INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

State of the Art Report

Deliverable

Deliverable data

Deliverable Name	**State of the Art Report**
Work Package	*WP1*
Partners involve	*Addalot, Kugler Maag Cie, Lero, Lund University, Softhouse, Sigrun*
Date	*9/12/2015*
Status	*Approved*

Document history

Version	Date	Author /Reviewer	Description
0.1	17.03.2015	Ulf Asklund	All individual drafts joined into this document.
0.2	23.05.2015	Anders Sixtensson	Added Disciplined Agile Delivery
0.3	12.06.2015	Martin Höst	Revision of existing text
0.4	16.06.2015	Klaas-Jan Stol	Overall clean up of text, lay-out, logos etc.
0.5	17.07.2015	Miguel A. Oltra	Revision and added comments for further improvement.
0.6	2.9.2015	Ulf Asklund	Merge of review comments
0.7	9.12.2015	Klaas-Jan Stol	Restructuring of the document; revise text overall; add introduction section.
0.8	18.12.2015	Martin Höst	Rewrite of the Experience Factory section
0.9	11.01.2016	Klaas-Jan Stol	Further textual improvements; add figure by Horst Hientz / Kugler Maag that provides an overview of how the different models relate to one another; add justification for the selected models that are discussed in this document.

Table of Contents

1 Introduction

This document presents a review of the most relevant existing models and frameworks. The purpose is to give a short overview of existing models and to evaluate the extent to which they are suitable to achieve the objectives that the SCALARE project has set forth, namely the support of organizations in scaling their software development and delivery capacity. The focus is on characteristics relevant for the purpose of SMF. As the SMF covers three large domains – organization, processes and methods, and products and services – many other existing models have some relationship to the SMF, however, often focusing on one of these aspects.

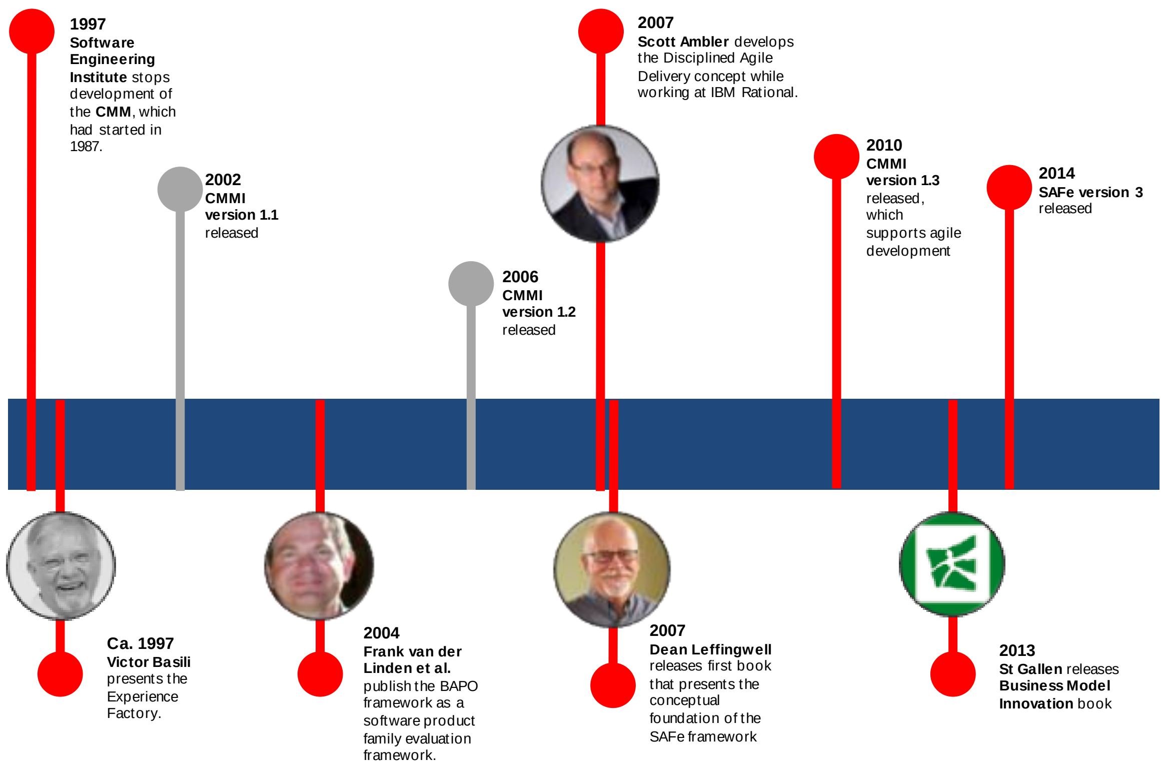
1.1 List of models included in this document

The selected models that are considered to be most relevant to the SCALARE project are:

- Business, Architecture, Process, and Organization Model (BAPO)
- Business Model Innovation (BMI)
- Capability Maturity Model (CMM)
- Experience Factory (EF)
- Scaled Agile Framework (SAFe)
- Discipline Agile Delivery (DAD)

These are well known and as such it is useful to explain these models in sufficient detail so that their relationship to the SCALARE project and the SMF is clear. The software engineering literature defines many software process improvement frameworks, but the majority of these are not widely adopted, well established, or known. Hence, we did not discuss these models, also because it is clear that software process improvement frameworks cover only a single dimension of the scaling software phenomenon (which is discussed in more detail later in this chapter).

While all models are evaluated in contrast with the SMF, this deliverable does not discuss the SMF in detail. For this, we refer to a separate deliverable that presents the SMF—at the time of writing (Winter 2015/2016), the SMF is still being developed. Figure 1.1 presents a timeline of selected key events with regards the models that are presented in this report.



1997
Software Engineering Institute stops development of the CMM, which had started in 1987.
2002
CMMI version 1.1 released
2006
CMMI version 1.2 released
2007
Scott Ambler develops the Disciplined Agile Delivery concept while working at IBM Rational.
2010
CMMI version 1.3 released, which supports agile development
2014
SAFe version 3 released
Ca. 1997
Victor Basili presents the Experience Factory.
2004
Frank van der Linden et al. publish the BAPO framework as a software product family evaluation framework.
2007
Dean Leffingwell releases first book that presents the conceptual foundation of the SAFe framework
2013
St Gallen releases Business Model Innovation book

Figure 1.1. Timeline of key events: development of frameworks and models over time.

1.2 Structure and Definitions

Models are created for a variety of purposes, one of which is to communicate a complex topic. Models help to explain the world and to communicate information with others. To create a model, we rely on abstractions of the real world and focus on the most important characteristics in order to fulfil the purpose we have using the model. It is hard, or even impossible, to evaluate models and frameworks and explain what is good and not good without knowing the context in which the model is to be used. That is, different models can be used in different situations. Evaluating models should be done in a specific context and focus on how well the models support the goals defined by that context. In the SCALARE project, this context is the ability to help organizations understand what they need to do in order to “scale” and how this can be done, and it is in this broad and general context that we evaluate the different models.

The focus of the SCALARE project is the ever-increasing role and importance of software for organizations, even those that did not previously consider themselves to be software organizations. The following is a list of scenarios that are typical for the SCALARE context (although it is not exhaustive):

- **Transitioning to software-based platforms**. Companies that are traditionally delivering products based on hardware platforms are now moving en masse to software-based solutions. A clear example of this is Husqvarna, who are moving away from petrol-based garden equipment to electronically driven equipment, which require software for their control. This in turn means that these companies now have to learn how to develop software-based platforms and solutions, but they may not be aware of the specific strategies that are available to them, or how to adopt those. One such strategy could be a product family, for example.

- **Tailoring existing software development methods to new environments**. Companies that are already developing software may find themselves in need to adopt and adapt new methods to their specific development context. An example of this is QUMAS, a company operating in a regulated environment and consequently they are subject to regular audits by regulatory bodies. As the need to deliver software more frequently and maintain a steady pace, QUMAS has adopted an agile approach for their software development, but had to tailor this approach to their specific needs, as agile methods are traditionally not suitable for regulated environments.

- **Extending the software development capacity.** Companies are also faced with a need to deliver increasingly more software, or take advantage of software that is already available, whether as internally developed components or externally available open source projects. An example of this scenario is Sony Mobile, who are relying extensively on open source components for delivery of software in their mobile phones, which are based on the open source Android operating system. Consequently, Sony Mobile needs to participate in a wider community of open source developers, and learn how to interact with them in a sustainable manner. Also, to leverage the use of internally developed assets, Sony Mobile are seeking to adopt the open source development paradigm internally – or inner source as it is termed. This introduces the concept of internal “open source” projects with ad-hoc development teams delivering assets. This scenario represents the scaling of software development capacity by scaling up the organization in different directions.

