of both aspects: the costs of bringing about the results and the result in revenues this generates

The fact that in principle both aspects can be applied to all four strategies leads to the following table, which we filled in with some refinements.

Perspectives	Success of the Strategy	Success of Strategy Implementation
Customer	We gain market share, particularly with the products that have unique features	We are able to offer products with new features that no other company has
Internal Processes	We ship more products	The total time from project start to delivery of new features is reduced
Learning and Growth	We become better at identifying features that lead to sales boost	Already existing features from other products can be reused with very little effort
Financial	The margin and overall revenues of products increase	The overall costs of new products decreases, a larger share of the money is spent on developing new products

Table 4: Example of Strategy Refinement for Market Leadership in Time-To-Market

Of course the different metrics already imply some measures for achieving the strategic goals (cf. Section 3.4). Here, we assume that the key idea is to achieve the goal by embarking on a product line approach, which should enable the organization to focus on the development of new features and to save effort on existing features.

Besides the four perspectives that are used in the traditional Balanced Scorecard approach, some authors propose an additional perspective: the people or employee perspective. We do not follow this approach, but rather go a step further. We add a **viewpoint** to our analysis schema, similar to the viewpoint in

Copyright © Fraunhofer IESE 2003

GQM. This enables us to take into account the multitude of different viewpoints that are relevant to a strategy implementation and it enables us to make the strategy consistent across the various viewpoints. This is particular important in the context of product line development, as already many product line attempts did fail due to inconsistencies among different perspectives (e.g., the direct goals of the project manager, who focusses on immediate delivery of his products, do not fit with the goals of the core asset manager, who focusses on long term growth of his asset base). Only through comprehensively eliciting, understanding, and transforming these goals is it possible to define an overall consistent strategy.

The specific viewpoints that are developed in the operationalization of a strategy are organization-specific as they are strongly related to the target organizations structure. Thus, we provide only some prototypical roles, here:

- Company management
- Financial management
- Customer relationship management
- Software development management
- Product line management
- Project manager
- Project development management
- Project sales representative
- Core asset development management
- Key account manager
- ...

Of course such a list can never be complete on one hand, while on the other hand generating a complete goal operationalization for all possible viewpoints will in practice be forbidding. Thus, specific viewpoints for an organization must be selected for the specific situation at hand. In this process, however, care should be taken to be complete in the following ways:

- A sufficiently broad range of roles need to be covered, if the strategy is only described for the top-management, we will not be able to identify where the strategy breaks down within the organization (if it breaks).
- Even people serving the same roles may have strongly different viewpoints. Thus, if more than one person serves a role, the input from more than one person should be sought (in particular, for controlling, we need of course acquire the input from all people serving a role).

Copyright © Fraunhofer IESE 2003

- If a sub-ordinate role (of another role) is brought into play, all sub-ordinate roles must be represented (e.g., if project management, which is sub-ordinate to product line management, is brought in, than core asset management should also be involved).

Of course some roles are more specific to product line development than others. Based on the roles above, prototypical roles are: product line manager, project manager, and core asset development manager. Let's suppose for the sake of the example that these roles are filled from a software development perspective in our example. If we assume this, we can refine the internal process perspective from the point of view of effectiveness of product line strategy implementation in the following manner for the three viewpoints:

- **Product Line Management:** How many new features with market relevance are available in products per year?
- **Project Manager:** How long does the development of my project take, how fast can we react to new features demanded for this project?
- **Core Asset Manager:** How many new features become available in the asset base per year?

Of course, these goal decompositions are just examples, but they are appropriate to demonstrate the main points: the various viewpoints together are required to ensure an adequate implementation of the strategy and the views of the lower level managers (here project and core asset manager) are actually refinements of the higher level view (here product line manager). We call this *refinement by organization* as the underlying schema is based on the organigram.

4.3.2 Operationalization of the Strategy

The steps outlined above already bring about a strong refinement of the core strategic goals. However, it is here not our key goal to define good company strategies, rather we want to specifically focus on the product line situation. This entails a much stronger focus on software development. On the other hand how much detail is appropriate in a specific situation strongly depends on the availability of certain metrics. For example, in an organization that is already at CMM level 3 and thus has a strong metrics program [Dym96], we can expect that most metrics we will need are already available. On the other hand for most organizations at CMM level 1 providing the necessary metrics will be very hard. In particular, process metrics on a more fine-grained level are often very hard to come by for such an organization.

Especially for the quality of the strategy implementation and for the improvement of it, we need such information, as we need to analyze how good product

Copyright © Fraunhofer IESE 2003

line software development works in the organization and how well it is aligned with the overall strategy. Thus, we are focusing from here on mainly on the strategic goals that take the internal point of view (i.e., controlling whether the strategy is adequately implemented).

In order to introduce more strongly the software development perspective, we need to refine the various strategy goals from the point of view of:

- product vs. process perspective
- life-cycle phase

Regarding the life-cycle phase we use the different process steps according to Table 3. One should note that not all process steps will be compatible with all viewpoints. For example the core asset development manager will mostly focus on the *development for reuse* perspective.

In order to derive specific metrics for the analysis of product line development, we first need to concisely define the characteristics of a strategy aspect we want to formalize:

Strategy Goal: what exactly is the goal that shall be achieved by the strategy; is it an optimization or an innovation goal; which part should we measure

Market Aspect: what is the market situation (expanded, contracting, new or stable); do we perform market-driven or customer-driven development

Metrics Level: do we aim at a market or software development type of metric; at which level according to the reference model (cf. Section 2.2) should the metric be defined

Company perspective: what is the specific goal of the measurement from a company point of view; to which BSc dimension does it belong?

Role: from the perspective of which role shall the results be quantified (this also defines the organizational granularity)

Product/Process: shall the characterization be performed from a product or a process point of view

Granularity: at which level of granularity (of the product / process metric) shall the metric be defined

Using this schema we can characterize a product line measurement goal and derive metrics in a similar way as this is done in the GQM-approach [vSB99]. In a similar way the specific metrics needs to be adapted as well to the specific situation of the organization. Here we provide some simple guidelines on defining the specific metrics:

Copyright © Fraunhofer IESE 2003

- The awareness of measurement in the organization will influence the possible metrics, the less awareness exists, the more coarse grained will the measures be. Usually measurement awareness can be correlated with the CMM level.
- In general some metrics will already be available like the times for product releases, the overall effort available for an organizational unit – and so forth. It will in general be necessary to overlap these already available metrics, with the required metrics in order to be sufficiently effective.
- The specific form of component development has a strong impact on the development of the necessary metrics. If there is a dedicated organizational unit for the development of reusable components, then it is much more easy to gather specific metrics on development for reuse.

In general one should aim at a good coverage of the various perspectives in order to develop a comprehensive set of metrics that enables to catch any deviations from the strategy early on and to identify the reasons for this.

Copyright © Fraunhofer IESE 2003

5 Evaluating the Future

So far, we dealt with prodcut line strategy as if the future could be planned for. Of course this is not true. This forces us to make strategic decisions in the face of uncertainty. In this section we will describe analysis techniques for determining the future value of decisions. This will require a transition from evaluation a single scenario to evaluating a complete range of scenarios

5.1 Overview of Approaches

Whenever we want to look ahead into the future, we need to face the concept of risk: things may not turn out as we hope or expect them to be. In order to come up with a realistic evaluation of our strategy we need to take risk explicitly into account and must determine the value of the strategy under this constraint.

There are two different types of risk that we must address:

Systematic risk or market risk — these are risks that are of a general nature and are extrinsically to the organization performing the analysis. These risks usually affect all players in a market in a similar manner. Typical examples are an economic downturn or a raise in interest rates.

Unique risk — these risk are specific to the organization and particularly to the strategy.

Many different approaches to risk analysis have been developed. However, most approaches in software engineering focus on the identification of unique risk [Boe89, Kon01]. In its qualitative form, we can address risks by identifying the different scenarios that may occur if the risk manifests. This point of view is very close to *game theory* [Sim00].

