Economics of Software Product Lines

January 2004

CONVERGYS

TABLE OF CONTENTS

Author

Dale Peterson, Vice President Architectural Development, Convergys

Republished from PFE-5 © Springer-Verlag
http://www.springer.de/comp/lcns/index.html

Economics of Software Product Lines

Dale R. Peterson

Convergys Corporation
Cincinnati, OH 45202 USA
dale.peterson@convergys.com

Abstract. Software Product Lines (SPL) offer tremendous potential to improve a company's competitive position and responsiveness to changing market needs. However, unless the full benefits and costs are well understood by executive-level stakeholders, the probability of success will be low. An SPL economic analysis framework is presented that is based on the author's experience with SPL adoption at Convergys. While derived from the Convergys experience, it is hoped that this framework will be of value to other companies contemplating a transition to an SPL.

1 Introduction

The transition from traditional software development where products are developed independently from one another to an approach that is based on multiple development groups working in a collaborative fashion to deliver products to market requires a massive change in how things are done. The investment costs and risks, as well as the downstream benefits can be significant. Gaining executive stakeholder buy-in to making the investment requires a business case that translates the qualitative benefits so often cited for software product lines into concrete business benefits. The business case process can be an effective decision making tool for not only deciding *whether* an organization should transition to an SPL, but also *how* best to make the transition a success. An understanding of the size and timing of the cash flows will enable the organization to assess its transition strategy, and determine the degree to which it should adopt an incremental and iterative approach. This paper describes the approach that was used at Convergys[1] to assess the financial impact associated with its transition to an SPL. While it is based on the Convergys experience, it is hoped that it will be of value to others considering a similar transition.

This paper is organized as follows. Section 2 provides the context behind the Convergys initiative. This context should enable others to determine its potential applicability outside of the original setting. Section 3 contains a high level description of an heuristic model for quantifying the economic benefits. Section 4 provides a qualitative discussion of the major cost elements, and time and cost drivers associated with the transition project. Section 5 contains an

[1] Convergys (R) is a registered trademark of Convergys Corporation or its wholly owned subsidiaries.

example economic analysis that is based on a simple transition project scenario. Section 6 describes a set of metrics that could be used to track business case results against the economic model predictions. Section 7 contains a summary and set of conclusions. The Appendix contains a more detailed derivation of the benefits model.

2 Context

The scenario that this framework is based on is as follows.

Existing Product Family There is a family of products already in existence. While there may be an opportunity to deliver new products in the future, the focus is on improving the way existing products are developed and delivered.

Independent Product Development The starting point for the transition is a set of products that are independently developed by different organizations that utilize a diverse set of platforms, technologies, development processes, and product architectures.

Large, Complex, Mission-Critical Systems The products are large, complex systems with footprints that span multiple application domains. The systems must meet stringent requirements for performance, scalability, availability, and operational cost.

Preserve Existing Software Assets There continues to be a significant investment in the existing products. The time and cost to build a product line based on new components is prohibitive and/or the risks are too high. Therefore, the decision is made to mine the existing asset base, possibly with new development in selected (and small) domains.

Component-Based Product Line Architecture The strategy is to adopt a component-based product line architecture, consisting of a common set of platform technologies, middleware components/frameworks, and business-level application components.

Very Large-Grained Components Because the strategy is to mine existing assets, the approach is to start with very large-grained components, and to subsequently break down the large components into smaller components.

Component Factory The vision for the software factory is a set of component groups that act as suppliers to the product groups.

Figure 1 shows a conceptual model of the envisioned software factory. Individual products serve specific vertical markets. The market needs are translated into requirements against the various products, and new features are allocated by a common architecture group to the different components (in general, a given feature will span multiple components). The common component groups deliver new releases to the product groups, which then extend, adapt, configure, and integrate the components to deliver new product releases.

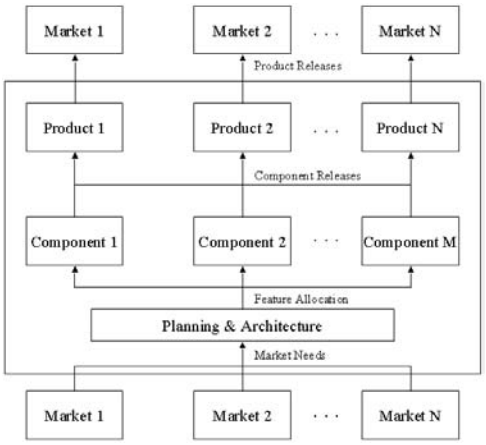

Fig. 1. Conceptual model of the component-based software factory.

3 Financial Flexibility

The benefits that motivated the SPL approach at Convergys are:

R&D Investment The ability to leverage the R&D investment across the product family by reusing a common set of components.

Subject Matter Expertise The ability to leverage subject matter experts across the product family by concentrating domain subject matter experts with similar skills and knowledge into centralized groups that serve all products.

Productivity and Quality Improving productivity and quality by breaking large, monolithic applications into smaller, more manageable projects, and by utilizing components that encapsulate their functionality.

Time to Market Increasing the rate of delivery of new capabilities to market and enabling new products to be delivered faster by reusing well established components.

People Mobility Provide employees with more career development opportunities by standardizing on the development environment and processes, thereby reducing the learning curve associated with a move to a new project.

Supplier Relationships Standardizing on the platforms and development environment enables a more effective leverage of supplier relationships.

Geographic Flexibility Standardizing on component-based development facilitates the distribution of development responsibilities across locations.

Sourcing Flexibility A modular architecture enables greater flexibility to build, license, or acquire software.

Product Refresh A modular architecture facilitates the process of refreshing the product family as new technologies and/or software components become available.

Given the anticipated size of the investment, it was necessary to go beyond the qualitative arguments listed above, and to develop a quantitative statement of the benefits. The approach taken was to focus on the benefits associated with productivity improvement and leverage associated with establishing a common set of components upon which members of the product family would be based. Over the years, numerous software reuse economic models have been developed (cf. [1–3]). Most of the models focus on the benefits associated with cost reduction or cost avoidance. Others have taken a more strategic view of product line economics in the context of a company's overall business strategy and market goals[4]. The motivation for developing yet another model was the desire to relate the concepts of commonality and variability, which are fundamental to software product lines, more directly to the economic benefits.

