Business Value of Software Reuse in the Digital Enterprise: An Industry Perspective

by

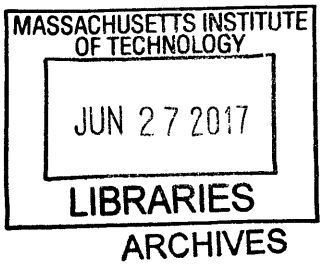
Edwin R. Suarez

Eng. Computer Science
Universidad Industrial de Santander, Bucaramanga 2000

M.S. Computer and Information Science
Gannon University, Erie 2010

Submitted to the System Design and Management Program
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Engineering and Management at the
Massachusetts Institute of Technology

June 2017



Signature redacted

Signature of Author: _____

Edwin R. Suarez
System Design and Management Fellow
May 22, 2017

Signature redacted

Certified by: _____

Jeanne W. Ross, Thesis Supervisor
Principal Research Scientist, MIT Center for Information Systems Research

Signature redacted

Accepted by: _____

Joan Rubin
Executive Director, System Design and Management Program

Business Value of Software Reuse in the Digital Enterprise:
An Industry Perspective

by

Edwin R. Suarez

Submitted to the System Design and Management Program on May 22, 2017
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Engineering and Management at the
Massachusetts Institute of Technology

ABSTRACT

Many companies are accelerating investments in digital technologies to streamline internal operations and create new business models. To achieve these goals, digital enterprises are adopting management and software development practices from traditional software products and service companies. Software reuse is one of those practices.

Despite being a well-documented method for increasing productivity, quality, and agility, there is a lack of good empirical data to validate the business value of reuse. This research paper compares available literature on the subject of software reuse to a set of interviews with business and technology executives on the benefits they receive from software reuse and their management practices.

The analysis concludes that the most valuable benefit of software reuse is competitive advantage through differentiation. Therefore, implementing software reuse is crucial for digital enterprises to protect their long-term sustainability. The more traditional benefits of reuse are important to gain financial and operational efficiency, but they are difficult to measure and validate against profit gains. To maximize the value of software reuse, digital enterprises should connect reuse with their business strategy and apply system thinking to assess the full impact of its implementation.

Thesis Supervisor: Jeanne Ross, PhD
Title: Principal Research Scientist, MIT Center for Information Systems Research

DISCLAIMER

This thesis is being submitted to the System Design and Management (SDM) program in partial fulfillment of the requirements for the degree of Master of Science in Engineering and Management at the Massachusetts Institute of Technology (MIT).

The views, opinions, and conclusions expressed in this research paper are those of the author and do not reflect the position of MIT, the SDM program, or any of the interviewees who participated of the industry data collection.

ACKNOWLEDGEMENTS

First, my wholehearted thanks to Dr. Jeanne Ross for her guidance and wisdom throughout the creation of this intellectual work product and to Jorge Frausto and Joseph Vajda, my managers at General Electric (GE) during the time I was attending MIT, for their sponsorship and coaching. I must also thank my mentors, Ben Wilson and Nancy Anderson, for inspiring me to follow my dreams.

My sincere gratitude to all the participants in the industry data collection for their time and contributions to this research.

Likewise, I would like to acknowledge the management team of the SDM program: Joan Rubin, Pat Hale, Steven D. Eppinger, Warren Seering, William "Bill" Foley, Jonathan Pratt, and their teams. Their support was central to the conclusion of my fellowship.

This section would not be complete without recognizing my friends and colleagues Mina Botros, Ahmed Altayyar, and Idalides Vergara, who accompanied me through my journey at MIT and provided much needed emotional support.

Finally, I must thank my wife Marcela, and my children, Ben and Martin, for their infinite patience, love, encouragement, and understanding during the days and nights I spent producing this research and the many months I spent away from home. I am here because of you.

TABLE OF CONTENT

TABLE OF FIGURES

ACRONYMS

BPO	Business Process Outsourcing
CEO	Chief Executive Officer
CIO	Chief Information Officer
COO	Chief Operating Officer
CPO	Chief Product Officer
CRM	Customer Relationship Management
CTO	Chief Technology Officer
ERP	Enterprise Resource Planning
GE	General Electric
GM	General Manager
HP	Hewlett-Packard
IP	Intellectual Property
IT	Information Technologies
KPI	Key Performance Indicators
MIT	Massachusetts Institute of Technology
OT	Operational Technologies
ROI	Return On Investment
RQ	Research Question
SDM	System Design and Management program
SG&A	Selling, General and Administrative Expenses
TCO	Total Cost of Ownership
USD	United States Dollar
VP	Vice President

CHAPTER 1: INTRODUCTION

Software is eating the world.
– Marc Andreesen

1.1. Background

For a few years now, many organizations across multiple industries have been experiencing what is being called the "Digital Economy". Under this new paradigm, businesses are transforming themselves into "Digital Businesses", empowering their organizations with more contemporary technology-enabled tools to streamline their operational processes, build agility, and gain market share through differentiation in the market place. Many of these digital tools are software applications in areas such as mobile, social networks, and big data and data analytics.

Historically, companies have pursued process standardization and digitization with traditional information technologies (IT) such as Enterprise Resource Planning (ERP) systems, to digitize their financial, supply chain, and manufacturing processes, and Customer Relationship Management (CRM) systems, to optimize their sales and marketing activities. And in the early years of this century, many organizations invested significant amounts of their budgets in building electronic commerce capabilities, or e-Commerce, to engage customers and execute commercial transactions through new online channels over the Internet. Although these technologies produced significant benefits to the organizations that implemented them, they did not create the transformational change that digital technologies are now introducing.

This conjecture was made convincingly clear by GE's Chief Executive Officer (CEO), Jeff Immelt, during Minds + Machines industry keynote event in 2012. During his opening presentation, Mr. Immelt shared with the audience: *"If you went to bed last night as an industrial company, you are going to wake up the next morning as a software and analytics company"* [1]. With this eloquent message, GE's CEO acknowledged that their business model had fundamentally shifted due to digital technologies. GE, like most big old companies, found that vast amount of data and new technologies are creating new business opportunities. A great example is GE's ability to harness the operational data coming from aircraft engines to predict failures and maintenance needs by using data analytics frameworks.

A crucial point is embedded into Mr. Immelt's message: companies will need to excel at using technology to change the way they do business, from internal operations to business models, and at building strong leadership skills to lead change [2]. Because creating digital products is different than manufacturing an airplane engine or a locomotive, mastering these new capabilities implies a major transformation.

Consequently, the key decisive challenges for a digital company are to discover new revenue streams through software-enabled solutions and to apply new management practices to build them. An important practice in the production of software systems is software reuse. Reuse promotes faster time to market or agility, higher quality, and lower costs [3]. However, achieving true financial impact through software reuse requires a deep understanding on how reuse enables such outcomes.

1.2. Purpose of the Research

The main purpose of this thesis is to answer two key research questions (RQ) by complementing and contrasting available literature in the subject of software reuse with industry data collected through interviews with executives from digital and software companies:

RQ #1: What business benefits impacting the bottom line are generated by software reuse in a digital enterprise?

RQ #2: How is software reuse successfully implemented in a digital enterprise?

1.3. Scope and Limitations

This research is focused on the business aspects of implementing software reuse without detailing specific implementation roadmaps. Specific technical topics, such as software reusability methodologies, tools, frameworks, business process reuse, etc. are beyond the scope of this research.

The analysis is based on literature review and industry data collected through interviews with 17 executives across eight different companies representing three specific company types. The literature review will summarize prior research on software reuse focusing on the established benefits of implementing a software reuse strategy. The analysis of the interviews assesses the experiences of business and technology executives as they

attempt to generate those, and other, benefits. The thesis summarizes the benefits they received and what practices they relied on to generate those benefits.

This research is not meant to be an exhaustive treatise on how to implement software reuse. Rather, the thesis intends to understand the business reasons behind the decision to implement a software reuse strategy and how to generate business benefits from reuse.

1.4. Thesis Organization

This thesis is organized in five main sections. Chapter 1 introduces the purpose and structure of the research paper as well as the scope and limitations of the research.

Chapter 2 presents a literature review on software reuse and the benefits from implementing a software reuse strategy. This chapter also introduces the business aspects of implementing reuse.

Chapter 3, industry data collection, summarizes each of the interviews with business and technology executives who have experienced the implementation of software reuse strategies.

Chapter 4 encompasses the analysis and discussion of the findings from the previous two chapters, comparing the findings from the literature with the experiences of the

executives. Chapter 5 presents the conclusion and highlights the contributions of this research.