Thus, we have a context where organizations must introduce novel strategies to scale up their software development capacity whether it is in the form of the introduction of software-based solutions (moving away from non-software solutions), the tailoring of existing software development methods, or adopting new sourcing strategies such as opensourcing and innersourcing. Figure 1.2 below captures this context of scaling in multiple dimensions.

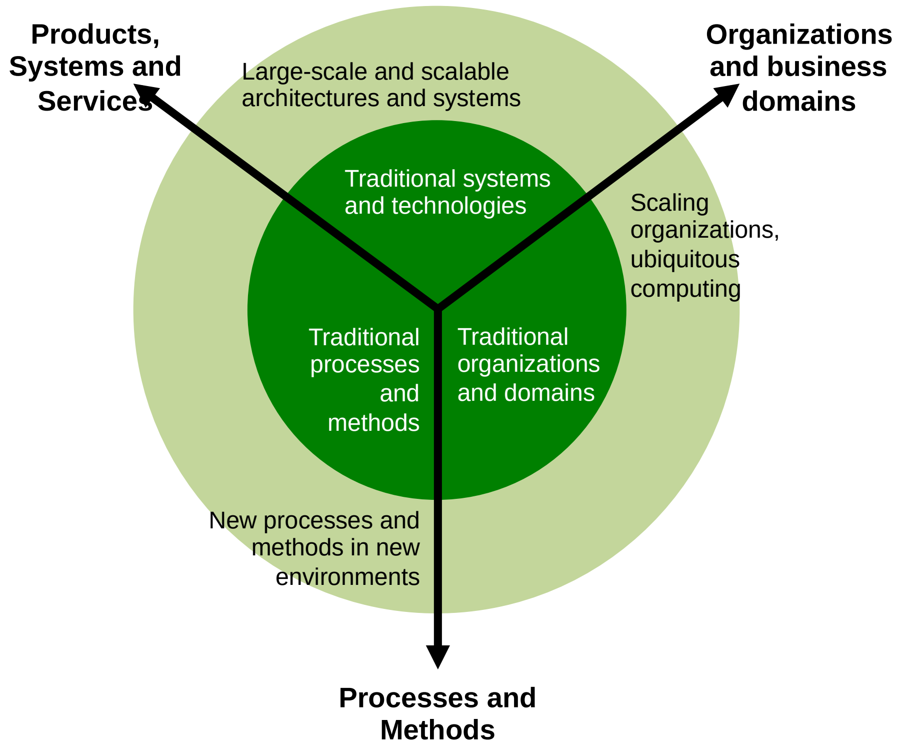

Products, Systems and Services
Organizations and business domains
Large-scale and scalable architectures and systems
Traditional systems and technologies
Scaling organizations, ubiquitous computing
Traditional processes and methods
Traditional organizations and domains
New processes and methods in new environments
Processes and Methods

Figure 1.2. Multi-dimensional scaling of software organizations

In each scenario, organizations are **driven** to embark on these scaling transformations by specific drivers. General drivers are the need to deliver software more quickly, at lower cost, and of higher quality. Specific drivers include the need to deliver new and innovative solutions, and adopt better strategies that secure the future ability to deliver new customer value quickly.

Figure 1.3 illustrates (a) the drivers that an organization may have to scale (top); (b) the current capabilities that an organization has (left); (c) the desired, or 'wanted' abilities (right); and (d) the software model that sits in between, and considers the three key dimensions, namely product, process, and organization. These three dimensions are recurring in the software engineering literature; indeed, the BAPO model (discussed I more detail in Chapter 2) includes them as does the software product line literature published by the SEI (e.g. Northrop 2004).

The challenge of this multi-dimensional scaling can be broken into the need to cross the chasm between the "current" state (left-hand side) and the "desired" state (right-hand side). In other words, this deliverable aims to present the other models and frameworks that have been suggested to achieve the desired abilities of organizations to deliver software. Therefore, this report does not focus on business drivers, nor does it describe the SMF itself.

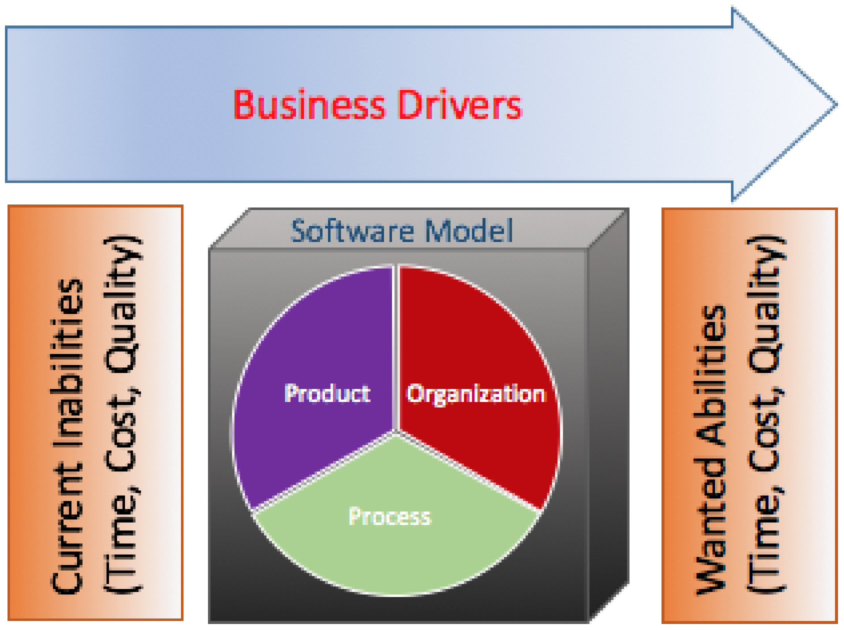

Business Drivers
Current Inabilities (Time, Cost, Quality)
Software Model
Product
Organization
Process
Wanted Abilities (Time, Cost, Quality)

Figure 1.3. Overview of the Scaling Management Framework: Business Drivers, Current inabilities, Wanted abilities.

It is important to define what "scaling" means in the context of the SCALARE project. Traditionally, the term is interpreted as "growing in size" – organizations 'scale', meaning they grow in number of departments, employees, teams, projects, products, etc. We refer to this as a narrow definition of scaling. The SCALARE project adopts a wider interpretation, namely not only scaling in size, but also the adaptation of an organization's ability to deliver software in different contexts, under new constraints, in collaboration with a variety of partners, which may be internal or external (for example through outsourcing strategies). Thus, we define 'scaling software' as shown in Figure 1.4:

The ability of an organization to:

A. **Deliver software that is growing in size, complexity and other product-related features,**

B. **Adopt and adapt software development processes and methods to produce that software,**

C. **Grow its capacity to deliver the software in terms of required organizational and business structures.**

Figure 1.4. Definition of Scaling Software as a multi-dimensional phenomenon.

Clause A refers to the traditional definition. Clause B refers to the ability of scaling processes and methods, for example, the adaptation and tailoring of agile methods to new contexts. Clause C refers to the ability for an organization to grow its capacity either internally or externally through the forging of partnerships, outsourcing, and other strategies.

The goal of this deliverable is to evaluate some of the most prevalent models available in industry, proposed by practitioners and researchers, for their ability to support organizations in their "scaling" transformation according to the definition provided above. The context and exemplary scenarios presented above suggest that any model to support industry in this endeavour offers support on the following aspects:

- **Scope and focus:** what are the context and aspects that the model focuses on? The three main dimensions defined above are products, systems and services, processes and methods, and organizations and business domains. Some models such as the Scaled Agile Framework (SAFe) focus exclusively on one dimension, such as the "scaling up" of using agile methods for example. The scope and focus define the **goal** of the model in supporting organizations to scale.

- **Assumptions underpinning the model.** Each model has certain assumptions which may either be implicit or explicitly stated. These **assumptions constrain the context** within which the model might be applicable, and explain the "paradigm" that the originators of the model might have had in mind when defining the model. For example, the SAFe framework (Scaled Agile Framework) mentioned above, assumes a certain structure of organizations that consists of teams, projects, programs and portfolios. Not all organizations, however, may be structured in this way, which could render the SAFe framework less or not suitable for a specific context.

- **Approach and vision of the model.** Each model suggests a certain set of steps, or otherwise that **provides guidance** to an organization. The level of this guidance may vary from being very concrete and detailed to providing an overall roadmap that still facilitates a certain level of tailoring. Models tend to have a certain "vision" – a suggested destination to pursue, although many models acknowledge that any scaling transition is a continuous and infinite process of improvement that cannot be completed.

Each of the models listed earlier in this chapter will be described along these three aspects: (1) scope and focus, (2) underpinning assumptions, and (3) approach and vision.