In game theory, we see the economic situation as a game with different players involved (the different risks correspond to moves of certain players). Through analysis of the different situations and potential game outcomes we can identify the next move with the highest value. In game theory the different moves are usually treated as having the same probability. This approach of analyzing potential actions and appropriate counteractions is also the basis for decision tree analysis [Wit96].

Of course the different risks are not equal. Rather the probability with which they can occur can vary considerably. This can be taken into account in *decision tree analysis* as we will discuss below. Besides analyzing the different possible cases directly, we can as well perform *option analysis*. This approach is more

Copyright © Fraunhofer IESE 2003

appropriate for analyzing specific possible, individual occurrences or chains of events.

5.2 Decision Tree Analysis

5.2.1 The Concepts of Decision Tree Analysis

The key idea of decision tree analysis is to develop a game tree of possible risk situations and counter-measures that can be taken. This is shown in Figure 11.

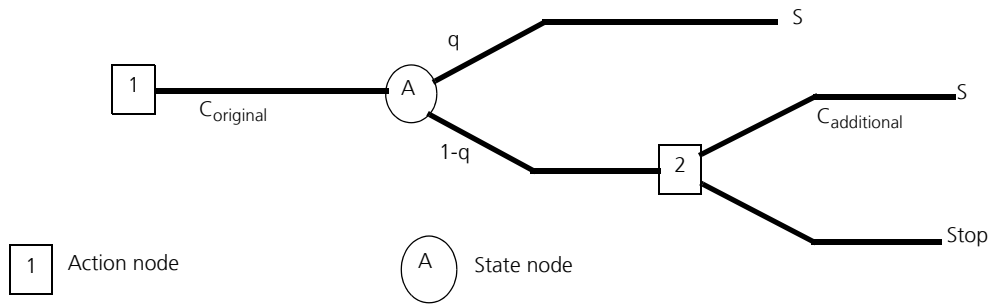


Figure 11: Example of a decision tree [Wit96]

In a decision tree we can distinguish two main types of nodes:

- Action node — at these points certain decision and actions must be taken
- State node — at these points certain information becomes available

Next to a state node we also write the expected probability for the corresponding alternative. At the leave nodes, we write the values we expect to arrive at.

Decision trees are typically evaluated in a step-wise manner, working backwards from the leaves towards the root node [HES+02]. In the example shown in Figure 11 this leads to the following computation:

If $S > C_{additional}$ we would choose the upper branch at decision node two, which would give this option the value of $S - C_{additional}$. More precisely this would have the net present value (NPV) of this case (with using the time it takes for realizing the benefit S after reaching state A. Similarly, if $S < C_{additional}$ we would of course make the decision to not invest thus, in this case the value of two would be 0 (no cost, no return). Now at the state node A, we need to combine the two possibilities, leading to:

- $q*S–(1-q)*NPV(S - C_{additional})$, in case $S > C_{additional}$
- $q*S–(1-q)*0$, in case $S < C_{additional}$

Copyright © Fraunhofer IESE 2003

In addition at the point of action node 1, we need to add $-C_{original}$ and to discount it additionally by the corresponding time. Of course this process can become rather complex for larger decision trees.

5.2.2 Applying Decision Tree Analysis to Product Line Strategies

Decision tree analysis can now be applied to the various decision situations we face in product line development, e.g., the need to set up a product line, the potential to add a further product to the product line, the possibility of developing a specific asset for reuse, etc. While, of course, the specifc decision trees are highly situation dependent, we can outline some typical nodes that may occur.

Setting up a Product Line

This is of course the most complex and the most comprehensive kind of decision tree that might be developed. Here the different *action nodes* might correspond to:

- building certain products
- developing certain assets for reuse (in specific ways)

the *state nodes* will typically correspond to:

- different market conditions
- successes in different product line development steps

Of course the setup of the product line can in turn be defined based on the decisions necessary for adding a product to a product line and the decisions required to developing reusable assets.

Adding a Product to a Product Line

Adding a product to a product line is often seen as an indivisible step. This is also how it will be represented in a decision tree for setting up a product line. However, if we look at the situation in more detail, it is usually possible to develop a specific product in certain sizes, which may depend on the costs incurred, technical difficulties and so forth. Correspondingly we will usually see the following *action nodes*:

- Developing certain (reusable) assets
- Adding certain user-visible functionality (in different ways)

the *state nodes* will typically correspond to:

- technical problems
- changing market conditions

Copyright © Fraunhofer IESE 2003

- financial problems (e.g., changes in licensing costs)
- most types of risks identified by risk assessments can be listed here

Just like in a risk assessment [Kon01], it is virtually impossible to fully identify all different possibilities. Consequently, it always represents an approximation of the real value. However, it implies the advantage that it allows to quantify the impact of the identified risks on the value of the project.

Developing a Reusable Asset

Developing reusable assets is an intrinsic part of the product line development situation. However, this can again be seen as a sequence of steps. This becomes particularly visible if we take into account that the necessary functionality may be developed incrementally, perhaps even with a prototyping approach. Correspondingly we will usually see the following *action nodes*:

- Generalize the assets
- Develop prototype or partial implementation

the *state nodes* will typically correspond to:

- technical problems
- changes of customer requirements
- expanded generalization required
- most types of risks identified by risk assessments can be listed here

This decision tree describes the incremental development of reusable assets. Of course, in order to describe the costs required for building the various steps of the assets we can use the model of product line development, we described in Section 4.2.

Decision tree analysis is a very powerful tool as it allows to perform a detailed analysis of the potential outcomes of a project and enables us to identify an adequate strategy from a value point of view. Moreover, it supports the use of NPV (net present value). However, it is simplistic in at least two ways:

- it makes the assumption that certain probabilities can be adequately estimated up-front
- it assumes that all necessary possibilities are known up-front

These problems of decision tree analysis can be partially resolved by using options approaches. Moreover, option approaches can be combined with decision tree analysis to value individual paths, thus leading to a more appropriate picture of the overall value of the decision tree. This we will discuss in the following section.

Copyright © Fraunhofer IESE 2003

5.3 Options

Options are a well known construct from financial theory. They represent *"the right without the obligation to perform a discretionary action in the future"*. In the context of finance this is usually the right to buy or sell a financial asset (e.g., a stock). This fundamental idea has been extended beyond the financial markets into the concept of *"real options"* [AK99], implying any kind of discretionary action on non-financial assets. Some authors even propose that a key basis for a successful strategy is the disciplined development of options [Wil99b].

The value of financial options has been thoroughly studied in finance, leading to a rather elaborate theory. With the extension to real options, the full power of this theory can now be brought to bear on real options as well. As some authors did point out [FF99, Erd02], reuse investments can also be seen as investments into real options.

5.3.1 A Brief Overview of Options

There are two main forms of options: *put* and *call options*. A put option enables the buyer to sell an asset at a later time at a fixed price. A call option enables the buyer to buy an asset at a later time at a fixed price. Consequently, the price of both types of options depend on the price of the underlying asset (*underlying*) in an inverse manner. This is shown in Figure 12. As this graphic shows, the value of the option is non-symmetrical relative to the value of the underlying asset.