To motivate the benefits model, consider Fig. 2 below. It depicts the set of

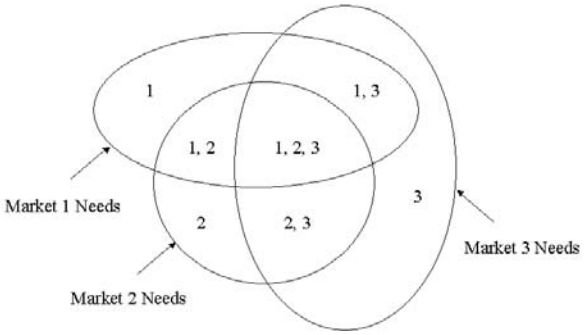

Fig. 2. A Venn diagram illustrating market needs overlap for a family of three products.

needs for three distinct markets. In general, there will be needs that are unique to each market. However, there will also be an overlap between those needs. Some needs will be common to two of the three markets, while other needs will be common to all three markets. The question then becomes, is there an architecture that enables common components to be built that incorporate the common needs, but also allow for variations and extensions to accommodate the unique needs? Based on an understanding of the requirements commonalties and variabilities, as well as on the component architecture, it should be possible to quantify how much of the development work will be performed by the component groups versus the product groups.

The Appendix describes a model that quantifies the benefits associated with improved productivity and the use of common components, based on the con-

cept of overlapping market needs. The first version of the model considers the people savings associated with eliminating redundant software development and increasing productivity. The result is (for a single component):

$$\Delta S = \left[1 - \frac{1}{(1+\delta p)\,(1+(N\lambda-1)\,\omega)}\right]\tilde{S} \qquad \text{(Reduce Cost Version)} \quad (1)$$

The notation is as follows. First, $\tilde{S}$ stands for the number of developers required in the case of independent development (no reuse, no productivity gain). Next, N stands for the number of products using the common component. The parameter δp stands for the *relative* change (positive or negative) in productivity (for example a value of zero means no improvement, while a value of 1/4 means a 25% increase). The parameter ω quantifies the *commonality* among the product requirements. In terms of Fig. 2, a commonality of one would mean a totally overlapping set of requirements, while a commonality of zero would mean the set of requirements for the different markets would be completely disjoint from one another. Finally, the parameter λ is a measure of the *leverage* that the product groups gain from the work performed by the component groups. The reason for this parameter entering the model is that not all features developed by the component groups will benefit all of the product groups.

Note that the cost to reuse a component is not explicit in the above equation. The cost of reuse in this model enters in two ways. First, the productivity change, δp may be negative. This would be the case if the productivity improvement associated with the component approach is offset by the resulting overhead due to the component groups building for reuse rather than a single product, as well as the management, governance, and coordination overhead that accompanies the collaborative development paradigm. Second, the costs associated with establishing a common set of reusable components enters by way of the investment needed to reengineer the existing product family, which is the subject of the transition cost model.

This version of the model corresponds to the "decrease the development effort per product" hypothesis discussed in [5].

We note that for the special case where $\delta p = 0$, $\omega = \lambda = 1$, this equation reverts to the simple form:

$$\Delta S = \frac{(N-1)}{N}\tilde{S} \quad (2)$$

The second version of the model looks at the benefits associated with increasing the throughput of the software factory (i.e. by doing more work with the same number of people):

$$\Delta\tilde{S} = \left[(1+\delta p)\,(1+(N\lambda-1)\,\omega) - 1\right]\tilde{S} \qquad \text{(Increase Throughput Version)} \quad (3)$$

This expression is derived by allocating the staff savings from equation (1) back to the component and product groups, and comparing the resulting throughput with the baseline (independent development) case. The expression above then is a statement of how many additional people would be needed in the baseline case

to obtain the equivalent level of throughput as is realized by the new software factory. The figure below illustrates the relative benefits associated with the two versions of the model. Intuitively, increasing throughput should result in a more competitive set of products, which in turn should drive revenue growth and market share. However, we have taken the more conservative approach to quantifying the benefits. This version of the model corresponds to the "develop more features with a given amount of money" hypothesis discussed in [5]. In

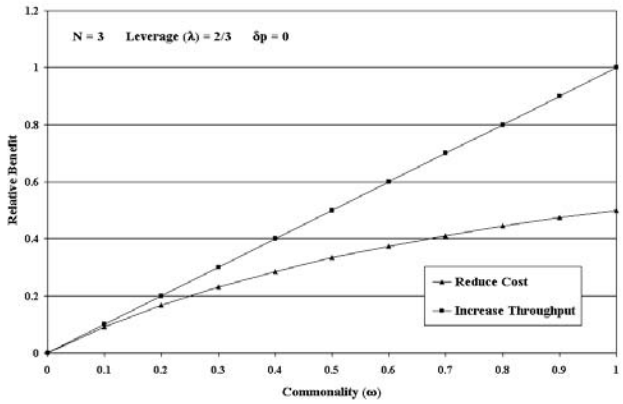

Fig. 3. Graph showing relative cost avoidance benefit for selected parameters.

general, the savings associated with increasing throughput are greater than the savings associated with staff reductions.

To summarize, the model suggests that the adoption of a common set of components provides *financial flexibility*[2], i.e. the ability to reduce investment while maintaining the same rate of delivery of new capabilities to market; or the ability to increase that rate of delivery without increasing investment levels. Combinations of the two approaches are also possible.

4 Transition Time and Cost

For many organizations, transitioning to an SPL is a complex and investment-intensive undertaking (cf. [7–13]). The complexity derives from the many, simultaneous changes that must occur to the organization, processes, architecture, and software assets associated with the product family. These changes must be coordinated in such a way as to minimize the disruption to ongoing product development projects.

This section contains a qualitative discussion of the types of costs that the Convergys business case had to account for, along with a discussion of the major

[2] See reference [6] for another view of flexibility.

factors that we have seen drive the size of the effort, and their *duration*. Time is important because the business case in general will be sensitive to the relative timing of the costs and benefits. We begin with a high-level description of the major cost elements. Next, we outline some of the key factors that have driven the transition cost level and time duration. Finally, we show the mapping between the cost elements and the drivers.

4.1 Cost Elements

The cost elements are grouped into four categories: architecture, process, organization, and implementation.

Architecture This category covers all costs associated with establishing a target product line architecture, technology standards, and an architectural evolution strategy and plan.

Domain Analysis Domain analysis is concerned with understanding and documenting the requirements commonalties and variabilities. This is a key input to defining the target architecture.

Target Architecture The target architecture specifies the common component boundaries, scope, interfaces, and variation points.

Technology Standards A common set of technology standards must be agreed on that deal with operating systems, programming languages, data base management systems, graphical user interface technologies and user interface look and feel standards.

Current Architecture In order to establish realistic architecture migration plans, an understanding of the current architectures employed across the product family are needed.

Migration Strategy and Plan The architecture migration strategy and plan is the roadmap for evolving the existing product family to the target architecture. It addresses how and when architectural modifications will occur across all products and domains.

Process This category covers all costs associated with establishing a target process architecture and set of process standards, as well as establishing a process migration strategy and plan.