This thesis also includes four appendices. Appendix A contains a list of approaches that support software reuse and reusability. Appendix B visualizes the reuse factors identified by a study of software reuse practices at Hewlett-Packard (HP). Appendix C shows a visual example of the cumulative cost of building software systems with and without reuse. And Appendix D lists the interview questions used during the industry data collection.

CHAPTER 2: LITERATURE RESEARCH

A successful book is not made of what is in it, but what is left out of it.
– Mark Twain

Chapter 2 presents a literature review on software reuse and its business benefits. First, the concepts of digital enterprise and software reuse are explained. The next step is to describe the business benefits from implementing reuse. This chapter ends with lessons learned from implementing a software reuse strategy.

2.1. The Digital Enterprise

According to Gartner [4], many enterprises use the term "Digital Business" to refer to any initiative elicited by the convergence and mutual reinforcement of social mobility, cloud, and information patterns to conduct new business scenarios. These business initiatives may impact every aspect of the organization, from new products and services to business operational processes.

Gartner provides insights into six areas that organizations must master to become a digital enterprise:

1. Business and technology visions: To create new business models through technology, organizations need to adopt the right business and technology visions and strategies.
2. Technology convergence: The capacity to identify and capture business value through the convergence of IT and operational technologies (OT). For instance,

the creation of new revenue streams using analytics (IT) and operational data (OT) from industrial assets.

3. Algorithms and analytics: Creating new capabilities to monetize operational data and to better understand customer behaviors.
4. Bimodal operations: Continue supporting traditional IT initiatives and frameworks while fostering a new environment for agile prototyping and learning.
5. IT Infrastructure: Introduce technologies capable of supporting the velocity, variety, and volume attributes of real-time data from business operations as well as the scalability and elasticity required by the new business models.
6. Organization and culture: Establishing the appropriate incentives, values, and behaviors in the organization.

Soule et al [5], at the MIT Center for Digital Business, argue that digital capabilities go beyond the ability to harness any set of technologies and they supplement Gartner's insights with three more focus areas:

7. Customer experience, where technology is used to address ever-changing customer expectations.
8. Operational efficiency, where business operational processes are streamlined, optimized, and automated.
9. Workforce enablement, where employees can collaborate better and be more productive.

The business outcomes of mastering digital capabilities are summarized by Westerman et al [2] as significantly higher levels of profit, productivity, and performance. These benefits are captured by rethinking business processes and business models, getting closer to customers and improving their engagements, empowering employees, and creating the right leadership capabilities inside the organization to actively drive a digital transformation.

Becoming a digital company entails an evolution, regardless of the organization's native business model, that is powered by contemporary digital technologies and that compels the creation of new capabilities that are software-enabled at the core. These new demands challenge a digital company to learn how to manipulate those technologies to create differentiating, internal or commercial, digital solutions. Software reuse is a known method for achieving this goal faster, with higher quality, and at lower cost [3].

2.2. Software Reuse

There are multiple definitions of software reuse across literature. For Leach [6], software reuse refers to a situation in which a software element is used in more than one project. Freeman [7] also defines reuse as any procedure that produces or helps produce a software system by reusing something from a previous development effort. And Savolainen et al [3] present a more holistic approach by defining software reuse as a prominent method for achieving faster time to market, higher quality, and lower costs in

the production of software systems by applying reusable assets[1] into a new context to create a software product. In all definitions, almost any work product or software artifact can become a reusable asset.

The concept of software reuse originated at the University of Cambridge in 1949 when Maurice Wilkes first recognized the need for minimizing redundant efforts in writing subroutines for computer programs, thus recommending the creation of software libraries for general use [9]. Then, almost twenty years later, McIlroy introduced the concept of software mass production, bringing industrial manufacturing processes like standard parts, product assembly, among others, into software development [10]. McIlroy's revolutionary approach, accelerated the need to standardize and reuse. Nowadays, because of the open source movement, there is a diverse range of reusable software components broadly available for integration, tailoring, and adaptation into other software products [11].

Prieto-Diaz and Freeman [12] argue that there are two levels of reuse to consider: the reuse of ideas and knowledge and the reuse of particular artifacts and components. The first level would include ideas, concepts, and even algorithms. Per Otchere [13], the know-how of building software is also a reusable asset. For the second level, Northrop et al [14] state that reusable artifacts and components, also known as reusable core assets, often include software components (implemented software code), requirements

[1] A financial asset is defined as a resource controlled by an organization as a result of a past event from which future economic benefits are expected to flow into the organization [8].

statements, documentation, specifications, system architecture, performance models, schedules, budgets, test plans, test cases, and process descriptions.

Similarly, Leach [6] presents 18 different reusable artifacts: architecture, requirements, design, programs and common systems, modules, transformation systems, cost models, plans and schedules, measurement data, products or systems, data, documentation, negotiation with customers, negotiations with software vendors, algorithms, classes and instances, interface specifications, inputs to application generators, and inputs to very high level languages. This more inclusive approach is connected to the phase of the software development process where the reusable software artifacts can be reused.

In addition to software artifacts, software-enabled business processes are also reusable. The reuse of business process models is the act of designing business processes by using existing process models. The reuse of software-enabled business process depends on the availability of the business process as a combined work product of software and a process model [15].

An interesting fact in the literature of software reuse is the existence of the term “software reusability”, which is often used interchangeably with software reuse. Software reusability can be defined as the extent to which a software component can be used, with or without adaptation, in multiple solutions [16]. In simple terms, reusability is the ability of a software component to be reused. In a larger system scale, reusability would belong to the group of “-ilities” of a software system. Considered non-functional requirements,

these "-ilities" are desired system properties that often manifest themselves after a system has been put to its initial use [17]. Therefore, software reusability is an attribute of a software system that implicitly suggests planned efforts towards such reuse [18].

For the purposes of this research, all the definitions discussed in this section will be considered in the analysis presented in Chapter 4.

2.3. Business Benefits of Software Reuse

Software reuse is one of the most attractive approaches to the improvement of agility, quality, and productivity in software production [13][19]. In practice, reuse can decrease product development costs up to 12%, reduce long-term maintenance costs up to 50%, and drop defect rates by 10% [20]. Leach [6] provides a few examples of successful software reuse:

- Reductions of 75% in overall development effort and cost reported by NASA.
- 312 projects in the aerospace industry, with averages of 20% increase in productivity, 20% reduction in customer complaints, 25% reduced time to repair, and 25% reduction in time to produce the system.
- A Japanese industry study that noted 15-50% increases in productivity, 20-35% reduction in customer complaints, 20% reduction in training costs, and 10-50% reduction in time to produce the system.
- A simulator system developed for the US Navy with an increase of nearly 200% in the number of source lines of code produced by hour (productivity).

Agility refers to the ability of the organization to introduce software products or solutions faster than the competition. The impact on the bottom line of the company is measured in orders and revenue. A straightforward connection from practice to business outcomes.

Quality is mainly measured through reliability[2]. By reusing software that has already been tested and proven effective in previous implementations, the reliability of the final software product is improved [9]. To achieve such outcomes, the company needs to put in place the quality process required to measure and manage the key process indicators (KPI) for reliability. The impact of improved quality on the bottom line manifests in the form of reduced costs from less errors, minimized rework, and reduced liability.

Productivity is more complicated. In the production of software, productivity is typically measured at personal levels (i.e. more lines of code per employee) [20][19]. Also, as a performance indicator, productivity has the potential to include the outcomes achieved by agility and quality. For instance, by increasing quality, the company can experience reduction in the cost of producing the software products, thus generating productivity. Producing more at lower cost is the traditional performance measure of output per unit of input attributed to productivity. At the company level, productivity gains have the potential to contribute to an increase in business profit[3] along with other factors [23].

[2] The definition of reliability depends on the context. In the case of software reuse, reliability is measured in the consistency of results and behavior under similar operating conditions [21].
[3] The so-called "bottom line", profit measures performance over a period of time and is calculated as revenues minus expenses [22].

Despite all the documented benefits, there is a lack of good empirical data to validate them [18]. This gap of experimental evidence of connecting business benefits of software reuse to the bottom line of the company is another motivator for this research. One of the main challenges is capturing the actual cost of implementing reuse to calculate return on investment (ROI):

1. The cost of making software reusable, which could be up to 60% of additional cost.
2. The cost of reusing software components.
3. The cost of defining and implementing a process to manage reuse.

