

Enterprise Scrum

Definition:
Business Agility for the 21st Century

Authored, Developed and Sustained

By

Mike Beedle

Enterprise Scrum Inc.



May 18, 2017

Release 3.3



CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

- a. **"Adaptation"** means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.
- b. **"Collection"** means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined above) for the purposes of this License.
- c. **"Distribute"** means to make available to the public the original and copies of the Work through sale or other transfer of ownership.
- d. **"Licensor"** means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.
- e. **"Original Author"** means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or

legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.

- f. **"Work"** means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address, sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.
- g. **"You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.
- h. **"Publicly Perform"** means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.
- i. **"Reproduce"** means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights. Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

- a. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections; and,
- b. to Distribute and Publicly Perform the Work including as incorporated in Collections.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats, but otherwise you have no rights to make Adaptations. Subject to 8(f), all rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Section 4(d).

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

- a. You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(c), as requested.
- b. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.
- c. If You Distribute, or Publicly Perform the Work or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work. The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Collection, at a minimum such credit will appear, if a credit for all contributing authors of Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.
- d. For the avoidance of doubt:
 - i. **Non-waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;
 - ii. **Waivable Compulsory License Schemes.** In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License if Your exercise of such rights is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(b) and otherwise waives the right to collect royalties through any statutory or compulsory licensing scheme; and,
 - iii. **Voluntary License Schemes.** The Licensor reserves the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You

of the rights granted under this License that is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(b).

- e. Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

- a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
- b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

- a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
- b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
- c. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
- d. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the

Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

- e. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

Creative Commons Notice

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, Creative Commons does not authorize the use by either party of the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time. For the avoidance of doubt, this trademark restriction does not form part of this License.

Creative Commons may be contacted at <https://creativecommons.org/>.

Acknowledgements

First and foremost, I would like to thank Jeff Sutherland for inventing what we now know as modern Scrum from experience and knowledge gained from circa 1983 through the fall of 1993. Without his work and inspiration Enterprise Scrum will not be possible, mixing Complexity Science, the subsumption architecture, org patterns and Nonaka and Takeuchi's ideas was a stroke of genius.

Secondly, I want to thank all of those of you – customers, practitioners, students, conference attendees, fellow Scrum trainers and coaches, and Certified Enterprise Scrum Trainers and Coaches:

- 1) For all the interesting conversations and shared insights
- 2) For all the unconditional support over so many years
- 3) For all the feedback on presentations, class materials and coaching
- 4) For all the questions about problems on the field over the years, and
- 5) For the opportunity to help you in a myriad ways to implement Enterprise Scrum in the trenches

Thirdly, I want to specially thank all of our other CESTs (Certified Enterprise Scrum Trainers and Coaches) for their very special support and feedback during or around the Enterprise Scrum classes: Jon Jorgensen, Juan Saldana, Brett Palmer, Javier Haro, Michael Hermann, Richard Cheng, John McFadyen, Abid Quereshi, Simon Roberts, Karim Harbott, Catherine Louis, Daniel Mezick, James Schiel, Jaya Pattaswamy, Mark Sheffield, Ulises Salvador Aguila Ledezma, Hriday Keni, Marina Simonova, Savannah Rayat, Lowell Lindstrom, Perez Emilio, John Miller, Stacia Heimgartner, Rick Waters, Bob Sarni, Alex Mezinsky, Jay Goldstein, Pierre Neis, and Michael Leber. Without their help and support, we would not be where we are today.

I also want to specially thank all the people that helped me review the Enterprise Scrum documents and ideas and which provided valuable feedback, or supported the Enterprise Scrum ideas: Peter Stevens, Luis Mulato, Johnny Ordonez, Kris Mir, German Pellejero, Deniz Ozdemir, Ram Srinivasan, Steve Spearman, Doug Shimp, Peter Green, Gene Gendel, Martine Devos, Martin Alaimo, and Pablo Tortorella.

Lastly, I also want to acknowledge the deep influence that some authors had on me; without their ideas and opinions, Enterprise Scrum would not be the same: Steve Denning, Peter Drucker, Alex Osterwalder, Kim Chan, Renee Mauborgne, Owen Harrison, Daniel Mezick, Salim Ismail, Clayton Christensen, Adrian Slywotzky, Gary Hamel, Michael Hammer, Frederic Laloux, Jeremy Hope, Robin Fraser, Eric Ries, Steve Blank, Jurgen Appelo, Geoffrey Moore, Christopher Alexander, Henrik Kniberg, Alistair Cockburn, James Coplien and Jim McCarthy.

To all the countries, skies, waters and cities that hosted my writings, in their restaurants, airports, cafes, boats, coffee shops, yachts and bars – THANK YOU:

Madrid, Wiechlice, Międzyzdroje, Lake Forest, Lviv, Munich, Frankfurt, Warsaw, Krakow, Barcelona, Brussels, Copenhagen, Elmhurst, Oslo, Chicago, Tisvildeleje, Vienna, Lansing, Buenos Aires, Nowe Miasteczko, Santiago, Atlanta, Cochabamba, Boston, Lima, Rio de Janeiro, Philadelphia, Prague, Washington DC, Montevideo, Fort Lauderdale, Miami, Los Cabos, London, Oak Brook, Toronto, New York, San Francisco, Phoenix, Shanghai, Seattle, Schaumburg, Key West, Nassau, Dallas, Mallorca, San Diego, Kansas City, Guadalajara, Paris, Tokyo, Nowa Jablona, Glogow, Monterrey, Berlin, Mexico City, Park Ridge and Chicago – where this document was finished to this version.

I hope Enterprise Scrum is useful to all present and future users seeking Business Agility and Agile Management so that they can use Scrum for:

- 1) in a Generic way, 2) in a scaled way, and with 3) more synergistic techniques business or otherwise.

This work would have never been completed without the support of my wife and my family. Thank you Barbara, you really deserve most of the credits!

Change Log

3.3 – Released to public on 5/18/2017.

- Added a better Enterprise Scrum purpose.
- Added a better community-driven Business Agility Definition.
- Included Business Agility patterns.
- Added better explanation of Agile Management.
- Expanded and included a better example of Business Agility.
- Expanded and included a better example of Software Scaling.
- Completed documenting Enterprise Scrum parameters.
- Added canvases for many other domains.
- Improved Enterprise Scrum history.
- Included OSA working with ES.

3.2 – 1/20/2017 Added and reorganized configuration parameters, and integrated them with the example chapters.

3.1 – 1/20/2017 Added the Business Agility definition. Edited the Business Agility chapter.

3.0 – 1/7/2017 – Added many conceptual sections and technical parameters to explain co-evolutionary and subsumption nature of Enterprise Scrum: surfer (important aspects), Cycles, waves of Cycles, etc. For example, there are now technical parameters on surfers, canvases, domain mappings into canvases and surfers and cycles. Added VLI structure, time, and action modifiers to VLIs officially. Added many cultural notes and parameters on teams. Added many other debts beyond “technical debt”. Added subsumption parameters to Cycles.

2.1 – Changed the introduction. Moved the patterns to the Enterprise Scrum Overview. Made grammar and spelling corrections.

2.0 – 11/09/2014 – Added connections to the Connected Company, the Creative Economy, Cognitive Science, Dave Snowden’s work, and underlined the importance or culture in Enterprise Scrum.

1.03 - 10/25/2014 - Added Adaptation, Improvement and Growth as part of the Piecemeal Growth pattern; which provides the fundamental basis to better understand how to achieve better business models, team and organization improvement, business transformation and product or service team scaling. Renamed all the high-level patterns. Added images. Added and reorganized the references. Highlighted empirical process management throughout and underlined it’s connections and relevance to piecemeal growth and the subsumption architecture.

1.02 - 8/4/2014 - Added more business references and linked Enterprise Scrum to them, such as Running Lean, Business Model Generation, Beyond Budgeting, Smart Tribes, Tribal Leadership, etc. Added Architecture-related parameters. Abstract 4 high-level Enterprise Scrum patterns: Scrum Team, True Business Value, Plan by Measurement, Adapt and/or Grow through Improvement Cycles.