Target Process The transition may require a major overhaul to existing processes. Critical areas to examine include the software release planning, specification, and commitments process (a given product group will depend on the timely delivery of new component releases from multiple components groups, and the component groups will find themselves having to meet the needs of multiple product groups); capacity planning and

management (allocating staff between the product groups and component groups is a complex production scheduling problem); software support (defining the technical support tiers and support service level agreements between component and product groups); and technology coordination (coordinating the upgrade to third party software, such as operating system upgrades, new releases of data base management systems, etc.).

Current Process An understanding of the suite of existing processes across the organization is needed to develop realistic process migration strategies and plans.

Process Standards Standards that must be agreed to by the various component and product groups to ensure the smooth operation of the software factory (for example, component documentation and packaging standards, establishment of a common set of artifacts, especially artifacts that cross component-product boundaries, such as requirements specifications, and project status tracking and reporting).

Tools Standards Establishment of development environment standards, such as requirements and design documentation, testing, and configuration management.

Metrics Establishment of a common set of process metrics.

Technical Governance The establishment of a set of processes and infrastructure to ensure conformance to the architecture, technology, and process standards.

Migration Strategy and Plan Establishment of a process migration strategy and plan that addresses how and when the new processes will be deployed within the organization.

Organization This category covers all costs associated with organizational architecture, training, change management, and overall transition planning and management.

Strategic Planning Development, communication, and enrollment in the SPL vision and umbrella transition strategy.

Program Management Managing the inter-organizational and inter-project dependencies during the transitioning effort, communicating status, progress and jeopardies to the major stakeholders.

Change Management Focused on enrolling the organization in the need for change, fostering awareness, communicating success, and working organizational change management issues throughout the transition.

Training Curriculum Curriculum development that addresses critical skills gaps within the organization, such as domain analysis, component-based development methods, and process design, as well as curriculum development associated with the new software development processes and process standards.

Organizational Design Mapping the target process architecture into an organizational structure.

Migration Strategy and Plan Developing the overall organization migration strategy and plan.

Implementation This category covers all costs associated with implementing the strategies and plans for transforming the organization, process, and software assets to conform to the target architecture.

Components Instantiation of the components defined in the target component architecture through a combination of reengineering, mining, new development, and licensing/acquisition.

Products Instantiation of members of the product line based on the target architecture and common components. This work involves some combination of reengineering existing products to use the common components, and/or retiring existing products and building new ones from the common components.

Process Implementing the new processes, conducting process pilots, training the organization on the new processes, process standards, and tools.

Organization Restructuring the organization to operationalize the new software factory.

4.2 Cost Drivers

The factors that we have seen influence costs are as follows.

NP - Number of Products The number of products within the existing family will have an impact on all four of the cost categories.

NC - Number of Components The number of components that are within the scope of the product line have an impact on the architecture and implementation categories.

PC - Number of Product-Component Combinations There are some elements that are sensitive to the number of product-component combinations.

AQ - Architectural Quality Adherence of the existing software assets to such architectural qualities as modularity, maintainability, etc. can have a major impact on the architecture and implementation efforts.

TS - Technology Standardization Level The level of technology standardization across the existing products (the number and variety of platforms, programming languages, use of third party software, adherence to industry standards, etc.) can have a major impact on the architecture and implementation costs.

PS - Process Standardization Level The level of software process standardization will impact process and implementation costs.

OS - Organizational Size The size of the development organization will impact the process, organization, and implementation costs.

DP - Software Development Practice There are a number of software development practice areas that can have a major impact on the effort to architect the product line. In particular, the existence of good requirements and architecture documentation practices will greatly facilitate the domain analysis, current architecture, and architecture migration planning efforts. Lack of good practice in these areas may require that these artifacts be re-constructed as a precursor to making forward progress.

4.3 Time Drivers

In general, the cost drivers discussed above also have a direct impact on the time its takes to transition to an SPL. In addition to these drivers, there are drivers that, while not having a major impact on the cost, nevertheless can have a dramatic impact on the time it takes to effect the transition. Since the business case depends critically on the timing of the cash flows, these drivers should be factored into the SPL timeline as much as possible. The factors that we have seen influence the transition timeline are as follows:

SM - Subject Matter Expertise Availability of subject matter experts (people with specific skills, such as architects, or people with specific domain knowledge) is a major factor across all categories. It is insufficient to have the investment dollars. Subject matter experts will be needed in the architecture, process, organizational, and implementation areas. However, subject matter experts are typically a scarce resource that are needed to support the existing product development projects. The inability to assign these resources to the SPL transition can have a negative impact on the rate of progress.

RD - R&D Constraints The company will have to decide between the level of investment in the SPL versus investing in its existing business. This issue tends to surface when the transitioning effort becomes investment intensive, i.e., during the middle phases of the project.

CL - Culture The SPL approach requires the organization to establish a common set of cultural values and a common way of doing things. Organizations that have a tradition of independence can find it difficult to let go of the established norms, and resist change on any kind. Thus, lack of a culture that is conducive to the SPL paradigm will have a major impact across all categories.

LE - Lexicon Organizations that have worked independently from one another tend to establish their own terminology to describe concepts relevant to their application. Thus different terms are used by different groups to describe the same concept. When these groups attempt to work together to conduct, for example, a domain analysis, a communications barrier may exist due to a single term having multiple conflicting interpretations, or due to multiple terms that refer to the same concept.

GE - Geography A highly distributed development organization presents a number of challenges that are not unique to SPL, yet need to be factored

into the transition timeline. In particular, communications and coordination can be a major challenge if the organization is highly distributed.

LC - Learning Curve The SPL approach requires a new set of skills not typically found in the development community, such as domain oriented software engineering, component-based development, and formal architecture methods and practices. The lack of experience in these areas can have a major impact on the architecture and implementation timelines.

4.4 Element to Driver Mapping

The table below summarizes the mapping between the various cost elements and time and cost drivers.

Cost Element → Mapping														
Category/Element	**NP**	**NC**	**PC**	**AQ**	**TS**	**PS**	**OS**	**DP**	**SM**	**RD**	**CL**	**LE**	**GE**	**LC**
Architecture														
Domain Analysis		x						x	x		x	x		x
Current Architecture	x	x	x	x	x			x	x				x	
Target Architecture	x	x	x	x	x				x		x	x	x	x
Technology Standards					x				x		x			
Migration Plan	x	x	x	x	x				x					
Process														
Current Process	x					x		x	x			x		
Target Process	x					x			x		x	x	x	x
Process Standards						x			x		x		x	
Tools Standards									x		x		x	
Metrics									x		x			
Technical Governance					x	x	x				x		x	
Migration Plan	x					x	x		x				x	
Organization														
Strategic Planning	x	x	x		x	x	x		x				x	
Program Management	x	x	x				x						x	
Change Management	x	x	x	x	x	x	x	x	x				x	
Training Curriculum					x	x	x		x				x	x
Organizational Design	x	x					x		x		x		x	x
Migration Plan	x	x					x		x		x		x	
Implementation														
Components		x		x	x				x	x			x	x
Products	x			x	x				x	x		x		x
Process	x					x	x			x	x		x	x
Organization	x	x					x			x	x		x	x

Weighting factors have not been applied in the table above, therefore not all "x's" are of equal importance. The major cost drivers are architectural quality, technology standardization level, organization size, and the number of components. As can be seen from the table, availability of Subject Matter Experts (SME's) is a major driver of transition time. An important benefit of the SPL approach is that it alleviates the problem associated with a limited supply of SME's, however, during the transition period, the transition project actually worsens the

problem since it results in an additional drain on these limited resources. In addition, R&D constraints have a major impact during the implementation phase, since the investment requirements become most intensive during this period. As with subject matter experts, the SPL goal is to lessen the R&D burden, however, the investment requirement during the implementation phase has the opposite effect.