This non-symmetrical property is key to the value of an option. However, it also makes it rather complex to determine the value of an option. Below we summarize the key terminology relevant to discussing the option concept:

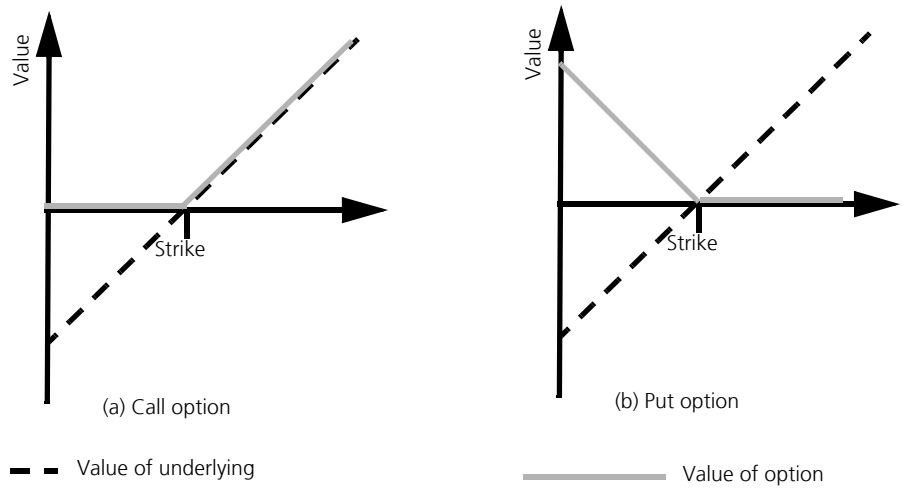


Figure 12: Value of options relative to the underlying asset

Copyright © Fraunhofer IESE 2003

Option: The option can either be a put or a call option.

Underlying: the underlying is the asset, the option refers to (e.g., for a IBM Call, the underlying is the IBM stock).

Strike price: the price at which the option can be exercised.

Strike time: the time at which the option can be exercised. One needs to note an important distinction here: there exist *European* and *American options*. While the former can only be exercised at the strike time, the latter can be exercised at any point in time up to the strike time.

The key question is now: how can the option concept help us in valuing a product line strategy?

5.3.2 Options in a Product Line Situation

Product line options can be seen as a particular form of *real options* [AK99, FF99, Erd02]. Real options in turn come in many different forms:

Growth / expansion option: a growth option characterizes a situation where the possibility for future investment exists. For example a small investment now enables a further investment latter. This is modelled as a call option.

Exit / Abandon option: the possibility to abandon a certain path of actions (e.g., investments in products that turn out to be non-profitable creates value in its own right). This is typically modelled as a put option.

Defer / Wait-to-invest option: the defer option enables an organization to perform a certain investment later. This option is created by a situation where deferring a path is possible. (This does not necessarily take an investment to acquire.)

Flexibility / Switch: the switch option enables to exchange certain parts. This might be the flexibility of switching certain components against different implementations, as well as the possibility to switch suppliers. This is typically modelled as a (set of) call options.

Learning option: the learning option is an investment that allows to perform some initial steps first before fully immersing into the next step. This is typically modelled as a call option.

In the reality of business decisions options appear in many different disguises. However, if we step back and examine them from a more abstract point of view, they are typically a form of put or call option, sometimes actually a combination of several options.

Copyright © Fraunhofer IESE 2003

If we look at the product line situation, then we see a large number of different activities that can be interpreted as options.[1] Some of these activities are:

- Component replacement [Erd02]
- Prototype first, generalize later
- Product Elimination
- Learning about the products

One should note that the addition of a product or a feature will usually not be modelled as a an option, although it carries potential (an option) for further revenue. Rather it is usually more appropriately modelled as an invesment and valued based on its NPV.

Component replacement refers to the situation that in a structured architecture one component is replaced by a different component, that can at least supply the same functionality. This may happen for several reasons like the replacement of a custom-developed component by a COTS component or the replacement of one COTS component by another. The following situations are particular to product line development:

- replacement of a component by a more general implementation. This is particularly relevant if some functionality is first build for a single product and later generalized *(see also prototype first, build later)*
- provision of an alternative implementation for a feature (e.g., a component for Fax communication while an E-mail component already exists). This is also related to the situation of feature addition.

Component replacement is always modelled as a call option. The price of the option corresponds to the additional investment required to achieve the necessary modularity for easy replacement of the component. The underlying is the benefit that can be achieved from such a replacement. The strike price is the cost of acquiring the replacement component. Consequently, the achievable benefit is characterized by the amount by which the benefit exceeds the costs.[2]

Prototype first, generalize later refers to the strategy that is often used in practice to first develop components specifically for a single product that is currently under development and to generalize these components later when the value of additional generalization becomes clear. However, in order to successfully perform such a strategy sufficient additional investments into the modularity of the platform must be made. This investments can be seen as the price that needs to

1 Here, we will focus only on the options that are developed in the continuous evolution of the product line. The transition to a product line is beyond the analysis of this study.

2 A very interesting treatment of the value of being able to replace components in an adequately structured architecture has been given in [SGCH01].

Copyright © Fraunhofer IESE 2003

be paid for the option. Again the strike price corresponds to the cost of developing the general component. While the underlying asset corresponds to the cost saving in developing the additional products.

Eliminating the products is a typical part of any product life-cycle. In a product line situation this occurs as well, however, the platform itself is not lost, but can be used in further product development. These additional value aspects are captured by the growth options that result from them. Under certain circumstances there might be further value aspects that must be managed (e.g., if we offer the platform itself to second source partners, etc.). In these cases a put option or abandon option exists that manifests the fact that some value pertains although the basic product has been abandoned.

Learning about the products is enabled by the fact that in a product line situation we will usually start out with developing general infrastructure, which by definition is rather flexible with respect to the specific products that can be built on top of it, while later on, we will develop the specific assets that determine the individual products. This delays the decision on the specific product features considerably. Time that can be used to understand much better the market conditions and thus, the specific products that need to be built. We discussed this in more detail in [SJ01].

After qualitatively discussing the various options that arise in product line development, we will now turn to the detailed qualitative treatment of options.

5.3.3 The Option Formula

Besides the basic parameters discussed above, further parameters are necessary to determine the value of an option:

- V, the current price of the underlying
- S, the strike or exercise price of the option
- T, the strike time (number of days until expiration)
- RF, the risk-free interest rate (e.g., the interest rate of the treasury bills)
- STDDEV, the instantaneous standard deviation of the underlying's returns

We did already discuss S and T. As we are dealing here not with traded assets (like stocks at the stock market), we will usually need to estimate the current price of the underlying. As this poses no problem particular to option analysis, we refer to the typical estimation processes that are made in product development. Typically, this will lead to an NPV analysis. The parameter RF is also very easy to estimate, as this actually refers to aspects of widely traded assets (i.e.,

Copyright © Fraunhofer IESE 2003

treasure bonds). The only severe problem here is the need to determine STD-DEV, i.e., the standard deviation of the underlying's returns.

This standard deviation can only be estimated by adequately looking back in time. How did the expected value from such an asset, e.g., from such a product, change. This is very difficult to estimate as usually, the current moment of evaluation is the first time people think about the value of the asset and there is no real track record in the real options case (as opposed to a stock). However, some general references to estimates of the market at large will usually help (e.g., how did the market of internet products develop over the last 5 years). As a higher standard deviation increases the value of an option, when in doubt the standard deviation should be estimated smaller so as not to exaggerate the value contribution of the option. A typical rule of thumb is to use 30%-60% for a project.

Let us now analyze a specific situation in order to illustrate this approach to valuation. Company A has already a product line in place. Now, a new customer comes in, however, he insists that he needs a different kind of user interface, which is not yet covered by the asset base. This leaves the management wondering whether it should generalize the user interface in a way that allows easy exchange. How much would be the value of performing such a generalization (or modularization) of the architecture over a specific adaptation for the next customer?

So the benefit that can be accrued here (the growth option) actually comes from the potential of saving costs in developing further products with new user interfaces. (Note, that we can not use the whole value of the product as the underlying as it would still be possible to perform a specialized adaptation of the product again. However, we can take into account the added complexity of future maintenance.) Usually, we will take here the NPV of the expected value of the cost savings as a basis.