1.01 - 4/09/2015 - Added more scaling references. Cross Functional Skill Matrix.

1.00 - 3/29/2014 - Base Definition. Enterprise Scrum naming conventions.

Business-Oriented: Multiple nested Improvement Cycles, insertion of techniques, calculations: budgets, schedules, fixed-date, risk-management, other cumulative metrics.

Genericity: business value, DOR, DOD, VLI types, Metrics and charts, generalized velocities, etc.

Scaling: meeting options, rules and participants, parent, contributors, dependOn, dependsOnUs, value list parent, global and local velocity, cumulative metrics, etc.

Contents

.....	2
<i>License</i>	2
Creative Commons Notice	6
1. What is Enterprise Scrum?	15
Executive Summary	15
Business Agility	16
Innovation Revolution	17
Faster Change	18
Growing Percentage of Revenue and Profit from NEW products and services.....	19
Maximizing Stock Value through Disruptive Innovation	20
The consequences of Faster Change.....	20
NPE == New Product Efficiency	21
Agile Management and Business Agility	22
Business Agility	22
Agile Management	23
Disruptive Waves	23
Higher Industry Uncertainty	24
Change Source by Process Type	25
Defined Processes	25
Development Processes	26
Subsumption	28
The Scrum Framework high-level description	29
Enterprise Scrum Framework high-level description	29
Brief history of Agile and Scrum	32
Agile	32
Scrum	33
Patterns	35
Agile Transformation.....	37
Additional Guidance.....	38
2. Enterprise Scrum Overview.....	40
Enterprise Scrum Features and Goals	40

Enterprise Scrum Concepts and Names	42
Enterprise Scrum Instances.....	43
Enterprise Scrum Templates.....	44
Agile Mindset and Agile Culture.....	45
Enterprise Scrum Team.....	46
Business Owner	47
Enterprise Scrum Coach	48
Team	49
Customer	49
Business Value	50
Knowledge	50
Patterns	50
Techniques.....	51
Information Architecture.....	51
Canvas	51
Value List	53
VLIs	53
VLI-DOR	54
VLI-DOD	54
Enterprise Scrum Process	54
Vision	55
Initial Value List	56
Cycles and Cycle Structure – PCRI	56
Metrics	57
Calculations	58
Expected Behavior after multiple Cycles	59
Iterative Incremental Growth Principle	60
Recursive Cycles.....	63
Wave Principle	64
Reports	65
ES - ScrumBoard	65
Charts.....	66
Debts	66
Enterprise Scrum Parameters	67
3. Business Agility	68

What is Business Agility?	68
Enterprise Scrum Organizational Taxonomy	69
Business Agility Patterns	71
All-at-once Management.....	72
Preserved Agile Growth	73
Customer Segment Team - Mirror	75
Supporting Processes	77
Enterprise Scrum Unified Portfolio and Metrics	78
Accelerated Innovation	82
4. Enterprise Scrum configuration	83
Enterprise Scrum parameters.....	83
Template parameters	89
Enterprise Scrum Team Parameters	91
Customer parameters.....	95
Business Value	96
Information Architecture parameters	98
VLITypes	105
Knowledge Parameters	110
Value List Item.....	112
Process Parameters	114
Cycle Parameters	116
Cycle Instance Parameters	121
Structure Parameters	123
Feature Teams - Scrum Team per Business Area	125
Add a parent Scrum Team to represent Big Picture	125
Delegation/Centralized.....	126
Collaboration	126
Subsumption	126
5. Enterprise Scrum Example – Startup or Innovative Business Unit	127
NAME.....	127
ADVANTAGES	127
Motivation	127
Examples	128
Framework - techniques, patterns.....	128
Enterprise Scrum Industry Impact.....	128
CONFIG.....	131
Enterprise Scrum Team	131
Enterprise Scrum Team: Business Owner	131
Enterprise Scrum Team: Coaches	131

Enterprise Scrum Team: Team	131
Customer	132
Customer: Business Value	132
Information Architecture: Canvas	133
Information Architecture: Principal VLI	133
Information Architecture: DORStandard	133
Information Architecture: DODStandard	133
Process: Cycles.....	134
Process: Optional Activities.....	134
INSTANCE.....	134
Vision	135
Initial Value List	135
Define	135
Prioritize.....	136
Size.....	136
CYCLES	137
Cycle.....	138
Planning	138
Collaboration	138
Review	138
Improve	139
Metrics.....	139
Reports.....	140
Plans and Forecasts	140
6. Conclusions – a call to GREATNESS!	141
Innovation Revolution	141
Invitation – a call to Greatness	141
References	143
Appendix A. Other Canvases	156

1. What is Enterprise Scrum?

Executive Summary

“The purpose of business is to create and keep a customer.”

– **Peter F. Drucker**

We need Business Agility and Agile Management in the 21st Century because our well-trusted until recently 20th century management techniques – strategic planning, CXX driven, Sloan managed, department-based, phase-based processes, power/approval hierarchies, are not working all that well any more. In the 21st Century we still need to create and serve customers, but now in a world of exponentially increasing rate of change [KotterForbes]:

- Product and service lifecycles are getting shorter and shorter.
- Half of the companies from the Fortune 500 are gone since the year 2000 (Pierre Nanterme, CEO of Accenture)
- 75% of the S&P is scheduled to change in the next 10 years
- Average company lifetime was ~60 years in 1970, now it's about 30 years [BCG-DieAnotherDay]
- Business growth cycle is approaching 7 years, so re-invention has to be started every 3.5 years [Zhexembayeva]
- Unicorn valuations (1 billion or more) are achieved faster (~1.5 year in 2017 instead of 7 years around year 2000)
- More Unicorns are created every year
- The “percentage of Profit and Revenue from NEW products and services in the last 5 years” [NonakaTakeuchi] -- which we will call from now on generically **PRIG5** (Profit/Revenue Innovation Growth in 5 years), keeps growing every year. It was around 25% in 1978, but now is 50% on the average.
- 93% of US multinationals are undergoing a company-wide transformation [KPMG]



Figure 1. Half of the companies in the Fortune 500 are gone since the year 2000

And as a consequence:

Wealth has never been created, transformed or destroyed at a faster pace than today.

Faster change creates opportunity for those companies with higher Business Agility, that can continue to delight customers with fast efficient innovation and adapt quickly despite rapid changing markets, technologies, or social changes. But for those that can't innovate or adapt fast enough, or accurately, the future looks dismal. To compete in the 21st Century, a company has to adapt and innovate faster than ever. This means that companies must do more **disruptive innovation**, rather than sustaining innovation – improving old products, or seek more operational efficiencies – reducing costs.

This environment defines what I have been calling the **Innovation Revolution** – an era in business driven by competition through innovation and faster more efficient adaptation of existing markets, technology and social changes, that has violently crept on us in the last 30 years or so. Simply said, we live in an innovation warzone where companies outcompete each other through innovation for survival.

Business Agility

We therefore define Business Agility as:

"Business Agility is the ability to adapt quickly and effectively to all forms of change to deliver maximum value and customer experience."

Editor: Mike Beedle

Contributors (listed alphabetically):

Brad Appleton	Dan Leeds
Alexandria Contes	Lowell Lindstrom
Guillermo Gomez Hdez	Steven Mak
Jim Hannon	Madan Naidu
Scott Hereford	Pierre Neis
Michael Herman	Carlos Ruiz Vazquez
Jon Jorgensen	David Sabine
Johanes Klose Andersen	Jay Virzi
Greg Kramer	Ilja Vishnevski
	Rick Waters

Where adapt is "collaborate, effect, experiment, innovate, lead, learn, manage, and organize itself"; and change is "customer, market, regulation, social, technology, and changes". Also, we prefer to have purposeful and sustainable Business Agility.