The remainder of this report is structured as follows Sections 2 to 7 present analyses of each of the models listed above, respectively. Section 8 concludes by summarizing the findings and re-iterating the need for an overall Scaling Management Framework, which is one of the key outputs of the SCALARE project.

2 Business Architecture Process Organization (BAPO)

2.1 Introduction

The BAPO model for product line engineering was a result of the ITEA-CAFÉ project (2001-2003) (van der Linden 2002; van der Linden et al. 2004; van der Linden et al. 2007). BAPO assumes that there are four key concerns in software engineering that need to be addressed to develop software products: *Business, Architecture, Process and Organisation,* hence the acronym BAPO.

The BAPO model argues that software product line development must address four types of concerns: Business concerns, Architectural concerns, Process concerns and Organisational concerns. Van der Linden et al. (2004, 2007) describe these as follows:

- **Business**: the costs and profits of the software, the strategy of applying it and the planning of producing it.
- **Architecture**: the technical means to build the software.
- **Process**: the roles, responsibilities and relationships within software development.
- **Organisation**: the people and organisational structures that execute the software development.

The first version of the BAPO framework considered the four factors (B, A, P, O) in a single dimension, as shown in Figure 2.1.

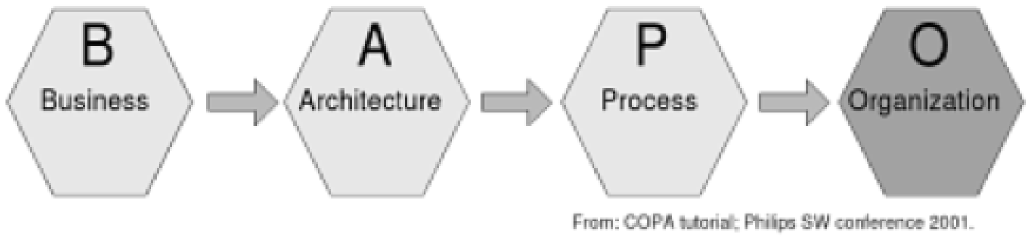


Figure 2.1. Four factors of the BAPO framework (source: COPA Tutorial, Philips SW Conference 2001)

A revision of the BAPO framework extended the relationships between the different concerns. In this version, while all categories (B, A, P, O) are connected, the BAPO model has a primary "path" follows the letters BAPO in that order; Business considerations affect a products architecture, which affects the process to follow, which affects the organisational structure. This is shown in Fig. 5 below.

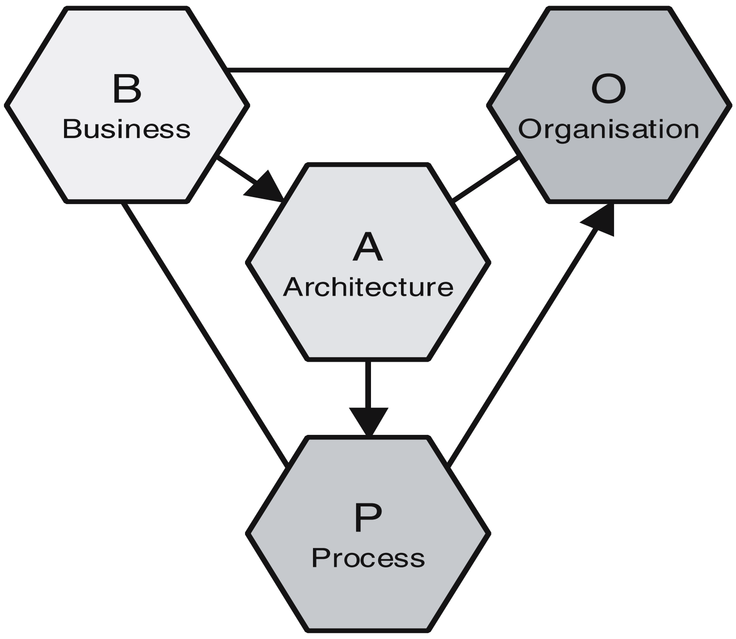


Figure 2.2. Architecture has a central position in the BAPO framework

2.2 Evaluation

2.2.1 *Scope and Focus*

The BAPO framework has an exclusive focus on *software-based systems* (that contain software, which includes embedded software systems), and its key goal is to align an organization's business concerns with the product architecture, the development processes and the organizational structures within the company. Having emerged from ITEA projects on software product lines (CAFÉ, FAMILIES, etc.), the BAPO framework is strongly architecture-focused, which is also implied by the central position that the architecture concept takes in Figure 2.2.

2.2.2 *Underpinning Assumptions*

The BAPO framework considers both the “problem space” and the “solution space” to be constant. The problem space is ‘constant’ in that it deals with the need to manage a line of related products that may be constructed from a common set of assets (e.g. components). The solution space is ‘constant’ in that, while solutions vary as new products are developed, the strategy, or “formula” remains the same: developing an appropriate systems architecture based on business considerations that is developed following an appropriate process within an organizational structure that facilitates the development of the system. These four categorical concerns are recurring in each product development in the product line. In this sense, the BAPO framework offers a single type of mechanisms for organizations to manage their software products and make appropriate business, process, and organizational choices. BAPO is one of the many guiding frameworks that can be used by organizations, but only *after* a decision has been taken to adopt a product line strategy.

The BAPO framework can be used to evaluate an organization’s current state-of-practice (as described extensively in the case studies presented (van der Linden et al. 2007). However, the BAPO framework assumes a *linear evaluation*, starting with business aspects, followed by architectural, process and organizational aspects.

The BAPO model is useful for those organizations that seek to adopt a software product line strategy – however, the drivers that organizations facing may address very different forces, for example, the need to move from a hardware-based to software solutions. The example of Husqvarna is typical, as they are moving from a fuel-based product offering to equipment running on electronic platforms (driven by software), with an ultimate goal of connecting all equipment into a “Connected Garden”. Thus, while a product line approach may be a suitable strategy, this choice is not straightforward; should Husqvarna select this strategy, the BAPO framework can offer guidance. However, many organizations similar to Husqvarna are still evaluating their “problem space” and have not yet decided to embark on a SPL strategy.

Business drivers affect software organizations in different ways. For some organizations, a certain business driver (e.g. shortening time-to-market) may result in changes relating to a product (for example, changes to its architecture – such as transforming to a software product line approach), while for others it may result in process changes (e.g., adopting agile methods). For others still it might result in changes that address organizational aspects, such as outsourcing initiatives (incl. innersourcing and crowdsourcing) and establishing an ecosystem of different parties around a single keystone platform product (such as Apple's iPhone, Google Android, and the WordPress content management system).

BAPO does not address trends such as the 'softwaretization' and 'servitization' of products. BAPO already assumes that the solution is software-based, whereas the SMF takes into consideration those scenarios such as found in Husqvarna, who are moving from petrol-based platforms to electronic (software-driven) platforms.

2.2.3 *Approach and Vision*

Figure 2.3 shows the Family Evaluation Framework (FEF) (the term "product family" is used as a synonym for "product line") that defines a number of different levels of maturity in each of the BAPO dimensions. The FEF is based on the BAPO model by refining each BAPO dimension and defining a set of maturity levels. This can be used by organizations to assess and evaluate their product line approach.

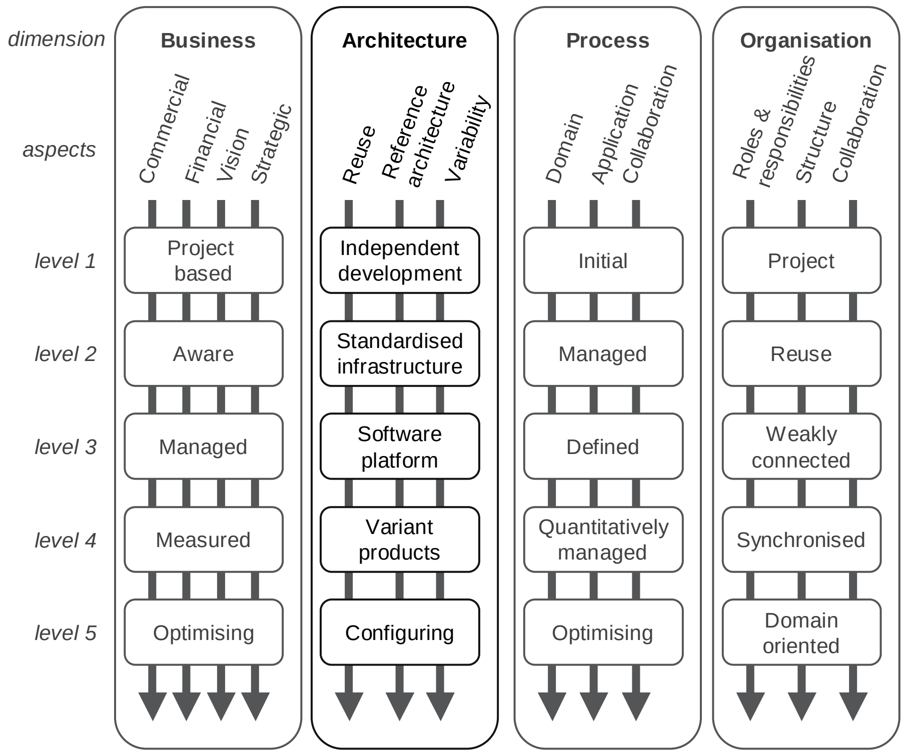


Figure 2.3. Family Evaluation Framework (FEF) defines maturity levels within the BAPO model (van der Linden et al. 2007)

3 Scaled Agile Framework (SAFe)

3.1 Introduction

The Scaled Agile Framework (SAFe) offers an overall guiding framework to organizations that wish to scale up the use of agile methods. The SAFe is developed by Dean Leffingwell, and the current version is 3.0.