In summary, software systems and products are built faster, more reliably, and at a lower cost, by reusing high quality software artifacts. The benefits of software reuse could translate into competitive advantage resulting from the development of specialized products and services that competitors cannot easily replicate [24]. Thus, a successful software reuse strategy is potentially a competitive differentiator.

2.4. Implementation of Software Reuse

As mentioned in the previous section, software reuse needs to be planned. There are multiple technical methodologies that facilitate software reuse and enable reusability; a comprehensive list of approaches is available in Appendix A.

Software development is not a simple manufacturing activity but more like a product design activity requiring a combination of art, science, engineering, and management skills. As a result, the entire production process, including reusability, is very difficult to

manage and control [25]. From a technical perspective, Otchere [13] found that 40% to 60% of all code is reusable from one software application to another, 60% of all design and code on all business applications is reusable, 75% of program functions are common to more than one program, and only 15% of the code found in most programs is unique to a specific application.

On the other hand, a study of HP's reuse practice highlighted that the impediments to improving software reuse are predominantly non-technical and socioeconomic. These include cultural, organizational, business, and process factors. Appendix B presents these factors at a greater level of detail. The same study found that these inhibitors can be overcome with management commitment, education, process, standards, policy, and incentive [20]. More recent studies also propose changes to the organizational culture, specifically the belief system[4], towards accepting and facilitating software reuse [26].

As discussed earlier, cost is a key factor to be considered when implementing software reuse. A software reuse strategy is a long-term investment, requiring upfront funding and time before an actual economic return is realized [27]. Appendix C shows an example of the cumulative cost of building software systems with and without reuse. Because of the extra effort required to prepare software for reuse, the implementation of software reuse is only financially viable when considering the long-term benefits [16].

[4] A belief system is a set of beliefs which guide and govern a person's attitude. A belief is a conviction that an individual hold as true [26].

Another potential pitfall in implementing a reuse strategy is the tendency to place too much emphasis on ensuring all products use every reusable artifact to enforce a common structure and behavior. Savolainen et al [3] call this phenomenon the "tyranny of the reuse organization". Tyranny of reuse occurs when an organization prioritizes reuse maximization in conflict with the business needs, causing slower introduction of products into the market and lack of product diversity. A potential solution to this challenge is to achieve a balance between reuse and differentiation by implementing complementary practices such as product line development and by aligning reuse to the overarching business strategy.

According to Leach [6], the benefits of reuse are best achieved when the design and development of reusable components and their utilization are systematically promoted. Organizations with effective reuse strategies established a disciplined process to analyze, measure, and manage software systems and the libraries of core reusable software assets. In those organizations, the transition to practicing reuse required changes in the way they worked and the behavior of practitioners involved. The key success factors in their adoption of reuse were instituting a reuse culture, providing training, adhering to standards, and securing management commitment [26].

As with most technology adoption in business organizations, effective reuse requires a holistic approach where people, business strategy, technology, and processes are well considered. The success of a software reuse strategy is the consequence of good technical and managerial practices [20].

CHAPTER 3: INDUSTRY DATA COLLECTION

You can learn more about a person from an hour of play than from a lifetime of conversation.
– Plato

This chapter summarizes 17 interviews with business and technology executives from eight different companies who have experienced software reuse. The objective of this section is to complement and contrast the literature review presented in Chapter 2 and to serve as foundation for the analysis in Chapter 4.

3.1. Company Types

Three types of companies are included in the collection of industry data for this thesis:

1. Digital companies: Companies that improve business performance by streamlining internal operations and creating new business models using digital technologies.
2. Software products companies: Organizations that have over 60% of their reported revenues coming from new software product sales. These companies typically generate higher margins because of economies of scale that come from selling multiple copies of the same software product [25].
3. Software services companies: Companies that get most their revenues from consulting, custom software development, integration work, technical support, systems maintenance, and related activities. Software services companies are labor intensive and benefit from economies of scope that come mainly from knowledge management and client account management. As a result, these companies are generally less profitable than software product companies [25].

Table 1 shows the number of participating companies and the number of interviews by company type.

Type of company	Number of companies	Number of interviews
Digital	3	10
Software products	3	3
Software services	2	4
TOTALS	**8**	**17**

Table 1. Number of participating companies and interviews by company type

3.2. Functional Areas and Job Titles

The other dimensions captured during the industry data collection are the functional areas of the interviewees and their roles inside the organization. Table 2 lists all the participating functions and job titles. The responsibilities under each function and role provide a different perspective on the definition, scope, and expected benefits of software reuse [6]. However, an in-depth study by these dimensions is not in the scope of this thesis.

Function	Job Title
Engineering	Technical Director
IT	Chief Information Officer (CIO)
IT	Enterprise Data Architect
IT	Vice President (VP)
Operations	General Manager (GM)
Operations	Global Director
Operations	Head of Application Development
Product Management	Head of Product Management
Services	VP

Table 2. List of participating functions and job titles

3.3. Summarized Data

The interviews are organized by company type and summarized in a few paragraphs with emphasis in the concepts used for the analysis and discussion in Chapter 4.

3.3.1. Digital Companies

This section includes ten interviews across three different companies. The firms categorized as digital companies include a provider of packaging solutions for the food industry with revenues above $4 billion dollars[5], a global conglomerate with a diverse portfolio of industrial products and services with more than $100 billion dollars in sales, and an international corporation specialized in developing technologies and solutions for energy management with more than $20 billion dollars in revenue.

3.3.1.1. Interview #1: Enterprise Data Architect, IT

Software reuse is the ability to solve a business problem with an existing available solution. The reused solution does not need to be a 100% fit to the requirements but rather a "good enough" solution or an equivalent to 80% match between the needs and the available solution. Quoting the interviewee: "*Applying Pareto, 20% of the top requirements come with an 80% fit*". For example, an integration platform can be reused to combine multiple ERP systems and data sources under the "good enough" rule.

The main benefits of software reuse are agility to respond to dynamic changing business conditions and reduced total cost of ownership (TCO). In the example of the data

[5] The source of all financial information is the most recent annual report. All currencies in USD.

platform, adding a new data source would incur a marginal cost increase. These benefits are connected to the bottom line of the organization through cost avoidance, by eliminating the need for additional headcount or new solutions, and through scalability, by increasing capacity without significantly increasing costs. Scalability is another way to look at productivity.

Implementing software reuse requires balancing short-term goals and long-term strategy, in both business and technical aspects. Most of the challenges in software reuse come from the lack of clear requirements, the need for strong governance processes, and the software development expertise in the organization. However, the level of maturity required to assist with reusability does not need to be at the same level of a software company. The goal is to ensure "good enough" support to core business processes.

3.3.1.2. Interview #2: VP, IT

Software reuse can be explained in three different yet related definitions:

- A catalog of software components linked to a specific functionality, like GitHub[6].
- A common group of functionality or core capabilities to be used many times.
- Standard packages to manage a non-differentiating business activity.

Agility (time-to-market) and improved quality (with resiliency and robustness as quality KPIs) are the main benefits of software reuse. These benefits impact the bottom line of the organization by reducing TCO and by accelerating the introduction of differentiating

[6] GitHub is a cloud-based version control repository highly used by the open source community.

capabilities that could enable revenue opportunities: "*Digitalization is part of our strategy. We decided to invest with a long-term mindset, prioritizing working capital and resource allocation to support reuse for differentiating capabilities.*"

Productivity is another benefit of reuse. One view to productivity benefits is the year-over-year reduction in selling, general, and administrative expenses (SG&A) from lowering TCO; a more traditional view to productivity. If managed correctly, these SG&A reductions will have an impact on the bottom line of the company. The other viewpoint comes from productivity generated by business process reuse. When these business processes are enabled or supported by software, the productivity generated by reusing these digital processes constitutes a real business outcome that can be mapped to profit. Personal productivity, on the other hand, is not considered a business outcome.

Leadership and vision are necessary to implement a strong software reuse strategy as well as a high level of maturity in technical capabilities and governance processes. Specifically, for differentiating products and processes, the technical maturity level needs to be at par with that of a software company. Non-differentiating processes do not require the additional investment. Since software reuse is linked to differentiation, the reuse strategy must be linked to business priorities and to the organization's innovation model.

3.3.1.3. Interview #3: VP, IT

The goal of software reuse is to make software artifacts, from software components through final products to business processes, easily consumable at any level of the

technology stack. The likelihood of reuse depends on the capability of configuring, instead of customizing[7], the software components by the development team.

Software reuse allows faster development time and optimizes the creation of business-agnostic and configurable software products. Through reuse, the software product portfolio is also simplified, consequently reducing TCO and improving the cost-to-revenue performance ratio. Specifically, in the software development process, reuse creates personal productivity in the form of more software work products per person and fosters good software development practices.