As such, Business Agility includes:

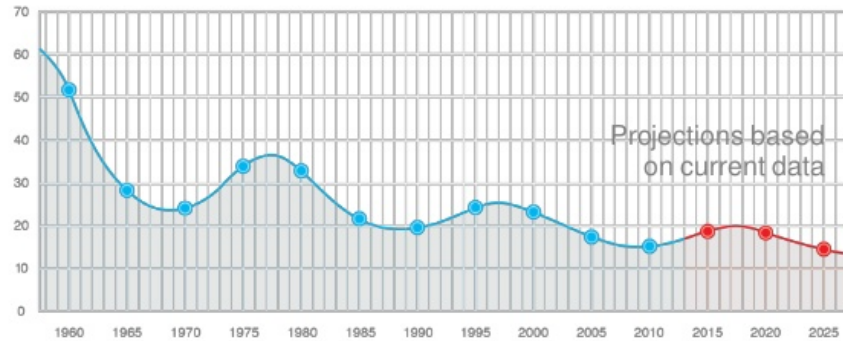
- Agile Culture/Mindset – a new set of agile values and attitudes about what is important at work
- Agile Leadership – a new way to lead organizations into the future based on agile thinking
- Business Agility Patterns – a new set of patterns that lead to business agility
- Agile Management – a new way to manage based on agile principles
- Modern Business Techniques – advanced business and technical techniques that compliment agile management: Lean Startup, Blue Ocean Strategy, Beyond Budgeting, Zone to Win, Exponential Organizations, Design Thinking, Business Model Generation, Profit Patterns, Management 3.0, Cynefin, etc.
- NEW Technology – AgriTech, FinTech, InsureTech, MediaTech, MarketingTech, EnergyTech, MaterialTech, Genomics/Proteinomics, Biotech, PharmaTech, HealthTech, new technologies
- NEW Business Models
- NEW Organizational Designs

Innovation Revolution

The Innovation Revolution, as I define it, is different but related to The Creative Economy [CreativeEconomy], which is increased economic activity due to creative work, like in music, movies, books, and inventions, etc.; or even the Third Wave, the internet of everything as Steve Case defines it [ThirdWave]. Instead, the Innovation Revolution means that every company in every industry - not just those involved in creative work, ***needs to innovate, adapt and reinvent itself faster and more efficiently to stay thrive and survive.***

I have been talking and lecturing on this Innovation Revolution since 2009 but others are having similar thoughts. For example, the WEF (World Economic Forum) recently announced that we are entering what it calls the *The 4th Industrial Revolution* [4thIndustrialRevolution].

At the Current Churn Rate, 75% of the S&P Will Be Replaced by 2027



Average company lifespan on S&P 500 Index (in years)
Year (each data point represents a rolling scale 7-year scale average of average lifespan)

Data: INNOSIGHT, Richard N. Foster, Standard & Poor's

Rally Software

©2014

Figure 2. S&P lifetime projections (source INNOSIGHT)

Faster Change

John Kotter, which wrote the book *Leading Change* [LeadingChange], has concluded that the rate of change in business is not only growing, it is growing exponentially [KotterForbes]. Evidence for this argument, he says, is for example, *shorter product and service life cycles* and therefore the need for faster and effective more agile management, business agility and agile innovation. Traditional management methods, such as Sloan management, process management, or product or service-oriented management based on *phased long-term defined processes* can no longer be used effectively because these management methods assume that there is little or no change. Instead, when we have open systems – more information coming in at all times, we need a different type of management – agile management.

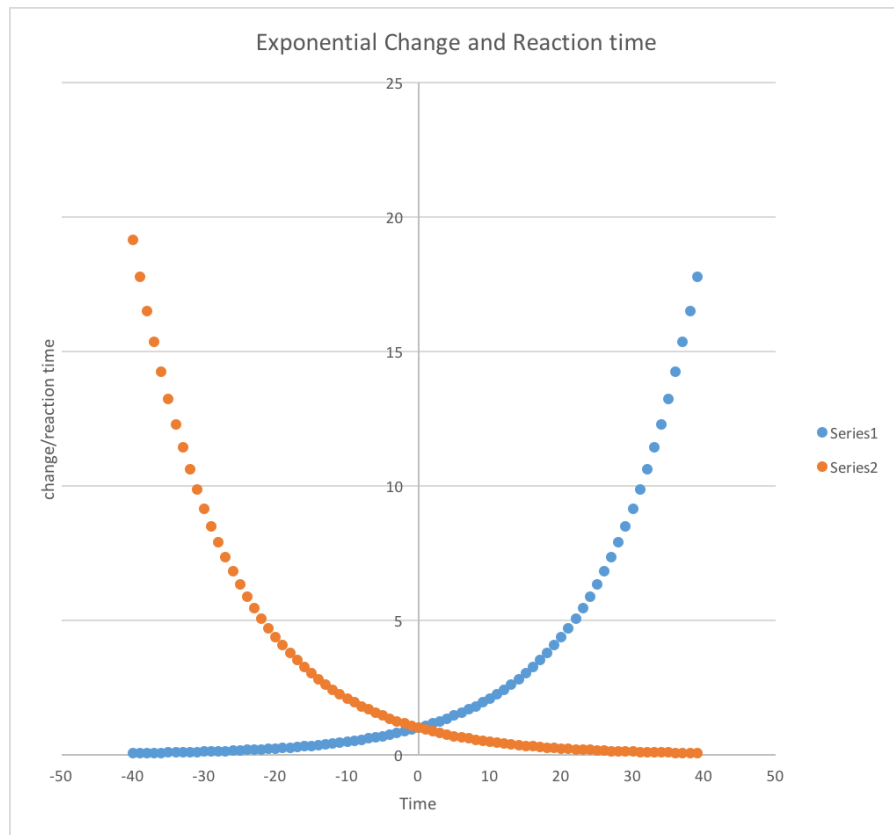


Figure 3 Exponential Change means Exponential smaller reaction time

Growing Percentage of Revenue and Profit from NEW products and services

When Nonaka and Takeuchi wrote their paper first describing Scrum, *The NEW new Product Development Game* [NonakaTakeuchi], the percentage of profit and revenue for products invented in the last 5 years was measured in 1978 to be 28% on the average, with a prediction of 33% in 1986. But these numbers keep getting higher every year. In *Best Practices in Product Innovation: What Distinguishes Top Performers* in 2011, Cooper, Edgett and Kleinschmidt [CEK], point to an average of 45% of profit and revenue come from products and services invented in the last 5 years (projected from their numbers), and of 70% of profit and revenue come from products and services invented in the last 5 years for the top 20% performers (projected from their numbers).

This means that if your company stops innovating today, it will lose on the average 45% of its profit and revenue, and 70% in more competitive industries.

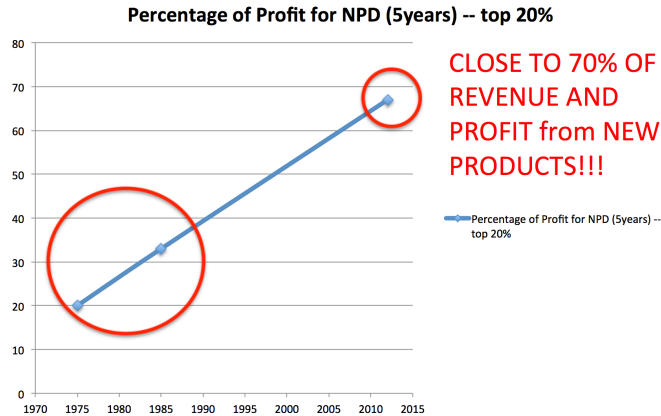


Figure 4. Revenue and Profit for top 20% competitors from NEW products in 5 years

Maximizing Stock Value through Disruptive Innovation

Everyone's goal is to make a company profitable, but because of the higher rate of change and the **PRIG5 ratio** described above, companies cannot maximize stock value today the way they used to maximize stock value in the past: by cutting cost or selling more of the same stuff. Instead, they need rebalance and rethink their portfolios, and generate more revenue from **NEW disruptive innovation**, without assuming that sustaining innovation or operational efficiencies alone will make them successful [NonakaTakeuchi], [CEK], [Christenssen], [Kanter], [Kaplan].