3.2 Evaluation

3.2.1 *Scope and Focus*

SAFe is a framework for large-scale software development. SAFe does not assume any assessment of the current state of practice within an organization. The gap between the starting organization and the stepwise change towards a more "SAFe-like" organization is not part of the framework itself; however, assessing such a gap is an important aspect in any scaling and transformation scenario envisaged by the SCALARE project.

3.2.2 *Underpinning Assumptions*

The Scaled Agile Framework (SAFe) addresses the social dimension of the scaling problem by organizing group cohesion around the "Program." In the SAFe, the Program represents a value chain that links customers, suppliers, business participants, and technical staff in a coordinated effort to deliver new solutions to meet business demands. The Program coordinates the efforts of multiple agile teams synchronized to a common cadence to optimize communication, reduce bottlenecks, and deliver value at a continuous and sustainable pace.

Business drivers for a SAFe-driven change typically include low throughput, long lead times and low quality, and the desire to work more agile. The SAFe model does not address the need to formulate business drivers.

The SAFe framework also does not address the various ways to package and deliver an offering to the market using open-source components, integration with legacy systems, and so on. SAFe's 'home ground' that seems to be most suitable is that of software-intensive products developed by an organization without dependencies on other sourcing forms including outsourcing, opensourcing, etc.

3.2.3 *Approach and Vision*

As Figure 3.1 shows, the SAFe identifies different layers within the organization. The bottom layer is the “team” level, which explains how teams are organized and the way their development process is coordinated. The Program layer is defined at a higher level, and links the activities from different agile teams (at the Team Level) together into a ‘Program”. Additional roles and ceremonies are defined in this layer. The top layer is called “Portfolio” and considers how the set of projects together that is delivered by an organization offer value to customers. At this level an organization can orchestrate the projects that are embarked on, which are evaluated on a business-strategic level. That is, projects are not evaluated purely on their own merit, but primarily from a strategic perspective, by answering the question whether or not the project is critical to the organization.

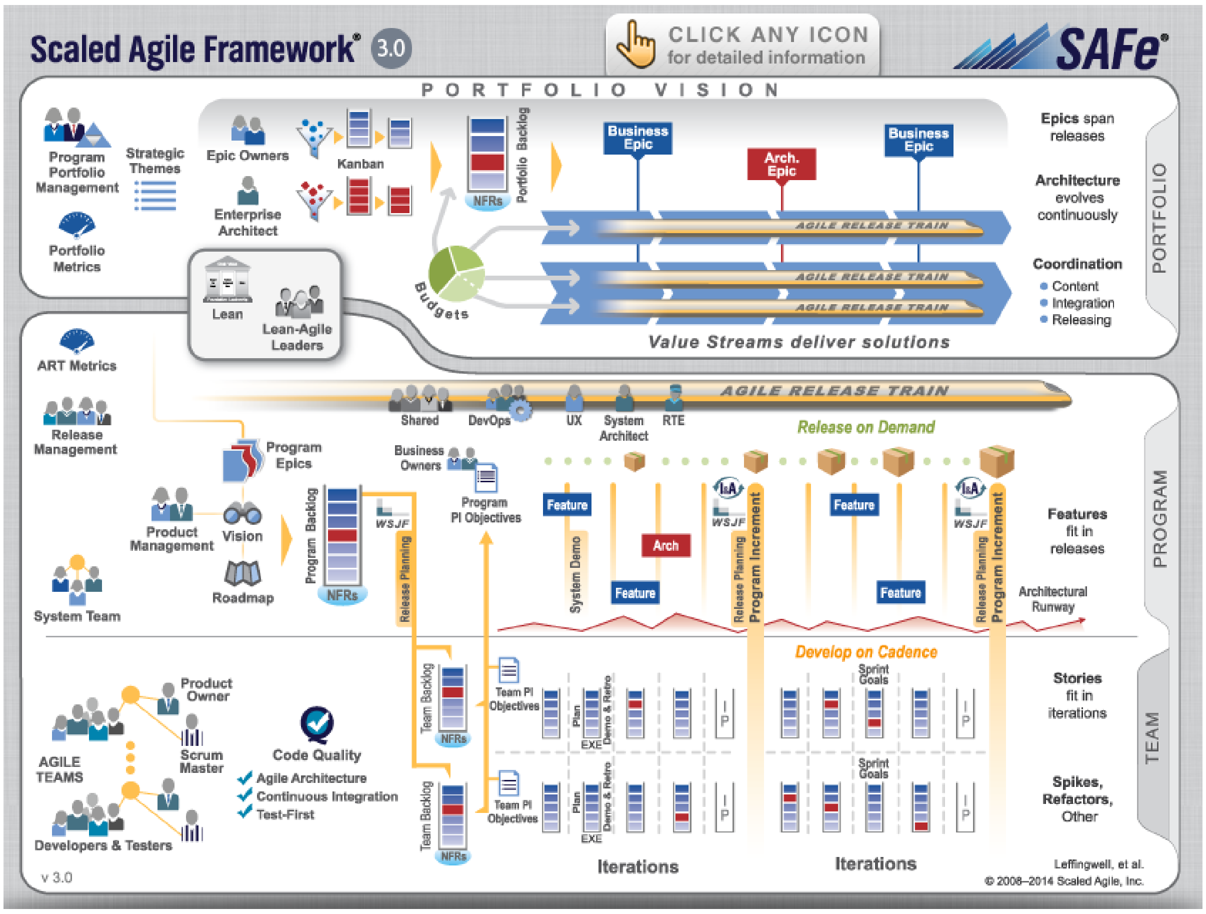


Figure 3.1. The SAFe framework (from http://www.scaledagileframework.com)

The organizational dimension of the scaling problem is addressed by abandoning the project metaphor and instead allowing and encouraging agile teams to be self-organizing, self-managing and cross functional. Additional teaming is necessary at the Program level to integrate the work produced by agile teams into value-producing features and to coordinate the activities required to deliver these changes into the business.

Also, the SAFe addresses the management and governance of the Program by describing an agile approach to portfolio management in the Portfolio layer. Having abandoned the project metaphor in favour of a continuous flow approach, the Portfolio layer integrates lean concepts by describing business strategy and technology oversight as a pull-based, Kanban approach. Kanban is a signalling mechanism that originated in the Toyota Production System, in which 'downstream' processes indicate to 'upstream' processes that they are ready for processing the next 'product' (Ohno 1988). This downstream to upstream signalling mechanism causes the 'pull' in pull-based systems, which is a key characteristic of lean manufacturing (based on the Toyota Production System).

Management and governance is achieved by paying attention to the flow of work through the Program. By watching the work queues, feedback is regularly provided to the teams at the Program and Team levels. Opportunities for improvements are continuously encouraged to increase flow where desired.

SAFe is a framework for scaling up the use of agile methods within large organizations, and presents a desired position for large-scale agile software development. It is becoming an industry standard for how to scale the use of agile methods at the enterprise level, that involves many different teams, products, and components. However, while the SAFe is proposed by Dean Leffingwell, a renowned agile advocate, other agile experts such as Ken Schwaber has criticized the SAFe for having introduced certification which he argues defeats the *reason d'etre* of agile methods, namely that of being a lightweight approach.

4 Business Model Innovation Map (BMIM)

4.1 Introduction

The Business Model Innovation Map (BMIM) was developed by the BMI Lab at the University of St. Gallen, Switzerland (Gassmann et al. 2015). The BMIM is based on the premise that prominent companies can lose their capability to innovate if they fail to adapt their business models to their environment that is constantly changing. The effectiveness of business models is critical to the survival of companies. However, at the time that the BMIM was proposed, little agreement existed on what constitutes a business model. The BMIM aims to fill that gap by providing a conceptualization of business models and defining the core components.