The other source of benefits from software reuse comes from the actual implementation of the final product built with reusable software. These benefits, like variable cost productivity[8], sales price increase, and more efficient use of working capital materialize from improvements in the business processes supported by software products and they are expected to have an impact on the bottom line.

Reusing software requires a delicate balance between business results and technology innovation. "*We need to keep a delicate balance to move the technology needle while delivering on business commitments*". On the technology side, software development methodologies like Agile[9], the use of open source components, and standard software repositories are key to making software reuse a success. On the business front, culture,

[7] Software customization refers to the adaptation of a software product by making code changes. Configuration is the ability to adapt the product using available functionality without code changes [28].
[8] Productivity from improvements in the manufacturing process affecting variable cost.
[9] Agile is an inclusive term for a set of software development principles, methods, and practices [29][30].

alignment with priorities, organizational empowerment, and education of executives are essential to reuse. A digital company needs to create the right skillsets to build commercial-grade software products, including design for reusability, as well as to implement the appropriate standards and governance models.

3.3.1.4. Interview #4: CIO, IT

Software reuse is the process of categorizing existing assets and identifying opportunities to leverage those assets in different use cases, with minimal to zero rework effort. These assets are available at different levels of the software architecture, from components up to full software products, and they should be adapted via configuration.

The key benefits of software reuse are: faster time-to-market, reduced TCO, and improved quality of the software assets. Additionally, reuse provides the flexibility to reallocate resources across products and to cross-pollinate domain expertise.

Investment, incentives, and better tools and processes are more important than policies and regulations when implementing reuse. In practice, software reuse is difficult to fulfill, requiring the appropriate technical, cultural, and management skills in the organization. Recognizing the gaps in the software development process when compared to a software product company is imperative in a digital company: "*A digital company is not a software company, but there are similar behaviors*". Because the business model is not commercial software products, there are new skillsets required to design for reusability.

3.3.1.5. Interview #5: VP, IT

Software reuse can be viewed from different perspectives: "*you get different answers depending on who you ask*". Software reuse is both a capability and a feature of software components. The former comes from the adaptability of the component to be used across different use cases. The latter is the potential for the component to be configured without changing code.

The known benefits of software reuse are increased speed in development time and implementation (time to value), personal and team productivity, and improved quality. For example, a development team of five to seven people can save more than 20 weeks of development time a year, equivalent to half a million dollars, by reusing software artifacts. Ultimately, these benefits create the foundation for agility and differentiation by introducing new and better features faster and at a lower cost.

A successful implementation of a software reuse strategy requires upgraded technical and managerial capabilities. On the technical side, software developers need to be retrained in more contemporary software development frameworks like Agile. On the management side, the organization needs to recognize the need to build technical leadership, make long-term investments, and create the right performance indicators to support reuse.

3.3.1.6. Interview #6: CIO, IT

Software reuse can be explained through three basic definitions:

1. Common packaged applications and business process workflows implemented by multiple business units. For example, the business process for material planning is reused horizontally across different manufacturing plants.
2. Custom applications developed by the same type of users, or personas, across multiple business units, for instance, a software product built for field engineers. By focusing on personas, common requirements are grouped for reuse.
3. Reuse of software components in the underlying infrastructure for products and business process workflows, a more traditional view of software reuse.

Productivity, agility, and cost reduction are the most prominent benefits of software reuse. The value of reuse comes from building software artifacts once and replicating them multiple times across software products and business processes. For example, software reuse can increase return on investment by 400% by reducing the cost of development and delivery of software solutions: "*With re-use, the return is $3-4 per every $1 invested in a 12-month period*".

Achieving productivity benefits comes from accelerating the deployment of the software products into the business units (up to 40%) and from the expected outcomes of the actual implementation. The ability to build configurable products from reusable components accelerates the introduction of new products into the market or their implementation into internal operations.

To successfully drive reuse in digital organizations, new business leaders need to be technically savvy, prioritize investment ("*starve what's not important*"), and put in place the right policies and incentives. And to compete in the market, digital companies need to achieve a good level of maturity in their software development capabilities, including reuse, just like software companies.

3.3.1.7. Interview #7: CIO, IT

Software reuse stands for the ability to solve multiple customer requirements by applying features from existing software components. Software-enabled processes are also reusable artifacts when the underlying software elements can be replicated to reinforce standardization.

Because of standardization, quality and productivity are the main benefits of software reuse. More software and process standardization implies less maintenance and cost over time, thus reducing TCO. Productivity is accomplished by reducing the cost of re-implementing the same solution multiple times and by standardizing the know-how required for the implementations. In other words, build once and replicate (adopt) many times. By standardizing, variation is also minimized thus improving quality. Additionally, data can be capitalized once software and processes are standardized. For instance, a set of standard installed base and customer data could be mined to identify opportunities for upselling products or introducing new services.

Implementing software reuse requires changing the culture of the organization and strong leadership to drive change. In the case of a digital company, this change is more profound, implying a transformation that requires long-term vision, new technical capabilities, governance, and education. A simple tool to help obtain momentum is the use of best practice sharing and success stories that can propagate the message of change.

A common challenge when implementing reuse is people trying to solve the same problem in different ways, in part because of the "I am different" mindset where each business unit believes the problems they experience, and the solution to these problems, are unique. Standardization, through reuse, offers an alternative way of thinking.

3.3.1.8. Interview #8: CIO, IT

In general, software reuse is the ability to configure software, already developed and implemented, for other use cases. The landscape of reusable software is very broad, from software artifacts, such as micro-services, to a final software product.

The main benefits of reuse are agility (speed to value), reductions in cost (to produce and deliver), and productivity. These first two business outcomes come from delivering a commercial software product or implementing software-supported business processes more efficiently.

The case of productivity is more interesting. Software reuse can generate personal and business productivity. On the personal side, reuse allows the production of more software code in less time. This level of productivity may have an impact on the bottom line if the features and functions included in these lines of code are considered differentiating. Otherwise, personal productivity is not considered a business outcome. Business productivity is the result of improvements and standardization in business processes that are enabled by reused software.

A strategic long-term vision is mandatory to successfully implement software reuse, particularly when defining differentiation and changing the culture around software development. By truly delimiting what matters and what needs to be reusable, the organization can put in place the appropriate software reuse strategy and free up resources and capital to fund reuse. Ultimately, the strategy of a digital organization needs to include software reuse just like in a software product company.

3.3.1.9. Interview #9: Head of Product Management, Product Management

Software reuse is the ability to configure an existing software artifact into a product with commercial value. Reuse can be enabled through many layers of the software development cycle, from functions through libraries to micro-services. Regardless of the level, reuse requires the infrastructure (people, processes, and tools) to support the reusable software artifacts. Without this support, software reuse is not possible.

The main business benefit expected from reuse is productivity in the form of scalability and financial efficiency. Scalability comes from additional implementations of the software artifacts at nominal cost. Financial efficiency, measured in margin[10], originates from developing the product with less effort and cost. Up to 70% of the margin of commercial software products is predicated in software reusability. "*Higher margin is the ultimate outcome*".

Implementing reuse is costly. As a rule of thumb, a software component needs to be reused with at least five customers to pay off the cost of designing the software for reuse. Besides cost, the right product strategy must be in place to support reuse. This strategy includes product management capabilities, processes, tools, commercial and infrastructure support, and partnerships with the rest of the ecosystem — nothing too different from what software product companies have in place. And obviously, funding and resources are also needed to make reuse happen.

3.3.1.10. Interview #10: CIO, IT

Software reuse is the ability to leverage existing concepts or ideas, approaches or methodologies, and actual software artifacts to solve a business problem. Conceptual reuse is the easiest to implement and includes practices like best practice sharing. Reusing an approach enables the identification and adoption of solutions that are more optimal and efficient than the incumbent practice. Reusing software essentially means building components once to be replicated multiple times.

[10] Margin is a measurement of profit.

The real impact of software reuse on the bottom line happens after the implementation of an existing software-based solution across multiple business units. However, the more traditional benefits of reuse are productivity, quality, and agility. Productivity gains come from not having to invest resources to build a new solution ("*re-work*") and from more scalable solutions that can enable bigger impact with less effort. Quality is improved through better reliability and scalability without introducing more failure modes. And agility is represented through reduced time to market.

According to the interviewee, implementing reuse should not be complicated. The key is to lay out the reference architecture, decide on what to build and what to reuse, and establish a governance model. Additionally, the right reward and incentive systems need to be in place to promote reuse.

