

1. ENTERPRISE SCRUM

We define Enterprise Scrum as - business definition:

Enterprise Scrum is a framework that seeks to quickly deliver the most business value and balanced benefits to all people involved.

Here is the Enterprise Scrum -- full definition – skip this if you are not a super nerd:

Enterprise Scrum is a generic, customer-centric, iterative-incremental, all-at-once, scalable, results-oriented, subsumption-based management **framework** that seeks **to quickly deliver the most business value and balanced benefits to all people involved**, through autonomous, self-DMOS teams.

Where self-DMOS means self-directed, self-managed, self-organizing and self-selected. We use the ES acronym to mean Enterprise Scrum throughout this document.

Notice that in these definitions above, that we say that Enterprise Scrum “seeks” to deliver some benefits, but these benefits are not guaranteed.

That is mouthful, so let’s explain this a little bit.

ENTERPRISE SCRUM DETAILED EXPLANATION

Let’s define each one of the terms above which describe Enterprise Scrum, and why they are important.

- Enterprise Scrum is **generic** management, meaning, it can work for many different purposes, in many domains, and at different levels of structure. Although Enterprise Scrum as presented here, is focused on **business** management, it can also be applied to many other things; for example, managing a soccer team, doing projects for college students, or improving the classrooms for elementary school students. This *genericity* is based on abstraction, generalization, extension, parametrization and being able to plug things in, like techniques. For example, Enterprise Scrum can be used for company, customer segment, business model, marketing and sales, product development, software development, research, or compliance management; or at the executive, middle management, program, or project levels; in any industry. Let’s explain how this genericity works.
 - Enterprise Scrum is an **abstraction** of Scrum because it is a deeper description of what Scrum really is. This better understanding makes it possible to generalize the concepts. This abstraction really includes all of the concepts in this list, which are all important.
 - Enterprise Scrum is a **generalization** of Scrum, because it uses the same concepts in Scrum but in a more general way. For example, Scrum is a 2-level subsumption architecture – although this is not described very well in the Scrum Guide; while Enterprise Scrum uses a configurable n-level subsumption architecture. For example, the subsumption levels in

Business Agility are: company, business units, customer segment, product and services, etc.

- Enterprise Scrum is an **extension** of Scrum, because it adds many things that were not in Scrum either explicitly or not at all before. For example, in Scrum we co-evolve “development type” activities like requirements, architecture, code, tests, plans, infrastructure, but we don’t explain this explicitly specially in the Sprint cycle. Enterprise Scrum extends this behavior, and can co-evolve *anything that is important* together for any domain – but that’s something that wasn’t there in Scrum before.
- Enterprise Scrum is a **parametrization** of Scrum, because it adds parameters to explicitly track things that were in Scrum and additional things added in Enterprise Scrum. For example, Cycle length, DOR, DOD, were in Scrum before, but not parametrized, but in addition there are 150+ other things that are important in Enterprise Scrum, for example, the information architecture from Enterprise Scrum, like VLITypes, DODStandard, DORStandard, etc.
- Enterprise Scrum is technique-pluggable, because we can insert explicitly **techniques** specific for different domains. For example, in Enterprise Scrum we can use techniques like Lean Startup, Design Thinking, Business Model Generation for company management; or User Stories, Release Planning, or UTDD for software development; or Internal Controls for compliance management, etc.
- Enterprise Scrum is **customer-centric**, it aims to deliver the most business value to customers and the most balanced benefit to all people involved, once team at a time. That is the purpose of creating a Value List – to delight the customers, which Peter Drucker says is the fundamental “source of all goodness”. This may appear to be a marketing-like statement, but there are in fact, fundamental constructs in Enterprise Scrum to do this. This is why it is presented here as a feature.
- **Iterative-Incremental**. In Enterprise Scrum, we get things DONE, grow and improve everything iteratively and incrementally through Cycles, from business models, to software products, marketing campaigns, compliance, or increased customer experience. The reason I am capitalizing DONE, is because getting things DONE in Enterprise Scrum has a special meaning: passing DOD (definition of done). We manage things assuming rapid change rather than assuming a static world where long-term plans and predictions can be made without changes; and the Cycles help us to make adjustments after evaluating things. This also allow us to take “smaller bets” and “fail faster”.
- Enterprise Scrum is **all-at-once** management, or co-evolutionary management, because we get all aspects of things DONE in ONE cycle not through multi-cycle phases. For example, instead of having a sequenced semi-parallel phase-based approach such as strategy, marketing, product development, sales, customer service, we prefer to deliver testable business models, that can be iterated and improved through feedback from the market and the customers. This leads to faster and better learning, growth, adaptation, visibility, risk management, and eventually higher confidence and certainty.

- **Scalable.** Enterprise Scrum has the necessary constructs to be **scaled** in different structural, collaboration, delivery modes, delivery targets and contract types.
- **Results-Oriented.** Enterprise Scrum is empirical-process management, because it's management is based on measurements.
 - **Multiple Metrics.** We are not just interested in productivity, cost or profit; but instead, we are interested *in balancing everything* that is important: customer experience, user experience, employee experience, profits, and a purpose in the world. To this may require multiple metrics to be tracked simultaneously.
 - **Projections.** In Enterprise Scrum, plans, forecasts and decisions are also based on measurements made on metrics -- in what is proven that can be done, not on what we imagine that can be done. For example, if we are managing a sales process, future sales projections are based on the current sales measurements.
- Enterprise Scrum is **subsumption-based** because at any level where we use Enterprise Scrum, we make decisions and adjustments based on the feedback and measurements of the "immediate surrounding reality" associated with it. For example, in startup management, we make marketing changes or sales changes, based on the immediate feedback of our messages and campaigns.
- Enterprise Scrum is a true **framework**, meaning, there are specific parameters that we configure to implement the Enterprise Scrum framework. For example, we may configure explicitly Business Value, metrics, charts, Value List (type of work, frequency, structure, etc.), DOD, DOR, Planning type, Review type, Improve type, canvas, surfers (aspects that are important), subsumption levels, etc. Unfortunately, the word "framework" has been abused, but Enterprise Scrum is a true framework.
- Enterprise Scrum promises to deliver the most **business value**, but business value itself, means different things for different people: better conversion cohorts, profit, compliance levels, features delivered, etc.; therefore Business Value itself, must be configured.
- **Balanced Management.** We seek to balance the benefit for ALL the people involved in whatever we are managing, improving or building. For example, in growing a startup company, we may want to find a nice balance: 1) delighting the customers by delivering more business value, or a better customer experience, 2) creating a good employee experience, 3) sharing a purpose to make a better world and 4) making profit for our stakeholders.
- **Autonomous self-DMOS Teams.** At a high level, this means greater autonomy, decentralization and decoupling. One of our goals is that each Let's explain what self-DMOS is: it's an acronym for self-directed, self-organized, self-managed and self-selected as much as possible instead of working through command and control, hierarchical, bureaucratic, functional, phased-based, process oriented, or product or service-oriented organizations. This allows us not only to expedite things, since there aren't any vertical or horizontal dependencies, but also to

develop a co-creation participatory culture, with more ownership and pride. For example, we want to decouple a single business unit, a customer segment or a product or service and manage them as independent as possible.

Again, notice that when we are defining these terms above, we say that we *aim* or *seek* to deliver “more business value”, and “balanced benefits to everyone in the Enterprise Scrum team – these things are not guaranteed, as well as any other benefits.

ENTERPRISE SCRUM BENEFITS

Now, let's switch modes. If you do are able to implement the features described above; then it is possible, but not guaranteed, that you will get some of these benefits for a single Enterprise Scrum team, or the organization as a whole.

Company

- agile management for most activities: company, business units, customer segments, products and services, marketing and sales, software development, hardware, compliance, etc.
- accelerated improvement for any activity
- faster delivery of value for any activity
- lost-lasting organizational and cultural change
- preserved agile growth or evolution
- accelerated innovation
- reduced exposure (money, effort, etc.)
- accelerated time to market
- reduced change management expenses
- less dependency on consultants and external resources
- improved alignment between the working teams and business objectives
- more flexible and reliable scaling
- more reliable distribution
- higher probability to make anything Agile or Business Agility to work

Team

- more fun
- improved CX (customer experience) and Customer Satisfaction!
- improved UX (user experience)
- improved passion, engagement, and EX (employee experience)
- better effectiveness
- more of a learning experience
- reduced risk
- improved visibility
- improved efficiency
- enhanced quality
- better reaction time
- improved ownership
- enhanced ability to manage changing priorities

- improved sense of purpose
- improved productivity
- improved team morale
- improved participation and co-creation
- improved team dynamics

ENTERPRISE SCRUM CONCEPTS AND NAMES

If you already know Scrum, this section may make it easier for you to get Enterprise Scrum faster.

In Enterprise Scrum, we use “more generic concepts and names” than those used in Scrum because we are generalizing Scrum to either different levels of scale or other business processes other than product development, where there may not be any Products, and where the concept of Sprints as 1-4 week time-box duration may be insufficient to describe longer-term iterations or nesting of Cycles. We hope that the Enterprise Scrum “concepts and names” will make it easier to understand and easier to relate to by workers with different purposes.

Software/Product	General Purpose
Scrum – for product development	Enterprise Scrum – for general-purpose scaled agile management
Product Owner – owner of a product	Business Owner – owner of a business area. Role, not necessarily a single person. Extended to provide explicit integrated Agile Leadership: vision and support.
Scrum Master – coaches Scrum Team to do Scrum for product development	Enterprise Scrum Coach – configures Enterprise Scrum and coaches an Enterprise Scrum Team.
Team – does product development	Team – does any kind of work
Business Value, not defined explicitly	Business Value – defined explicitly
Product Backlog – list to develop a product	Value List – list to deliver value for any activity
PBI (product backlog item) – feature or something else to be DONE in product development by passing a DOD	VLI (value list item) – anything that gets DONE and delivers value passing DOD for ANY activity or domain
Sprint – 1-4 week time box with Planning, (unnamed section to do work), Review, Retrospective and Refinement	Cycle – configurable time box of any length (1 hr. – 1 yr.), with configurable explicit options for the cycle (Planning, Collaboration, Review, Improve), with nesting allowed in time and structure e.g. one week Cycles contained in quarterly Cycles, contained in 1 yr.

	Cycles.
Daily Scrums – mandatory	Daily Scrums – optional under some circumstances where subsumption is achieved anyhow
DOD – for product development	DODStandard, DORStandard, DOD, DOR – for general purpose
Scrum Board – for product development	Enterprise Scrum Scrum Board – for general purpose. Similar workflow than that of ScrumBoard, but has a “hat” of Vision and Initial Value List
Velocity – single productivity metric	N Metrics – many possible metrics: customer experience, customer satisfaction, user experience, employee experience, velocity (effort), profit, etc.
Product Increment – an increment for a product	Value Increment – where value is delivered for ANY purpose
PSP (potentially shippable product)	Potential Value or Actual Value – potential or actual value
Sprint Burndown	Charts to graph Metrics, including but not limited to Burndowns
NONE	Configuration and Configuration Parameters
NONE	Configurable Techniques
NONE	Scaling options – 1024 combinations
NONE	Calculations – there can be one or more calculations for different Cycles. For example, recalculation of Release Plan: schedules, budgets, etc.