4.2 Evaluation

4.2.1 *Scope and Focus*

Researchers from the University of St Gallen defined a business model to contain the following four components:

1. **The Who** – who is the customer;
2. **The What** – what is being offered to the customer; also referred to as the value proposition;
3. **The How** – process and activities to implement the business model;
4. **The Value** – explains why the business model is financially viable.

Figure 4.1 presents what Gassmann et al. call the ”magic triangle” of the business model definition, which explains how these four components are related to each other.

BMIM does not focus at a particular driver or enabler for innovation, such as software technology, but takes a broader view when addressing needs of (future) customers and markets. Technology innovation, possibly disruptive, is an enabler for any Business Model to create value for the customer; it does not create value on its own.

BMI is an iterative (change) process in cycles of designing, prototyping, and testing the new Business Model ideas. A new Business Model affects and aligns all three dimensions towards a new value creation proposition.

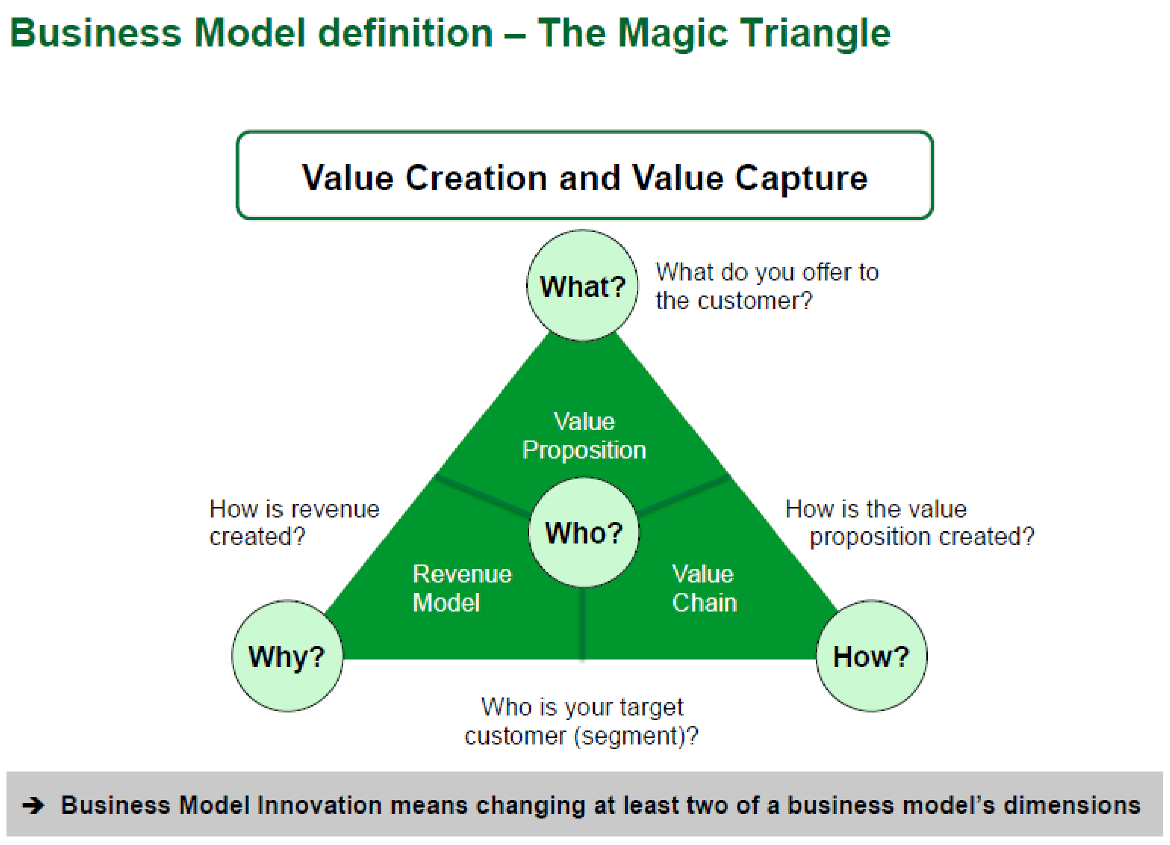

Business Model definition – The Magic Triangle
Value Creation and Value Capture
What?
What do you offer to the customer?
Value Proposition
How is revenue created?
Who?
How is the value proposition created?
Revenue Model
Value Chain
Why?
How?
Who is your target customer (segment)?
➔ Business Model Innovation means changing at least two of a business model's dimensions

Figure 4.1. Business model definition (source: Gassmann et al. 2015)

4.2.2 *Underpinning Assumptions*

Business drivers for change are captured by looking on the factors the industry competes on (following the dominant industry logic - "red ocean strategy") to create new, raise, eliminate, or reduce those factors relative to the industry standard.

BMIM is analysing in a structured way the ecosystem in which the old business is operating; it looks at market players and market change drivers for value creation and value capturing. BMI explicitly focuses at following dimensions of the business:

(1) **Who** is the target customer (segment)?

(2) **Why** is the business profitable?

(3) **What** is the offering to the customer?

(4) **How** is the value proposition created?

BMI requires changing at least two of a business model's dimensions.

Table 4-1. Selection of business model patterns (source: Gassmann et al. 2015)

Pattern Name	Business Model components	Example companies	Description
Add-on	What, Value	Ryanair, SAP, Sega	The core offering is priced competitively, but there are numerous extras that drive the final price up. In the end, the costumer pays more than he or she initially assumed. Customers benefit from a variable offer, which they can adapt to their specific needs.
Affiliation	How, Value	Amazon Store, Pinterest	The focus lies in supporting others to successfully sell products and directly benefit from successful transactions. Affiliates usually profit from some kind of pay-per-sale or pay-per-display compensation. The company, on the other hand, is able to gain access to a more diverse potential customer base without additional active sales or marketing efforts.
Auction	What, Value	eBay, Google, Elance	Auctioning means selling a product or service to the highest bidder. The final price is achieved when a particular end time of the auction is reached or when no higher offers are received. This allows the company to sell at the highest price acceptable to the customer. The customer benefits from the opportunity to influence the price of a product.

Gassmann et al. argue that organizations have difficulty developing new business models because ”thinking out of the box” is difficult. To understand how business innovation works in practice, Gassmann et al. have studied 250 business models that have been used in the last 25 years, which resulted in a set of 55 patterns.

The BMIM leverages on the observation that 90% of all business model innovations are *recombinations*. It makes use of the 55 Business Model Patterns to innovate the business model through creative imitation and recombination. A selection of these 55 patterns is listed in Table 4-1.

4.2.3 *Approach and Vision*

BMIM's strategic move ("blue ocean strategy") is applying a combination of 55 patterns by transferring, combining, and leveraging those in order to generate new Business Model ideas. The BMIM methodology consists of three main steps:

1. **Initiation**. In this first step, the "transformation journey" is defined. A starting point must be defined as well as an approximate direction.
2. **Ideation**. In the second step, the 55 business model patterns are considered one by one in a group setting, to discuss what that pattern might mean if it were applied to the current situation. This process is called "pattern confrontation". This step is to explore and understand the implications of applying one or more specific patterns.
3. **Integration**. In the final step, the selected patterns must be implemented and tailored according to the specific context of the product and the company. The St. Gallen Business Model Navigator™ provides checklists and analytical tools (e.g. the value network methodology) that can assist in this task.

5 The Capability Maturity Model (CMM)

5.1 Introduction

The Capability Maturity Model (CMM) is a process improvement framework (Paulk et al. 1994; Raynus 1998). It can be used as an assessment model, as an improvement model or as software engineering training material. The CMM provides a set of practices for improving processes, building upon an organization’s attributes. The CMM framework does not provide a single process; instead, it suggests how to improve an organization’s processes, but does not *define* the organization’s processes—one implication of this is that the CMM can also facilitate adoption of agile methods such as Extreme Programming (Paulk 2001). The CMM is designed as a maturity model for an organization to improve its existing processes according to proven best practices developed by members of industry, government, and academia.

The CMM has been now integrated into the CMM Integration (CMMI) which defines three areas of interest:

- Product and service development — CMMI for Development
- Service establishment, management — CMMI for Services
- Product and service acquisition — CMMI for Acquisition

However, for the purpose of this report, topics such as services (different from “software as a service” or the trend of *servitization*) and acquisition (of complete products and services as opposed to “sourcing”) are not within the scope of the SCALARE project. Hence this report focuses on the CMM.