Of course, it is again difficult to estimate the standard deviation. As two extreme scenarios we estimate that it can either never occur again that such an adaptation is necessary or that it may happen twice a year. We use a very simplistic approach to determining an estimate for the standard variation: we use a sequence of 0, 40, 0, 40 (a projection over four years). This leads to a standard deviation of ~23.[1] This standard deviation of course grossly overestimates the to be expected returns, as we used rather extreme values. As could be expected from our initial estimate (it could happen twice a year or not at all), the standard

1 Note, that an more appropriate modeling would be the sequence would be 80, 120, 80, 120; meaning at 80 return there would be no benefit in this year, while at 120, the benefit of modularization could be accrued twice. This modeling, however, would lead to an identical standard deviation. Note, that the most accurate modeling of gaining several times the full value for a lower cost cannot be modeled directly with a single option.

Copyright © Fraunhofer IESE 2003

deviation is rather high. This point illustrates a very important issue: a high standard deviation may easily result from unclear information of the situation and not only from an intrinsic risk of the situation.

If we now assume that the additional cost for a modularization that would allow the easy replacement of the user interface is about 0,1 of the cost of replacing the user interface directly (leading to a total value of 1.1 times the value for a direct replacement of the user interface), then we get the following parameters:[1] V: 100, E: 80, T: 360 (days), RF: 0,05, STDDEV: 0,23 (we have to give the percentage). This yields an appropriate option value of 25. This value, however, already includes the value for the already contracted product of 20. This return was already included in the original estimate, thus we need to subtract it here, leading to a value of 0.05 the cost of the user interface to match with the cost of the modularization of 0,15. Thus, the value of the option is actually lower than what we would pay for it.[2] Thus, in the analyzed situation we would usually not perform the additional investment, as the value of the option turned out to be lower than the cost for acquiring it over the expected time horizon.

As we will discuss in Section 5.4, this may for example change if we change the analyzed time horizon as the option value is strongly dependent on the parameters (especially the standard deviation).

The situation gets even more difficult if we take into account that additional options are available.

5.3.4 Combining Options and Decision Trees

In the situation above we actually have more than one option. We could just as well plan to develop a generic component that would enable us to develop the various products at even lower costs. For example, let's assume that we would be able to develop a generic component, which would enable us to develop a specifically adapted user interface for only 30% of the cost. However, on top of any specific user interface development, this would cost us about 200% of a single user interface. If we take a time horizon of two years and an expectation of about 1 product per year that needs such a user interface (consistent with our estimate above), than we would arrive at a value for the corresponding option (i.e., reuse investment) of about 150% (V: 200, E:60, T:720, RF:0,05, STDDEV: 0,81)[3]. This is lower than the required investment, thus, no such investment would occur. But, of course, this estimate critically hinges on our estimate of the

1 Note, that in order to get the full value of the next implementation, we still need to pay 80% of the price, i.e., in order to get 100, we still need to invest 80. In order to simplify the presentation, we will not take the need for discounted cash flows into account, i.e., usually we would need to perform a discount on our base values.

2 Computations were made with the options plug-in by Ray Steele [Ste], based on european options.

Copyright © Fraunhofer IESE 2003

frequency of such products. Thus, we need also to take into account the situation that our estimate is too low. If for example, this situation occurs on average 1,5 times a year, than this will of course impact the overall value strongly.

If we compare this possibility, then we get the following parameters (we assume a smaller standard deviation in this case as well): V: 300, E:90, T:720, RF:0,05, STDDEV: 0,7. This leads us to an option value of 225%. This would be a clear reason to perform the infrastructure investment. However, this is not compatible with the current estimates, thur from the current perspective it is not a recommended path of action.

This opens a third possibility: perform the modularization now, then observe the market over the next year and determine the most adequate estimates. Based on those determine again whether the necessary reasons for infrastructure investments exist. This is shown in Figure 13.

This possibility of performing the investment into a reusable UI implementation has again a value of about 24% of the total value of an UI implementation. (V: 140, E:200, T:360, RF:0,05, STDDEV: 0,7). However, one may not simply add the two options, as they we can only exercise them in one of the two ways. Thus, we need to estimate as well the likelihood of an increase in the estimate. In Figure 13 we estimate this to be about 20%. Thus, we need to fold back the tree, as we discussed earlier. This leads to a total value at point A (and as we disregard discounting for now) at point 1 of 0.3*24+0.7*5 = 10.7. This is actually higher then the required investment. Consequently, we would rationally decide to perform this action.

Then after a year, we can reevaluate the situation and either continue to perform the single development or develop a reusable implementation.

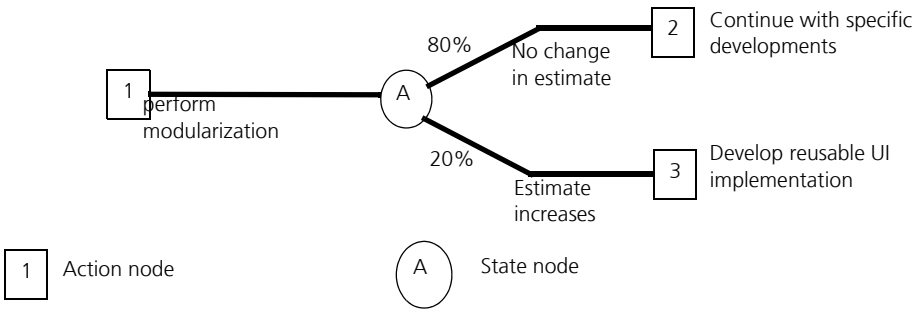


Figure 13: The staged investment with options

3 Here, the value would be value of a full component 100 (in two years), at a cost (E: 60); further we get STDDEV: 0,81, if we analyze a sequence of (60, 200, 60, 200).

Copyright © Fraunhofer IESE 2003

5.3.5 Product Line Options from a Market Perspective

So far we did only take into account cost savings in our options analysis. However, especially, if we look at time-to-market the impact of product line development is much broader: it can lead to more rapid development, which in turn may raise the worth of the products, particularly, if we are in a market-driven situation. In our options terminology this implies that the acquisition of the product line itself can increase the value of the underlying. Current option pricing theories do not address such cases.

Thus, in order to take this value increase into account in our analysis above, we would need to add the additional product value to the cost saving. This would of course increase the value of the corresponding option. It should be noted that sometimes the main value contribution for product line development may come from this time-to-market advantage. In this case the option modeling needs to include the potential variation of the product values.

5.4 Consequences of Option Creation for Strategy Implementation

As discussed in the previous section, the influence of the created options on the value of a certain strategy can be decisive for determining the appropriate path of action. However, working out the detailed mathematics for each situation is not a simple thing to do. Thus, we will focus in this section on some general guidelines that can be deduced from our option analysis.

A key distinction that we need to make in order to discuss option value is the distinction between options *in the money*, i.e., where the underlying is currently at a higher value than the strike price, and options *out of the money*, i.e., the underlying has currently a lower value than the strike price. (For put options these terms are reversed.) In addition we call options where the underlying has the same value as the strike price *at the money*.

Some initial heuristics emerge, once we analyze the parameter sensitivity of the option formula:

Strike price vs. Underlying: For options in the money the value of the option is mostly dominated by the difference between strike price and value of the underlying (the further in the money, the smaller in proportion is the added value). For options out of the money, the further the option is out of the money the smaller is the value of the option (however, it always remains positive).

Standard deviation: The value of an option strongly depends on the volatility of the underlying. For an option which is at the money, we can easily get a fifth of the value of the underlying at a standard deviation of the underly-

Copyright © Fraunhofer IESE 2003

ing of 0,5. Moreover, for at the money options the value of the option is roughly proportional to the standard deviation.

Time: The further away the expiration time of the option, the higher is the value of the option.

Risk-Free Rate: The higher the risk-free rate, the higher is the value of the option.

The above parameters are roughly sorted based on their impact. (The exact impact may depend on the exact relation among the values.)