In essence, Enterprise Scrum is an abstraction, generalization, parametrization and extension of Scrum; that is, preserving the basic Scrum rules:

- we still want to do the work that delivers the most value first
- we still work, build, learn and improve iterative-incrementally
- you still want to pass DOD for as much work every Cycle
- team members still need to self-organize and collaborate with other team members
- Etc.

ENTERPRISE SCRUM INSTANCES

Enterprise Scrum is a *true framework*:

- it can be explicitly *configured* via parameters, and

- *instantiated*, and yielding specific **instances**.

We typically name different instances using the template:

“Instance of Enterprise Scrum for doing X”;

We can create Enterprise Scrum **instances** to manage most everything in an agile way. For example, these are Enterprise Scrum instances to do very many different things:

- Instance of Enterprise Scrum for Startup Management with Lean Startup, Design Thinking, Blue Ocean Strategy and Business Model Generation
- Instance of Enterprise Scrum for Company Management with Business Unit Portfolio management
- Instance of Enterprise Scrum for Business Agility
- Instance of Enterprise Scrum for Scaled Software Development with User Stories and Release management
- Instance of Enterprise Scrum for Marketing and Sales
- Instance of Enterprise Scrum for Compliance Management
- Instance of Enterprise Scrum for Real State Sales
- Etc.

In a recent poll in our Facebook Enterprise Scrum group, we found 50+ instances. You can join our FB and LinkedIn groups here:

Facebook:

<https://www.facebook.com/groups/EnterpriseScrum/>

LinkedIn:

<https://www.linkedin.com/groups/7488992>

Morphing Instances

You can also have advanced instances that morph and evolve, for example:

Instance of Enterprise Scrum for Business Agility

that starts with overall company management, adds portfolios of business units and customer segments, then becomes transformation, and ends up in a never-ending management of improving portfolios. We teach in our Business Agility classes using this concept.

Spawning Instances

There are many times when the instances morph, but in a way that they spawn other instances. For example, for a small startup we typically start with the Enterprise Scrum – Business Model Canvas, but as the company grows, and the business becomes more repeatable and scalable, we can then spawn other instances for software development, marketing and sales, etc.

Configuration

As stated above, we can configure Enterprise Scrum for different purposes and create instances. Enterprise Scrum has an explicit configuration that is detailed in full, at the **Enterprise Scrum Configuration Guide**. However, we include at the end of this document the **Enterprise Scrum Configuration Guide – Quick Reference**, which lists all of the available parameters and their relationships. You don't have to configure each and every parameter to configure an instance – instead, *configure the least parameters that make sense for that instance*.

Here is a list of all the parameter categories:

instance – defines and configures the instance as a whole

template – configures which template if any are we using

ESTeam – configures the Enterprise Scrum Team

leadership – configures the supporting leaders

customer – configures the customer, market, and Business Value

informationArchitecture – configure the information architecture including VLITypes, surfers, canvas, DODStandard, DORStandard, etc.

knowledge – configure the techniques and patterns used

valueList – state the Initial Value List and the ongoing Value List

process – defines and configures the overall process in terms of events, based on the domain and techniques used for the Initial Value List and Cycles

schedule – tracks all the events, including the Initial Value List and all the Cycle instances.

cycleTypes – defines and configures what Cycles are we using

metrics – defines and configures which metrics are we using

reports – defines and configures which reports are we having

calculations – defines and configures which calculations are we doing

scaling – configure the connections to other Enterprise Scrum instances

structuralPattern – configures which organizational pattern are we using

collaborationMode – configures which collaboration mode are we using

deliveryMode – configures the frequency in which we deliver value

deliveryTarget – configures the delivery target – type of entity we delivery value to

contractType – configure the contract type

Enterprise Scrum Configuration				
instance	ES Team	IA	Process	Scaling
template	Leadership	VLTypes	cycleTypes	Structural Pattern
	Customer	canvas	metrics	Collab Mode
		knowledge	reports	Delivery Mode
			calculations	Delivery Target
				Contract Type

Figure 1. Enterprise Scrum Configuration Canvas

Configuration never really ends, because some things change or we need improvement. We define the “ongoing configuration”, evolution.

Enterprise Scrum Configuration and Evolution

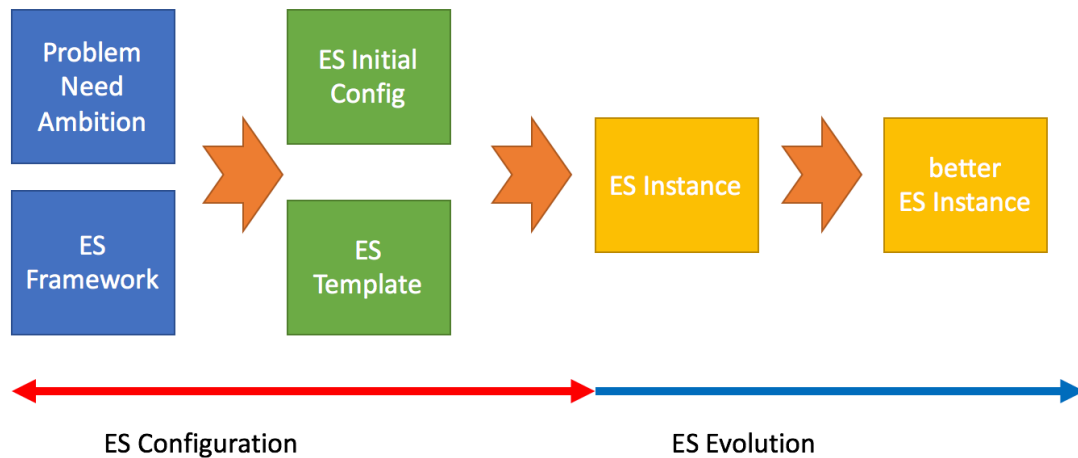


Figure 2. Enterprise Scrum Configuration Evolution

However, in Enterprise Scrum, you can use a shortcut to configure an instance by using a Template, as we show in the next section.

ENTERPRISE SCRUM TEMPLATES

In Enterprise Scrum we can use templates for different things:

- **Instance of Enterprise Scrum** – the whole configuration but, it is by definition or a template, incomplete
- **Enterprise Scrum Canvases** – just the canvas. There are currently 14 canvases, but they will grow in number fast. Since there are many applications of Enterprise Scrum, each one will get its own canvas eventually.
- **Initial Value Lists** – template for the Initial Value List
- **Cycle** – template for the Cycle sequence
- Etc.

These parameters *explicitly* configure a new instance of Enterprise Scrum by copying all the available parameters in the template to the new instance. See the **template** section of the configuration.

For example, a template may have some entries with values already for: **ESTeam**, **Customer**, **InformationArchitecture** (including VLITypes, surfers, canvas, etc.), **ValueList**, **valueListSequence**, **Techniques**, **Cycles**, **cycleSequence**, *etc.*; but not for the initialValueListEvents, cycleInstances, or the schedule since those hold unique calendar data for the instance.

AGILE MINDSET AND AGILE CULTURE

Culture and mindset are the most important aspects of Scrum and Enterprise Scrum. No process, framework, methodology is any good ... if it is not powered by the will of people. And you really can't force people to do things they don't want to do -- you will always lose, even if some of these things are for their own good.

"When the Spirit of a people is strong, focused, and vibrant, wonderful things can happen. When the Spirit is down, it makes very little difference how good your reputation, how much money you have in the bank, or how strong the need for your goods or services. Not too much happens."
[Owen-PowerOfSpirit].

Enterprise Scrum, Scrum and Agile, work a lot better if the people doing the work, are competent, engaged, and collaborative.

People are much more collaborative, engaged, and want to become more competent, only if:

- 1) they believe in the purpose of what they are doing,
- 2) they like doing that kind of work, and
- 3) they like working with the people that they work with.

This 3 conditions above, will make it much easier to have the right values to be more Agile; in other words, it will be easier for them to act with an Agile Mindset.

But be careful, an Agile Mindset, can't be imposed or forced onto individuals, because

you can't impose or force values – they must come from the heart to be long-lasting.





Figure 3. Enterprise Scrum Values

An Agile Mindset – operating with the right attitude and values, among several people leads to higher trust.

When many people that work together have an Agile Mindset, we say that they have an Agile culture; therefore, in an Agile Culture, there is more trust among the members of a team or organization. Nonaka describes this as BA in Japanese – a “sharing place”, where people can: share knowledge, collaborate, cooperate, ask for help, and provide help to others.

I have recently come up with a concept that captures all the above thoughts as a more “selective space” in the overall Agile space, as Resonant Agility. This concept represents the Competence-Engagement-Collaborative Model of Enterprise Scrum.

 **Mike Beedle**
@mikebeedle

 **RESONANT AGILITY** - the agility I believe in:

- ✓ INVITE highly-CAPABLE people to co-CREATE something GREAT we:
- 🔥 BELIEVE in the PURPOSE of
- 🔥 WANT TO DO
- 🔥 do with PEOPLE we LIKE and RESPECT
- ✓ to benefit PEOPLE: CX, UX, EX
- ✓ all-at-once RESULTS-driven
- ✓ always IMPROVING

4:07 PM - 16 Dec 2017

Figure 4. Resonant Agility

ENTERPRISE SCRUM TEAM

As in Scrum we only have 3 roles: **Enterprise Scrum Team**, the **Team** (people that do the work), a **Business Owner**, and a **Coach**, and they are true roles.

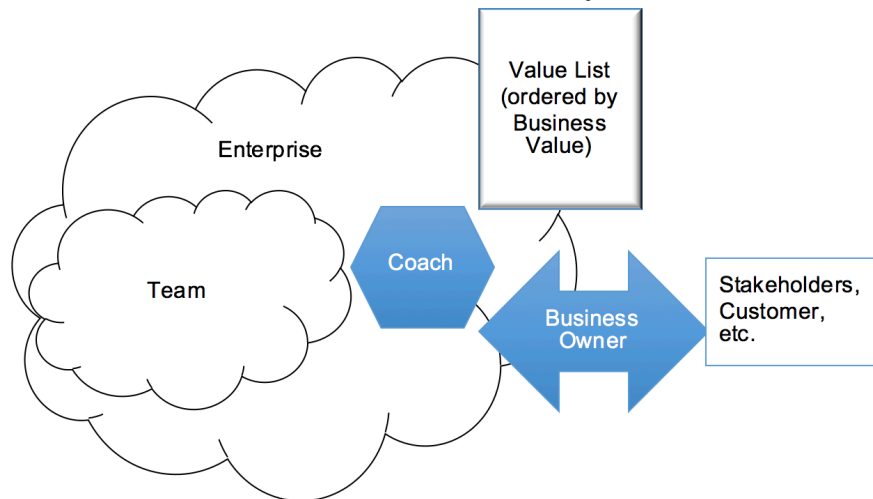


Figure 5. Enterprise Scrum Team

As opposed to Scrum, these are roles, meaning they can be distributed among several people. We will see though the example, that there are many ways to implement these roles, some which are more fitting to some environments than others. See the **ESTeam** configuration section.