The consequences of Faster Change

The consequences of the increasing exponential rate of change on a business in a competitive environment are very many:

- **Process More Information and Decide Faster.** As company managers, we have much more changing market information (competitors, suppliers, technology, etc.) to process in a shorter time. So important decisions must be taken much faster. Since the amount of information and change are growing exponentially; then we have an exponentially shorter reaction time to adapt, decide and learn. We need to live with the fact, that we now live in a more uncertain world, where may have to "change our minds" and "make more corrections on the go".
- **Innovate and Delight Faster.** Despite the rapidly changing environment, we need to develop a good coherent vision of what the customer wants or needs and bring NEW products and services that can delight them faster to market, and adapt as the customer changes.
- **Fast and Effective Adaption to Change.** Despite the higher amount of changing information, we must adapt quickly and effectively making the right decision, to any important changes to compete or survive, regardless of the

source of those changes: market, competition, technology, regulatory or internal changes.

- **Shorter Plans and Predictions.** With the larger amount of information and change we also have exponentially shorter predictable time horizons. That doesn't mean we should plan long term, we can and should plan long-term. But what we cannot do, is stop making adjustments to our plans in any time horizon.
- **Sharper Competition.** A rapidly changing business environment leads to stronger sharper competition – products and services either make it or break it faster with an ever-decreasing product and service lifecycles. In this environment, you are already dead out of the competition if you can't produce NEW products and services; but to make it even more challenging, innovation must be on target and produced with higher efficiency.
- **Process Feedback Faster** We must process all feedback faster: Customer Feedback, Market Feedback and Technology changes.
- **Accurate Timeframes** We need to have more certainty in development timeframes, because time-to-market is so ever so important, but compromising by varying scope since everything tends to change more: markets, competitors' offerings, customer needs, technology, etc.
- **Better Portfolio Management** As competition stiffens, we need better and faster managed portfolios of business units, customer segments, and products and services, so that we can cut down anything that is not as profitable, obsolete or non-competitive.
- **Experiment More and Succeed or Fail Fast** Another one of the consequences of faster change is that there is less certainty in being successful. In fact, the times where we so-proudly boasted "we never failed", are gone. In a less certain environment, our products and services may need to either succeed or fail faster – because there is little or no pay off with either late or unneeded products or services. We simply can't afford to get too stubborn on a failing business idea, product, or service.

NPE == New Product Efficiency

There is also a wide gap among players within an industry. Arthur D. Little which defines NPE (New Product Efficiency) as New Products Sales divided by R&D investment, in their *Innovation Excellence Study* [ArthurDLittle], reveals that the best innovators are 12X more productive. **This means "efficient innovators" get twelve times more sales for the same R&D investment.**

But because of a larger PRIG5, as we saw above, **NPE is critical to the company success**, because the world has turned into an innovation competition within every industry and market segment, and companies not only need to innovate fast – they also need to innovate efficiently.

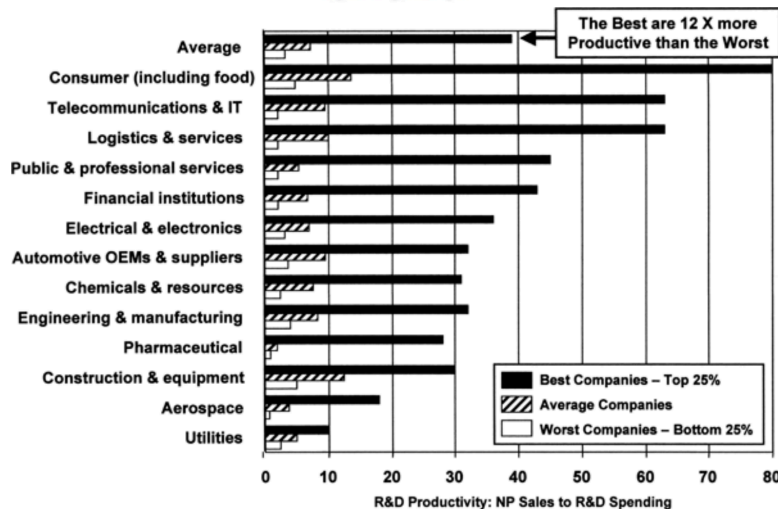


Figure 1.—NPD productivity varies greatly among companies, with huge differences between the best and worst Companies in each industry. Source: A. D. Little Innovation Excellence Study (5). Source: A.D. Little Innovation Excellence Study (5).

Figure 5. New Product Efficiency by Industry

What are the consequences of this gradient in NPE?

- **Larger NPE** can mean a combination of better sales, faster innovation or less expensive innovation. But either way, winners profit more with lesser investment.
- **Larger NPE can be achieved anywhere in the business lifecycle:** strategy, marketing, product and service development, technology, sales, or customer feedback. But also, any “stage” can be the bottleneck; therefore, some companies have opted to remove bottlenecks by doing ***all-at-once innovation and business management*** i.e. deploying testable business models [BusinessModelCanvas], instead of following a phase-based approach. We will see that this is one of the key patterns of Enterprise Scrum, Business Agility and Agile Management.
- **Winners often get larger market share faster** market share is not necessarily a measure of profitability but is still a measure of competitive advantage [ProfitZone].
- Because everything changes faster, *we can think of a company not being just managed, but of a company being developed.* This is also a key concept in Enterprise Scrum – the more change, the more activities look like development activities [Blank].

Agile Management and Business Agility

Business Agility

Given our conclusions from the 1) exponential rate of business change, 2) the PRIG5, and 3) the New Product Efficiency sections, one clear picture emerges:

To successfully lead our companies in the 21st century, we need Business Agility.

And this is what Enterprise Scrum provides – a way to agilize everything within a company:

The focus of Enterprise Scrum is to give a company, or any part of it Business Agility.

Enterprise Scrum means Scrum applied to any part or the enterprise as a whole, so it means continuously reinventing, improving and adapting the company and everything it does. There is a deeply rooted explanation of this that we will examine in detail in the next sections: Enterprise Scrum is subsumption-based empirical business management at any level, for any structure, which is the foundation of Business Agility and Agile Management.

Some companies have already started to practice and write about this new style of management. For example, Salesforce.com uses a style of management that eerily resembles Enterprise Scrum. Also, Steve Deming – a former Scrum Alliance board member, wrote the book *Radical Management* [RadicalManagement], clearly pointing into this direction.

Agile Management

Agile Management is a style of management where deliver something of value to the customer e.g. a product, service or business model; and then when we get feedback from the customers, we can *change anything that is important to improve the value and customer experience to them*.

For example, if we are delivering business model to the customers involving a mobile app and a service, when we get the feedback from the customer we can change:

- The business model – business model pattern
- Marketing – messages, campaigns, channels, relationships, etc.
- Sales – closing, payment options, terms, guarantees, contracts, etc.
- Products or Services – features, quality, levels of service, processes, etc.
- Order Fulfillment – distribution, delivery, packaging, etc.
- Customer Service – maintenance, support, availability, etc.
- Etc.

Disruptive Waves

The world is now under the effect of very many disruptive waves approaching every industry and every sector in every industry:

- AgriTech,
- Biotech,
- EnergyTech,
- FinTech,

- Genomics/Proteinomics,
- HealthTech,
- InsureTech,
- MarketingTech,
- MaterialTech,
- MediaTech,
- PharmaTech,
- Retail
- Tech (IoT, VR, AR, AI, Robots, 3D printing, drones, IT, etc.)
- TelecomTech

Technology alone is not the only change, there are also major waves of change from social changes, regulatory, and market changes; although, yes, they are all interrelated.