We can now turn to the question of what these parameters mean for product line reuse:

- Whenever the strike price is lower than the current value, it means, that the investment into reuse is expected to pay off right away. So it will usually be appropriate to invest in reuse right away. However, this may easily destroy some other options, thus lowering the value.
- When the value gained from reuse is expected to be lower than the necessary investment, than it is key to analyze the implied options. A value difference of a fifth of the investment can easily be made up by the generated option value for realistic parameters (STDDEV: 0.5; T: 3 years; RF: 5%).
- The higher the uncertainty involved in a product line situation, the higher is the value of the generated options! Thus, any investment that increases the future flexibility (and thus creates options) is encouraged, while any investment that reduces the flexibility is discouraged. In this sense any refactoring activity and any development of reusable assets, that are still open to a large number of products leads to a strong increase in the value. On the other hand, even the development of reusable assets, that can only be applied in a limited number of systems actually reduces the value as it will usually destroy additional value. We can thus conclude:
  - High uncertainty favors general infrastructure investments
  - Low uncertainty favors the investment in specific reusable assets that have a limited range of applicability
- The further we can look ahead in time (the pay back horizon for investments usually depends on company policy), the higher will be the value of any options.

The rules given above have been derived from analyzing the parameters from a single call option. This corresponds to the basic growth option. We can also look at other types of options:

- The waiting and the learning option, typically go hand in hand. We wait until events unfold in order to learn more about the market situation. Sometimes

Copyright © Fraunhofer IESE 2003

we will explicitly prototype some technology or some products in order to learn more. In order that here an option is created, we need to leave our investment open in the sense that we focus on building rather general assets. This has been already illustrated in the example in the previous section, where developing specific assets for a reusable user interface were not worthwhile, while the investment in some modularity that would later on allow such an expansion lead to positive results. The advantage of such a deferring strategy is the higher, the higher the uncertainty is.

- An exit option is typically created by being able to abandon a certain path while still being able to recover part of the investment. This is typically the case when the components built as a basis for a certain product can be reused in a wide range of circumstances. The worth of this exit option strongly depends on the uncertainty of this specific kind of product (or type of product). Moreover, when valuing it one needs to take care not to value as well the additional possibilities that are created as this would entail to take the option into account twice.
- A switch option, i.e., the possibility to replace a component by another one is very important if we do not yet know the specific functionality that is important for a future product. Thus, if we need to commit, but have uncertainty about the specific functionality that is needed, we are better of when creating a switch option when building the component.

While the above heuristics give some insight in the potential benefits of creating options when building systems, there is still the difficulty of exactly determining the benefits. In general, we thus propose the following path of action: determine the various actions that are possible (including deferring the decision) and determine the options that are created in the course of action. If some path has some high positive pay-off, these are the key paths. Out of those, select the one with the highest number of options (unless the pay-offs are too different).

If some of the options are at the money, expect them to contribute about 10% of the total value (for high uncertainty increase this value, for low uncertainty decrease it). Usually, in this situation again the path of action with the highest number of contributing (at the money) options will be the most favorable ones. It should be noted, that this heuristic would have led to the same approach that we determined by thorough mathematical analysis in the previous section.

Copyright © Fraunhofer IESE 2003

6 Conclusions

Successfully establishing a product family is not an easy feat. Supporting it consistently over many years is even less so. In this deliverable with addressed the basis for the successful, continuous operation of a product family by addressing the need of an underlying strategy in which the product family must be successfully embedded. As the situations in which a product family can be successfully applied are rather diverse, there can not be a single, fixed model of costs and investments, that describes them all. Rather, a framework is required that enables a product family organization to establish its situation specific criteria for product family investments. This report provides a basis for this.

We propose a model of product line economics that collects the major influence factors that exist for product family value and describes their interrelation. This model provides a basis for the development of product family strategies. We discuss approaches for the derivation of product family strategies, focussing on portfolio approaches and the balanced scorecard technique for characterizing the strategy systematically in a numerical way. We even extend this technique by providing a systematic manner for deriving the necessary metrics that are required for controlling adherence to the product family strategy. As it is necessary in the evaluation of future benefits to take uncertainty into account, we also discuss the existing state of the art in evaluating future benefits.

In order to identify appropriate investments, it is not only required to estimate the potential benefits, we also need to determine the necessary costs. We did also address this aspect by providing a reference model for product family development, that can be used as a basis for the analysis, especially when combined with the GQM-approach [BCR94].

As a result, this report provides a basis for the strategic management of ongoing product family development. However, as transitioning to a product family adds major concers that have not been addressed in this deliverable, we see the need to focus on this aspect in the context of our future work.

Copyright © Fraunhofer IESE 2003

A Bibliography

[AK99] Martha Amram and Nalin Kulatilaka. Disciplined decisions — aligning strategy with the financial markets. *Harvard Business Review*, pages 95–104, January–February 1999.

[BC96] Lisa Brownsword and Paul Clements. *A case study in successful product line development*. Technical Report CMU/SEI-96-TR-016, Software Engineering Institute, Carnegie Mellon University, 1996.

[BCH+95] Barry Boehm, Bradford Clark, Ellis Horowitz, Chris Westland, Ray Madachy, and Richard Selby. Cost models for future software life cycle processes: COCOMO 2.0. *Annals of Software Engineering*, 1995.

[BCR94] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. Goal Question Metric Paradigm. In John J. Marciniak, editor, *Encyclopedia of Software Engineering*, Volume 1, pages 528–532. John Wiley & Sons, 1994.

[BEB98] Lionel C. Briand, Khaled El Emam, and Frank Bomarius. COBRA: A hybrid method for software cost estimation, benchmarking, and risk assessment. In *Proceedings of the Twentieth International Conference on Software Engineering*, pages 390–399, April 1998.

[BEM+98] Lionel C. Briand, Khaled El Emam, Katrina Maxwell, Dagmar Surmann, and Isabella Wieczorek. *An assessment and comparison of common software cost estimation modeling techniques*. Technical Report 045.98/E, Fraunhofer IESE, 1998.

[Bey01] Horst-Thilo Beyer. *Innovative product management*. Available on the Internt, in German., 2001.

[BHJ+] Andreas Birk, Gerald Heller, Isabel John, Stefan Joos, Klaus M" uller, Klaus Schmid, and Thomas von der Maßen. *Bericht des GI Arbeitskreises Requirements Engineering f¸r Produktlinien*. Klaus Schmid (Ed.).

[BJJ+01] Jan Bosch, Michel Jaring, Susanne Johnsson, Klaus Schmid, Steffen Thiel, Bernhard Thomé, and Siegfried Trosch; Klaus Schmid (Ed.). *Task 1.2: Domain analysis: Consortium-wide deliverable on scoping*. Deliverable of the ESAPS Project, Eureka Σ! 2023 Programme, ITEA Project 99005, 2001.

[Boe81] Barry W. Boehm. *Software Engineering Economics*. Prentice Hall PTR, 1981.

[Boe89] Barry W. Boehm. Risk management practices: The six basic steps. In

Copyright © Fraunhofer IESE 2003

Barry Boehm, editor, *Tutorial: Software Risk Management*, pages 115–147. IEEE Computer Society Press, 1989.

[BS96] Sergio Bandinelli and Goiuria Sagarduy. *A unifying framework for reuse economic models*. Technical Report ESI-1996-Reuse03, European Software Institute, 1996.

[Bug] Luigi Buglione. *Ict balanced scorecards*. http://www.geocities.com/lbu_measure/bsc/bsc.htm. As seen in November 2002.

[CGKS01] Paul Clements, Cristina Gacek, Peter Knauber, and Klaus Schmid. Successful software product line development in a small organization. In Paul Clements and Linda Northrop, editors, *Software Product Lines: Practices and Patterns*, Chapter 11. Addison Wesley Longman, 2001.

[Chu99] Sunita Chulani. COQUALMO (COnstructive QUALity MOdel) — a software defect density prediction model. In *Proceedings of the ESCOM — SCOPE'99*, pages 297–306, 1999.

[Cle01] Paul C. Clements. *On the importance of product line scoping*, 2001.

[CN01] Paul Clements and Linda Northrop. *Software Product Lines*. Addison–Wesley, 2001.