We form **Enterprise Scrum Teams**, to deliver the most value in the shortest amount of time. And we do this while we balance other important things like: employee experience, profit and have a purpose in the world. In general, we can use Enterprise Scrum to improve, build or manage in any context. For example, we can use Enterprise Scrum for a soccer team, where we can plan how to play one or more games and then inspect and adapt.

Enterprise Scrum teams can range from all roles distributed among the team members; all the way to the traditional Scrum roles, where the **Business Owner** and the **Coach** are a unique single person. These one-person role implementations are indeed useful patterns to be followed in the appropriate contexts, but limiting for all possible circumstances. For example, in Enterprise Scrum it is possible to have: companies without CEOs, scaled development teams without Chief Product Owners, or self-direct, self-managed and self-organized teams where the Business Owner and Coach are distributed among the team members like they often are in a startup.

An **Enterprise Scrum Team** is as autonomous, independent, intelligent, self-directed, self-managed, self-organized, self-correcting and self-selected as much as possible. An **Enterprise Scrum Team** is akin to Smart Tribe with intense customer focus

[SmartTribes].

Once we liberate the enterprise from the chains of functional hierarchical management and handoffs, the different Enterprise Scrum Teams can operate much more autonomously. Granted, if there are dependencies among business units, customer segments, or products or services; there needs to be agreements among the different teams to coordinate these dependencies.

When doing Enterprise Scrum, we want to decouple and decentralize teams as much as possible to seek higher autonomy. Scale by de-scaling.

Enterprise Scrum Teams operate at a much higher operational efficiency than traditional teams. This is based on reducing waste, and avoiding hand-offs, functional silos or deep delegation hierarchies. We can understand this, because Agile and Scrum work, is “reengineered” or compressed work. Also, from an implementation perspective, Scrum and Enterprise Scrum, use many Lean principles and techniques, which originated at Toyota, of course. For example, the **Value List**, is a queue of work akin to kanban, where the inventory pulled into the team are the **VLI**s (**value list items**), which deliver granules of **Business Value**. The **Cycles** by which we accomplish work are Just-In Time work cycles using small batches of work. And the **Retrospectives** in every **Cycle** are Kaizen (or Deming) Circles.

We have some empirical evidence that Scrum and Enterprise Scrum can deliver anywhere from 100% - 10,000% productivity gains. But again, the main goal of a Scrum Team is to deliver a *balance* of important things: it is useless to achieve a 2000% productivity if the customers are not satisfied, or the employees feel unhappy and leave the company.

An **Enterprise Scrum Team** may be formed to manage a company, a portfolio, a customer segment, a functional area like strategy, marketing and sales, compliance, etc. – although we much prefer to use all-at-once organizational structures.

Enterprise Scrum Teams can work in many different scaled modes using the **scaling** configuration options explained later.

Business Owner

The **Business Owner**, helped by the stakeholders, defines and orders an **Initial Value List** – a list of **VLI**s (**value list items**), that when **DONE** according to a **DOD** (definition of done), each contributes with some visible and measurable **Business Value** directly or indirectly to the customer.

The responsibilities of the **Business Owner** are:

- Manage the vision through the **Value List**
- Manage the **Value List** at all times by defining, prioritizing, or changing it. All of these actions are termed “refinement of the Value List”.
- Working and interacting with the customer and/or the stakeholders.

- Delivering **Business Value** to a customer (or a Customer segment).
- The **Business Owner** is the only person that can modify the **Value List**.
- The **Business Owner** is also responsible for the success and failure of the project, and its ROI.
- The **Business Owner** coordinates, interacts and represents the stakeholders and makes consensus about what they need or want.
- The **Business Owner** approves the work, or not, after each **Cycle**.
- The **Business Owner** always chooses what to deliver
- The **Business Owner** always chooses what to work on – even within a **Cycle**.
- The Business Owner can also provide support to the **Team** and the **Coach**

There is a tradeoff with the people that make up the **Business Owner**:

- **Many people in Business Owner**: with more people there is more knowledge and arguably better more balanced decisions, but there is also more coordination, and potentially, more politics
- **Single Person Business Owner**: with one person, we get expedited decisions, because there is less coordination, but there is less knowledge.
- **Distributed Business Owner**: it is possible that the team distributes the **Business Owner** role among all team members, the risk there is that the team either may not have the necessary knowledge, or the time to envision a “better future” while they do the work. However, this arrangement works well for startups.

When the work for the Business Owner is too large, it is also customary and convenient to build a **Business Owner Team**, a group of people that can help accomplish all the work of the **Business Owner**. For example, the **Business Owner Team** can have SMEs (subject matter experts, power users, business analysts, etc.)

Enterprise Scrum Coach

The **Coach**:

- helps configure the Enterprise Scrum framework for a specific purpose: Marketing and Sales, Software, Software Scaling, Business Agility, etc.
- coaches the **Enterprise Scrum Team** to implement the best Enterprise Scrum instance possible
- proposes initial team members, that hopefully select other team members.
- helps the team improve technically, socially, emotionally, collaboratively, etc.
- finds gaps and provides viable strategies to continuously improve everything: the team, the process, the configuration, etc.
- constantly empathizes, encourages, motivates, communicates, assesses, mentors, teaches, etc.
- matches people with what they really want to do, and to work with people they like
- schedules all Enterprise Scrum meetings and activities
- sends invites and agendas to all Enterprise Scrum meetings
- shows up to each Enterprise Scrum meeting and facilitates each one of the meetings in Scrum
- facilitates and encourages communications among all parties involved:

Business Owner, stakeholders, and team members.

- ensures all the deliverables for every meeting are **DONE**
- removes impediments for the **Enterprise Scrum Team**
- solves issues for the **Enterprise Scrum Team**
- etc.

Just like the case of the **Business Owner**, you can implement the **Coach**, with one or more people, and there are tradeoffs as well:

- **Many people in Coach**: with more people there is more knowledge and arguably better more balanced decisions, but more coordination and the possibility of inconsistencies
- **Single Person Coach**: with one person we get consistent coaching, and there is less coordination, but less knowledge, experience, and problem solving abilities. With a single coach, there is also less time to solve issues.
- **Distributed Coach**: it is possible that the team distributes the Coach role among all team members, the risks there are that, 1) the team can't see "beyond the forest", and could get stuck specially in social issues, 2) they just don't have the ability to be a "good coach" collectively.

Team

The **Team** is one or more people that do the work, but the preferred size is 3-9 people, like in Scrum. The **Team**:

- collaborates, cooperates, shares knowledge, and anyone in the team should help other members i.e. is in Nash equilibrium.
- gets the work defined in the **VLIs** (Value List Items) of the **Value List** **DONE** according to their **DOD**, and delivers **Business Value**
- chooses and *pulls* the amount of work into every Cycle
- always decides the scope to work that the team can get done as changes take place
- Teams accomplish their work in the most self-organizing way possible. But in Enterprise Scrum, we realize that not all new teams are ready to self-organize to do their work completely, so it is possible, that they get a mentor or architect within the team to get started, and evolve into self-organization.
- Ideally, the team tells the team how to do their work.
- Uses its cross-functional expertise to get the work **DONE**.
- Ideally, selects other team members as team evolves i.e. is self-selecting. That's not always possible, but it is still ideal.

LEADERSHIP

Enterprise Scrum provides a built-in Agile Leadership model, that is localized for each agilized activity. We try to make organizations as independent, both horizontally and vertically; but realistically, there are always some leadership relationships and dependencies.

Agile Leadership can be applied in two different forms independently, or combined.

These two ways are by:

- **External.** Having leadership participate in the individual canvases with the purpose of providing support. For example, participating and providing support participating in a customer segment ES – BMC. The Business Owner role here is critical, as it can front-end much of the communication with leadership making them stakeholders. A well-designed organization will not have too many stakeholders – that is already a red flag that the organization is probably not designed very well.
- **Integrated.** The leadership team having their own canvas with activities that provide support for other teams. For example, being part of an **Enterprise Scrum Team** that is managing a company – a portfolio of business units in a company, and providing support by communicating and collecting customer segment feedback. In other words, the leadership team is/are Business Owner(s), of a supporting organization, and it is assumed that the Business Owner of the organization

Ideally, the leaders associated with an Enterprise Scrum instance, are also using Enterprise Scrum for a different purpose (integrated); but we can't assume that in general – that's why the leaders may be external.

Ideally, the leaders in the leadership team would provide support in many ways:

- resolve leadership issues: e.g. conflicting direction, removing political situations, or other impediments the team can't.
- resolve issues with others in our behalf:
 - o provide funding
 - o approve hiring people
 - o get other types of approvals or resources
- provide direct support for the team:
 - o make some decisions that only leadership can make
 - o communicating important information
 - o resolving team issues that only leadership can resolve
 - o help the team make some decisions
 - o help the team resolve dependencies
 - o help us establish partnerships and collaborations
- adapt their leadership style depending on the needs of the team(s). Different teams may need, in fact, different styles of leadership.

In terms, of leadership style, here is the Enterprise Scrum Leadership Model:

Enterprise Scrum Agile Leadership Model

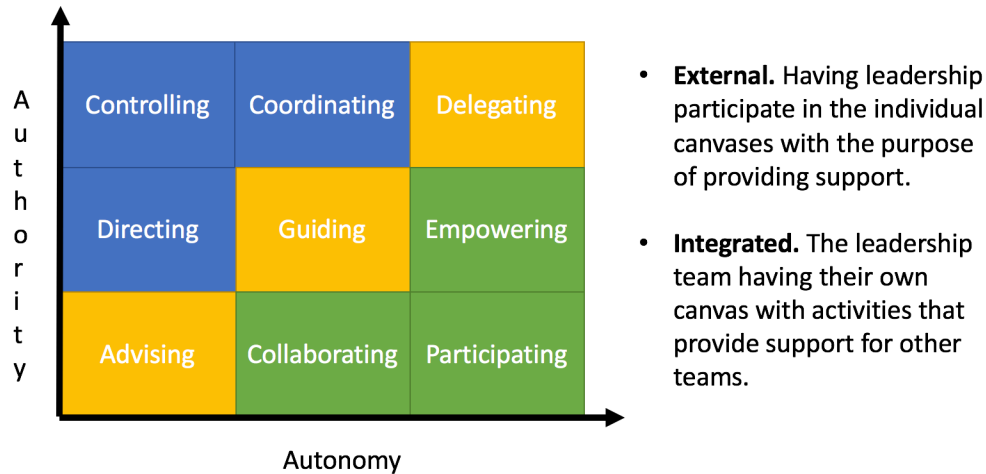


Figure 6. Enterprise Scrum Agile Leadership Model

In Enterprise Scrum, in terms of the freedom that the team has in “doing their work”, we assume that the Team can at least be **empowered** to self-organize, and prefer the leadership **collaborating** and **participating**. So we clearly prefer to be in the green area; but we realize that sometimes other options are needed like temporary controlling, coordination, directing, advising, guiding or delegating actions.