The main challenge with implementing reuse is the effort to raise awareness on the reusable assets and to establish the processes to manage them. A software reuse strategy is important but its implementation needs to be balanced with business priorities and goals: "*business outcomes always come first*".

3.3.2. Software Products Companies

Next is the summary of three interviews from three different software product companies: a group of businesses that provides customers with infrastructure components to store, manage, protect, and analyze data with revenues north of $20 billion dollars; a technology company that develops, licenses, supports, and sells software and hardware products

with revenues of more than $80 billion dollars; and a multinational technology company specializing in Internet-related services and products, with reported revenues of more than $90 billion dollars.

3.3.2.1. Interview #11: CIO, IT

Software reuse allows an organization to take available capabilities and artifacts to assemble new software products. Reusable components include libraries and standard objects but not final products such as an ERP system. Assembling a software product implies some level standardization and a good portfolio of reusable core assets.

Software reuse done right increases agility, by accelerating the delivery of solutions and products, and simplifies application portfolio management, with less cost structure and complexity. Reuse also generates productivity gains by lowering TCO and releasing working capital to work on other initiatives (financial efficiency). However, most of these benefits ("*soft benefits*") are hard to measure and most likely will not have an impact on the bottom line of the organization. Only "*hard benefits*", such as real savings and competitive differentiation, will have such impact.

Although there is enough literature to prove the value of reuse, in practice software reuse is difficult to implement. A big-enough library of software components need to be in place, technical people need to be open to changing their practices, and business leadership must be bought-in to lead adoption. And then, there is the challenge of capital funding. Reuse is a long-term investment competing with short-term results.

3.3.2.2. Interview #12: GM, Operations

Software reuse is the capability of leveraging existing software artifacts, from atomic components through full solutions to business processes and their enabling software, to solve different business problems. The goal of reuse is to maximize existing assets and resources to make a positive impact on business KPIs. There are four levels of reuse: software artifacts (in all layers), data, operational processes, and business strategy.

Reuse drives speed and cost avoidance in building software components. However, the real value comes from the execution of the software-enabled business processes that are built with reused software: "*It's about business process reuse and its implementation. Software plus a process model*". For instance, if the goal is to reduce backlog (a business outcome), the reuse strategy must be aligned with building the right solution with reusable software assets and positively impacting the backlog KPIs.

Implementing reuse is very challenging. Issues and opportunities need to be prioritized based on business value just like in any other business initiative. In the same way, reusability needs to be driven, as part of the business strategy, by the Chief Executive Officer (CEO) or the Chief Operating Officer (COO) of the company, not from the technology leaders.

3.3.2.3. Interview #13: Technical Director, Engineering

The definition of software reuse depends on perspective. For a CTO, software reuse is leveraging existing software components for multiple deployments (atomic view). For a

Chief Product Officer (CPO), reuse is creating new solutions from existing products to add value to customers and differentiate in the market (commercial view). And for a CIO, the view of reuse is around catalogs and the enabling infrastructure (architectural view).

The same notion of perspective applies to the benefits of reuse. For a CTO, the benefits gravitate around agility because of faster deployments, quicker integration, and more innovation. For a CPO, the key metric of success is incremental revenue from accelerating the introduction of differentiating products into the market. And for a CIO, the key benefit is cost reduction from lower TCO and productivity gains.

Implementing reuse requires a long-term vision to allocate capital and human resourcing. Reuse also involves a business strategy to manage the trade-offs between agility and cost and their inherent risks. The biggest challenge to a successful software reuse strategy is "non-technical people", referring to business leaders who do not fully understand those tradeoffs and risks.

Nevertheless, nothing special is required to incentivize software reusability. The recipe for success is good product vision, great foundational technology processes and software development methodologies, coupled with the right people to manage reusability and differentiation.

3.3.3. Software Services Companies

Four interviews are covered in this section with representation from two software services companies: A $13-billion-dollar global leader in consulting, systems integration, and professional and outsourcing services with expertise across multiple industries and the largest private IT company in Latin America, offering application software development services, business process outsourcing (BPO), and IT application and infrastructure management services.

3.3.3.1. Interview #14: VP, Services

For this VP, software reuse is the capability of configuring developed software assets, instead of customizing, for other projects and clients. Only the components that have been implemented for a client (value added) are considered reusable assets.

According to empirical data, projects that took advantage of software reuse experienced 30% of productivity gains and 3 to 5% margin improvements. The same data showed increases of 20% in the win/loss ratio for commercial proposals, which generated additional revenue.

A business and technology framework needs to be in place to implement software reuse: "*Our company started with a pilot to validate the strategy along with a central reusability global team and a steering committee board to decide on vision, investment, and financial targets. Once the pilot validated our assumptions, we embedded reusability in the services delivery methodology and nominated client partners at the top of the organization*

to drive reusability, followed by changes to incentive compensation". The big organizational impact happened when reuse was connected to career growth opportunities and reusable assets were considered intellectual property (IP) of the company.

3.3.3.2. Interview #15: VP, Services

"*Not software reuse but software use*". The goal of software reuse is to design software to serve as core components for multiple use cases and maximize their use. The word reuse implies doing the same thing multiple times, which is not completely appropriate in the case of business software.

The business value of software reuse comes from standardization in both software and business processes. Reuse allows organizations to mitigate cost and risk by removing variation in software-enabled business processes. And standardization complements and compels the case for differentiation: "*Standardize where differentiation adds no value*".

Software reuse is a journey that requires vision and strategy aligned with the business model. In other words, the overall software strategy needs to be motivated by the business strategy. Although digital companies are not software manufacturers, they should mandate the reuse of software assets.

3.3.3.3. Interview #16: Global Director, Operations

Reuse is very complex. In the software services business, software code may be owned by customers, limiting what can be reused. When possible, reuse happens in three different capacities:

1. Reusing knowledge by sharing practices to improve existing software artifacts. Crowdsourcing is a good example of reusable knowledge.
2. Reusing business applications, where the focus is to reuse designs and software applications that can be configured to support business processes.
3. Reusing infrastructure, including libraries and standards.

The main benefits of software reuse are cost savings, agility, and quality. From experience, reuse can affect the margin of a project by reducing costs up to 40%. Quality gains come from improvements in reliability. And agility is experienced by accelerating delivery of software products into the market or expediting the deployment of software-enabled business processes to capture value from the actual implementation. In the end, the ability to reuse software is a competitive differentiator. However, many of the benefits are difficult to quantify. Other than project margin improvements, all other benefits are difficult to connect to profit.

People are at the core of a software reuse strategy. Therefore, creating the right culture and advocating the right leadership support is fundamental for success. In the software services industry, there are compliance, security, and legal requirements that also need

to be considered. From the technical perspective, software generators and translators, code libraries (like GitHub), and the right architecture are also required to institute reuse.

3.3.3.4. Interview #17: Head of Application Development, Operations

Software reuse is the ability to develop software assets that can be implemented in multiple software products or applications, enabling resource savings (time and cost) and minimizing software development risks. Usually, from 5% to 15% of functionality can be reused for systems of engagement. Reusable software assets include application frameworks, design patterns, support subsystems (i.e. authorization and authentication), and functional components.

The expected benefits of software reuse are greater agility, reduced cost, improved quality, and productivity gains. Agility is achieved by accelerating the delivery of new applications. Cost reductions come from savings in design, coding, and testing activities. Software reuse can reduce coding efforts up to 10%, lower software development cost up to 5%, and increase velocity up to 10%. Improvements in quality are evaluated through higher reliability and better performance and security. Productivity is the result of developing software faster, at a lower cost and with higher quality. For information-driven businesses and digital companies, the agility to deliver new applications is key for competitive differentiation and long-term sustainability of the organization.

Implementing software reuse requires significant effort and investment: "*We put in place application development practice managers who are responsible to manage reusable*

software assets, look for productivity gains using innovative technologies, processes, and tools, and continuously improve product and service quality. The company also established a software asset management group to capture reusable assets, package them for reuse, and facilitate their deployment".

However, software services companies do not need the same level of maturity in their software reuse strategy as a software product company. Software product organizations own the entire lifecycle of their products as well as their IP, whereas a service company transfers both to the client who then determines product evolution. This business model places constraints on the software reuse strategy of a software services company.

CHAPTER 4: DISCUSSION

Not everything that counts can be counted and not everything that can be counted counts.
– Albert Einstein

Chapter 4 outlines the observations resulting from the analysis of the literature review and the interview data in support of the research questions introduced in Chapter 2. This chapter is organized in two sections. The first section presents general observations from contrasting the literature review and the interview data. The second section summarizes the analysis by company type.