5 Economic Analysis

The derivation of economic measures such as Net Present Value, Internal Rate of Return, and Payback Period involves modeling the cash flows associated with the investment[1]. The cash flows are based on the benefits as discussed in Section 3, while the costs are those discussed in Section 4. In this section, we present a simple transition scenario in order to illustrate the application of the benefits model. While simplistic and idealized, it nevertheless provides insight into the types of issues that the business case must address.

We start with some assumptions that simplify the analysis. First, we assume that the number of products is constant over the planning period. Further, we assume that the loaded cost per person per year is fixed. Finally, we assume that the investment is additive to the existing investment, and that people with the necessary skills can be added to work the transition without impacting the existing product development efforts. In what follows, we will use the more conservative approach to quantifying benefits (constant throughput, cost avoidance in terms of staff reductions).

We divide the business case planning period into four sequential phases. The first phase ("Ramp") involves a linear ramp up of staff to work the SPL transition. This phase is all costs - there are no benefits. The duration of this phase will depend on how quickly people with the right skill set can be added to the project. The next phase ("Engineering") is where most of the transition effort occurs. We assume a constant staffing level during this phase. No benefits accrue during this phase - only costs. The third phase ("Deployment") involves the successive introduction of the reusable components into the product line, and the completion of the SPL transition. During this phase, costs decrease from their maximum run rate at the beginning of the phase to zero at the end. On the other hand, the benefit run rate progressively increases from zero at the beginning of the phase to its maximum value at the end. The last phase ("Steady-State") begins with the completion of the SPL transition and runs until the end of the planning period. During this phase, there are no transition costs, and the benefit run rate is constant. Figure 2 shows the different phases along the planning timeline. We assume a five year planning horizon. We consider the scenario summarized in the table below.

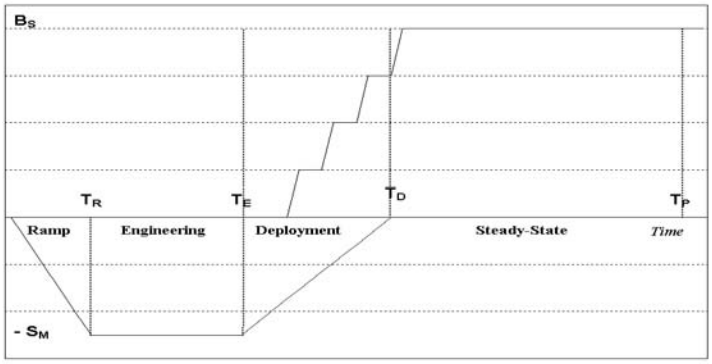

Fig. 4. Planning timeline and milestones.

Model Example - Baseline Parameters		
Parameter	**Value**	**Units**
Number of Products	4	-
Total Independent Staff Level	100	People
Loaded Cost	10,000	$ Per Person-Month
Planning Horizon	60	Months
Total Transition Effort	600	Person-Months
Maximum Transition Staff	25	People
Ramp-Up Time	6	Months
Component Deployment Interval	3	Months Per Product
Cost of Capital	12%	Per Year
Commonality	2/3	-
Leverage	3/4	-
Productivity Improvement	1/4	-
Independent Investment	12	($1,000,000) Per Year
Total Transition Cost	6	($1,000,000)
Engineering Complete (T_E)	21	Months
Deployment Complete (T_D)	33	Months

The economic measures are summarized below.

Model Example - Baseline Measures		
Measure	**Value**	**Units**
Net Present Value	7.7	($1,000,000)
Internal Rate of Return	64	% Per Year
Payback Period	41	Months

The Net Present Value and Internal Rate of Return are satisfactory. However, the Payback Period of 41 months suggests a high risk associated with the investment.

We next consider the sensitivity of the results to variations in the key input parameters. The table below summarizes the sensitivities for selected variations. The parameter being changed, along with its new value are shown in the first

two columns. The remaining columns show the new values of the measures as well as their change from the baseline case. The top half of the table shows downside scenarios (more pessimistic assumptions), while the bottom half shows the upside scenarios (more optimistic assumptions).

Model Example - Sensitivity Analysis							
Parameter	**Value**	**NPV**	**ΔNPV**	$\mathbf{i_{IROR}}$	$\mathbf{\Delta i_{IROR}}$	$\mathbf{T_{PB}}$	$\mathbf{\Delta T_{PB}}$
Total Effort	750	3.7	-4.0	34.8	-29.0	51	10
Maximum Staff	20	5.0	-2.7	47.3	-16.6	48	7
Ramp-Up Time	12	6.4	-1.3	59.2	-4.6	45	4
Commonality	1/2	6.5	-1.2	59.8	-7.1	43	2
Leverage	1/2	4.1	-3.6	40.9	-22.9	47	6
Productivity Improvement	0	5.8	-1.9	52.5	-11.3	44	3
Total Effort	450	12.0	4.3	110.0	46.1	32	-9
Maximum Staff	30	9.7	1.9	76.1	12.3	37	-4
Ramp-Up Time	3	8.2	0.4	63.8	0	41	0
Commonality	3/4	8.2	.0.5	66.7	2.9	41	0
Leverage	7/8	8.9	1.2	70.8	6.9	40	-1
Productivity Improvement	1/2	8.5	0.8	68.0	4.2	41	0

Not surprisingly, the results are highly sensitive to the total transition effort. An increase of 25% in the total effort yields marginal results with a very long (four year) payback period. Note that, even with a 25% reduction in the total effort, the payback period is still 32 months. Changes to the staffing level also result in significant changes to the results. The baseline scenario assumed a staffing level equal to 25% of the total size of the development organization (a rather optimistic assumption). A decrease of 25% in the maximum transition staffing level versus a 25% increase results in a swing of almost $5,000,000 in the net present value, and an eleven month swing in the payback period. This result must be kept in mind in the business case process, since diversion of skilled and knowledgeable developers from existing product development efforts to the transitioning effort is problematic at best - the transition projects are best carried out by people who have an in-depth understanding of the existing code base. Unfortunately, these are also the same people who are in greatest demand to support existing projects. The results are also sensitive to the productivity improvement, commonality, and leverage parameters. Since the entire business case rests on a reasonably accurate estimate of these parameters, the impact of over estimating them should be closely examined to determine how robust the business case is against estimation error.