The leadership group is a different group than the stakeholder group in Enterprise Scrum, because by definition, a stakeholder is a person having influence i.e. “say so” in Value List, but a leader, the way it is defined in Enterprise Scrum, is about providing supporting. There could be an overlap between the stakeholders and the leaders, of course; but we don’t make any general assumptions of overlap between the leaders and stakeholders, that’s why there are 2 separate lists.

CUSTOMER

Enterprise Scrum seeks to deliver the most Business Value in the shortest amount of time to a customer or customer segment. Therefore, it is a good idea to know what industry, what area of the industry, what segment we are serving, and what **Business Value** is for our customers. That way we can better organize our **Value List** and help them seek a better purpose. See the **Customer** configuration section.

Business Value

Business Value in Enterprise Scrum is measured through a business relevant **metrics** in that process, domain or industry such as one or more of the following: customer satisfaction, revenue, profit, ROI, market share, a measure of competitive advantage; but we prefer that this is *realized business value* not *potential business value*.

Business Value can be anything of value in that domain or for that industry. For example, for compliance management, it could a higher level of compliance or less incidents; but for a software development effort, it is typically features done.

We must measure **Business Value** appropriately to have empirical evidence that we in fact, have obtained some **Business Value**. This will also give us a metric to improve the delivery of **Business Value** in the future through **Cycles**. It is also typical to create **charts** for the **metrics** for our measurements or calculations.

KNOWLEDGE

In Enterprise Scrum we can configure the specific **patterns** and **techniques** used and for what purpose. Also, we have the option to gather knowledge over time, as we learn and convert “chaos into order”: chaos into complex, then complicated and then simple [Cynefin].

See the **knowledge** section of the configuration.

Patterns

Patterns for different domains are available as “best practices”. Enterprise Scrum can also help us to list and remember patterns in our instances, either learned or discovered.

For example, If we are building a startup, we may want to use Business Model Generation patterns [BusinessModelGeneration]. If we are doing product strategy, we may want to use some of the Profit Patterns available [ProfitPatterns]. If we are building an Enterprise Architecture, we may want to use enterprise architecture patterns. If we are building a software architecture, we may want to use the best possible design and architectural patterns [DesignPatterns], [POSA1], [POSA2].

Techniques

Likewise, we can introduce a number of **techniques** to get the work done for different domains. For example, for Company Management:

- Blue Ocean Strategy - Strategic Canvas, ERRC [BlueOcean]
- Design Thinking [DesignThinking]
- Lean Startup- pivoting [LeanStartup]
- Business Model Generation - BMC, VPD [BusinessModelGeneration]
- Profit Zone – Profit Patterns [ProftZone], [ProfitPatterns].
- Exponential Organizations - MTP [ExponentialOrganizations]
- Scenario Planning – Scenarios [ScenarioPlanning],
- Red Ocean Strategy – competitive analysis, 5-force analysis [Porter1], [Porter2],
- Beyond Budgeting – rolling budgets [BeyondBudgeting].
- Etc.

Techniques can be high-level, and define extra steps in the **Initial Value List**, for example, Strategy Canvas from Blue Ocean Strategy; or they could be extremely low-level techniques through execution, for example the usage of Design Patterns in software development.

Techniques may also result into many specific steps in the process of an instance, which can be accounted, configured and remembered in Enterprise Scrum. For example, using User Stories in software development triggers: User Story Workshops, Planning Poker, User Story Refinement, etc. This is important information that we can capture, if we wish to re-instantiate the instance for some other team.

INFORMATION ARCHITECTURE (ADVANCED SECTION)

The information architecture of the domain breaks down what we do:

What do we deliver and to who?

What are the list of deliverables?

This translates into VLITypes.

What is the principal way we deliver value?

What are the aspects of what we deliver and who is responsible for them? These are the Surfers.

What are the attributes of the surfers?

These translate into the additional attributes of a VLIType.

The information above derives:

- the canvas for the activity,
- the initial value list for the activity, and
- the cycle structure

Canvas

In Enterprise Scrum, we coevolve the Surfers on the Canvas (all important aspects or attributes of ALL important things == VLITypes), through waves of Cycles in subsumption mode. In Enterprise Scrum, we create different canvases for different activities, to make it easier to see all the aspects at once -- our coevolved Surfers.

For example, if we are using Enterprise Scrum to manage a startup, we may want to use the Enterprise Scrum – Business Model Canvas:

ES – Business Model Canvas

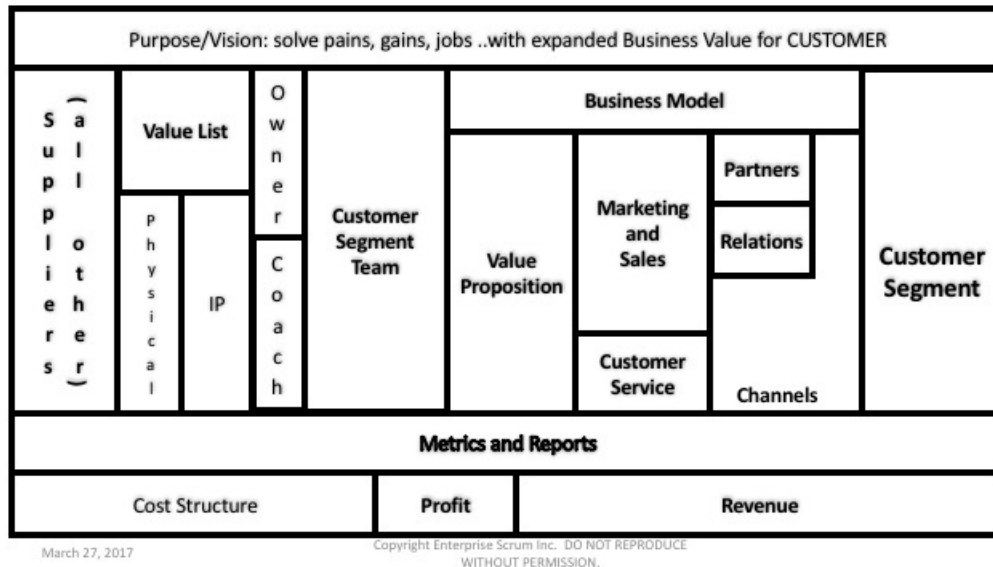


Figure 7. Enterprise Scrum Business Model Canvas

All of the boxes above, are all the VLITypes, by definition, and each one of them has one or more attributes – we call these the surfers.

Similarly, we can build canvases for any activity: company management, executive management, startups, portfolio management, business units, customer segments, products, services, marketing, compliance, software development, etc. See the included canvases.

This is in fact such a good practice, that we have added this as a general technique: **Build Your Canvas Pattern.**

It is actually really easy. We just ask the right questions:
 What do we deliver after each Cycle? Those are the VLITypes.
 What attributes do these things have? Those are the “surfers” or attributes of the VLIs.

For everything you manage in Enterprise Scrum you can build a custom Canvas that includes “everything that is important” and co-evolve it together through Cycles.

For example, in software development it is important to track customer satisfaction, requirements, architecture, the plan, the development team, etc. The Canvas, by definition, must include all important things – all the “surfers”, attributes of the VLITypes.

The list of surfers, then map to the scheduled events to build the Initial Value List, and the PC3R Cycles, if in addition to the above questions we ask: Who delivers these aspects? Through which activities?

That's how we get from "important things" to an Initial Value List process definition in Enterprise Scrum.

Within an instance, we can use the Canvas to:

- Fill in the "initial big picture" through the Initial Value List
- Then, we can coevolve everything, including the dependencies among VLITypes, through the different Surfers through the PC3R **Cycles**, until we reach our objectives.

Canvases are useful as a **template** to get started faster using Enterprise Scrum.

VALUE LIST

The **Value List** represents the vision at any one time, and targets a purpose, through a list of **VLIs**, each of which bring **Business Value** to a customer, internal or external. There is one **Value List** per **Scrum Team** which is composed of **VLIs**. This is how an Enterprise Scrum team manages all of its work.

Enterprise Scrum, like Scrum, is a system for getting things done all-at-once in each **Cycle**. For different types of work, we will have different **VLITypes**, that coevolve different important aspects. See the **VLITypes**, **ValueList** and **ValueListItem** configuration options.

For example, in financial trading, a trade may have 1) business requirements, 2) an implementation strategy, 3) compliance requirements, 4) reporting requirements, etc.

We may have other **VLIs** that add direct, or indirect **Business Value**, but we typically have what we call a **primary VLI** – a primary activity that brings business value, and potentially several other activities. For example, in a real-state process, the properties to be sold, are the primary VLI.

VLIs

Each **VLI** delivers granular, direct or indirect, **Business Value** for a specific domain. For example, a VLI for real-state sales would be a property sold, and we coevolve 1) the price, 2) the sale terms, 3) the property, 4) the marketing of the property, etc. In software development a **VLI** is a feature or "user story", and we coevolve 1) the requirements, 2) the team, 3) the plan: budget, schedule, risk; 4) the architecture, 5) the DOD, 6) the deliverable product increment, etc.

VLIs can have many different attributes depending on their type. The VLI attributes are a very important and key aspect of Enterprise Scrum, because they generalize work. We call the aspects or attributes of a VLIType, the **Surfers**.

For example, a VLI will eventually have information about:

- the plan to get it DONE,
- its DOR (Definition of Ready), different than the DORStandard

- its DOD (definition of done), different than the DODStandard
- volunteers that completed the work,
- dependency with other VLIs,
- domain-specific “surfer type” attributes,
- administrative information: who created it, when was it added, approved, what it’s budget, revenue, profit, or other related metrics.
- time constraints, VLIs can be cycle, un-selectable, scheduled, repeatable;
- structure information: singleton, collection, workflow or conditional;
- regarding action type, they can be work, refine, decide, test, answer, monitor.
- Etc.

As stated above, once we map the extra **VLI** attributes for a VLIType, indirectly we will need to map the Initial Value List activities, and what gets **DONE** in the PC3R **Cycles**. Two very important attributes that deserve additional clarification are the DOR and DOD.

VLI-DOR

In Enterprise Scrum, there must be an explicit DOR (definition of ready). The DOR tells you when a VLI is ready to be worked on.

VLI-DOD

In Enterprise Scrum, as we do in Scrum, **DONE** is not a fuzzy-defined thing. We must specify “when are we done”, for any business process where we want to use Enterprise Scrum as specifically as we can.

We prefer that **DONE** is not “partially done”, “almost done”, “I think it’s done”, and that instead, it is a well-defined concept. For example, in a real-state process, a sale, has a definite **DOD** (definition of done):

- 1) we must have a completed inspection
- 2) we must have a signed sales contract
- 3) we must have a cashier’s check that includes the total price, fees and closing costs
- 4) etc.