4.1. General Observations

4.1.1. Definition of Software Reuse

My first observation is the consistency between the definitions of software reuse in the industry data collection and the definitions captured during the literature review. As expected, multiple definitions were provided across the interviews influenced by the point of view of the participant.

The differences referred mainly to the scope of reusable assets. The range varies from the atomic view, like a function or procedure, through work products across the software development lifecycle, to finished software systems. However, interview #12 disagreed with the idea of considering a full system as reusable. I believe this divergence from the literature comes from looking at software reuse as an activity exclusive to new software development whereas full system reuse is more implementation centric.

Interviews differed from the literature on some details. For example, many of the interviews discussed the idea of minimal effort in the process of reusing existing software components. In all cases, the goal is configuring, instead of customizing, the software artifacts to extract the benefits of reuse.

A few of the interviews included the concept of business process as part of the scope of reuse when these processes are enabled by reusable software. And within those interviews, a subset discussed the concept of standardization as a benefit of reuse. I would argue that some standardization is needed to adapt reusable artifacts into solutions and to effectively build and implement digital business processes. Thus, thinking of standardization as a requirement for reuse, rather than a benefit, may be more applicable.

To the extent there are differences in the definitions subscribed by the interviewees, I would also expect some differences in the benefits generated by reuse and the management practices applied.

4.1.2. Business Benefits of Software Reuse

Like with the definition of reuse, the literature review and the industry interviews led to similar descriptions of the benefits of reuse. I would argue that these findings illustrate that the business aspects of a software reuse strategy and the benefits from implementing reuse are well known.

As expected, the main benefits of software reuse discussed across the interviews are agility, quality, and productivity. In general, agility is targeted to the value generated by introducing software products into the market at faster speed. Quality is a function of more reliable tested software artifacts and related work products. And productivity is an umbrella term that covers more outcomes with less effort and consequently less cost.

A key insight obtained from contrasting the literature and the interview data is the impact of reuse on the bottom line of the organization. The benefits of reuse are evident, yet I recognize the lack of good empirical data to validate them, as mentioned in the literature review. The main challenge, I believe, comes from the difficulty of tracing the business outcomes back to reuse. Proving that profit is increased by introducing a product faster, lessening the number of software defects, or reducing cost would require adding metrics and overhead to track these benefits on top of the costs of implementing reuse. And there are benefits that are not considered true business outcomes like personal productivity or non-commercial reuse. In the words of one of the CIOs who participated in the interviews, most of the benefits of reuse may be "soft benefits".

From my point of view, the business value of software reuse originates from the benefits generated by implementing digital business processes and when reuse is applied to commercial products or services. In the case of the business processes enabled by software, the value of reuse to the bottom line of the organization is generated by the business outcomes that such digital processes provide. The business value would not come from the benefits in the software development process. Examples of this scenario

provided in the interviews include variable productivity, sales price increases, and more efficient use of working capital that materialize from improvements in the manufacturing, commercial, and other business processes across the company. Applying software reuse to improve agility, quality, and productivity within a business process may have an impact on the bottom line.

The other case to consider is reuse in commercial and services products. In this scenario, because of the direct connection to revenue, the organization is already collecting and measuring the impact of reuse on margin. Productivity gains in the development of software products will reduce the overall cost throughout the product lifecycle, promptly impacting profit. The same could be said for quality and agility. The impact on profit is clear and well documented. As stated by the Head of Product Management in one of interviews: "*70% of the margin of commercial software is predicated in software reusability*".

Another important takeaway is found in the idea that software reuse is a competitive differentiator. The literature review touched briefly on this potential scenario. However, many of the interviews validated this notion. Interview #2 referred to the importance of reuse when linked to differentiation. Similar opinions were provided in interviews #5, #6, #8, #11, #13, #15, and #16. Considering the importance of agility in software services companies, I would expect differentiation to be a key business outcome from reuse in these organizations. I also believe that interview #7 inferred the value of differentiation

when discussing the opportunities to capitalize data resulting from process standardization.

In summary, I would argue that the most valuable benefit of software reuse is competitive advantage through differentiation. Differentiation would enable the development of software products and services that competitors would be challenged to replicate. Differentiation should have a direct impact on the bottom line of the organization because of revenue increase and cost reduction from productivity and quality gains.

4.1.3. Implementation of Software Reuse

According to industry interviews, there seems to be consensus on the significant effort and investment required to implement a software reuse strategy. Only a couple of interviews were not fully onboard with the magnitude of the change. Nevertheless, all the interviewees agreed on the need to put in place the right business and technology leadership as well as the appropriate technology processes and tools to be successful at the reuse endeavor.

The literature review also revealed the tendency to place too much emphasis on reuse conflicting with differentiation as a potential pitfall of implementing reuse. Except for one comment in interview #10, the topic of managing the trade-off between reuse and differentiation was not discussed. This insight could mean that most interviewees did not consider this balancing act to be relevant or that the trade-off is an inherent challenge of implementing reuse. In either case, I would argue that managing the trade-off between

reuse and differentiation is a key success factor for effectively implementing and extracting value from reuse and should be included in the overarching business strategy.

Consistent with the literature, interviewees stated that people, processes, and tools are at the heart of implementing reuse. The interviewees did not provide details on how they addressed each of these elements. The only exceptions are the information provided in interviews #14 and #17 where the discussions included changes in the practice of products and projects delivery, enhancements in the organizational structure, and modifications to compensation models. Time constraints for the interviews and inexperience in the field of research could have led to this lack of information.

To close this section, I would like to highlight that implementing a reuse strategy would benefit from the perspective offered by system thinking[11]. As discussed before, an organization needs to consider various multi-disciplinary factors such as economic value, culture, business priorities, management and technical skills, compliance, security, legal, external market, among many others. System thinking provides a mental model for assessing the impact and extracting the value of reuse in a holistic manner, beyond each individual factor.

[11] System thinking is the cognitive process of reasoning about a question, circumstance, or problem explicitly as a system, a set of interrelated entities [31].

4.2. Analysis by Company Type

4.2.1. Digital Companies

As noted earlier, there are two sides to a digital enterprise. The commercial side, focused on creating new complementary revenue streams and the operational side, dedicated to improving internal business processes.

On the commercial side of digital, software reuse is an enabler for higher margins in the development of software products and services. The traditional benefits of reuse drive efficiency and speed in achieving those margin improvements. For internal operations, the main value of reuse comes from the benefits of digitalizing business processes with reusable software. Reuse does accelerate agility and generate productivity gains, but as mentioned in the previous section, these benefits are difficult to connect to profit. On the other hand, the benefits extracted from the improvements in business processes are directly tied to business KPIs.

Digital companies are evidently becoming more interested on implementing reuse as they transition from focusing on internal operations to building software-enabled commercial offerings. These organizations highly value the productivity and agility benefits of reuse.

4.2.2. Software Products Companies

For the two interviews in this section, the software reuse strategy is intimately connected with the overall business strategy. Although the definition of software reuse differed

between the interviews, they are both closely related to the concepts described in the literature review.

In software products companies the traditional benefits of reuse are apparently better connected to the bottom line of the organization. I would argue that this condition is present because building products at scale is the raison d'être for these organizations. That may not be the case for digital companies, where economies of scope and mass production are predominant. Just like increasing productivity in an industrial manufacturing company (i.e. building more widgets with less cost) would have an impact on profit, software reuse could have a similar effect in a software products company. However, these benefits could be hard to measure and validate.

Interestingly, both participants agreed that software reuse is difficult to implement in an organization. The main impediments are the support from business leadership, building the right business case to secure funding, and putting in place the foundational technical framework to facilitate reusability in day-to-day operations.

4.2.3. Software Services Companies

Software services companies introduced the concept of commercial value of reuse. Because of their business model, reuse is only notable when client deliverables are built with reusable software components. And when applied to differentiating capabilities, reuse becomes a competitive advantage.

By reusing software components these companies can also experience agility, productivity, standardization, and quality gains in the planning, development, and delivery of software solutions and services. Because these benefits are easily connected to margin, thus to the bottom line, they are very well documented and understood.

The agreement across these companies is that software reuse requires the implementation of the right business and technology frameworks, which translate into effort and financial investment. Additionally, because of the commercial nature of their operations (i.e. customer ownership of the final software product), the reuse capabilities may be limited by legal, compliance, and even security requirements.

According to the interviews, software service companies have the highest level of maturity in applying reuse to their business operations. In my opinion, reuse adds efficiency and performance to their labor-intensive operations, where small improvements in productivity and speed can generate significant gains in profit.