In addition to examining the effects of changing a single parameter, we have developed a practice of defining "scenarios" by which groups of parameters are changed simultaneously to see their combined impact on the measures.

In summary, the results indicate a long pay-back period, even though the net present value and internal rate of return are reasonable in the baseline scenario. Of course, the major drivers behind the pay-back period are the total transition effort and staffing profile. In order to mitigate the risks associated with the

investment, a different strategy may be in order. The transition strategy that this example is based on can be characterized as the "big bang" approach, since all of the costs are incurred up front, with the benefits following during the later phases. This suggests the need to look for alternative strategies that spread the costs out over time, and that deliver smaller, more incremental benefits earlier. This type of strategy will result in a much lowered investment risk, which will facilitate gaining stakeholder buy-in to the project. On the down side, an incremental approach could result in a very long investment cycle, along with compromises to the product line architecture and associated software assets. Further, the incremental approach can be more costly in the long term, since multiple iterations are needed, with each iteration requiring re-integration and certification of the deliverables. There is no clear-cut strategy that is optimal in all cases. However, the above example does stress the need to carefully examine the effect that the strategy will have on the cash flows and end results.

6 Metrics

Critical to success is a set of metrics that allow the organization to track progress against the predictions documented in the business case. By providing a feedback mechanism, the business case can be updated based on positive and negative variances from the anticipated results. The benefits model is based on the concept of "cost avoidance." Unfortunately, it is not feasible to directly measure whether or not costs were avoided without conducting some sort of experiment with a control group that continues to develop the old way, and comparing the control group results with the SPL results. Furthermore, business conditions are constantly changing, R&D investment levels fluctuate from year to year, and new products are brought to market while others retire, etc. Finally, the company may have other initiatives underway that target areas that overlap with the transition, such as quality and productivity improvement projects. However, there are some metrics that follow directly from the economic model that allow for an indirect measure of the economic impact. The metrics are summarized in the table below. In addition to these metrics, standard quality metrics such as defect density and defect removal effectiveness are essential, however, since these metrics are not unique to the our approach, they are not considered here.

Recommended SPL Economic Metrics		
Metric	**Units**	**Scope**
Staff	-	Product-Component
Throughput	FP/Year	Product
Throughput	FP/Year	Component
Productivity	FP/Person-Year	Product
Productivity	FP/Person-Year	Component
Commonality	Percentage	Component
Commonality	Percentage	Product-Component
Leverage	Percentage	Component
Leverage	Percentage	Product-Component

The implementation of the commonality and leverage metrics requires a measurement program that properly accounts for the demand across all product - com-

ponent combinations. In particular, the front end of the development process will need to be modified to include a categorization of new enhancement requests according to which products and components are impacted. Function point counts need to be tracked for component-product pair. The Convergys approach is centered on large-scale components, therefore, the number of component-product combinations is manageable.

7 Conclusions

The stakes associated with a transition to an SPL are high - the investment requirements and risks, as well as the downstream benefits can be significant. Therefore it is essential that a business case be developed that properly accounts for the costs, benefits, and risks. A good understanding of the size and timing of the cash flows will enable the organization to assess its transition strategy, and determine the degree to which it should adopt an incremental and iterative approach. An understanding of the cost elements, and the cost and time drivers is an important input to this process. During the early phases, the business case will by necessity be at a high level. As the transition progresses, the strategies, plans, costs, and benefits will become more specific and accurate. Therefore, the business case should be updated on a regular basis to incorporate new information and feedback in terms of actual versus predicted results. A good set of metrics and associated measurement program will be needed to implement the feedback loop. In summary, the business case can be an effective tool for not only helping an organization decide *whether* to transition to an SPL, but also *how* to make the transition in such a way as to maximize return and minimize risk.

Going forward, the SPL benefits model requires validation. Additional work is needed in defining a standard measurement process for the metrics that have been recommended.

In closing, it should be stressed that there are additional considerations beyond pure economics that must be taken into account before the decision is made to transition to an SPL. The most important consideration is that the SPL approach requires *massive change* to the organization. In particular, the culture typified by the Independent scenario, where each development organization controls all of the resources, and is used to making decisions totally optimized for their particular product, is not conducive to an approach where development groups must depend on one another for their success. Organizations usually resist change of any significance. An effective awareness and enrollment program is needed to address stakeholder concerns, and to articulate why the change is needed. A clear vision and practical implementation strategy for achieving the vision are essential to addressing the SPL skeptics. Therefore, effective change management, organizational leadership, and *persistence* are critical to success.

References

1. J. Poulin. Measuring Software Reuse: Principles, Practices, and Economic Models. Addison-Wesley, 1997.
2. J. Gaffney and R. Cruickshank. A General Economics Model of Software Reuse. Proceedings of the International Conference on Software Engineering, Melbourne Australia. ACM, 1992.
3. R. Malan and K. Wentzel. Economics of Software Reuse Revisited. Hewlett-Packard Technical Report HPL-93-31, 1993.
4. K. Schmid. Integrated Cost- and Investmentmodels for Product Family Development. IESE-Report No. 067.03/E Version 1.0, 2003.
5. P. Knauber, J. Bermejo, G. Böckle, J. Sampaio do Prado Leite, F. Van der Linden, L. Northrop, M. Stark, and D. Weiss. Quantifying Product Line Benefits. Software Product-Family Engineering, 4th International Workshop, PFE 2001, Bilbao Spain, October, 2001.
6. J. Favaro, K. Favaro, P. Favaro. Value Based Software Reuse Investment. Annals of Software Engineering 5, 1998.
7. P. Clemens and L. Northrop. Software Product Lines: Practices and Patterns, Addison-Wesley, 2001.
8. I. Jacobson, M. Griss, and P. Jonsson. Software Reuse: Architecture, Process and Organization for Business Success, ACM Press, 1997.
9. G. Böckle, J. Muñoz, P. Knauber, C. Krueger, J. Sampaio do Prado Leite, F. van der Linden, L. Northrop, M. Stark, and D. Weiss. Adopting and Institutionalizing a Product Line Culture. Software Product Lines: Second International Conference, SPLC 2, San Diego, CA, USA, August 2002 Proceedings. Springer-Verlag, 2002.
10. D. Fafchamps. Organizational factors and reuse. IEEE Software, September, 1994.
11. J. Sametinger. Software Engineering with Reusable Components. Springer-Verlag, 1997.
12. T. Jolley, D. Kasik, and C. Kimball. Governance Polarities of Internal Product Lines. Software Product Lines: Second International Conference, SPLC 2, San Diego, CA, USA, August 2002 Proceedings. Springer-Verlag, 2002.
13. J. Bosch. Maturity and Evolution in Software Product Lines: Approaches, Artefacts and Organization. Software Product Lines: Second International Conference, SPLC 2, San Diego, CA, USA, August 2002 Proceedings. Springer-Verlag, 2002.