[Coh95] Lou Cohen. *Quality Function Deployment*. Addison–Wesley, 1995.

[Coh01] Sholom Cohen. Predicting when product line investment pays. In *Proceedings of the Second ICSE Workshop on Software Product Lines: Economics, Architectures, and Implications*, 2001.

[Dea] E. Dean. *Conjoint analysis from the perspective of competitive advantage*. Available online at http://akao.larc.nasa.gov/dfc/ppt/cja.html.

[DeM82] Tom DeMarco. *Controlling Software Projects*. Prentice Hall PTR, 1982.

[DK88] Gregory Dobson and Shlomo Kalish. Positioning and pricing a product line. *Marketing Science*, 7(2):107–125, 1988.

[DSF95] Department of Defense — Software Reuse Initiative, Version 3.1. *Domain Scoping Framework, Volume 2: Technical Description*, 1995.

[Dym96] Ken Dymond. *A Guide to the CMM — Understanding the Capability Maturity Model for Software*. Process Inc. US, fourth edition, 1996.

[EMT98] Michel Ezran, Maurizio Morisio, and Colin Tully. *Practical Software Reuse: The Essential Guide*. European Systems and Software Initiative (ESSI), 1998.

[Erd02] Hakan Erdogmus. A real options perspective of software reuse. In

Copyright © Fraunhofer IESE 2003

International Workshop on Reuse Economics (IWRE), 2002.

[Esq98] Manuel Eduardo Jaime Esqueda. *Product line streamlining: A methodology to guide product costing and decision-making*. Master's thesis, Sloan School of Management, June 1998.

[Fav96] John M. Favaro. A comparison of approaches to reuse investment analysis. In *Proceedings of the Fourth International Conference on Software Reuse*, pages 136–145, 1996.

[FF99] John M. Favaro and Kenneth R. Favaro. Strategic analysis of application framework investments. In M. Fayad and R. Johnson, editors, *Building Application Frameworks: Object-Oriented Foundations of Framework Design*, pages 565–597. John Wiley & Sons, 1999.

[FFF98] John M. Favaro, Kenneth R. Favaro, and Paul F. Favaro. Value based software reuse investment. *Annals of Software Engineering*, 5:5–52, 1998.

[FHR00] Stuart Faulk, Robert R. Harmon, and David M. Raffo. Value-based software engineering (VBSE): A value-driven approach to product-line engineering. In Patrick Donohoe, editor, *Software Product Lines: Experience and Research Directions, Proceedings of the First Software Product Lines Conference (SPLC1), August 28–31, 2000, Denver, Colorado, USA*, pages 205–223. Kluwer Academic Publishers, 2000.

[Fra92] Giovanni De Fraja. Product line competition and market structure. *Economic Notes*, 21(3):511–525, 1992.

[Fre00] Bernd Freimut. *Reliability prediction models for product lines - a brief overview*. Technical Report 046.00/E, Fraunhofer IESE, 2000.

[Gab97] Gabler Verlag. *Gabler Wirtschaftslexikon*, 14 edition, 1997.

[GK85] Paul E. Green and Abba M. Krieger. Models and heuristics for product line selection. *Marketing Science*, 4(1):1–19, 1985.

[HES+02] Warren Harrison, Hakan Erdogmus, Kevin J. Sullivan, Barry W. Boehm, and Don J. Reifer. *Software engineering economics: Background, current practices and future directions*. Tutorial 3 at the International Conference on Software Engineering, 2002.

[HSM00] Georg Herzwurm, Sixten Schockert, and Werner Mellis. *Joint Requirements Engineering*. Vieweg Verlag, 2000.

[ISO98] *ISO/IEC FCD 9126* "*Information Technology — Software Product Quality-Part 1: Quality Model*", 1998.

[KB01] Philip Kotler and Friedhelm Bliemel. *Marketing Management*. Schäffer-Poeschel, 2001.

[KLD02] Kyo C. Kang, Jaejoon Lee, and Patrick Donohoe. Feature-oriented

Copyright © Fraunhofer IESE 2003

product line engineering. *IEEE Software*, 19(4):58–65, 2002.

[KMR99] Marc I. Kellner, Raymond J. Madachy, and David M. Raffo. Software process simulation modeling: Why? what? how? *Journal of Systems and Software*, 46:91–105, 1999.

[KN96] Robert Kaplan and David Norton. *The Balanced Scorecard*. Harvard Business School Press, 1996.

[Kon01] Jyrki Kontio. *Software Engineering Risk Management*. PhD thesis, Helsinki University of Technology, 2001.

[KVC96] Vrinda Kadiyali, Naufel Vilassim, and Pradeep Chintagunta. Empirical analysis of competitive product line pricing decisions: Lead, follow, or move together? *Journal of Business*, 1996.

[Lim96] Wayne C. Lim. Reuse economics: A comparison of seventeen models and directions for future research. In *Proceedings of the Fourth International Conference on Software Reuse*, pages 41–50, 1996.

[Lim98] Wayne C. Lim. *Managing Software Reuse*. Prentice Hall PTR, 1998.

[MCGZ00] A. Mili, S. Fowler Chmiel, R. Gottumukkala, and L. Zhang. An integrated cost model for software reuse. In *Proceedings of the 22nd International Conference on Software Engineering*, 2000.

[MM94] Charlotte H. Mason and George R. Milne. An approach for identifying cannibalization within product line extensions and multi-brand strategies. *Journal of Business Research*, 31:163–170, 1994.

[MS02] Dirk Muthig and Klaus Schmid. Balancing evolution with revolution to optimize product line development. In *NATO/RTO Symposium on Technology for Evolutionary Software Development, Bonn, September 23-24*, 2002.

[MZ88] Richard D. McBride and Fred S. Zufryden. Integer programming approach to the optimal product line selection problem. *Marketing Science*, 7(2):126–140, 1988.

[Pfl96] Shary Pfleeger. Measuring reuse: A cautionary tale. *IEEE Software*, 13:118–127, 1996.

[Pin93] B. Joseph Pine. *Mass Customization — The New Frontier in Business Competition*. Harvard Business School Press, 1993.

[Por99] Michael E. Porter. *Wettbewerbsstrategie*. Campus, 1999.

[Pou97a] Jeffrey S. Poulin. *The economics of product line development*. http://home.stny.rr.com/jeffreypoulin/Papers/IJAST97/ijast97.html, 1997.

[Pou97b] Jeffrey S. Poulin. *Measuring Software Reuse*. Addison–Wesley, 1997.

[RCL99] Ioana Rus, James Collofello, and Peter Lakey. Software process sim-

Copyright © Fraunhofer IESE 2003

ulation for reliability management. *Journal of Systems and Software*, 46:173–182, 1999.

[Rei97] Donald J. Reifer. *Practical Software Reuse*. John Wiley & Sons, 1997.

[RFH00] David M. Raffo, Stuart Faulk, and Robert R. Harmon. A customer value-driven approach to product-line engineering. In Peter Knauber and Giancarlo Succi, editors, *First Workshop on Software Product Lines: Economics, Architectures, and Implications, Limerick*, pages 3–7, 2000.

[RU98] David Robertson and Karl Ulrich. Planning for product platforms. *Sloan Management Review*, 39(4):19–31, 1998.

[Sch00a] Klaus Schmid. *A framework for product line quality model development*. Technical Report 047.00/E, Fraunhofer IESE, 2000.

[Sch00b] Klaus Schmid. Scoping software product lines — an analysis of an emerging technology. In Patrick Donohoe, editor, *Software Product Lines: Experience and Research Directions; Proceedings of the First Software Product Line Conference (SPLC1)*, pages 513–532. Kluwer Academic Publishers, 2000.

[Sch01] Klaus Schmid. People issues in developing software product lines. In *Proceedings of the Second ICSE Workshop on Software Product Lines: Economics, Architectures, and Implications*, 2001.