4.2.4. Comparative View

As mentioned at the beginning of this chapter, the similarities of the industry data and the literature review are evident. The analysis of the industry data also shows considerable similarities across the interviews, with distinctive key insights by company type. These similarities and differences are illustrated in Table 3.

Finally, I believe the interviews have complemented, rather than contradicted, the literature review. The insights from the commercial aspects of reuse contributed to expand the scope of reuse and its benefits in the organization. In my opinion, the main contribution from the industry data to this research is the concept of competitive advantage through differentiation as the most valuable benefit of software reuse.

	Digital Companies	**Software Products Companies**	**Software Services Companies**
Definition of software reuse	• Ability to solve similar business problems by configuring existing software assets. • Configuration instead of customization; minimal rework effort. • Artifacts include atomic level components through finished software products to standard business processes. • Commercial value from implementation in commercial products.	• Capability to assemble new products from available artifacts; maximize existing assets. • Scope ranges from ideas to components, excluding final products. • Business processes and enabling software are reusable. • Some level of standardization and portfolio management required. • Categorization required; premeditated planning.	• Capability to configure developed software assets for other projects and clients. • Maximization of assets by building reusable core components. • Scope includes knowledge, software infrastructure, and finished applications. • Intentional design and architecture required.
Benefits of software reuse	• Agility, quality, and productivity; difficult to quantify and validate. • Traditional benefits applicable to the process of building	• Agility and TCO reduction; financial efficiency. • Innovation. • Commercial value from differentiation.	• Commercial value by delivering client services built on reusable artifacts. • Cost reduction, agility, quality, and risk mitigation.

	software; "Soft benefits". • TCO reduction, improved cost-to-revenue performance; "Hard benefits". • Value from implementing software-enabled business processes built with reusable software. • Competitive advantage through differentiation; impact on the bottom line. • Faster delivery of commercial products at a lower cost; higher margin.	• Most benefits are "soft", difficult to measure and connect to the bottom line. • Competitive differentiation is a "hard benefit"; direct impact on profit. • Real value comes from implementing reusable software-enabled business processes; impact to business KPIs.	• Margin is the main KPI impacted. • Standardization of software development and business processes. • Enabler for competitive differentiation.
Implementation of software reuse	• Aligned to business strategy; prioritization. • Trade-offs between long-term strategy and short-term results. • High technical maturity. • Significant cultural challenge yet not technically difficult. • Governance and supporting infrastructure.	• Difficult to implement due to cultural and management challenges. • Allocation of human and capital resourcing; long-term investment. • Business strategy to balance agility and cost. • Management skills to manage trade-offs between reuse and differentiation. • High level of maturity in software development processes.	• Significant investment and effort. • Technology roadmap aligned to business strategy; leadership and long-term vision. • Facilitated by governance and financial incentives. • Cultural adaptation. • Consideration to compliance, security, and legal requirements.

Table 3. Comparative view of key insights from interviews

CHAPTER 5: CONCLUSION

One worthwhile task carried to a successful conclusion is better than fifty half-finished tasks.
– B.C. Forbes

Unquestionably, reuse is beneficial for companies involved in building software applications and products. Nevertheless, the decision to implement a reuse strategy needs to be tightly connected to the business strategy and should be prioritized through its business value.

Because the dual goals of digital companies are to improve internal operations while creating new revenue streams, reuse is particularly critical for these organizations. On the internal operations side, software reuse supports business process improvement through efficient digitalization. In the commercial area, software reuse enables higher margins by developing products and services faster and at a lower cost.

Considering that competitive advantage through differentiation is the most valuable benefit of software reuse, all companies undertaking their digital journey should consider establishing a reuse strategy to protect the long-term sustainability of the organization. However, organizations should also consider applying system thinking to their reuse strategy to fully assess the impact of reuse and extract the full value of its implementation.

REFERENCES

[1] "Minds + Machines 2012: Jeff Immelt Keynote," 29-Nov-2012. [Online]. Available: https://www.youtube.com/watch?v=SvI3Pmv-DhE.

[2] G. Westerman, D. Bonnet, and A. McAfee, *Leading digital: turning technology into business transformation*. Boston, Massachusettes: Harvard Business Review Press, 2014.

[3] J. Savolainen, J. Kuusela, M. Mannion, and T. Vehkomäki, "Combining Different Product Line Models to Balance Needs of Product Differentiation and Reuse," presented at the High confidence software reuse in large systems: 10th International Conference on Software Reuse, ICSR 2008, Beijing, China, May 25-29, 2008, Berlin ; New York, 2008.

[4] P. Meehan and D. Scheibenreif, "2016 Strategic Roadmap for Digital Business Transformation," G00303322.

[5] D. Soule, A. Puram, G. Westerman, and D. Bonnet, "Becoming a Digital Organization: The Journey to Digital Dexterity," MIT Center for Digital Business, Working Paper #301, 2016.

[6] R. Leach, *Software Reuse: Methods, Models, and Costs*, Second Edition. Ronald J. Leach.

[7] P. Freeman, "A Perspective on Reusability," *Tutor. Softw. Reusability*.

[8] G. Lindfield and A. Hyland, "Classification of assets: Joint World Bank and IFRS Foundation 'train the trainers' workshop hosted by the ECCB, 30 April to 4 May 2012," presented at the "Train the Trainers" Workshop, Mar-2012.

[9] W. Tracz, "Software Reuse: Motivators and Inhibitors," *Tutor. Softw. Reuse Emerg. Technol.*, p. 50, 1988.

[10] M. D. McIlroy, "'Mass Produced' Software Components. NATO Software Engineering Conference," NATO Science Committee, Garmisch, Germany, Jan. 1969.

[11] I. Sommerville, *Software Engineering*, Tenth edition. Boston: Pearson, 2016.

[12] R. Prieto-Diaz and P. Freeman, "Classifying Software for Reusability," *IEEE Softw.*, vol. 4, no. 1, p. 6, 1987.

[13] V. Otchere, "A Methodology For Implementing Software Reusability at Ford Motor Company: A Pragmatic Approach," Massachusetts Institute of Technology, Cambridge, Mass, 1993.

[14] L. M. Northrop and P. C. Clements, "A Framework for Software Product Line Practice, Version 5.0." Software Engineering Institute, 2006.

[15] H. Kaindl, R. Popp, R. Hoch, and C. Zeidler, "Reuse vs. Reusability of Software Supporting Business Processes," presented at the Software Reuse: Bridging with Social-Awareness, Limassol, Cyprus, 2017.

[16] J. W. Hooper and R. Chester, *Software reuse: guidelines and methods.* Springer, 2013.

[17] O. L. De Weck, D. Roos, and C. L. Magee, *Engineering systems: meeting human needs in a complex technological world*. Cambridge, Mass: MIT Press, 2012.

[18] W. Tracz, "Software Reuse Myths Revisited," Loral Federal Systems Company, Owego, NY, 0270–5257/94, 1994.

[19] F. P. Brooks, *The mythical man-month: essays on software engineering*, Anniversary ed. Reading, Mass: Addison-Wesley Pub. Co, 1995.

[20] M. L. Griss, "Software Reuse: From Library to Factory," *IBM Syst. J.*, Aug. 1993.

[21] P. Bourque, R. E. Fairley, and IEEE Computer Society, *Guide to the software engineering body of knowledge*. 2014.

[22] S. A. Ross, R. Westerfield, and B. D. Jordan, *Fundamentals of corporate finance*, 6th standard ed. Boston, Mass: McGraw-Hill, 2003.

[23] E. Grifell-Tatjé and C. Lovell, "Profits and Productivity," *Manag. Sci.*, vol. 45, no. 9, pp. 1177–1193, 1999.

[24] C. Criscuolo, "Productivity Is Soaring at Top Firms and Sluggish Everywhere Else," *Harvard Business Review*, 24-Aug-2015. [Online]. Available: https://hbr.org/2015/08/productivity-is-soaring-at-top-firms-and-sluggish-everywhere-else. [Accessed: 24-Feb-2017].

[25] M. A. Cusumano, *The business of software: what every manager, programmer, and entrepreneur must know to thrive and survive in good times and bad*. Free Press, 2014.

[26] L. Amorim and M. Mendonça, "A Method to Support the Adoption of Reuse Technology in Large Software Organizations," presented at the Software reuse: bridging with social-awareness: 15th international conference, ICSR 2016, Limassol, Cyprus, June 5-7, 2016: proceedings, [Cham] Heidelberg, 2017.