A Model Concepts

The economic model is based on the comparison of software product development costs in two scenarios. The first scenario ("Independent") is based on a family of products developed independently from one another. In this scenario, each product has its own dedicated funding source and development organization. This scenario represents the baseline "business as usual" case for economic comparison purposes. The second scenario ("SPL") assumes that assets common to multiple products are developed and supported by a "component factory," which delivers new component[3] releases to the product groups. The product groups are responsible for integrating the common components, and adapting/extending the

[3] We use the term "component" to mean a set of software programs that encapsualtes it's functionality, and has well-defined and documented interfaces that define

common component functionality to deliver new products/product releases that meet the needs of specific vertical markets.

A.1 Requirements Demand

We begin by considering the requirements (both functional and technical) placed on a family of products. Let N denote the number of products in the product family. For now, we confine ourselves to a single application "domain." The extension of the model to multiple domains will be discussed below. We confine ourselves to some planning period of length T_P. In what follows, the index k will label products (k = 1, 2, ..., N). Let $\mathfrak{R}_k$ represent the set of requirements associated with product k. The set of requirements for the entire product family is the union of the requirements for the individual products:

$$\mathfrak{R} = \bigcup_k \mathfrak{R}_k \tag{1}$$

We partition the set $\mathfrak{R}$ into disjoint subsets in such a way that all requirements in a given subset apply to the same set of products. Let $\{\boldsymbol{\pi}\}$ denote the set of N - dimensional vectors that satisfy: $|\boldsymbol{\pi}| > 0$; $\boldsymbol{\pi}_k = 0$ or 1. Then a partition, $\mathfrak{R}_{\boldsymbol{\pi}} \subseteq \mathfrak{R}$, is defined as:

$$\mathfrak{R}_{\boldsymbol{\pi}} \equiv \left(\bigcap_{\{k|\pi_k=1\}} \mathfrak{R}_k \right) \bigcap \left(\bigcup_{\{k|\pi_k=0\}} \mathfrak{R}_k \right)^C \tag{2}$$

where the complement is defined relative to $\mathfrak{R}$. In general, there will be $2^N - 1$ such partitions. Fig. 1 below illustrates the case for $N = 3$. In this example, $\mathfrak{R}_{100}$ represents the requirements unique to the first product, whereas $\mathfrak{R}_{110}$ represents the set of requirements shared by products 1 and 2, but not shared by product 3. In general, a "1" appearing in the k^{th} position means the k^{th} product is a member of the partition, whereas a value of "0" indicates the product is not a member of the partition.

We define the partition "weight" as the number of products that are members of the partition: $n(\boldsymbol{\pi}) \equiv \sum_k \boldsymbol{\pi}_k$. Thus, in Fig. 1, the partitions ($\mathfrak{R}_{100}$, $\mathfrak{R}_{010}$, $\mathfrak{R}_{001}$) have weight $n(\boldsymbol{\pi}) = 1$, ($\mathfrak{R}_{110}$, $\mathfrak{R}_{101}$, $\mathfrak{R}_{011}$) have weight $n(\boldsymbol{\pi}) = 2$, and the partition ($\mathfrak{R}_{111}$) possesses a weight of $n(\boldsymbol{\pi}) = 3$. For a given N, the number of partitions with a given weight, n, is given by the binomial coefficient, $\binom{N}{n}$. Application domains for which there is significant overlap in requirements are good candidates for the SPL approach. However, in order to take advantage of these overlapping requirements, an architecture is needed that capitalizes on the commonalties, accomodates the variabilities, *and* still provides sufficient

services provided and services expected. There is no implicit assumption regarding component size. Components can be small-grained, or very large-grained. Further, there is no assumption regarding how components are implemented.

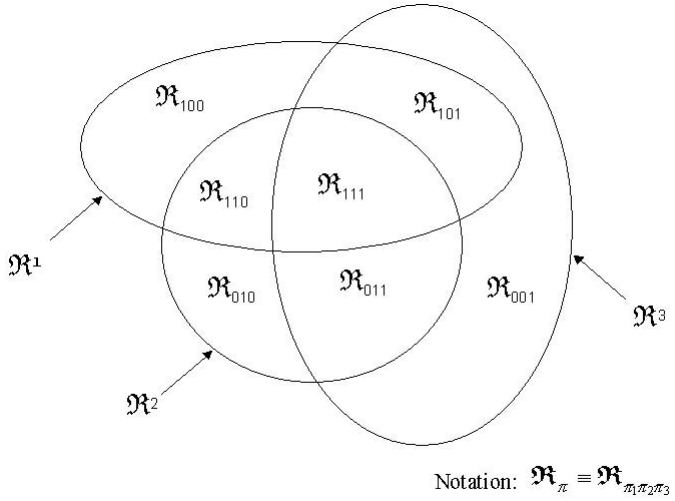

Fig. 5. A Venn diagram illustrating requirements partitions for a family of three products. There are three partitions ($\mathfrak{R}_{100}$, $\mathfrak{R}_{010}$, $\mathfrak{R}_{001}$) that contain requirements unique to a single product, three partitions ($\mathfrak{R}_{110}$, $\mathfrak{R}_{101}$, $\mathfrak{R}_{011}$) that contain requirements common to two of the products, and a single partition ($\mathfrak{R}_{111}$) that contains requirements common to all products.

system scalability, performance, and operational cost-effectiveness. Accordingly, we define a mapping which associates a number for each partition:

$$M_{\boldsymbol{\pi}} = \mathcal{M}\left(\mathfrak{R}_{\boldsymbol{\pi}}\right) \tag{3}$$

It is understood in the above equation that architectural/design constraints may limit the size of $M_{\boldsymbol{\pi}}$. The units that we shall adopt for $M_{\boldsymbol{\pi}}$ are function points (FP).

The *demand function* is then defined as the average rate of new requirements that are placed on the product family during the planning period:

$$D_{\boldsymbol{\pi}} \equiv \frac{1}{T_P} M_{\boldsymbol{\pi}} \tag{4}$$

The demand function has units of function points per unit time (e.g. FP/Year).