And the same is true in other business processes. Our goal is to get things **DONE** according to **DOD** each and every **Cycle**.

ENTERPRISE SCRUM PROCESS

The Enterprise Scrum process is very simple: it is an iterative incremental process through Cycles that starts with a Vision and an Initial Value List to accomplish that vision, but the activities of Initial Value List and the Cycles can be configured based on the canvas for that domain.

Vision
Initial Value List

Cycle
Cycle
Cycle
Cycle
Cycle

(... until we achieve our goals.)

In Enterprise Scrum we do iterative incremental empirical-process management by measuring one or more things after each **Cycle**, and then improving the work, the team and the vision through feedback as needed to achieve our goals. See the **InitialValueList** and **Cycle** configuration sections.

Cycles can be as short as one hour, and as long as a year. They can also be nested in time and structure. For example, we could have bi-weekly **Cycles** contained by quarterly **Cycles** (a very common pattern). Or have Business Unit cycles wrapping Customer Segment **Cycles**.

Cycles are also associated with the **Surfers** as define above -- important things that are coevolved in subsumption for that level, as we talked about when we introduced the **Value List**. Each cycle, we do **PCRI: Plan, Collaborate** to get DONE, **Review** or **Improve** anything in the canvas.

Sometimes is convenient to have a vision for what will happen after many cycles. We call this a **Release Plan** or **Roadmap**, and sometimes it is useful to document it.

The **Scrum Team** can perform **measurements** on the **metrics** on what got **DONE** according to a **DOD** such as: customer satisfaction, employee experience, profits, sales, compliance levels, etc.; and make **Plans** based on these **measurements** to improve, adapt, and manage.

We will use this process as a template to describe different processes such as Startup management, portfolio management of all sorts, compliance management, marketing and sales management, etc.

Vision

The **Business Owner** helped by the stakeholders define a vision. Typically to delight the customers. But again, keep in mind we must balance employee happiness, profits, and having a purpose in the world or whatever else is important in that domain.

Think of the Vision as a “fuzzy deformed ghost” that becomes real as you get more thing done: you can’t see the end exactly it doesn’t matter how smart you are. So, it is just that, a vision.

Initial Value List

The **Business Owner** helped by the stakeholders map the **Vision** to an **Initial Value List** by defining the individual **VLI**s to accomplish the **Vision**. This typically is achieved

through a series of activities to accomplish the DOR (definition of ready - see below), and through an **Initial Value List** process as described above.

For example, in Startup Management, a VLI is ready if it has been defined, prioritized, and sized in effort and cost – this is different than standard Scrum. In general, these activities may change depending on the DOR and depend on the activity, and therefore the Canvas for the activity. Generalizing, software development, sales, compliance, HR, finance or portfolio management, have different Canvases and therefore different Initial Value List processes and DORs.

Cycles and Cycle Structure – PCRI

In Enterprise Scrum, we generalize the 2-level subsumption architecture behavior of Scrum, into an n-level and m-things in subsumption for any domain. Subsumption simply means “checking with the surrounding reality of that domain” and adapting as needed by changing behavior.

Therefore, in Enterprise Scrum, we must decide, which things are coevolved in which Cycles, and at what level, and then coevolve them through appropriate Cycles. For example, for a real-state sales process, we may choose to coevolve the sales activities and the self-organization of the team at the day level, and the 1) overall sales, 2) sales projections, 3) sales infrastructure and 4) overall team improvements at the 1st Cycle level, usually with a duration of 1-4 weeks.

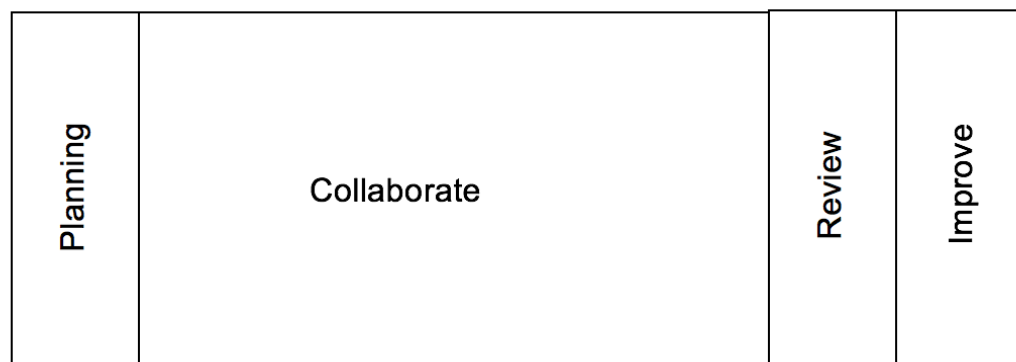


Figure 8. Cycle Structure in Enterprise Scrum

The Cycle structure in Enterprise Scrum is:

Planning – where the Team determines in **part 1**, how much work they think they can do for all Surfers. The workers PULL IN the work. In **part 2**, how they will do the work. For example, the team members can volunteer for the VLIs accepted and plan for each of them according to a **DOD** (definition of done). The Enterprise Scrum team plans for everything in the Canvas. For example, a sales team may want to plan for: 1) marketing message, 2) campaigns, 3) feedback techniques, 4) closing techniques, 5) overall sales, 6) sales projections, 7) sales

infrastructure, and 8) team improvements, etc. In Enterprise Scrum part 2, and even the whole planning can be configured as optional depending on the context.

Collaboration – the goal is to accomplish as much work that passes a **DOD** through collaboration, cooperation, helping each other and sharing knowledge for that Cycle. Ideally, the Business Owner is clarifying things as needed and pre-approving as many VLIs instead of waiting for **Review** to do so. The Daily Scrums are also optional, depending on context. If the team is interacting and know their dependencies all the time, they are not mandated.

Review – This is the last opportunity for the Business Owner to review and accept the work DONE in a Cycle to incrementally deliver some Business Value. It is much preferred that we actively interact with the Business Owner during the Cycle to approve things before we get to the Review. For example, a sales team may want to review: 1) marketing message, 2) campaigns, 3) feedback techniques, 4) closing techniques, 5) overall sales, 6) sales projections, 7) sales infrastructure, and 8) team improvements, etc. At **Review** we can also review anything of interest in the canvas, but we always review the **primary VLI** first - the **VLIType** that provides most **Business Value**. For example, in software development, we first review the features or requirements of the product increment.

Improve – At **Improve**, we have the opportunity to improve anything of interest on the Canvas including the team. The main idea is to choose a better direction to go to including everything we do. For example, a sales team may want to improve: 1) marketing message, 2) campaigns, 3) feedback techniques, 4) closing techniques, 5) overall sales, 6) sales projections, 7) sales infrastructure, and 8) team improvements, etc.

We call 1) marketing message, 2) campaigns the **Surfers** of Marketing and Sales for the 1st level Cycle, and the ES – BMC above.; and we call the 1) day-to-day sales activities and 2) the self-organization of the team the surfers of the day cycle.

METRICS

Metrics track important things that were DONE in a Cycle that are important to PEOPLE: the customer, the employees, stakeholders, and even people in the world. As such, **Cycles** can be used to improve one or more **metrics**, which may be of different metric types, for example:

- customer satisfaction
- customer experience
- employee experience
- profit
- cost savings
- productivity

- quality
- performance
- learning
- accomplishment
- building
- adaptation
- transformation
- improvement or
- growth
- etc.

or whatever is important for whatever we are trying to do.

To visually see the progress or evolution of the **metrics** within a **Cycle** we can use one or more **charts**. For example, we can use a Burndown chart or a different graph to track one or more things simultaneously.

CALCULATIONS

In Enterprise Scrum, we can have calculations related to the Cycles using the data measured before. For example, we can make projections of sales, customer satisfaction, compliance, effort or budget based on previous measurements in the previous Cycles.

This doesn't mean that we make "static future plans" in Enterprise Scrum. It only means that we can make the "best prediction" given what we know at some point in time. For example, we may calculate schedules, budgets, scope to be achieved by a certain date, customer satisfaction, total net sales, etc.

Here is an example for Schedules and Releases

The value list size is:

VL-Size = **size** (from the **valueListItem** in the appropriate **sizingUnit**)

Number of Cycles:

ICs (Cycles) = VL-Size (Value List size) / V (velocity in the appropriate **metric**)

Then we simply calculate time as:

Schedule = ICs * **length**

Where **length** is the **length** parameter in the Cycle.

Burn rates

The total cost is:

TC (Total Cost) = resources * average salaries * HR factor + proposal cost + other fixed costs

And therefore the burn rate is:

BR (Burn Rate) = TC/Number of Yearly Cycles

Budgets

Since we know the burn rate and the number of Cycles, we can then calculate the budget:

\$ (Budget with no profit) = ICs * BR

Even with a profit percentage X:

\$ (Budget with X% profit) = ((1 + X)/100) * \$

We can do similar calculations for sales projects, net profit, cumulative customer satisfaction, etc.

We can in fact include a “repeatable VLI” with the purpose of producing this calculation, as an explicit way to include this calculation in a given **Cycle**. This makes it *natural* to include Release Planning calculations, or any other projection.

EXPECTED BEHAVIOR AFTER MULTIPLE CYCLES

Rome wasn't created in one Cycle; and neither is an Agile business. Growing an Agile business is an evolutionary process. And it takes evolution in time to transform organizations, to grow products or services, and to find better business directions. In fact, a recurrent and common pattern in just about every subject that has hard activities or goals is that of “Iterative Incremental growth”.

Iterative Incremental growth or adaptation, is found as a fundamental principle in Cosmology, Architecture, Biology, Physics and Mathematics, Political Science, Medicine, etc., and it also relates to the metrics above. Iterative Incremental growth is at the center of revolutionary thoughts like the Alexanderian architecture concepts explained in the Timeless Way of Building [Alexander].

Henrik Kniberg makes the point that the Iterative and Incremental is also a risk management technique: we don't have to know everything up front, and we don't have to deliver the value all at once.

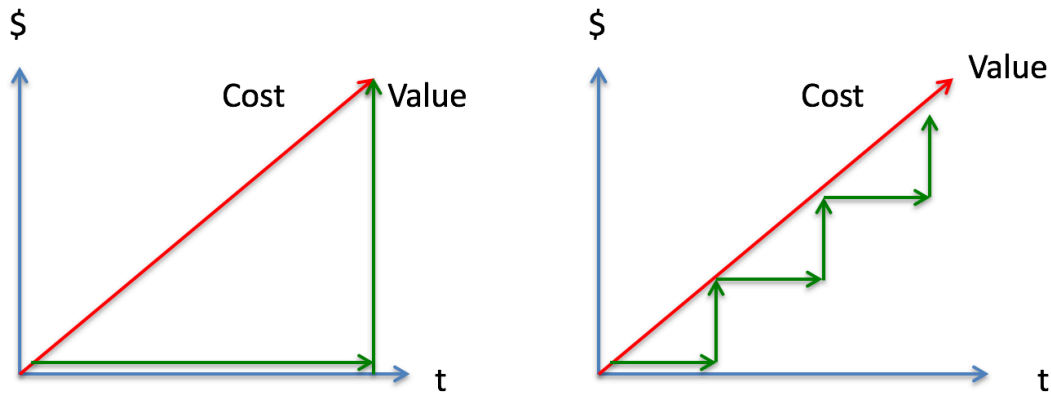


Figure 9. Comparison of business value delivery from Waterfall to Iterative Incremental

Enterprise Scrum brings *professional iterative and improvement management* to the whole or any part of organization; and if you do it right, it brings with it all of its cultural tradeoffs of sharing knowledge, cooperation, collaboration and helping each other.