[27] O. Lewis and W. Buchanan, "Performance Issues of Variability Design in Embedded System Application Families," PhD, Edinburgh Napier University, 2000.

[28] jeyakumar.rathinasamy, "Configuration Vs. Customization." [Online]. Available: https://archive.sap.com/discussions/thread/1027498. [Accessed: 29-Apr-2017].

[29] "What is Agile Software Development?," *Agile Alliance*, 29-Jun-2015. .

[30] K. Beck *et al.*, "Manifesto for Agile Software Development," *Agile Manifesto*. [Online]. Available: http://agilemanifesto.org/iso/en/manifesto.html. [Accessed: 28-Apr-2017].

[31] E. Crawley, B. Cameron, and D. Selva, *System architecture: strategy and product development for complex systems*. Boston: Pearson, 2016.

[32] P. F. Drucker, *The effective executive*. HarperCollins e-Books, 2014.

APPENDIX A

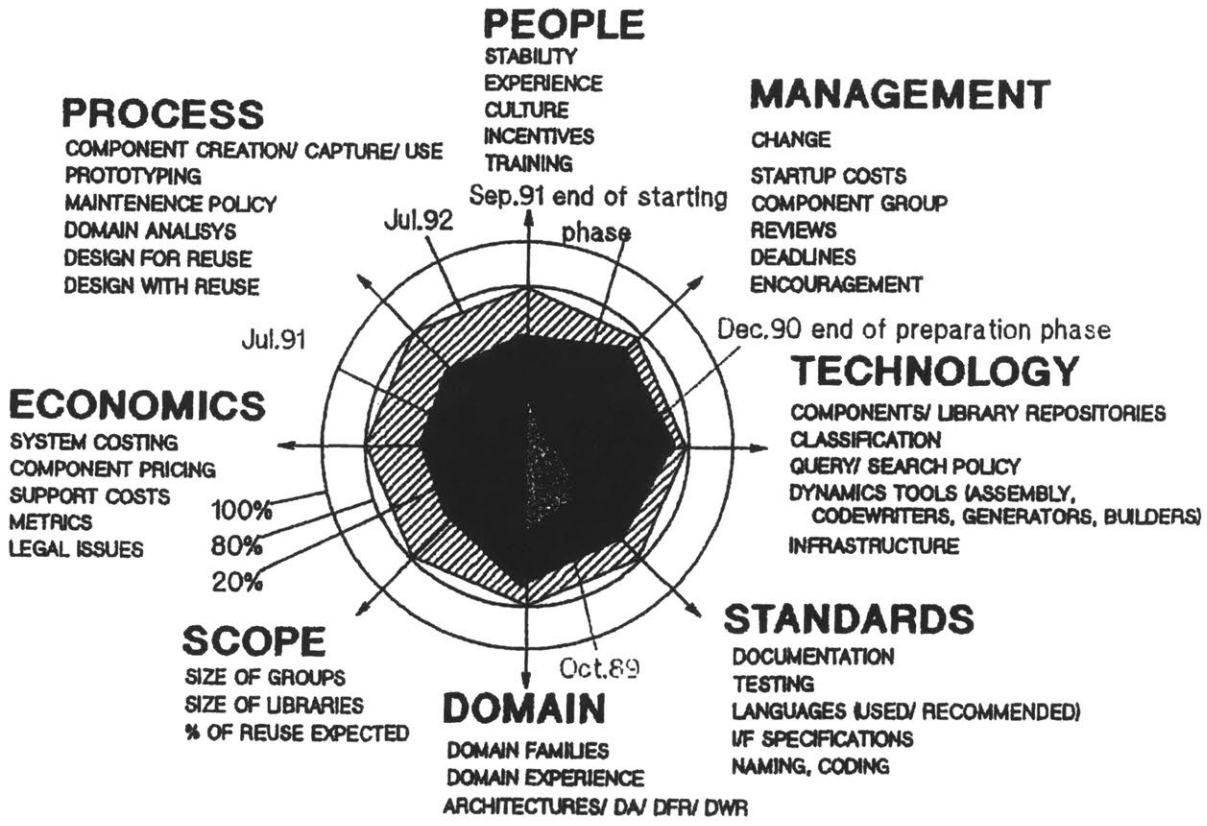
Approaches that support software reuse and reusability

Approach	Description
Application frameworks	Collections of abstract and concrete classes are adapted and extended to create application systems.
Application system integration	Two or more application systems are integrated to provide extended functionality.
Architectural patterns	Standard software architectures that support common types of application systems are used as the basis of applications.
Aspect-oriented software development	Shared components are woven into an application at different places when the program is compiled.
Component-based software engineering	Systems are developed by integrating components (collections of objects) that conform to component-model standards.
Configurable application systems	Domain-specific systems are designed so that they can be configured to the needs of specific system customers.
Design patterns	Generic abstractions that occur across applications are represented as design patterns.
ERP systems	Large-scale systems that encapsulate generic business functionality and rules are configured for an organization.
Legacy system wrapping	Legacy systems are 'wrapped' by defining a set of interfaces and providing access to these legacy systems.
Model-driven engineering	Software is represented as domain models and implementation independent models and code is generated from these models.
Program generators	A generator system embeds knowledge of a type of application and is used to generate systems in that domain from a user-supplied system model.
Program libraries	Class and function libraries that implement commonly used abstractions are available for reuse.
Service-oriented systems	Systems are developed by linking shared services, which may be externally provided.
Software product lines	An application type is generalized around a common architecture so that it can be adapted for different customers.
Systems of systems	Two or more distributed systems are integrated to create a new system.

Source: I. Sommerville, *Software Engineering*, Tenth edition. Boston: Pearson, 2016.

APPENDIX B

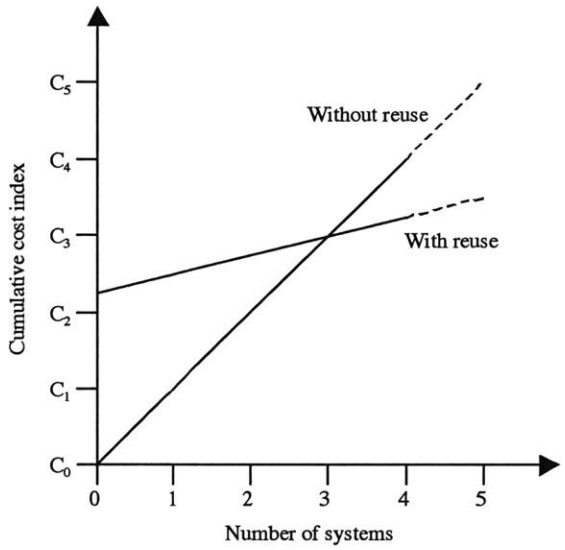
Reuse factors identified at Hewlett-Packard (HP)



Source: M. L. Griss, "Software Reuse: From Library to Factory," *IBM Systems Journal*, Aug. 1993.

APPENDIX C

Visual example of the cumulative cost of building software systems



This figure shows an example of the cumulative cost of building systems with and without reuse. There is evidence that suggests that the average convergence occurs after three systems.

Source: O. Lewis and W. Buchanan, "Performance Issues of Variability Design in Embedded System Application Families," PhD, Edinburgh Napier University, 2000.

APPENDIX D

Interview questions for the industry data collection

1. What is your definition of software reuse? What do you intend to reuse?
2. What benefits or business outcomes do you expect to achieve with software reuse?
3. How do you connect the benefits outlined in the previous question to the bottom line of the company?
4. Is well documented that the three major reasons for a software reuse strategy are speed-to-market, productivity, and quality. For productivity, a large part of the literature discusses personal productivity. What level of productivity do you expect to achieve with software reuse and how do you connect it to the financial health of your company?
5. Based on research, reuse is not as simple as one may think. Enabling software reusability could be expensive, especially when considering that impediments are predominantly non-technical and socioeconomic.
    a. How do you manage the long-term reuse strategy with short-term business results?
    b. What is your business case to implement a software reuse strategy?
    c. What are the biggest challenges encountered when implementing a software reusability strategy and how do you handle them?
6. In the technical side, what approaches (policies, methodologies, processes, tools) do you implement to facilitate and incentivize software reuse?
7. How does Predix fit in your software reuse strategy? (for GE executives only)
8. According to literature, reuse maximization accelerates time-to-market and lowers cost but then diminishes market differentiation. How do you approach this trade-off?
9. What level of maturity in software reuse would be needed by your organization when compared with software-product companies? (for non-product companies)
10. Who else in your organization do you recommend I talk to?