5.2 Evaluation

5.2.1 *Scope and Focus*

The CMM originated from research by Watts Humphrey and others from the Software Engineering Institute (SEI) (Humphrey 1988). The SEI was founded to develop software engineering expertise for the US Department of Defence (DoD). By observing successes and failures across a large number of software projects, “best practices” were distilled and organized within a five-level framework (Paulk et al. 1993).

The scope of the CMM is organizations’ software development processes and as such it focuses on following a disciplined approach in implementing a number of key practice areas (more details in section 5.2.3). Business considerations and product-specific concerns are not included—while the CMM prescribes extensive documentation related to the product being developed (e.g. requirements documentation, etc.), this is the same for any product, but no product-specific guidance is provided.

The CMM can be used as a framework for assessing an organization’s current state of practice, and indeed, CMM assessment and certification can be acquired. For US organizations wishing to deliver to the US DoD, such certification is necessary. Other organizations around the world use CMM certification to signal their claimed excellence with regards to their development processes.

5.2.2 *Underpinning Assumptions*

The CMM was developed based on the assumption that whatever practices were present in successful software projects, were contributing to the success of those projects. Likewise, failures were traced back to their root cause, which would be linked to “missing practices.” For example, failure to manage different versions of either the product or process documentation suggests a need to implement a sound version control system.

The key assumption that the CMM implies is what can be summarized as “one size fits all.” Whatever practices are useful in one project will also result in successes in other contexts. One important consequence of this is the assumption that a singular process improvement path can be adopted for any software organization, irrespective of the context that the organization operates in. Thus, a linear process improvement path is suggested. Figure 5.1 below illustrates this linear pathway clearly.

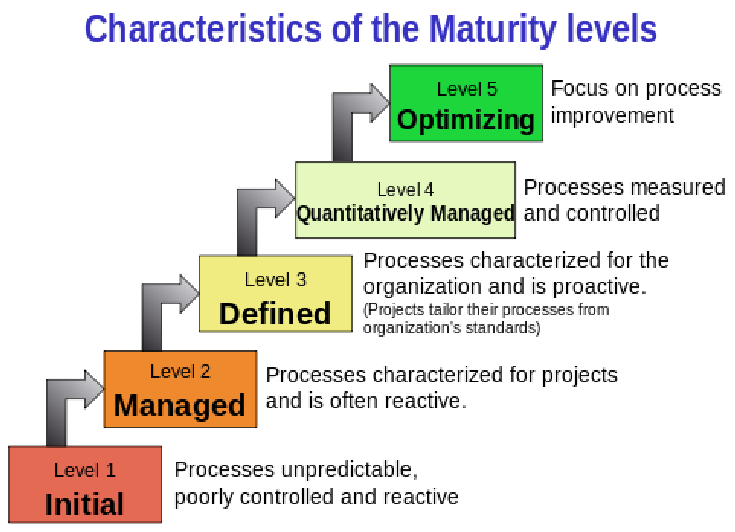

Characteristics of the Maturity levels
Level 5
Optimizing
Focus on process improvement
Level 4
Quantitatively Managed
Processes measured and controlled
Level 3
Defined
Processes characterized for the organization and is proactive.
(Projects tailor their processes from organization's standards)
Level 2
Managed
Processes characterized for projects and is often reactive.
Level 1
Initial
Processes unpredictable, poorly controlled and reactive

Figure 5.1. CMM Maturity levels

(source: https://en.wikipedia.org/wiki/Capability_Maturity_Model_Integration)

Another assumption is that, given a sufficiently rigorous software development process, the software product will be of high quality – or as the saying goes, *"the proof is in the pudding."* The CMM is not unique in this assumption—indeed, most manufacturing processes (software and non-software) that are subject to regulation (such as the US Food & Drug Administration, FDA), also assumes that as long as certain practices are implemented, the product will be of acceptable quality.

The CMM does not require any specific process. Traditional terminology placed so-called plan-driven approaches (waterfall, V-model, etc.) on one side of the spectrum with agile methods being placed on the other end, suggesting that agile methods do not follow any plan. However, this dichotomy of "plan-driven" versus "agile" is incorrect, and indeed, CMM advocates have clearly indicated the possibility to marry agile methods with the CMM framework (Paulk 2001). Version 1.3 of the CMM explicitly acknowledged agile methods.

CMMI users have to be very knowledgeable in software engineering, while the users of the SMF could also be non-technical people including managers and business analysts.

SMF addresses top managers in product companies, seeing a big growth of the importance of their software and their software organizations.

A requirement in CMMI is that top managers support and drive the software improvement initiatives, but users of CMMI are software engineers and software process improvement engineers.

5.2.3 *Approach and Vision*

The CMM defines a number of key process areas – Figure 5.2 below defines these 18 KPAs.

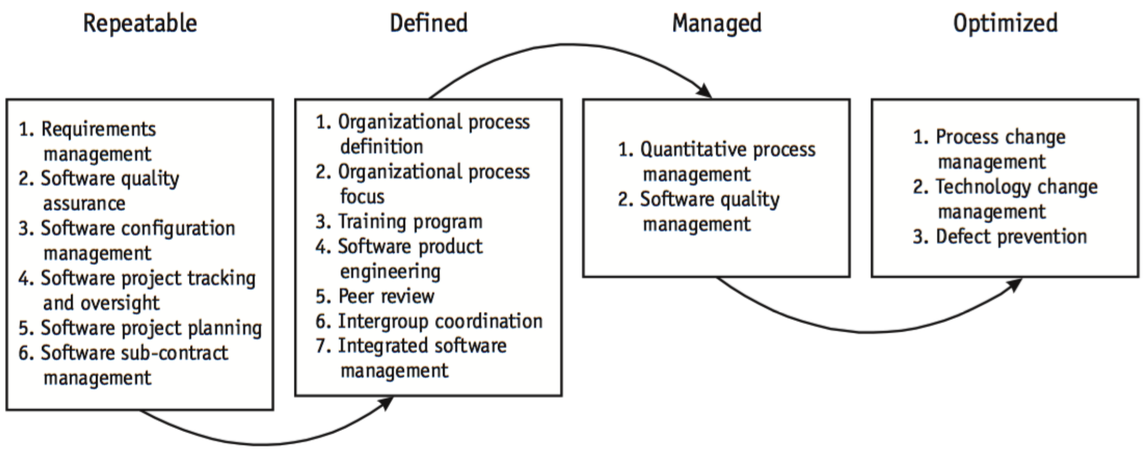

Repeatable
Defined
Managed
Optimized
1. Requirements management
2. Software quality assurance
3. Software configuration management
4. Software project tracking and oversight
5. Software project planning
6. Software sub-contract management
1. Organizational process definition
2. Organizational process focus
3. Training program
4. Software product engineering
5. Peer review
6. Intergroup coordination
7. Integrated software management
1. Quantitative process management
2. Software quality management
1. Process change management
2. Technology change management
3. Defect prevention

Figure 5.2. Key Practice Areas as defined in CMM levels 2 (Repeatable) to 5 (Optimized) ***(source: Raynus 1998)***

The CMM defines the criteria for an organization to 'mature' its process and be promoted to the next level on the maturity model. For example, in order to be assessed at Level 3 ("Defined"), an organization must successfully implement all processes as defined for the KPAs in Level 3 and all of those at lower levels (i.e. Level 2; Level 1 does not define any KPAs). This approach to "maturation" clearly indicates the next process improvement activities to undertake if an organization wishes to be assessed at a higher level.

6 Experience Factory (EF)

6.1 Introduction

The idea of the Experience Factory (EF) concept is to divide the organization into two different sub-organizations in order to manage reuse of experience in software process improvement. One of the sub-organizations is responsible for traditional execution of projects, i.e. the traditional organization. The other sub-organization, the EF, is responsible for collecting experience from projects in the traditional organization, generalizing it, and providing it back to the traditional organization when new projects are started. This results in the collection of experiences into a database using different methods is collected. Extensive research on the EF concept was conducted during the nineties, although after that less research has been conducted. Basili et al. (1994) describe the Experience Factory in detail.

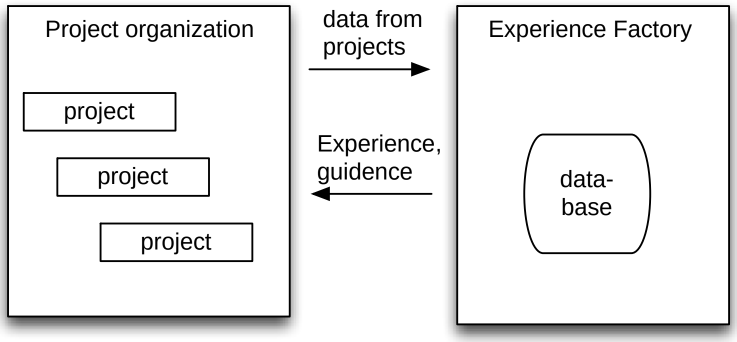


Figure 6.1 The Experience Factory

This is shown in Figure 6.1 where the projects reside in the project organization and the experience management in the Experience Factory part.