Improvement and Growth

There are many instances of Enterprise Scrum where the expected behavior is improvement and growth through exponential growth with saturation. This is because “growth” can only go so far, for example, velocity in a software development team, or profits (from sales) in a saturated real-estate market.

We can grow our business models, products, services, our teams or transformations. In Enterprise Scrum and Scrum everything is done iteratively and incrementally; so we **grow** everything: sales, revenue, profit, quality, customer satisfaction, organizations, transformations, quality, etc.

Growth typically follows an S-curve, or a sequence of S-curves, really.

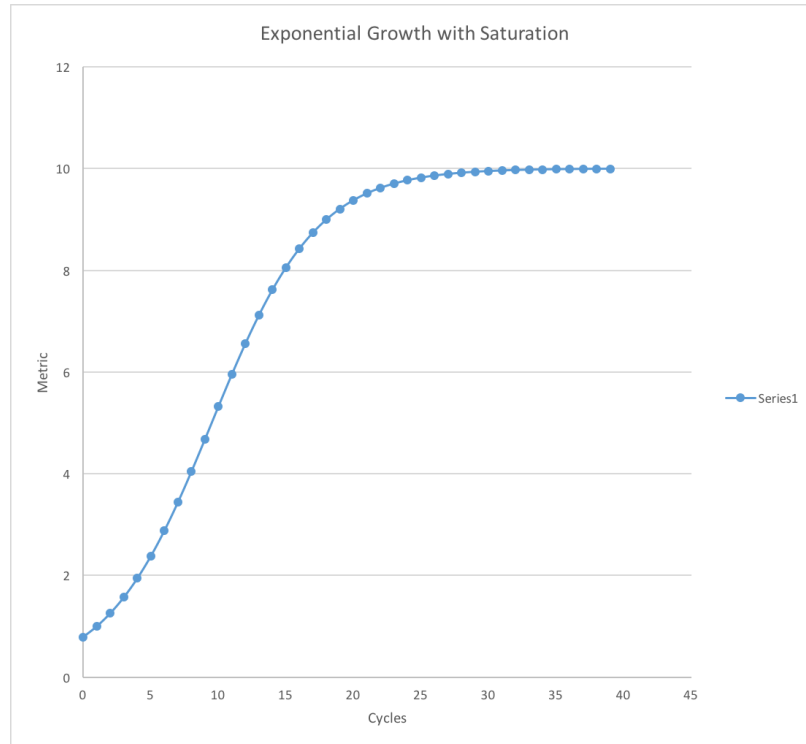


Figure 10. S curve expected growth or improvement over time.

One of Jeff Sutherland’s first Scrum articles advertised a measured 600% improvement. But of course, this improvement was not achieved in one Sprint – it was the iterative incremental growth process. By the way, I prefer not to call either Sprints or Cycles “iterations”. Iteration gives you the wrong impression that you are iterating “doing the same again and again”; but in Scrum as in Enterprise Scrum we don’t want to do the same – we want to improve and do it better after every Sprint or Cycle.

In Enterprise Scrum, we don’t limit ourselves to improve the “amount of work DONE” or velocity, and neither do the best Scrum teams. Instead, we rather seek to improve customer satisfaction, employee experience, quality, return on investment, sales, compliance levels – just about anything we want through the introduction of metrics. In fact, we prefer to improve, optimize or maximize not only one quantity but several of them together to seek a more “balanced management”.

We should expect a saturation curve or S curve for our growth and improvements: 1) at first it’s hard to improve, 2) then we quickly improve but 3) slow down as we reach the saturation point, where it may become more difficult to improve.

The Enterprise Scrum S-curve is modelled by a variant of the “logistic equation” or sigmoid curve:

$$F(x) = L / (1 + B * e^{(-k * (x - x_0))})$$

Equation 1. Exponential Growth with Saturation

where:

- e = natural logarithm base
- x_0 = the midpoint value
- L = the curve's maximum value
- $L/(1 + B)$ is the initial $f(0)$ initial metric
- k = steepness of the curve

Adaptation and Moving Targets

One universal aspect of open information systems, is that implicitly, they may cause our goals to shift or change over time. For example, a changing regulatory environment, could dramatically change the features of a hardware product, like an oil rig, a turbine, or telephone switch. Yet, we must adjust our work to “chase the moving target”.

Fortunately for us, Scrum and Enterprise Scrum are designed for that purpose: to easily adapt or change as NEW information changes or comes into the process. This makes Scrum and Enterprise Scrum a better fit for processes with changing information.

After an iteration at any one level we could make adaptations at different levels **to optimize what we defined as business value.**

- Change our business model, or
- Change our product, service or process – changing the concept of them, or
- Change or add features to an evolving product, service or process, or
- Improve our team, process, or customer service.

How can do we accomplish this? By making things transparent after each Cycle (weekly, monthly, quarterly, etc.), so that we can inspect and adapt after each Cycle choosing better behaviors for our company, program, project or team. This works is a nested multi-level subsumption architecture – choosing behaviors by analyzing context, at every level of scale; which has always been an important element in Scrum.

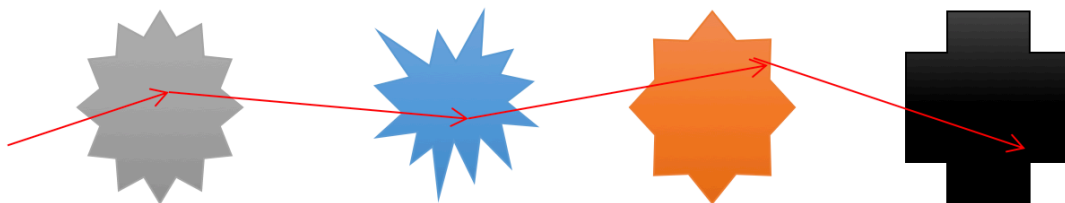


Figure 11. Change business models, products, services, features as you get feedback.

Recursive Cycles

A common thing to do among very many Enterprise Scrum instances, is to add an envelope **Cycle**:

Vision → Big Cycle (Long-term wave prediction)
 Small Cycle (corrective prediction)
 Small Cycle (corrective prediction)
 Small Cycle (corrective prediction)
 Small Cycle (corrective prediction)
→ Big Cycle (end)

In Software Development or Startup Management, this is called Release Planning. For example, for Startup management we are interested in finding a viable:

- Business Model
- Product or Service
- Effective Marketing
- Revenue/Profit

simultaneously, so we would first try an initial business model with a product or service, with an initial marketing in our “big Cycle”, use some metrics on customer satisfaction, profit and revenue and then adapt our initial predictions on Business model, Product and Service, Marketing in each of the small Cycles.

Another example, in software development we try to predict:

- Requirements
- Architecture and
- Plan

So, we can do a first pass making these long-term prediction “on the rough”, and then at the end of every small Cycle, make a correction to the original “long-term predictions. This is what leads to the release plan is then updated after every **Cycle**. You can nest as many **Cycles** as needed as long as you update the envelope plans after each of the smaller Cycles.

Wave Principle

In general, this behavior can be captured, through the **Wave Principle** is:

Wave Principle. *Any long-term “rough” predictions made in the Initial Value List of a longer Cycle, must be refined or recalculated after each and every shorter Cycles it includes.*

The Wave Principle allows iterative-incremental coevolution and projections of multiple aspects or entities.

The wave is a useful metaphor, because each of the enclosing cycles in the wave triggers a “rolling wave” of “predictions”. This is exactly how Beyond Budgeting works.

But this behavior is now generalized in Enterprise Scrum for any activity:

Software Development:	requirements, architecture, plan
Company Management:	profits, revenue
HR:	projections on number of hires
Compliance:	projections on compliance or incident levels
Etc.	

By the way, it is a common misconception that “long-term” plans can’t be done, or should not be done with Enterprise Scrum (or Scrum) but this is simply not true: *As long as the longer plans are updated after each small Cycle, there is no problem in making longer term plans.*

From the implementation point of view, the long-term Cycle plan is implemented by a Value List – by a specific list of VLIs to do to achieve the long-term vision.

REPORTS

ES - ScrumBoard

To track how much work we have done, we use a **ES - Scrum Board** across many **Cycles**. The **ES - ScrumBoard** shows us how much work we have selected to work on a **Cycle** (selected), how much work is work in progress (WIP), and how much work we got to **DONE**.

We describe how Enterprise Scrum **PCRI** cycles work above, but let’s work through how these Cycles fit into the ES - ScrumBoard.

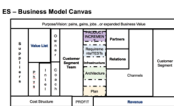
At the **Cycle Planning** we 1) move the chosen work from the “not selected” canvas or column into the selected/WIP column, and optionally, 2) we make an initial plan on how we are going to getting to **DONE**. For example, if we are doing compliance management, it could be an internal control that needs to be chosen and a plan on how is going to be tested. In Enterprise Scrum, however, we can choose 3 options to do planning: 1) traditional Scrum with part 1 and part 2, 2) just part 1, with the team self-organizing to make the plan on the go, 3) no planning at all, just self-organizing to choose the work and do the work working with the **Business Owner**.

At **Collaboration** we work on it and get it to **DONE**. For example, it could be an internal control that needs to be tested by generating a trading report.

At **Review**, we move everything that got **DONE** to the DONE Canvas or column, and then we meet with the **Business Owner**, and show or review what we got **DONE** in the entire Canvas at execution and explain anything that is important. Review sessions, can have many sub-parts – as many as boxes in the Canvas. But at the very least, they should have the traditional Scrum reviews on 1) what we got done, 2) how the team is working.

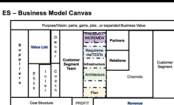
At **Improve** we improve anything we can that makes sense in the Canvas including 1) the things that deliver Business Value directly, 2) how to improve the team, 3) how to improve results and metrics, etc.