In fact, we can see that over the last 100 years, the adoption rates of technology continue to increase to record shorter times.

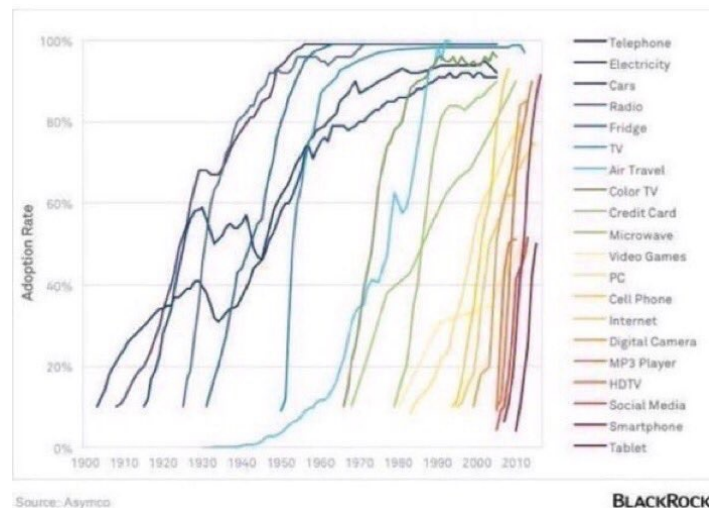


Figure 6. Adoption Curves

Higher Industry Uncertainly

The uncertainty may vary per industry and by business process, but it is increasing to new highs every year. In fact, we can see from the chart below that some of the revenue volatility is very high across the board. This is understandable from the perspective of the existing and upcoming disruptive waves in the world today – and it's only going to get worse. Although there are other factors depending on the industry, a large percentage this industry volatility is correlated to “inventing the right new products and services for an ever-more demanding customer”.

INDUSTRIES RANKED BY LEVEL OF UNCERTAINTY

INDUSTRY	R&D OF SALES	REVENUE VOLATILITY	FIRM TURNOVER*
1 Medical equipment	8.2%	90.7%	13.1%
2 Computers	5.8	98.8	12.0
3 Computer software	9.8	69.9	14.4
4 Pharmaceutical products	17.4	63.3	12.7
5 Measuring & control equip.	9.3	97.0	8.8
6 Machinery	3.2	100.5	9.3
7 Agriculture	10.8	123.3	4.9
8 Electronic equipment	5.2	61.5	10.5
9 Chemicals	3.0	71.2	9.2
10 Electrical equipment	9.8	35.0	9.2
24 Business services	3.2	46.2	6.5
40 Business supplies	1.4	34.8	5.0
41 Shipping containers	0.5	65.1	4.9
42 Real estate	1.3	57.6	3.0
43 Beer & liquor	2.3	12.8	3.7
44 Personal services	0.3	59.7	4.4
45 Tobacco products	1.0	20.3	5.2
46 Insurance	2.2	30.4	0.9
47 Wholesale	0.1	14.1	6.3
48 Utilities	0.2	45.6	0.2
49 Precious metals	0.1	40.7	1.5

*PERCENTAGE OF (ENTRANCE + EXITS)/TOTAL FIRMS IN THE INDUSTRY PER YEAR. SOURCE COMPUSTAT 2013

HBR.ORG

Figure 7. Industries by uncertainty

Change Source by Process Type

But where is the most change and uncertainty coming from in an enterprise? Change is everywhere, and it can come from very many different directions, but where does the most critical change come from? The systems thinking answer or process answer to this question is: from business processes that are **open information systems**. Understanding a system from its systems thinking perspective is hard, but add to that **change**, and things can get much more complex.

All of the processes of an enterprise can be broadly divided into: 1) production or defined processes – which are closed or almost closed information systems; or 2) development or empirical processes – which are open information systems.

Defined Processes

Defined processes, which are almost closed information systems (not much information changing while the process executes), are processes that can be changed at a slower pace; while development-like processes, which are open information systems, need faster, higher-frequency, often larger and more pointed changes in time. Defined processes execute with most of the steps known in advance, while development-like or empirical processes execute opportunistically first analyzing context and then taking

action through transparency, inspection and adaptation [ProcessControlTheory].

Defined processes are processes like production, manufacturing, accounting, payroll, or traditional customer service. A manufacturing process can be changed or improved but the steps it executes in that version of the process are known in advance. Production-like processes should be optimized through **Lean Production** techniques – Lean Principles and Lean Techniques applied to production-like processes [Ohno], [Wommanck1], [Wommanck2], [Wommack3]. Most Global 5000 Companies have been optimizing their production-like processes using these techniques for nearly 30 years outside Japan, and longer in Japan.

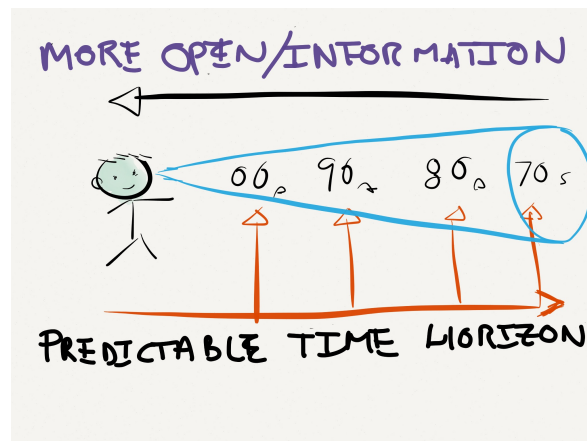


Figure 8. Development-Like processes are Open Information Systems

Development Processes

Development processes on the other hand, are processes like startup management, portfolio management, strategy, marketing, sales, compliance, finance, product/service development or R&D that can admit new information leading to new steps as the process executes. In the 21st century, these processes may even have to endure major changes even within a few days with a new partnership, a new product offering by a competitor, or a new discovery in R&D.

So in parallel to production-like processes, development-like processes should be optimized with **Lean Development** techniques.

But contrary to production-like processes, development-like processes, which are the source of most of the top-line and competitive advantage to firms in the 21st century, have not yet been optimized through Lean Development techniques hardly at all.

By the end of the 20th century, most Global companies spent most of their efforts optimizing their all of their processes seeking operational efficiency, with good results cutting costs and reducing the bottom line, assuming all of them could be treated like defined processes. This, in fact continues on today in 2016. But this is a grave conceptual mistake that can lead to many problems: development-like processes that admit NEW information cannot be managed well through defined process techniques.

Instead, development processes are better managed through development-like techniques like Enterprise Scrum.

Any process instance within the enterprise that operates while receiving NEW information, is by definition of a development-style flavor – because it needs to change and adapt; and therefore it's not any more a repeatable process, and can be better managed through Enterprise Scrum than by a defined style process management technique. A development style process instance is a process instance where:

- there is high uncertainty because NEW information is flowing into the system requiring adaptation to frequent changes (market, customer, technology, etc.).
- this information could be harder to understand as things get more complex: markets, technologies, interactions, etc.; and therefore, misunderstandings and misconceptions are easier to make.
- outputs and outcomes cannot be readily predicted in advance
- people and teams tend to change faster over time

Empirical process management assumes an **open information system**, where we must adapt as new information comes in, or as our understanding of the up-front conditions becomes better. Empirical process management is based on transparency, inspection and adaptation:

Transparency – so that everything is visible and understood

Inspection – so that we can determine the current state

Adaptation – so that we can adapt, improve and learn

For example, the true state of a company must be **transparent**, so that we can **inspect** its current market position, and then **adapt** to improve its strategic position. Empirical process management is one the fundamental concepts for iterative and incremental improvement, adaptation and growth. The other important is subsumption – the ability to manage appropriately in knowledge layers.