[Sch02] Klaus Schmid. *Planning Software Reuse — A Disciplined Approach for Software Product Lines*. PhD thesis, University of Kaiserslautern, 2002.

[SGCH01] Kevin J. Sullivan, William G. Grisworld, Yuanfang Cai, and Ben Hallen. The structure and value of modularity in software design. In *Proceedings of the Joint International Conference on Software Engineering and ACM SIGSOFT Symposium on the Foundations of Software Engineering, September 2001, Vienna*, 2001.

[Sim00] Hermann Simon. *Das Grosse Handbuch der Strategiekonzepte*. Campus, 2000. In German.

[SJ01] Klaus Schmid and Isabel John. Product line development as a rational, strategic decision. In Klaus Schmid and Birgit Geppert, editors, *Proceedings of the International Workshop on Product Line Engineering in Early Steps: Planning, Modeling, and Managing (PLEES'01)*. Fraunhofer Institute for Experimental Software Engineering (IESE), 2001. IESE-Report 050.01/E.

[SM99] Harvey Siy and Audris Mockus. Measuring domain engineering effects on software change cost. In *Proceedings of the Sixth International Software Metrics Symposium*, 1999.

[Spr00] Klaus Spremann. *Portfoliomanagement*. Oldenbourg, 2000.

Copyright © Fraunhofer IESE 2003

[Ste] Ray Steele. *Homepage of the excel options plug-in*. http://pilot.msu.edu/user/steelera/faqs.html.

[Ste91] Jeroen Steenbrink. Product line strategy. *EIU International Motor Business*, pages 78–92, October 1991.

[SU97] Susan Walsh Sanderson and Mustafa Uzumeri. *The Innovation Imperative — Strategies for Managing Product Models and Families*. Irwin Professional Publishing, 1997.

[SV02] Klaus Schmid and Martin Verlage. The economic impact of product line adoption and evolution. *IEEE Software*, 19(6):50–57, July/August 2002.

[TCO00] Peter Toft, Derek Coleman, and Joni Ohta. A cooperative model for cross-divisional product development for a software product line. In Patrick Donohoe, editor, *Software Product Lines: Experience and Research Directions; Proceedings of the First Software Product Line Conference (SPLC1)*, pages 111–132. Kluwer Academic Publishers, 2000.

[vSB99] Rini van Solingen and Egon Berghout. *The Goal/Question/Metric Method: A practical guide for quality improvement of software development*. McGraw-Hill, 1999.

[Wil99a] Edward Wiles. *Economic models of software reuse: A survey, comparison and partial validation*. Technical Report UWA-DCS-99-032, Department of Computer Science, University of Wales, Aberystwyth, U.K., April 1999.

[Wil99b] Peter J. Williamson. Strategy as options on the future. *Sloan Management Review*, 40:117–126, 1999.

[Wit96] Jim Withey. *Investment analysis of software assets for product lines*. Technical Report CMU/SEI-96-TR-010, Software Engineering Institute, Carnegie Mellon University, 1996.

[WL99] David M. Weiss and Chi Tau Robert Lai. *Software Product-Line Engineering*. Addison–Wesley, 1999.

[WN89] Lynn O. Wilson and John A. Norton. Optimal entry timing for product line extensions. *Marketing Science*, 1989.

[Wym93] A. Wayne Wymore. *Model-Based Systems Engineering*. CRC Press, 1993.

[ZP00] Xuemei Zhang and Hoang Pham. An analysis of factors affecting software reliability. *Journal of Systems and Software*, 50:43–56, 2000.

Copyright © Fraunhofer IESE 2003

B Identification of Quality Drivers

The quality drivers (or in this case more appropriate: the effort drivers) have been identified from a number of sources. First of all, we need to differentiate between situations that can be mapped to the classical situation of single-system development. For this case we made use of existing collections of effort drivers.

Based on the different reuse situations we subdivided the description of the quality drivers into four main sections:

- *Standard software development* — these cost drivers are non-specific, they apply for any software that is developed anew from scratch as system-specific, but they will also apply in addition in all the other situations mentioned here.
- *Adaptive reuse* — these cost drivers are supposed to be relevant in the context of white-box reuse, i.e., in case we take existing assets and modify them in an appropriate manner.
- *Development for reuse* — an implementation is developed which is supposed to be reusable in a generic manner.
- *Development with reuse* — given an existing asset, reuse does not come for free. The factors impacting the cost of reusing are captured here.

B.1 General Cost Factors

In this section we describe some cost factors that are of general relevancy. As this matches the context of classical single-system development, we do not aim at providing a comprehensive list, rather we refer the reader to existing surveys and cost estimation approaches [Boe81, BEM+98]. Rather, we give a short list of factors that we found particularly relevant in the context of setting up a feature-based cost-estimation effort.

As these factors can easily be found in other literature, we provide only a list without further description.

1 Novelty of functional requirement

2 Complexity of hardware-software interface

3 Importance of performance requirements

Copyright © Fraunhofer IESE 2003

4 Complexity of data structures and data management operations

5 Complexity of device dependent operations

6 Complexity of control operations

7 Importance of software reliability requirements

8 Importance of security requirements

9 Extent of memory constraints

B.2 Adaptive Reuse Cost Drivers

In this section, we describe some cost factors that impact the cost of white box (i.e., adaptive) reuse. These factors where identified based on interviews.

Throughout this appendix we will refer to P' as the product that was the basis for the development of the asset and P will denote the product P that is the aim of reuse.

1 Match of functional requirements for the Feature F in P and P'.
Definition: the degree to which the functional requirements for feature F in P and P' differ.
Clarification: The larger this difference, the more rework is required to adapt F from P' to suite in P, the smaller the percental savings from implementing F from scratch.
Externally visible from the user's point of view.
Consideration: Ways in which such differences can be manifested include:
- no differences, identical interfaces
- minor differences such as small cosmetic changes for the output
- some additional functionality has to be implemented
- profound differences.

2 Match of architectural embedding of features for implementation of feature f in P and P'.
Definition: The degree to which paradigms and implementation strategies underlying the design and implementation of F in P' match those in P.
Clarification: The larger this difference, the more rework is required to adapt F from P' to fit for P, the smaller the percental savings.
Consideration: Examples of issues that constitute a mismatch include:
Different design programming paradigms (e.g., procedural vs. OO)
Different communication patterns: event-based vs. direct interaction, CORBA vs. DCOM

3 Quality of documentation: scope / extent / completeness of documentation.
Definition: The degree to which design / implementation of feature F is doc-

Copyright © Fraunhofer IESE 2003

umented.
Clarification: The smaller the scope/extent/completeness of documentation, the more work is required to complete the documentation, the smaller the percental changes.
Consideration: Benchmark to assess the quality of the documentation are the documentation guidelines for the target system. I.e., we consider how much of the documentation that will be required for Feature F in P is already available in P'.

4 Quality of documentation: Clarity / understandability of documentation
Definition: The ease with which the documentation of feature F can be understood.
Clarification: The higher the understandability, the less effort is required to judge to which degree F in P' needs to be modified to fit in P, the larger the percental savings.
The higher the understandability, the easier it is to identify in the design / implementation of P'.
Consideration:

5 Quality of documentation: Traceability of design to code
Definition: The ease of which artifacts in the design can be traced to corresponding artifacts in the code.
Clarification: The higher the traceability, the less effort is required to identify the implementation of F in P', the higher the percental savings.
Consideration:

6 Coupling to environment
Definition: The degree to which the implementation of F in P' references external resources.
Clarification: The more the implementation of F depends on external resources, the more difficult it is to isolate the implementation and port it to the new environment in P.
Highly coupled features are more fault-prone, hence will require higher effort for testing, and fixing faults.
Consideration: external references can occur in the form of usage of global variables (data interactions), invocation of functions / methods (functional dependencies).
Only external references that are relevant to the performance of F's functionality required in P need to be considered here.