ES - ScrumBoard Vision

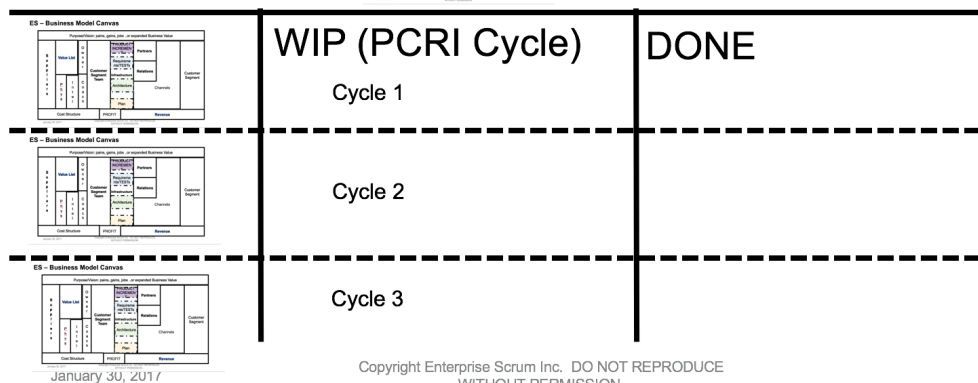


Business Owner and Stakeholders

Initial Value List



Business Owner, Stakeholders and Team



C
o
a
c
h

Figure 12. ES - ScrumBoard tracks how much work is getting DONE.

Typically, we define **metrics** on what we got **DONE**. For example, in compliance management we could keep track on

Charts

In contrast with regular Scrum, in Enterprise Scrum we are at liberty to use any chart that is useful to us, not just a “burndown chart”. This can be configured in the ES configuration.

For example, we could use a chart to track the metric “daily number of incidents” or even “cumulative number of incidents” within a **Cycle** in compliance. Or we could track “daily real state sales”, or “cumulative real state profits”, if we are tracking real-state sales process.

SCALING

Enterprise Scrum provides very many scaling options as well. These scaling options work not only for software development, but for any scaled Agile activity, such as marketing and sales, compliance, company management, transformation, etc.

These scaling options are organized in 1) structural patterns, 2) collaboration modes, 3) delivery modes, 4) delivery targets, 5) contract types. Each one of these options has 4 sub options, giving a total of $4^5 = 1024$ total number of combinations.

Why so many options? Because scaling is a very complex subject. Other frameworks trivialize the domain, and give the impression that canned, simple solutions can work; but this is a mistake that may lead to inappropriate solutions that may cause trouble, pain and harm.

Some of the options apply to single teams as well.

Apply to Single Team:

Delivery Mode: CDep, CDel, Cycle, Release

Contract Type: Fixed Price, Fixed Date (+DevOps), T&M, cost+

Apply to Single Team and multiple teams working together:

Structural Patterns: Centralized BO+C, Centralized BO, Collaboration V(T+BO), Chameleon

Collaboration Mode: centralization, delegation, collaboration, subsumption

Delivery Target: project/product/ process, program, portfolio, Enterprise

Let's explain each one of these options in detail.

Structural Patterns

Centralized Business Owner and Coach – works for 3-4 teams where there is a business expert. For example, implementing a trading system.

Centralized Business Owner – works well up to 7 teams, depending on context. Each team now has it's own Coach.

Potluck Scaling - Collaboration Virtual Business Owner – works well for portfolios where there is not a lot of number of contributors, and they can make agreements

Chameleon – same as above, but now with a coordinator that can work in different collaborations modes (see below).

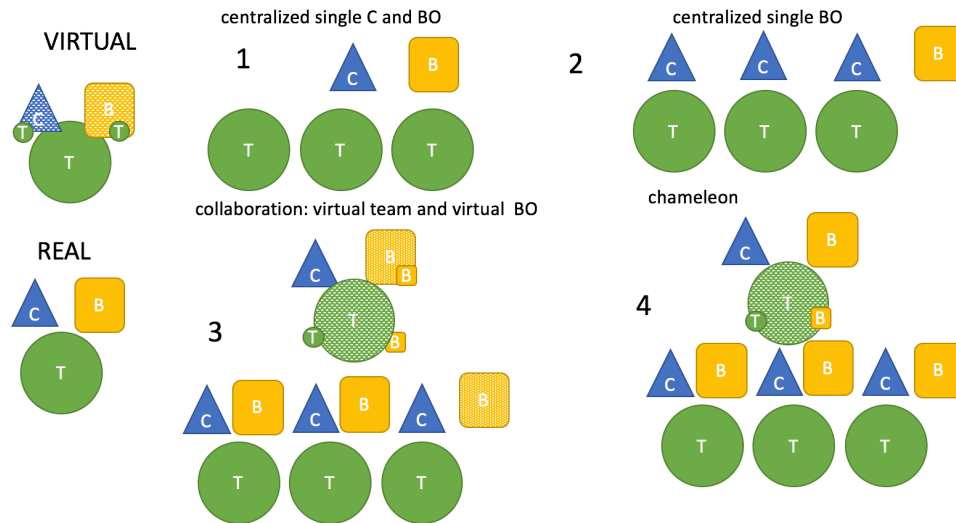


Figure 13. Enterprise Scrum Structural Patterns

Collaboration Modes

Centralization

There is a central **Business Owner** specifying and checking requirements.

Delegation

The teams are coordinated by a Chief Business Owner, and a Chief Coach in a hierarchical delegation mode. At the Initial Value List and at each PCRI Cycle, there are V-like meetings for Planning, Review, and Improve and then the work is “delegated to the lower teams”.

Collaboration

The work is done by the teams’ **Business Owner** making collaborative agreements with other teams. There is no one is a Chief anything.

Subsumption

Works like collaboration, but in addition, the **Business Owners** are part of a different subsumption level Enterprise Scrum Team. For example, all product and services teams for a Customer Segment, or a Business Unit, are part of the portfolio-level Scrum.

Delivery Modes

Continuous Deployment. Value Increment is continuously deployed as soon as possible by passing DOD and deploying immediately.

Continuous Delivery. Value Increment is continuously delivered by integrating and testing comprehensively and passing DOD as soon as needed, but not deployed to production.

Cycle Deployment. Value Increment is deployed by passing DOD each Cycle.

Release (Coincident multi-Cycle) Value Increment is deployed in releases but value can be delivered through continuous delivery or Cycle delivery.

Delivery Targets

Large Project – 2 or more teams working to deliver a large project

Program – 2 or more projects working through multi-disciplinary teams

Portfolio – 2 more teams contributing (dependent or independent)

Enterprise Architecture – ALL business areas included

This can work for any business process: SW, HR, HW, marketing and sales, compliance, etc.; or even the whole company, Business Agility.

Contract Types

T&M – time and materials, typically allows changes in both scope and release date

Fixed Date – typically means time and materials and variable scope. We want to deliver all the “must have” functionality and as much of the “want to have” and “nice to have”. When the fixed dates are recurring, we get an “operations”, “subscription like” recurring expense. Some people call this DevOps.

Fixed Price – typically means fixed scope and fixed date, but sometimes it can be variable date.

Cost+ - fixed cost for the initially agreed upon functionality, PLUS whatever changes additionally.

IMPEDIMENTS, ISSUES AND DEBTS

Enterprise Scrum provides a way to track, manage and resolve impediments, issues and debts. First, let's define these terms:

- **Issues** are problems with a range of severity.
- **Debts** are things that we should be doing but we are not.
- **Impediments** are issues that are blocking us to do work or get results.

Issues, impediments and debts can occur at many different levels of scale, from an individual, to the whole company.

Here are some examples of issues – some of them may be or become impediments:

- Not being able to connect temporarily to an LDAP server
- A problem with implementing a specific internal control
- Not getting enough feedback from the **Business Owner** in a Cycle
- Not having an answer for a specific decision: can we get the new servers by end of year?
- Etc.

Here are some examples of debts:

- technical debt
- refinement debt
- decision debt
- research debt
- monitoring debt
- change debt
- cultural maturity debt
- motivation debt
- learning debt

Issues and debts can be filled out at any time, but most of them typically come up at the Improve sessions.

We also provide a graphical representation that we call the **Debt-Issue Canvas** to make it easier for teams to see issues, impediments or debts. Impediments are issues that are marked typically with a red dot to highlight them.

	DEBT	ISSUES
technical		
refinement		
decision		
research		
monitoring		
change		
cultural maturity		
motivation		
learning		
collaboration		
leadership		
OTHER		

Figure 14. Debts and Issues Canvas

Enterprise Scrum Parameters

The full list and explanation of all the configurable parameters is in the **Enterprise Scrum Configuration Guide**, but we include in the next page the **Enterprise Scrum Configuration Guide Quick Reference**.

ENTERPRISE SCRUM CONFIGURATION QUICK REFERENCE

Shorthand Notation:

// comment

// assignment

=

// function calling with parameters

// all parameters are by reference

function calling and list of parameters = foo params[a, b, c, ...]

// allowed values

list of valid values = values<a, b, c, etc.>

// list of

list of a's = listOf(a)

// value of

value of parameter a = |a|

// relationships

// extends

A extends B = A inherits from B

// association

// class B uses or contains A

// or

// A is an attribute of B

// all associations are by reference.

B

contains{

A

}

ALL Parameters short list

Instance Parameters

EnterpriseScrum

contains{

// -----

// INSTANCE

// -----

instanceName,

description,

**morphing = values<yes, no>,
morphing Points listOf(morphing contains{id, description, changes listOf(parameters)})**

```
// -----  
// TEMPLATE  
// -----  
// IF using a template COPY the template  
// fields that are filled into THIS instance  
// -----  
template params[template name],  
  
// -----  
// ETeam  
// -----  
ETeam,  
  
// -----  
// leadership  
// -----  
leadership,  
  
// -----  
// CUSTOMER  
// -----  
customer,  
  
// -----  
// IA for DOMAIN  
// -----  
// information architecture → Cycles  
informationArchitecture params[customer],  
  
// canvas must include VLITypes and surfers “aspects” of the VLIType  
canvas = canvasFromIA params[informationArchitecture],  
  
// all VLITypes from canvas  
VLITypes listOf(VLIType) = VLITypesFromIA params[canvas],  
  
// ALL surfers for ALL VLITypes  
// ALL IMPORTANT THINGS – determines extra VLI attributes!  
surfers = surfers params[ listOf(VLIType)],  
vliExtraAttributes = getVLIExtraAttributes params[VLIType],  
  
// -----
```

```

// KNOWLEDGE
// -----
knowledge,

// -----
// Value List
// -----
// the value list with ALL the data they hold
valueList listOf(VLI),

// -----
// PROCESS
/ -----
process params[informationArchitecture],

// -----
// CYCLES
/ -----
cycleTypes = getCycleTypes params[process],

// -----
// SCHEDULE includes Initial Value List
// -----
schedule = getSchedule params[process],

// -----
// METRICS
/ -----
metricsForCycles = getMetricsForCycles params[cycleTypes],

// -----
// REPORTS
/ -----
reportsForCycles = getReportsForCycles params[cycleTypes],

// -----
// CALCULATIONS
/ -----
calculationForCycles = getCalculationsForCycles params[cycleTypes],

// -----
// SCALING