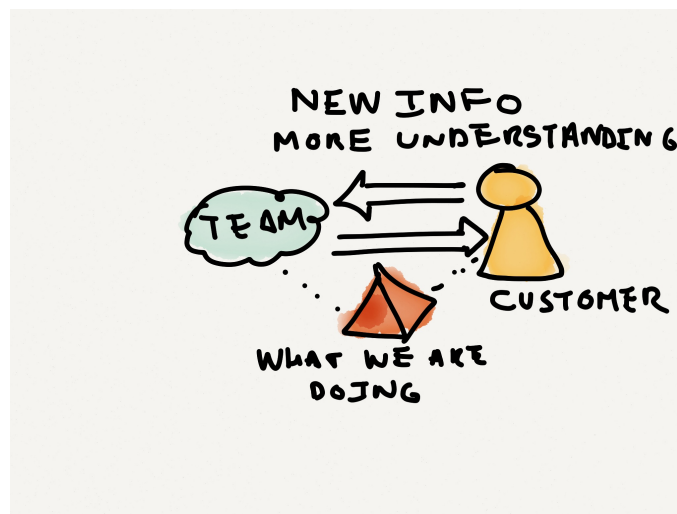


Figure 9. New Information and more understanding make Development-like processes uncertain

As such, all development-like processes, like company management, business unit portfolio, customer segment portfolio, product and service portfolio, a new or existing products or service, and even functional processes – if you are still managing by functions that is, such as finance, strategy, marketing, compliance, legal are better managed with Enterprise Scrum, rather than by traditional defined-process techniques.

Companies will get competitive advantage in the 21st century by developing more top line through agile innovation, agile management and agile adaptation – not by “operational efficiency”.

As we will see, collections of defined process instances also have an empirical flavor because of the statistics of the defined processes, so Enterprise Scrum is also a very useful framework for managing collections of defined processes and any kind of portfolios. Moreover, because business architecture or process redesign is a creative process for a given defined process, Scrum is also used effectively for business architecture and BPR (business process reengineering).

Subsumption

To make it slightly more real – pun intended, and you will see why in a second, the world doesn't operate in just “one layer of management”. For example, a company may have several divisions, each with several Customer Segments, each one of which may have different products and services; and at each level, we need to do empirical process management. This is what subsumption is: managing as independently as possible the different knowledge levels based on the “reality” of the problem”.

“Reality is our model.”

– **Rodney Brooks**, inventor of the subsumption architecture

Minimizing dependencies, as usual, is one of the key Business Agility patterns – this means insulating our bets not only horizontally e.g. many products and services, but also vertically e.g. business units, customer segments, value propositions. In the language of Geoffrey Moore, we need to “zone to win” everywhere in the enterprise [ZoneToWin]. Hierarchy is not necessarily a terrible thing, because hierarchies allow us to partition and organize “zoning”, what we cannot afford is too much couplings, dependencies among departments or phases, because this will cause many systemic problems and slow us down to a crawl. Therefore, Business Agility needs to be customer-centric, all-at-once, and as decentralized as possible.

Enterprise Scrum – Business Agility for Companies of ANY size!!!

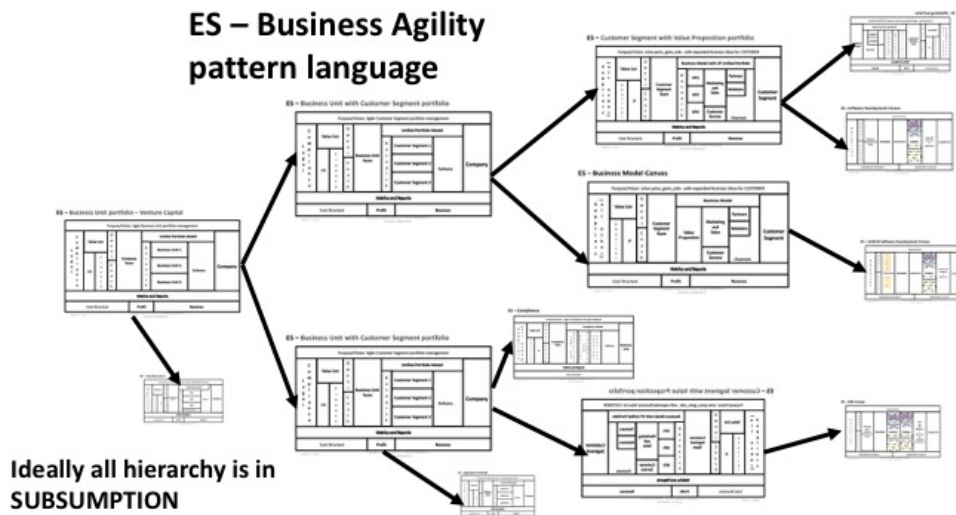


Figure 10. Company subsumption hierarchy

The Scrum Framework high-level description

Modern Scrum, as invented by Jeff Sutherland, is an empirical process management framework for **product development** – typically software and hardware, that delivers the most business value in the shortest possible amount of time. Because Scrum is a framework, we can insert many techniques for **product development**. We will not spend much time here describing basic Scrum; instead, we encourage the reader to learn Scrum by reading and studying the *Scrum Guide* [ScrumGuide]. However, applying Scrum to other things like company management, startups, marketing, financial trading, portfolio management, is hard, because custom ad hoc adaptations must be made. People in the trenches have “winged it” and made these adaptations in the trenches, and it is by gathering this experience that we have derived Enterprise Scrum.

Enterprise Scrum Framework high-level description

The purpose of Enterprise Scrum is to help creating great companies, with great purposes, with happier customers, happier employees, that can grow faster, learn faster and transform faster into whatever they need to be. In other words, the goal of Enterprise Scrum is to make Business Agility possible.

We define Enterprise Scrum to be:

Enterprise Scrum is a customer-centric, co-evolutionary, empirical, subsumption-based management framework to deliver the most business value in the shortest amount of time while balancing the benefits for all

***people involved**, based on an abstraction, generalization, extension and parametrization of Scrum for generic, business, technique-pluggable, and scaled management purposes.*

The more business definition is what is bolded:

Enterprise Scrum is a framework to deliver the most business value in the shortest amount of time while balancing the benefits for all people involved.

Let's explain this a little bit.

- Enterprise Scrum is a **co-evolutionary** because we coevolve everything that is important simultaneously. For example, in a startup we coevolve, the customer, the team, the product or service, the financials, the channels, the relationships, etc. together simultaneously. So we say that in Enterprise Scrum we coevolve the Surfers on the Canvas (all important aspects of things for different people), through waves of Cycles in subsumption mode. In Enterprise Scrum we create different canvases for different activities.
- Enterprise Scrum is **empirical**, because it is based on measurements. No, we are not interested in just productivity or profit, but instead, in everything that is important, hopefully balancing what is most important: people! For example, we can balance customer satisfaction, employee happiness, profits, and a purpose in the world.
- Enterprise Scrum is **subsumption-based** because at any level where we apply it, we require to measure the relevant reality associated with it, and manage making adjustments with the feedback of what we do from reality. For example, in startup management, if we make marketing changes or sales changes, we would use the feedback immediately from our campaigns to very quickly modify our messages.
- Enterprise Scrum is a true **framework**, meaning, there are specific parameters and techniques that we plug-in explicitly while we implement Enterprise Scrum. For example, we may specify explicitly Business Value, metrics, charts, Value List (type of work, frequency, structure, etc.), DOD, DOR, Planning type, Review type, Improve type, canvas, surfers (things that are important), subsumption levels, etc. Unfortunately, the word framework has been abused, but Enterprise Scrum is a true framework.
- Enterprise Scrum purpose is to deliver the most **business value** in the shortest amount of time to everyone involved, hopefully in balance to: customers, employees, stakeholders, and people in the world.
- Enterprise Scrum is an **abstraction** of Scrum because it better describes what Scrum is. No offense to previous people documenting Scrum – including myself, but we have not done a very good job.
- 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, Enterprise Scrum is an n-level subsumption

architecture. Scrum parametrizes Sprint length, Enterprise Scrum parametrizes 130+ other things