6.2 Evaluation

6.2.1 *Scope and Focus*

The scope of the approach is limited to software process improvement, and to a rather large degree on quantitative experience of using different process models. The approach was developed after an observation that there is a need to collect and generalize experience from using new development processes, which can be compared to that there is a need to generalize product artefact before they can be reused in future projects.

Compared to the SMF the scope is solely on software process improvement and the focus on quantitative data.

6.2.2 *Underpinning Assumptions*

The basic assumption is that it is possible to collect quantitative data in order to support software process improvement, and that this requires methods for defining which data to collect, i.e. goal based measurement, e.g. GQM as discussed for example by Van Solingen and Berghout (1999). Business drivers are not describes as explicitly, although they can, of course, be included in a GQM metrics analysis. The assumption is also that it is possible to set up one EF for every organization, whereas the assumption of the book from Scalare is more to generalize knowledge from a wide range of organizations.

6.2.3 *Approach and Vision*

The Experience Factory (EF) is one of the first approaches based on distilling “lessons learned” from empirical industry case studies into reusable practice patterns. Approach requires an improvement cycle where experience data is collected, e.g. the Quality Improvement Paradigm, where improvements are carried out for each project, and data is collected in the form of experiences: 1) Characterize the current project, 2) Set improvement goals for the project, 3) Choose development process, 4) Execute the project while collecting data and giving feedback, 5) Analyse the results, 6) Generalize and package the results in the experience database. It can be seen that the main responsibility for some steps are in the project organization (e.g. 3) and some in the EF (e.g. 6).

7 Disciplined Agile Delivery (DAD)

7.1 Introduction

Many organizations start their agile journey by adopting Scrum because it describes a good strategy for leading agile software teams—some surveys suggest that Scrum is the most popular agile approach adopted in industry. However, Scrum represents only part of what is required to deliver sophisticated solutions to stakeholders. Invariably, teams must look at other practices and techniques to close the gaps that Scrum purposely ignores. When looking at other methods there is considerable overlap and conflicting terminology that can be confusing to practitioners and customers. Worse yet, stakeholders are often struggling with seeking additional advice or are not cognizant of the key issues they must address to close the gaps that the Scrum framework offers. The Disciplined Agile Delivery (DAD) is a process decision framework that focuses primarily on people and "learning".

7.2 Evaluation

7.2.1 *Scope and Focus*

DAD is an approach that extends Scrum with proven strategies from other agile methods and practices including Agile Modelling (AM), Extreme Programming (XP), Unified Process (UP), Kanban, Lean Software Development, Outside In Development (OID) and several other methods. DAD is a non-proprietary, freely available framework. DAD extends the construction-focused lifecycle of Scrum to address the full, end-to-end delivery lifecycle from project initiation all the way to delivering the solution to its end users. It also supports lean and continuous delivery versions of the lifecycle: unlike other agile methods, DAD does not prescribe a single lifecycle because it recognizes that one process size does not fit all. DAD includes advice about the technical practices such as those from Extreme Programming (XP) as well as the modelling, documentation, and governance strategies missing from both Scrum and XP. But, instead of the prescriptive approach seen in other agile methods, including Scrum, the DAD framework takes a goals-driven approach. In doing so DAD provides contextual advice regarding viable alternatives and their trade-offs, enabling an organization to tailor DAD to effectively address the situation in which they find themselves. By describing what works, what does not, and more importantly why, DAD assists to adopt strategies that are appropriate for the context of an organization. Thus, DAD has been suggested as a framework to help organizations in solving the "puzzle" of selecting appropriate techniques and practices – this is suggested in Figure 7.1.

The DAD is exclusively focused on agile methods, and does not consider aspects related to business, architecture and organization. While it does not advocate a single solution for different problems, the solution space does assume the augmentation of an agile approach with other practices to ensure delivery of a full product. The DAD framework offers many different starting points; organizations can choose their point of departure based on their current environment. Whatever the current state in an organization's agile process, the DAD offers guidance in further improving this.

7.2.2 *Underpinning Assumptions*

The organizations that use the DAD framework are often organizations that have understood the complexity of lean and agile by doing this at small scale at first. The DAD framework describes several strategies for organizing large or geographically distributed teams. It describes a range of options for scaling your approach to agile and lean software development, giving you context-sensitive options that other models, such as the SAFe, do not.

7.2.3 *Approach and Vision*

An important feature of the Disciplined Agile Delivery (DAD) framework is that it provides a foundation from which to scale agile solution delivery. There are three levels for what it means to scale agile delivery.

The first level captures how organizations typically start with agile or lean methods such as Scrum or Kanban. The next level is where the DAD framework does the "heavy lifting" for you by showing how the various agile strategies work together, taking your existing agile teams to the next level and giving teams new to agile a head start by providing the process guidance they require. The third level is named Agility at scale. An organization can tailor its process, team structure, and tooling approaches as is appropriate for its current development context.

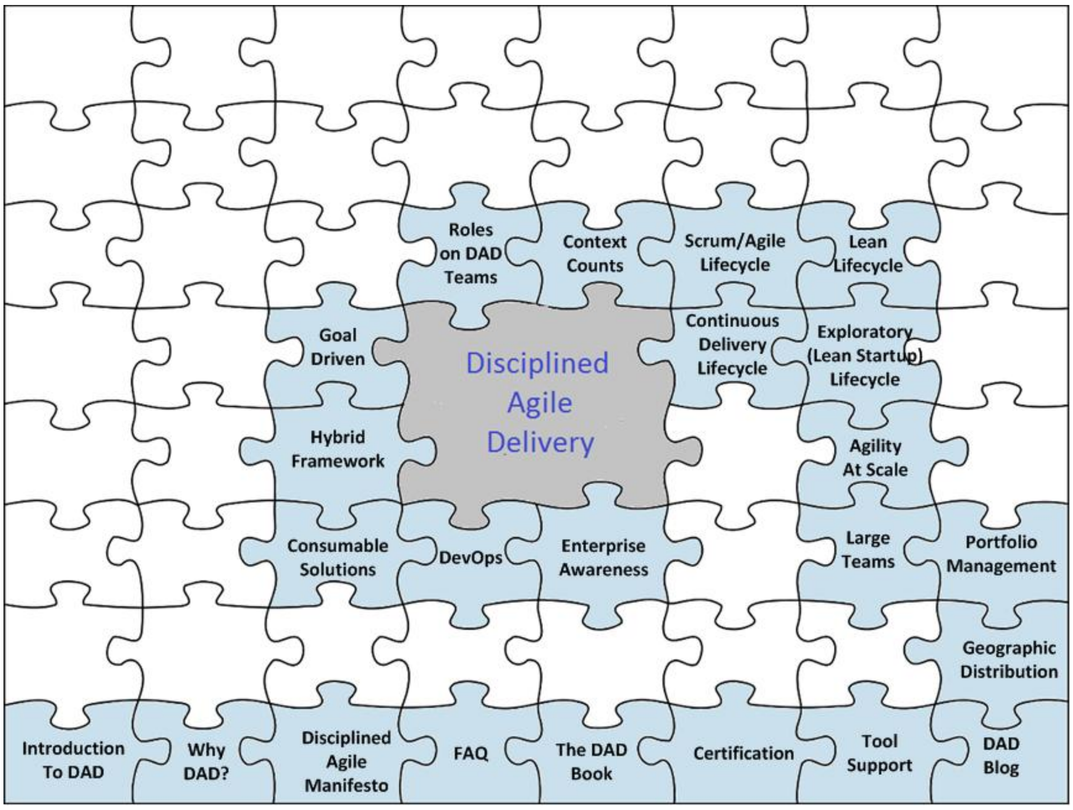


Figure 7.1. The Disciplined Agile Delivery process decision framework

(source: http://www.disciplinedagiledelivery.com/home/)

The DAD framework is not as end-state-focused as SAFe is, hence the idea is that one cannot anticipate which end-state is reached. The DAD framework claims that successful scaling of agile and lean techniques can be achieved in several ways. First, its full delivery lifecycles and breadth of software development advice answers how to successfully apply agile in practice. Second, its goal-driven approach provides the required flexibility for tailoring an agile process to meet the challenges faced by agile teams working at scale. Third, the DAD framework builds on many foundational concepts required at scale. Several new trends have emerged in recent years that can be included in the DAD framework; for example, DevOps has emerged as an important approach, which addresses the gap between developers on the one hand and operations teams on the other.