```



```
// -----
structure,
structuralPattern = getStructuralPattern params[structure],
collaborationMode = getCollaborationMode params[structure],
deliveryMode = getDeliveryMode params[structure],
deliveryTarget = getDeliveryTarget params[structure]
contractType = getContractType params[structure],

// -----
// Impediment, Issues and Debts
// -----
impediments listOf(impediment),
issues listOf(issue),
debts listOf(debt)

} // END of ES
```

Template Parameters

```
template
contains{
name,
instanceEnterpriseScrumTemplate params[name],
canvasTemplate params[name],
initialValueListProcessTemplate params[name],
initialValueListTemplate params[name],
cycleTemplate params[name]
}
```

```
templates listOf(template)
```

Enterprise Scrum Team Parameters

```
ESTeam
contains{
name,

// business owner
busiessOwnerType=values<single, many, distributed>,
businessOwner params[name],
busiessOwners listOf(businesOwner),

// coach
coachType = values<single, many, distributed>,
coach params[name],
coaches listOf(coach),
```

```

// team
team,
skillMatrix params[members, sillLevel, skills],

// architect
architect params[name],
architects listOf(architect),

// stakeholders
stakeHolders listOf(stakeholder),

// parent – just for delegation or centralization
teamRole = values< contributor, delegator, centralControl>,

relationship
contains{
ESTeam,
Type params[values<subsumption, collaboration, delegation, centralization>],
Domain params[ domain, values<work, knowledge>]
Direction params[values<weDependON, theydependON, mutualDependence]
}

relationships listOf(relationship),

} // END of ESTeam


// team
skill
contains{
name
description
level = values<1-10>
}

member
contains{
name,
skills listOf(skill),
emotional maturity,
participation = values<full-time, part-time>,

// ES - CEC Model – competence, engagement, collaboration
// relative to skills needed

```

```

competence = values<low, medium, high, virtuoso>,
engagement = values<distant, involved, participating, engaged, superEngaged>,
collaboration = values<independent, friction diplomatic, jelled>
}

training
contains{
name
description
level
}

team
contains{
// team membership and overall
teamFormationStage = values<forming, storming, norming, performing>,
demonstratedPerformance,
members contains{member},
timeTogether,

// skills, technical or otherwise
// -> add ALL from ALL members
teamCurrentTechSkills params[team]
teamDesiredTechSkills params[team]
technicalSkillsGAP params[team]

// collaboration skills
// -> add ALL from ALL members
teamCurrentSocialSkills params[team]
teamSocialGaps params[team]
teamDesiredSocialSkills params[team]

// engagement level
// -> add ALL from ALL members
teamCurrentEngagementLevel params[team]
teamEngagementGaps params[team]
teamDesiredEngagementLevel params[team]

// Trainings required from GAPS
trainingCoachingRequired params[team]

// Trainings performed
trainings listOf(training)
} // END of Team

```

Leadership Parameters

decision extends issue

```
contains{  
}
```

supportingAction extends issue

```
contains{  
}
```

leadership

```
contains{  
  leaders listOf(person), // it includes at least the Business Owner  
  issues listOf(issue),  
  pendingDecisions listOf(decision),  
  pendingSupportingActions listOf(supportingAction)  
}
```

Customer Parameters

customer

```
contains{  
  multiplicity = values<single, manyDefined, customerSegment>,  
  industry,  
  area,  
  segment,  
  customerBusinessOwner,  
  businessValue  
}
```

Business Value

businessValue

```
contains{  
  description,  
  whys=values<pains, gains, jobs, other>  
  optimizationVariables,  
  optimizationVariableToMetricMap,  
  optimizationGoals  
}
```

Information Architecture Parameters

informationArchitecture

```
contains{
```

```

// -----
// DELIVERABLES
// -----
//
// what is delivered by the ETeam?
//
deliverable,
deliverables listOf(derivable),

// -----
// VLI Types
// -----
//
// get VLITypes from deliverables 1-to-1
//
VLITypes = VLITypesFromDeliverables params[deliverables],

// what VLIType delivers most of the Business Value
//
principalVLIType = params[VLI-type1]

// -----
// SURFERS
// -----
// what are the aspects of a VLI?

VLIsurfers = surferfromVLIs params[VLIType],

// attributes from surfers
attributes = VLIAttributesToSurferMap params[VLISurfer],

// VLI unique surfers
//
VLIUniqueSurfers = uniqueSurfers params[VLIsurfers],

// common surfers among VLIs
//
commonSurfers params[ listOf(VLIType)]

//
// surfer dependencies – global and local
surferDepenenceMap params[ listOf(surfer)]

// -----

```

```

// INITIAL VALUE LIST
// -----
// initial Value List from surfers
initialValueListProcess = getInitialValueListProcess params[surfer]

// -----
// CANVAS
// -----
// get canvas from VLITypes
canvas = canvas params[VLIType]

// -----
// C8W Technique
// -----
deliverables = deliverablesFromC8W params[]
VLITypes = VLITypesFromC8W params[]
Surfers = surfersFromC8W params[VLIType]

// -----
// ARCHITECTURE MODEL AND MANAGEMENT
// -----
// What is the architecture for this domain?
//
architectureType

// What are the architecture models for this domain if any?
//
architectureModel
architecturalModels listOf(architectureModel)

// how are we doing architectural management?
//
architectureManagementType = values< self-org, agile architect, traditional architect>

}

VLITypes
VLIType
contains{
name,
description,
priority

```

**prioritizable,
orderingTechnique,**

**sizeable,
sizingUnits listOf(sizingUnit params[metric]),
sizeUncertainty,
size,**

**dependentVLIs,
DORStandard,
DODStandard,
DeliverableToVLIsMap,
requiredDODtime,**

cumulativeDOD

**surfers listOf(surfer)
}**

VLITypes listOf(VLIType)

Knowledge Parameters

**knowledge
contains{
techniques listOf(technique),
patterns listOf(pattern)
}**

**technique
contains{
name,
purpose,
processStep,
howItsUsed,
interfaces,
mappings,
references
}**

**pattern
contains{
name,
context,
problem,**

```

solution,
rationale,
resultingContext
}

```

Value List Item

valueListItem extends **VLIType**

```

contains{
DOR,
givenDODtime,
DOD,
planForVLI,
volunteers,
parentVLI,
dependsOnVLIs,
selectable,
actionMods = values<work, develop, refine, decide, test, experiment, sell,
research, answer, monitor, support>,
time = values<cycle, unselectable, scheduled, repeatable params[cycleName] >,
structure = values<singleton, collection params[order = values<unordered,
ordered, criteria, values<ascending, descending>], workflow listOf(orderedPair
contains{singleton, actor}) >,

```

// domain specific from VLITypes – DOMAIN SPECIFIC MODEL

```

Revenue,
Profit, Etc.

```

// admin

```

Created by,
Added in Date,
Budget,

```

```

}

```

valueList listOf(valueListItem),

Process

```

// -----

```

// PROCESS

```

/ -----

```

```

process
contains{

```



```

// -----
// Initial Value List sequence
// -----
// NOTE:
// we typically hold different meetings to fill different surfers
// ONE or more per event
// in Enterprise Scrum you decide how to structure
// the filling up of the canvas values
eventType
contains{
name,
agenda params[ listOfActivities],
typicalDuration params[hours],
techniquesUsed params[techniques]
},
eventTypes listOf(eventType),

event
contains{
eventType,
timeDate,
duration,
attendees params[ESteam]
},

sequenceStep
contains{
surfers,
event
},

InitialValueList
contains{
initialValueListSequence listOf(sequenceStep),

initialValueListSequence = initialValueListSequenceFromIA params[surfers],

//-----
// The events to build the Initial Value List
//
// This is valuable information even if you don't use an Initial Value List
// and just Cycle through to begin with
//
// Because then, these activities then can added as VLIs for the Cycles
//
//-----

```

```

initialValueListEvents listOf(events),
initialValueListevents = initialValueListEvents params[initialValueListSequence],
}

```

```

// -----
// CYCLES
// -----
cycles
contains{
  listOfCycles params[informationArchitecture],

// Cycle structure
cycleSequence listOf(sequenceStep),

cycleSequence = cycleSequenceFromIA params[surfers],

//
// The events to build a Cycle
//
cycleEvents contains{cycleSequence},

// Cycle instances with ALL the data they hold
cycleInstances listOf(cycleEvents),
}

// -----
// Schedule
// -----
schedule params[events],

schedule = schedule listOf(initialValueListEvents, cycleInstances),
} // end of process

```

Cycle Parameters

```
listOfCycles listOf(Cycle)
```

```

Cycle
contains{
  name,
  level,
  length,
  qualitative,

```

```

// subsumption
surfersForCycle listOf(surfers),

/// PCRI
recommendedPCRItimeFrames,
planningType = values<WBS, other>,
collaborationType = values<independent, collaborative, other>,
reviewType params[ surfers, values<cumulative, incrementOnly >]
improveType params[ surfers, values<wellImprove, other>],

// Cycle DOR/DOD
cycleDOR,
cycleDOD,
passedCumulativeDOD

// metrics, calculations and charts
metrics listOf( metric params[name, description, unit, frequency]),
calculations listOf( calculation params[description, formula] )
charts listOf( metricChart contains{name, description, metric, type} ),

// reports
scrumBoard,

// team dependencies for Cycle
dependencyMatrix,

// value
valueIncrementName,
usableValue
}

```

Cycle Instance Parameters

```

cycleInstances listOf(cycleInstance)

cycleInstance extends Cycle
contains{
icID,
initialValueListForCycle listOf(valueListItem),
doneValueListForCycle listOf(valueListItem),
retrospective params[ listOf(wells), listOf(improvements)],
measurements listOf(measurement),
chartInstances listOf(chartInstance),
calculationsInstances listOf(calculationInstance),

```

```

scrumBoardInstance,
dependencyMatrixInstance
}

```

Scaling Parameters

```

scaling
contains{
// -----
// Scaling
// -----
// connection mode – if more than ONE team
//
// the ESTeams hold the relationships
//
scaling
contains{
contributors listOf(EnterpriseScrum),
scalingStructuralPattern = values< centralized-B+S, centralized-B, collab-virtual-B,
chameleon>

connectionMode = values<subsumption, collaboration, delegation, centralized>,
}

// -----
// Contract
// -----
// contract type
contractType = values<fixed date, fixed price, time and materials, cost-plus,
ongoing>,

// -----
// Delivery
// -----
// delivery mode
deliveryMode = values<continuous deployment, continuous delivery, Cycle
delivery, Release delivery>,

// -----
// Delivery target
// -----
// delivery target
// -----
deliveryTarget = values<large project, program, portfolio, Enterprise
Architecture>

```

```

}

// -----
// ISSUES and DEBTS
// -----
issue
contains{
type =
values<
technical,
refinement,
decision,
research,
monitoring,
change,
cultural maturity,
motivation,
learning,
OTHER
>,

description,
resolution,
loggedBy,
loggedDateTime,
resolvedBy, // One of the leaders listed
resolvedDateTime
}

debt extends issue
contains{

}

```