- Enterprise Scrum is an **extension** of Scrum, because it adds many things that were not there in Scrum either explicitly or not there before. For example, Scrum did co-evolve development type of activities like requirements, architecture, code, tests, plans, but it didn't say that explicitly. On the other hand, Enterprise Scrum can co-evolve anything that is important together – but that's something that wasn't there in Scrum before.
- Enterprise Scrum is **parametrization** of Scrum, because it adds parameters to explicitly track things. For example, Cycle length, DOR, DOD, etc. and 130+ other things that are important.
- Enterprise Scrum is management in a **generic** way, because it can work for any purpose, at different level or knowledge domain of the organization. For example, executive, middle management, program, project, and for different purposes and business process, for example, company management, startups, business unit portfolio, customer segment, business model, marketing, sales, product development, software development, research, compliance, etc.
- Although Enterprise Scrum can be applied to anything, for example, managing a soccer team, or doing projects for college students, or improving the classrooms for elementary school students, the main driver for it has been in **business**.
- Enterprise Scrum is technique-pluggable, because we can insert explicitly **techniques** from different domains. For example, in Enterprise Scrum we can use explicit 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,
- Enterprise Scrum has the necessary constructs to be **scaled** in different collaboration and structural modes.

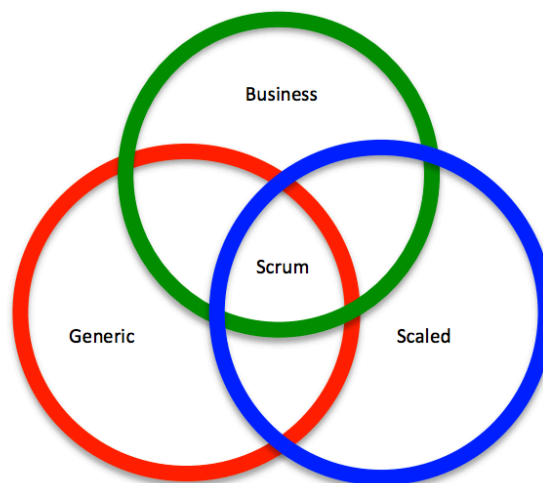


Figure 8. Enterprise Scrum: business-like, generic, scaled Scrum

Brief history of Agile and Scrum

Agile

In business, the origin of the use of the word Agile, started with the creation of the Agile Consortium and the publication of *The 21 Century Manufacturing Enterprise Strategy* in 1991 [Nagel1] and described in detail in the book *Agile Competitors and Virtual Organizations: Strategies for Enriching the Customer* [AgileCompetitors]. However, the position of these business pioneers was that there was no true guidance or patterns, on organizational structure, processes, culture, roles or anything else. That companies should adapt to whatever makes sense.

However, some of us that worked in the trenches re-engineering organizations and processes thought differently. I specifically wrote the paper “cOOherentBPR: A pattern language to build Agile organizations” [Beedle-cOOherentBPR-1997] as a rebuttal to [AgileCompetitors], because my experience differed significantly: there are much more Agile organizational structures than others. True Agile organizations use compressed processes, not phases or departments. Inadvertently, I created a new style of Agile for business management that was all-at-once, based on both my knowledge of reengineering and Scrum.

The Agile ideas were discovered independently in software and culminated in the creation of the *Agile Manifesto* [AgileManifesto], but eerily have the same identical guiding principles as those in business. In fact, I proposed the word Agile at the Snowbird meeting in February 11, 2001, because what we were discussing in those 2 days, reminded me a lot of the business agility which I had worked as described above in 1997. Martin Fowler, one of the other 16 participants at the meeting in Snowbird commented: “... but if we call it Agile, we will be successful” (paraphrasing). And here we are. HBR recently published this as the “true agile story” [HBR-AgileStory].

To add a little bit to the reengineering story, in 1995, I was writing a book on Reengineering the Application Development Process, but I stopped writing because when I got to know Scrum from Jeff Sutherland, I realized that Scrum was a way to “reengineer the development process”. It already provided a way to do **all-at-once management** of the application development process. This is one of the key patterns of reengineering, “process compression”, which is a key feature of Scrum, that later got inherited by Agile in software.

It was a fortunate and lucky choice of words to choose Agile, because now the previously defined “business agility”, and the newly defined “software agility” were eerily similar. There were also very many early ideas in software development that pointed out that this was a good direction. See for example, *Evo* [Gilb], *Quality Software Management* [Weinberg], *Mythical Man Month* [MMM], *Peopleware* [PeopleWare], and *Wicked Problems, Righteous Solutions: A Catalog of Modern Engineering* by Peter DeGrace, Leslie Hulet Stahl.

Scrum

The use of the word Scrum, comes to use from Nonaka and Takeuchi's HBR article *The NEW new product development game* [NonakaTakeuchi], and it was used to describe the type of development that Japanese companies started to do in the late 70s. This style of development started at Toyota, by applying TPS (Toyota Production System) techniques such as kanban, kaizen, JIT, etc. [Ohno], [Liker1], [Wommack1]; to development activities which resulted in a new way of doing development called TPDS (Toyota Product Development System) [Liker2]. Apparently, Nonaka chose this word as they watched Australian TV in Japan, and the overlapping of players in a Scrum, reminded him of the "all at once" way of doing development.

Curiously, TPS itself emerged from the TWI (The Workers Initiative) principles (Training Within Industry program) [TWI]; which was originally a program from the US to train women for work in World War II. After the WWII was over several TWI instructors including Edwards Deming were sent to Japan and Korea to help rebuild these countries.

The Toyota style of product development eventually made it into other Japanese companies, such as Honda, Fujitsu, Cannon, Panasonic, etc.; after these companies started hiring employees and consultants that worked at Toyota. The Toyota family knew about this and was very candid about it. They allow it thinking that these techniques would lead to a better world.

However, Jeff Sutherland invented what we now call "Scrum", "modern Scrum" or "Jeff Sutherland's Scrum" in 1993 by putting together the following four concepts:

1. **Complexity Science** – Jeff is a brain/cellular researcher well-versed in these complexity science since the early 80s.
2. **Nonaka and Takeuchi ideas** explained in *The NEW new product development game* [NonakaTakeuchi].
3. **Organizational Patterns**, recurring best practices in successful teams first documented by Jim Coplien after analyzing what hyper-productive teams were doing [Coplien] [OrgPatterns], and the
4. **Subsumption Architecture**, the concepts of artificial intelligence discovered by Rod Brooks [Brooks1], [Brooks2].
5. **Empirical Process Management**. When tasked to document Scrum Ken Schwaber found another good explanation to explain Scrum through "empirical process control" [ProcessControlTheory].

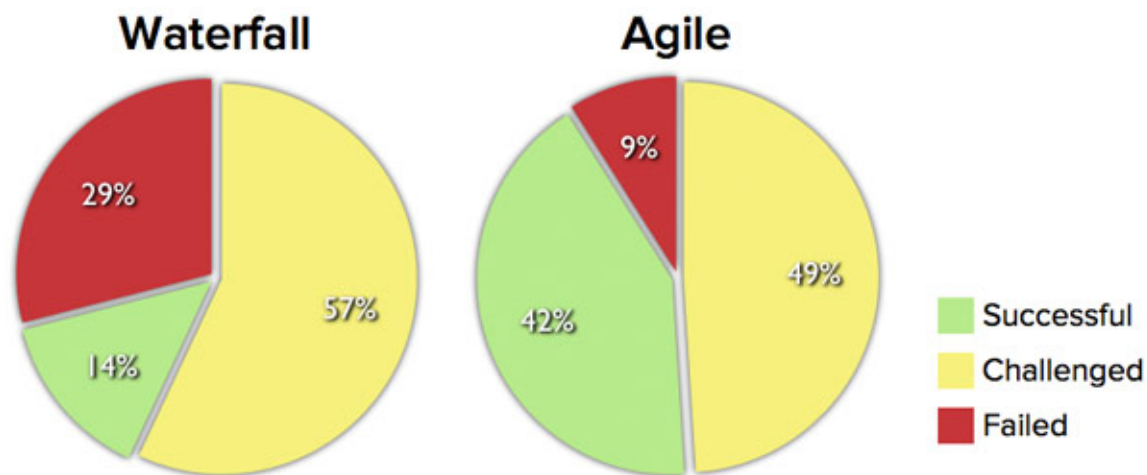
Enterprise Scrum is based on all these concepts and ideas.