In the Independent scenrio, each product group must develop, enhance, and maintain its own code base to meet the set of requirements, even though other products may share many of those requirements. For a given domain, the demand placed on a given product is given by:

$$D_k = \sum_{\{\boldsymbol{\pi}\}} \hat{k} \cdot \boldsymbol{\pi}\, D_{\boldsymbol{\pi}} \tag{5}$$

where the factor $\hat{k} \cdot \boldsymbol{\pi}$ ensures that only demand relevant to the k^{th} product is included in the sum. Here, $\hat{k}$ is the unit vector whose components are all 0 except for the k^{th} component which is equal to 1. The total *effective demand* across all products in the Independent scenario is then:

$$\tilde{D} = \sum_{k=1}^{N} D_k = \sum_{k=1}^{N} \sum_{\{\boldsymbol{\pi}\}} \hat{k} \cdot \boldsymbol{\pi} D_{\boldsymbol{\pi}} = \sum_{\{\boldsymbol{\pi}\}} n(\boldsymbol{\pi}) D_{\boldsymbol{\pi}} \tag{6}$$

By contrast, in the SPL scenario, functionality for a given partition is implemented once. The total demand in the SPL case is then:

$$D = \sum_{\{\boldsymbol{\pi}\}} D_{\boldsymbol{\pi}} \tag{7}$$

The Independent demand function carries a weighting factor equal to $n(\boldsymbol{\pi})$. This weighting is a measure of the demand redundancy inherit in the Independent scenario. In order to quantify this redundancy, we introduce the concepts of "commonality" and "leverage."

A.2 Commonality and Leverage

We start by assuming that, in the SPL scenario, functionality common to two or more products is developed by the common component factory, while functionality specific to a vertical market is implemented by the product group responsible for that vertical. Let $\{\boldsymbol{\pi}\}^c$ represent the set of partitions which apply to two or more products, and let $\{\boldsymbol{\pi}\}^v$ represent the set of partitions that are unique to a single vertical market product. Then $\{\boldsymbol{\pi}\} = \{\boldsymbol{\pi}\}^c \bigcup \{\boldsymbol{\pi}\}^v$. Next, we define the common demand and vertical demand functions as:

$$\tilde{D}^C \equiv \textstyle\sum_{\{\boldsymbol{\pi}\}^c} n(\boldsymbol{\pi}) D_{\boldsymbol{\pi}}; \qquad \tilde{D}^V \equiv \displaystyle\sum_{\{\boldsymbol{\pi}\}^v} D_{\boldsymbol{\pi}} \tag{8}$$

$$D_k^C \equiv \textstyle\sum_{\{\boldsymbol{\pi}\}^c} \hat{k} \cdot \boldsymbol{\pi} D_{\boldsymbol{\pi}}; \qquad D_k^V \equiv \displaystyle\sum_{\{\boldsymbol{\pi}\}^v} \hat{k} \cdot \boldsymbol{\pi} D_{\boldsymbol{\pi}} = D_{\hat{k}} \tag{9}$$

$$D^C \equiv \textstyle\sum_{\{\boldsymbol{\pi}\}^c} D_{\boldsymbol{\pi}}; \qquad D^V \equiv \displaystyle\sum_{\{\boldsymbol{\pi}\}^v} D_{\boldsymbol{\pi}} = \tilde{D}^V \tag{10}$$

Then:

$$D = D^C + D^V; \qquad \tilde{D} = \tilde{D}^C + D^V \tag{11}$$

Here, D_k^C is the size of demand on the k^{th} product that is served by the common component group, while D_k^V is the size of the demand on the k^{th} product that is met by the product group itself. D^C is the total demand that is common to two or more products, while D^V is the total demand unique to the various products. Similarly, $\tilde{D}^C$ is the size of the demand that is common across the product family, weighted by the redundancy factors, while $\tilde{D}^V$ is the size of the

total demand unique to the various products. The redundancy weighting factor for the vertical market partitions is one, therefore $\tilde{D}^V = D^V$.

Next, we introduce the *commonality* parameters:

$$\omega_k \equiv \frac{D_k^C}{D_k}; \qquad \omega \equiv \frac{D^C}{D}; \qquad \tilde{\omega} \equiv \frac{\tilde{D}^C}{\tilde{D}} \tag{12}$$

The interpretation of these quantities is as follows. ω_k is that fraction of demand placed on the k^{th} product that is shared by other products in the product family. ω is that fraction of demand that is shared by two or more products - it represents the total demand placed on the component factory in the SPL scenario. Finally, $\tilde{\omega}$ represents the fraction of demand in the Independent case that is shared by multiple products, properly weighted by the redundancy factors.

Next, we introduce the notion of *leverage*:

$$\lambda_k \equiv \frac{D_k^C}{D^C}; \qquad \lambda \equiv \frac{1}{N}\sum_k \lambda_k \tag{13}$$

The parameter λ_k represents the fraction of new capabilities produced by the component factory that benefit the k^{th} product, hence the term "leverage." λ is just the numerical average of the leverage factors across all products.

We are now ready to quantify the demand redundancy in the Independent scenario. Based on the above definitions, we have:

$$D^V = \tilde{D} - \tilde{D}^C = (1-\tilde{\omega})\,\tilde{D} = D - D^C = (1-\omega)\,D \Rightarrow \frac{\tilde{D}}{D} = \frac{(1-\omega)}{(1-\tilde{\omega})} \tag{14}$$

Also:

$$\lambda = \frac{1}{N}\sum_k \lambda_k = \frac{1}{N}\sum_k \frac{D_k^C}{D^C} = \frac{1}{N\omega D}\sum_k \omega_k D_k = \frac{\tilde{D}\tilde{\omega}}{N\omega D} \Rightarrow \frac{\tilde{D}}{D} = \frac{N\lambda\omega}{\tilde{\omega}} \tag{15}$$

Hence:

$$\frac{\tilde{D}}{D} = 1 + (N\lambda - 1)\,\omega \tag{16}$$

Note that the redunancy increases as the number of products, commonality, and leverage increase.

A.3 Productivity and Throughput

Software development organizations respond to market demand by delivering new releases of their product that enhance its value to potential customers. Let ΔF_k represent the average size of new product releases (in function points) over the planning period, T_P, for product k. Also, let $\tilde{S}_k$, S_k, represent the average staffing level associated with product k in the Independent and SPL scenarios, respectively. The *productivity* is then defined as:

$$\tilde{p}_k \equiv \frac{\Delta F_k}{\tilde{S}_k} \text{ Independent Case}$$

$$p_k \equiv \frac{\Delta F_k}{S_k} \text{ SPL Case} \tag{17}$$

The *throughput* (Independent scenario) is defined as:

$$\tilde{T}_k \equiv \tilde{p}_k \tilde{S}_k \tag{18}$$

Throughput has the same units as demand: function points per unit time.