8 Conclusions

Scaling software development is an important concern for the software industry. Several frameworks, models and guidelines have been offered by practitioners and researchers. This report has presented the most relevant and prevalent models that have been proposed. One observation from this review is that virtually all models assume that "scaling" problems can be solved by a linear growth model, and that the need to scale is merely in degree, rather than in kind. In other words, all models assume that scaling software refers to "more" software and "larger" systems.

Figure 8.1 positions the models reviewed in this report along two dimensions: Drivers, and Impact.

The first dimension, Drivers, is concerned with the "Why" – why do organizations wish to change, and subsequently adopt a certain model. Several reasons can be observed, including compliance (or "better" compliance), performance, business growth (through business improvement), and business innovation.

The second dimension is Impact, and is concerned with the question of which aspects of the organization are affected. The figure defines the three key dimensions that the SCALARE project has defined to be relevant (see Chapter 1), namely: Product, Process, and Organization.

The figure positions the various models within this two-dimensional grid, and highlights that models can be found in several positions within this grid. For example, the CMM can be used as a framework for process compliance as well as for process improvement.

Business Assurance	**Compliance** **(ticket-to-trade)**		CMM	
Business Improvement	**Performance Improvement** **(quality, cost, time)**	BAPO	CMM, SAFe, EF, BAPO	BAPO CMM
	Business Growth **(Product/Service Innovation)**	EF	EF	
Business Innovation	**Business Foundation** **(New Business, Company Transformation)**	BMI	BMI	BMI
Drivers	**Impact**	**Product** **(Systems & Service)**	**Processes** **(Development & Operations)**	**Organisation** **(Research & Development)**

Figure 8.1. Positioning of the various models presented in this report.

8.1 Summary of the findings

The need to scale agile methods has been recognized for more than a decade, soon after the presentation of the Agile Manifesto in 2001. Agile methods were originally thought suitable for development of small systems, with co-located teams in non-critical environments. Soon, however, practitioners and researchers made attempts to scale up the use of agile methods, leading for example to the SAFe framework (Leffingwell 2007). While such frameworks are very useful to guide companies in their attempts to deploy agile methods at scale, they do not address other dimensions of scale.

The DAD is a model that advocates a higher level of customization than the SAFe, but its scope remains limited to that of agile methods and practices focused on delivery of a product – hence a process-oriented model.

The CMM is also focused on the process, which assumes a linear progression model as it defines a number of Key Process Areas that any organization adopting the CMM should implement. While the CMM is a framework, and does not suggested specific software development methods, it remains limited to the process dimension.

Table 8-1 provides some of the key findings of the review presented in this report. What becomes clear is that all models offer guidance in specific directions, but they are all limited in that they either assume a ‘constant’ solution and/or problem space; they assume a linear progression model, or they focus only on one specific dimension, with most models focusing on process-related issues.

This report does not suggest these models are not useful – instead, the SCALARE project aims to present these initiatives within a larger encompassing framework, which recognizes that different types of growth (scaling) requires different approaches. What all scaling scenarios have in common, however, is that they involve scaling and tailoring in three key dimensions, namely that of products, systems and services, processes and methods, and organizations and business domains. These three dimensions unify all models reviewed in this document.

Table 8-1. Key findings of the review of existing models

	Scope and Focus	Underpinning Assumptions	Approach and Vision
BAPO	Multi-dimensional, but focused specifically on product lines / families	Single problem space, and single solution space. BAPO is very architecture-centric and assumes a software product line approach.	Family Evaluation Framework (FEF) that defines a set of maturity levels for each of the BAPO dimensions.
SAFe	Adopting agile methods in the enterprise; agile at large.	It assumes a common organizational structure of teams, projects, programs, portfolios for any organization adopting the SAFe.	Using agile and lean practices. The specific details are quite detailed and well documented.
BMI	Focus on business models. No attention for organizational aspects.	Business model innovations are difficult due to "thinking out of the box" challenge. Instead, new business model innovations can be developed by reusing and combining business model patterns.	Adopt, adapt and combine a subset of 55 pre-defined business model innovation "patterns"
CMM	Focuses on process improvement only; disregards product aspects, organizational aspects as well as business aspects.	Linear set of maturity levels, "one size fits all". It assumes that all key process areas are suitable in all contexts.	Companies increase their 'maturity' based on an assessment of their implementation of a predefined set of Key Process Areas.
Experience Factory	Software process improvement, quantitative knowledge	Lessons learned through empirical studies can be transferred to new settings.	"Bottom up" process improvement based on project needs.
DAD	Focus exclusively on agile methods.	Organizations understand agile/lean methods and that these have been successfully employed.	Tailor the 'solution space' based on the context of the organization.

8.2 Conclusion

The SCALARE is based on the premise that “scaling software” is not only a matter of “large-scale”, but that this also must address other dimensions, such as the tailoring of methods to new domains, the transformation of hardware-based to software-based solutions, and the need to grow organizations and source software elsewhere than merely within the organization.

The models reviewed in this report each offer a specific solution to a specific problem. Some solutions are more constrained suggesting a ‘one size fits all’ approach (e.g. the SAFe framework), whereas others recognize the need to tailor solutions (e.g. the DAD framework). Software organizations are, however, facing new challenges as they move into new domains, are confronted with new regulations and standards (software can be a “medical device” in its own right since a 2010 EU Directive, for example), and are experimenting with new sourcing strategies such as innersourcing, crowdsourcing and opensourcing.

The SCALARE project aims to deliver a framework that offers “solutions to problems”, recognizing that each organization’s software capability must take into account the organization’s business drivers, nature of the product, and suitability of the processes and methods that are adopted. An organization’s software delivery capacity, measured for example in the time to deliver or size of the workforce, can also be scaled up and down with new strategies such as crowdsourcing. The Scaling Management Framework aims to capture these “common” solutions for recurring problems into an overall framework. The framework is built on a set of “practice patterns” and organizations can select the appropriate patterns that are suitable for their context. In this sense, the SMF is akin to the DAD presented in this review, except that the DAD is limited to “agile delivery” whereas the SMF takes a much wider view of “delivery” including the transformation of hardware-based to software-based solutions.

9 References

SW Ambler. 2013. Going Beyond Scrum: Disciplined Agile Delivery. White Paper Series. October 2013. Available from: http://disciplinedagileconsortium.org/Resources/Documents/BeyondScrum.pdf

SW Ambler, M Lines. 2012. Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise. ISBN 978-0132810135.

VR Basili, G. Caldiera, H.D. Rombach. 1994. "Experience Factory". Encyclopedia of Software Engineering, edited by J.J. Marciniak, Vol. 1, John Wiley & Sons, 1994, pp. 469-476.

VR Basili, F McGarry. 1997. The experience factory: how to build and run one (tutorial). ICSE '97 Proceedings of the 19th international conference on Software engineering pp. 643-644

O Gassmann, K. Frankenberger, M. Csik. 2015. The Business Model Navigator: 55 Models That Will Revolutionise Your Business. FT Press.

WS Humphrey. 1988. Characterizing the Software Process: A Maturity Framework. 5(2) pp. 73-79

P Jalote. 1999. CMM in Practice: Processes for Executing Software Projects at Infosys. Addison-Wesley

D Leffingwell. 2007. Scaling Software Agility: Best Practices for Large Enterprises. Addison-Wesley

LM Northrop. 2004. Software Product Line Adoption Roadmap. Technical Report CMU/SEI-2004-TR-022/ESC-TR-2004-022. Carnegie Mellon Software Engineering Institute.

MC Paulk, B Curtis, M Chrissis, C Weber. 1993. Capability Maturity Model, version 1.1. IEEE Software 10(4) pp. 18-27.

MC Paulk, CV Weber, B Curtis, The Capability Maturity Model: Guidelines for Improving the Software Process, Carnegie Mellon University, 1994.

MC Paulk. 2001. Extreme Programming from a CMM Perspective. IEEE Software 18(6) pp. 19-26

T Ohno. 1988. Toyota Production System: Beyond Large-Scale Production. Productivity Press.

J Raynus. 1998. Software process improvement with CMM. Artech House.

F van der Linden. 2002. Software Product Families in Europe: The Esaps & Café Projects. IEEE Software 19(4).

F van der Linden, K Schmid, E. Rommes, 2007. "Software Product Lines in Action", Springer.

F van der Linden, J Bosch, E Kamsties, K Känsälä, H Obbink. 2004. Software Product Family Evaluation. In: Software Product-Family Engineering. Volume 3014 of the series Lecture Notes in Computer Science pp. 352-369

R van Solingen, E. Berghout, "The Goal/Question/Metric Method, A Practical Guide for Quality Improvement of Software Development", McGraw-Hill, 1999.