7 Percentage of new code
Definition: The percentage of code of the implementation of F in P that needed to be added to what was recovered from F.
Clarification: The higher the percentage of code that needs to be added, the higher the effort to develop the code and test the code. Thus, the smaller is the percental savings as compared to directly implementing F in P.

Copyright © Fraunhofer IESE 2003

Consideration: New code may be due to new functional requirements, or it may stem from interface considerations (e.g., implementation of a wrapper).

8 Complexity
Definition: The data, control, and functional complexity of feature F. This impacts cognitive complexity imposed on the developer when analyzing/ designing/implementing/testing feature F.
Clarification: The more complex the Feature, the higher the percental savings that can be expected, as we can reuse a more "effort-loaded" asset, that we do not need to re-implement.
Consideration: Understandability, structural complexity;
Interactions: with degree of difference.

9 Re-testing policy and need
Definition: Is there an established policy when a reused and modified feature needs to be performed.
Clarification: Such a policy helps to decide if a feature F in product P is tested or not, which of course affects the percental savings that are higher if the feature need not be retested. (This holds of course only under the assumption that the policy is meaningful, i.e., retesting is only skipped if only few errrors have to be expected.
Consideration:

10 Availability and knowledge of modification techniques and tools
Definition: Availability / Knowledge of modification techniques and tools to support modification of F in P' so that it fits in F.
Clarification: The use of such tools (e.g, for refactoring) can reduce the effort required to perform the necessary changes. Such tools must 1) be available and 2) the developers need to know them and how to use them.
Consideration:

11 Trust in implementation of F in P'
Definition: the degree to which the developers are convinced that the implementation of F is reliable and efficient.
Clarification: The higher the confidence in the implementation, the less resistance will there be to reuse the implementation of feature F, less effort will be spent to look for other solutions. Also, it is likely that less effort will be spent on regression testing then.
Consideration: Questions such by whom was the feature implemented, or how extensively was it tested will play a role here.

12 Locality of implementation of F in P'
Definition: Whether the implementation of F in P' is one contiguous block, or whether it is scattered over various places.
Clarification: The more the implementation is scattered, the more effort it takes to cull these bits and pieces together, and intergate in P, The smaller are the savings vs. implementing F from scratch.
Consideration:

Copyright © Fraunhofer IESE 2003

13 Formality of requirements
Definition: The degree to which the functional requirements are formally expressed.
Clarification: The higher the degree of formality, the easier it is to assess the match of functionality of F in P and P', and the easier the required modifictations can be implemented. (The more formality, the easier the reuse).
Consideration: This factor is likely to be equal for all features

14 Match of quality (nonfunctional requirements) for F in P and P'.
Definition: The degree to which the nonfunctional requirements for feature F in P and P' differ in that P has stricter requirements.
Clarification: The larger this difference, the more rework of F is required, the smaller the effort savings from reusing F.
Consideration:

15 Availability of a domain model or domain model concepts.
Definition:
Clarification: Availability of a domain model helps during the requirements phase (match candidate). The larger this difference, the more rework of F is required, the smaller the effort savings from reusing F.
Consideration: This is not feature specific.

16 Domain-specifity of feature F.
Definition: The degree to which feature F is specific to the domain of product P / P'.
Clarification: The more domain-specific the feature is, the easier the feature is to reuse (people are likely to be familiar with concepts and ideas underlying the design/implementation of F).
Consideration: This is especially true when a domain model is available. (Interaction with previous factor!).

17 Availability of experts for the domain, the old system, and the new system.
Definition: Degree to which experts of the domain, the old system, the new system can be meaningfully consulted (quality and availability of their expertise).
Clarification: If experts are available, their expertise will speed up the reuse process and reduce the risk of errors being made (avoiding pitfalls in the first place, false assumptions etc.)
Consideration: Interaction with "Quality of documentation". If excellent documentation is available, additional availability of experts is not as beneficial as when there is no or poor documentation.

Copyright © Fraunhofer IESE 2003

B.3 Cost Drivers for Reusing

Even in case we already have a rather generic implementation available, costs may be incurred from reusing. The cost drivers in this section deal with this situation.

1 Adaptability of the generic implementation
Definition: The ease with which the generic implementation of F can be integrated in product P.
Clarification: The easier the reuse is possible, the higher the realtive savings.
Consideration: Ease of adaptability, can reach from "verbatim reuse" - i.e., reuse without any modification, to parameterized reuse (where only some parameters have to be selected), to implementing wrapper/glue code (in varying amounts).

2 Residual variability (fine requirements)
Definition: Any functional "fine-grained" requirements for F in P that are not captured by the generic implementation of F.
Clarification: The more requirements the generic implementation is missing, the more effort is needed to integrate F in P, the smaller are the percentual savings.
Consideration: This aims at functional aspects, whereas the first factors aims more at the different reuse mechanisms.

3 Degree of complexity and abstraction of the domain context
Definition: The level of abstractions, objects behind them, that need to be considered to make the transition from the generic to the specific case.
Clarification: The more complex the identification of necessary adaptations is (e.g., for developing glue code), the lower the percentual saving.
Consideration:

4 Understandability
Definition: Degree to which the contents of the reusable asset is understandable..
Clarification: The harder it is to understand what is "inside" the reusable asset, the more effort is required to integrate it in P, the smaller the percentual savings.
Consideration:

B.4 Cost Drivers for Development for Reuse

In general the cost of development for reuse is higher than the cost for single-system development. How much higher is impacted by various factors. Here we capture these factors:

Copyright © Fraunhofer IESE 2003

1 Availablity of domain information
Definition: The degree to which information about the domain is available
Clarification: The fewer domain information is available, the larger the additional percentual effort for implementing F generically is, as this information still has to be discovered/deducted.
Consideration: Two important sources of domain information that need to be considered:
- Existence of a standard model for the domain
- Availablity of domain experts

2 Added functionality in the generic case
Definition: The added functionality of feature F in the generic case as compared to the functionality of F in P.
Clarification: The higher the added functionality, the larger the additional percentual effort for implementing F generically, and for testing the added functionality.
Consideration: Two things need to be considered:
- the amount of added functionality (consider functional requirements across all products in which F is implemented)
- the complexity of the added functionality

3 Feature complexity
Definition: The data, control, and functional complexity of feature F. This impacts cognitive complexity imposed on the developer when analyzing/designing/implementing/testing feature F.
Clarification: The higher the complexity, the more fault-prone is the the implementation, the more faults have to be identified and corrected, the higher the effort spent on this acitivity.
Consideration:

4 Wrapping the feature
Definition: The effort required to wrap the feature so that it can be used easily in all products.
Clarification: The higher the effort to wrap the feature to make it generic, the higher the overhead vs just implementing F in one product.
Consideration: Wrapping up the feature involves isolation of the interface, in order to generate a clean API.

5 Reuse implementation method
Definition: The method that is used to implement F generically.
Clarification: This can come in many different forms from domain specific languages, to Frames, to components. The specific approach chosen will considerably change the underlying effort model.
Consideration:

6 Performance constraints
Definition: Constraints on the CPU usage and memory usage of the implementation of feature F.

Copyright © Fraunhofer IESE 2003

Clarification: The generic feature must match the performance demands of all products in which it will be integrated. Therefore, the most restrictive memory/CPU usage requirements across all products must be applied. The stricter these requirements are when compared to those for a feature F, the larger the additional effort for implementing F generically.
Consideration:

7 Connectivity of F ("Vernetzung")
Definition: Degree to which F interacts with other features
Clarification: The more a feature F is connected, the more difficult it is to implement F generically.
Consideration:

Copyright © Fraunhofer IESE 2003

Document Information

Title:	Integrated Cost- and Investmentmodels for Product Family Development
Date:	July 21, 2003
Report:	IESE-067.03/E
Status:	Final
Distribution:	Public

Copyright 2003, Fraunhofer IESE.
All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means including, without limitation, photocopying, recording, or otherwise, without the prior written permission of the publisher. Written permission is not needed if this publication is distributed for non-commercial purposes.