B SPL Benefits

B.1 Relationship Between Independent and SPL Staffing Levels

Having introduced the basic concepts, we now quantify the SPL benefits. In the SPL case, the development staff is allocated between the common component factory and the individual product groups. Let S and p represent the total staff and average productivity, respectively for the SPL scenario. Let S^C denote the common component staff, and let S^V denote the product (vertical market) staff. The allocation of the two will be in accordance with the relative demand:

$$S^C = \omega S; \qquad S^V = (1-\omega)\, S \tag{19}$$

In addition, S^V will be split between the various products as:

$$S_k^V = \frac{D_{\hat{k}}}{D^V} S^V \tag{20}$$

To determine S, we require that the SPL scenario provide the same level of throughput as the Independent scenario:

$$T_k = \tilde{T}_k \tag{21}$$

The SPL throughput is:

$$T_k = p\left(\lambda_k S^C + S_k^V\right) = p\left(\lambda_k \omega + \frac{D_k}{D^V}(1-\omega)\right) S \tag{22}$$

In the above, the leverage factor, λ_k accounts for the fact that not all of the component factory output benefits the product in question. Setting the SPL throughput equal to the Independent throughput, and summing over all products, we obtain:

$$p\left(1 + (N\lambda - 1)\,\omega\right) S = \tilde{p}\tilde{S} \tag{23}$$

Hence:

$$\frac{S}{\tilde{S}} = \frac{\tilde{p}}{p}\frac{D}{\tilde{D}} \tag{24}$$

Thus we see that the SPL staffing is reduced due to the elimination of redundant demand (assuming $N\lambda > 1$), and due to an increase in productivity (assuming $p > \tilde{p}$). Letting $r \equiv \tilde{p}/p$, and using the result from equation (25), we obtain:

$$S = r\left[1 + (N\lambda - 1)\,\omega\right]^{-1} \tilde{S} \tag{25}$$

Equation (34) suggests that we can lower staffing levels and still maintain throughput equivalent to the Independent case. Expressing p as $p \equiv \tilde{p}\,(1+\delta p)$, and letting $\Lambda \equiv\ (N\lambda - 1)\,\omega$ we obtain the following expressions for the staff savings:

$$\Delta S \equiv \tilde{S} - S = \left[1 - \frac{1}{r\,(N\lambda - 1)\,\omega}\right]\tilde{S} \equiv \left[1 - \frac{1}{(1+\delta p)\,(1+\Lambda)}\right]\tilde{S} \qquad (26)$$

The break-even condition is obtained as follows.

$$\Delta S \geqslant 0 \Rightarrow (1+\delta p)\,(1+\Lambda) \geqslant 0 \qquad (27)$$

B.2 Financial Flexibility

The SPL approach has a lower staffing requirement than the Independent approach. Let L represent the fully loaded cost per person per year. The annual investment requirements are:

$$\tilde{I} = L\tilde{S}\ \text{ Independent;} \qquad I \equiv LS\ \text{ SPL} \qquad (28)$$

Therefore, the cost avoidance per annum due to improved productivity and the elimination of redundant development is:

$$\Delta I \equiv \tilde{I} - I = L\Delta S = \left[1 - \frac{1}{(1+\delta p)\,(1+\Lambda)}\right]\tilde{I} \qquad (29)$$

Thus, the obvious way to quantify the benefit is in terms of the cost avoidance implied by equations (39) and (40).

However, there is another way to look at equation (35). Instead of reducing the staffing levels, the staff can be re-allocated back to the component groups and product groups in order to *increase throughput*. Assuming that the staff is allocated in the same way as in equation (28), we obtain a higher throughput level:

$$T \longrightarrow T + \Delta T = (1+\delta p)\,(1+\Lambda)\,\tilde{T} \qquad (30)$$

To achieve this higher throughput level in the Independent scenario, we would have to add staff (and increase the investment level) by:

$$\Delta\tilde{S} = \frac{\Delta T}{\tilde{p}} = [(1+\delta p)\,(1+\Lambda) - 1]\,\tilde{S} \Rightarrow \Delta\tilde{I} = [(1+\delta p)\,(1+\Lambda) - 1]\,\tilde{I} \qquad (31)$$

Comparing equations (39) and (42), we see that:

$$\frac{\Delta\tilde{I}}{\Delta I} = (1+\delta p)\,(1+\Lambda) > 1 \qquad (32)$$

B.3 Time Dependence of the Benefits

During the period when the component is being deployed to the various products, the benefits will have a dependence on the number of products using that component. Let $n(t)$ represent the number of products using the component at time t. Then we have:

$$\begin{aligned} \Delta I(t) &= \left[1 - \frac{1}{(1+\delta p(t))\,(1+\Lambda(t))}\right] \tilde{I}(t) \\ \Delta \tilde{I}(t) &= \left[(1+\delta p(t))\,(1+\Lambda(t)) - 1\right] \tilde{I}(t) \end{aligned} \tag{33}$$

where:

$$\delta p(t) = \frac{n(t)\delta p}{N}; \qquad \Lambda(t) = \left[1 + (n(t)\lambda - 1)\,\omega\right]; \qquad \tilde{I}(t) = \frac{n(t)}{N}\tilde{I} \tag{34}$$

B.4 Multiple Components

For the case of multiple components, we attached a component subscript (α) to all parameters, where $\alpha = 1, 2, \ldots$ Number of Components).

$$\begin{aligned} \Delta I_\alpha(t) &= \left[1 - \frac{1}{(1+\delta p_\alpha(t))\,(1+\Lambda_\alpha(t))}\right] \tilde{I}_\alpha(t) \\ \Delta \tilde{I}_\alpha(t) &= \left[(1+\delta p_\alpha(t))\,(1+\Lambda_\alpha(t)) - 1\right] \tilde{I}_\alpha(t) \end{aligned} \tag{35}$$

Then the total benefits are given by:

$$\Delta I(t) = \sum_\alpha \Delta I_\alpha(t); \qquad \Delta \tilde{I}(t) = \sum_\alpha \Delta \tilde{I}_\alpha(t) \tag{36}$$

www.convergys.com

FOR INFORMATION ON OUR PRODUCTS AND SERVICES, PLEASE CALL:

1 800 344 3000 1 513 458 1300

CONVERGYS CORPORATE HEADQUARTERS
201 East Fourth Street Cincinnati, OH USA 45202
Tel: 513 723 7000 Fax: 513 421 8624 e-mail: marketing@convergys.com

CONVERGYS LATIN AMERICA
CENU - Av.das Nações Unidas, 12.901-34 andar - Torre Norte
CEP: 04578-000 - São Paulo - BRASIL
Tel: +55 115 102 1800 Fax: +55 115 102 1911 e-mail:convergys.brazil@convergys.com

CONVERGYS EUROPE, MIDDLE EAST & AFRICA HEADQUARTERS
Cambourne Business Park Cambourne Cambridge CB3 6DN UK
Tel: +44 1223 705000 Fax: +44 1223 705001 e-mail: europe@convergys.com

CONVERGYS ASIA PACIFIC
30 Cecil Street #11-08 Prudential Tower Singapore 049712
Tel: +65 6557 2277 Fax: +65 557 2727 e-mail: asia@convergys.com

Convergys provides billing, employee care and customer care services, bringing together world-class resources and expertise to help clients transform customer relationships into a competitive advantage.