The first book on "modern Scrum" (and Agile) was *Agile Software Development with Scrum* [BeedleSchwaber], but now there are many other good books on the subject – it was the first book with Scrum and Agile on its title. Curiously, the proposed title of the book was "Agile Management with Scrum" at some point, but our editor told us that there was barely a market for Scrum management in software development and that there really wasn't a market for general-purpose Scrum management at the time. But even back then, in 2001, we knew Scrum was used, could be used, and should be

used, for many things other than just software development. When I got the domain EnterpriseScrum.com in 2003, I was sure that this type of management would revolutionize the world one day. Well, that day has come. Others have made similar predictions. See for example Jeff Sutherland work, [Agile-3rdWave], or Radical Management by Steve Denning [Denning-RadicalManagement], [Denning-ScrumMajorDiscovery], Dan Greening's work [Greening].

As we will see, the understanding of this history will be very important to understand Enterprise Scrum, as Enterprise Scrum is a continuation of these original ideas, but now extended to any part of a company, at any level, or for the entire company as a whole.

There have been some early attempts to create frameworks for agile companies, see for example, *cOOherentBPR: A pattern language to build Agile organizations* [Beedle-cOOherentBPR-1997], or *Enterprise Architecture Patterns: Building Blocks of the Agile Company* [Beedle-EnterpriseArchitecturePatterns-1998].



Source: The CHAOS Manifesto, The Standish Group, 2012.

Figure 11. Standish Group, CHAOS Manifesto results 2012

Today we know Scrum is better than other styles of management in software development because it has a higher measured probability of success 42% vs. 14% – see for example the Standish Group's CHAOS report [StandishGroup], [StandishGroup – ChaosReport]; and we are starting to get empirical evidence that Scrum and Enterprise Scrum are better management for anything that requires a development style of management – and that's almost everything due to the rate of change in the world today.

But beware, the promise of Scrum or Enterprise Scrum management is not 100% success; it is simply a higher probability of success.

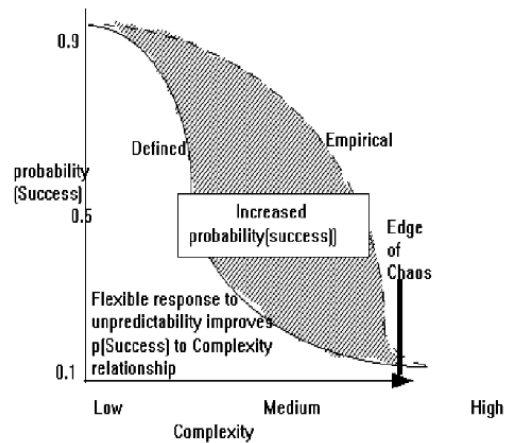


Figure 12. As complexity increases Agile Management and Scrum have a higher probability of success for any business process.

Patterns

Enterprise Scrum comes from the conclusions of observing, hearing about, discussing formally or informally, formally interviewing, consulting on, teaching, or mentoring on in very many companies, projects, instances of Scrum in different domains and industries since 1995, where people in the trenches either had already informally customized Scrum for the purpose of doing Agile Management, or wanted to do so in the future.

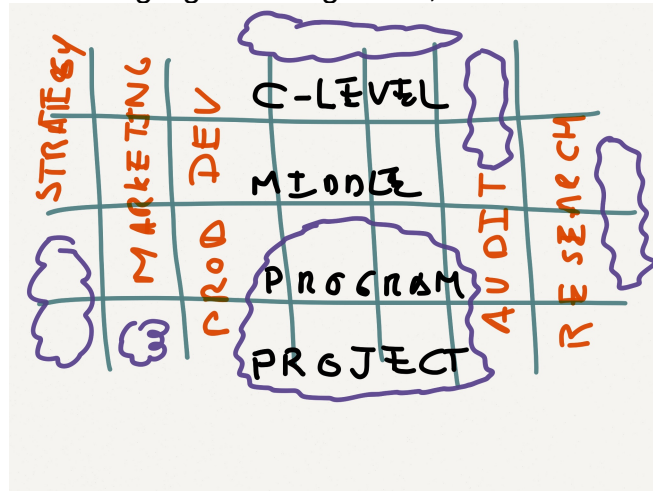


Figure 9. Scrum is being applied EVERYWHERE.

Here is a brief list of some of the early companies and projects where Scrum was first practiced in a generic, scaled or more business-like purposes:

- VMARK – senior management Scrum, 1995
- IDX – first scaled up Scrum, 1996

- Individual – scaled up Scrum, 1996
- Nike Securities, Scrum for BPR, 1997
- CVS/Caremark – scaled-up Scrum (25+ teams), 2001
- New Governance Inc. – company management, 2001
- PatientKeeper – company management, 2001

Today there are thousands of other examples like Salesforce.com, GE, Spotify, John Deere, NPR, Intronis, C. H. Robinson, ABN Amro, JP Morgan Chase, DOD, Enterprise Scrum Inc., Scrum Inc., Cars.com, Scrum Alliance, Hewitt, IBM, Standard and Poor, OpenView Venture Partners, Total Attorneys, Systematic, Trifork, Mission Bell Winery, etc. In a recent poll at the Enterprise Scrum group, we identified 38 different applications of Scrum for different domains, business processes, or activities, ranging from company management, marketing, sales, compliance, HR, finance, startups, all the way to political campaigns, asset management, and emergency management, etc.

These are the type of companies and the kind of processes, programs and projects that led to the concept of Enterprise Scrum. Jeff Sutherland gives us very many other examples in his book *Scrum: The Art of Doing Twice the Work in Half the Time* [Sutherland]. There are also very many examples in the recent HBR and Forbes articles on Agile [Denning-ExplainingAgile], [HBR-AgileStory], [HBR-EmbracingAgile], [Kniberg], etc.

There are thousands of examples in very many industries throughout most business processes, ranging from startups through Global 5000 companies. In fact, the Scrum Alliance has recently formed the Learning Consortium, which mission is to encourage the use of this style of management [LearningConsortium].

So the process by which these patterns and parameters have been found and validated is by observing the world and mining patterns, and then applying the patterns and verifying that the patterns work in the real world [Alexander]:

- **Pattern Analysis (or mining): analyze World → get Patterns from success stories**
- **Pattern Application: apply Patterns → test success stories in the world**

Many companies, projects and processes have benefited from the application of these patterns. In my estimation, there are at least 100,000 people doing Enterprise Scrum, and possibly as many as 1,000,000 by very conservative estimates. My prediction is that in the future, this style of management will grow rapidly in adoption and that it will cause what historians will call perhaps:

- “the Agile Management Revolution”,
- “the Knowledge Worker Revolution” as Peter Drucker predicted,
- the “4th Industrial Revolution” as Klaus Schwab from the WEF calls it, or
- the **Innovation Revolution**, as I prefer to call it.

Many of these patterns unfortunately have not been properly documented in the proper

organizational patterns form, which follows the seminal work started by Jim Coplien back in 1993 [Coplien]. However, some of us at the ScrumPLOP working group will continue to document Scrum and hopefully Enterprise Scrum patterns in the future. We are planning to publish a book on with these patterns [ScrumPLOP].

Agile Transformation

We are now witnessing now the 3rd wave of agile coming our way: Business Agility everywhere in the enterprise. The first agile wave was “single team” in product development that started circa 2001, and the second wave of Agile was scaled agile in product development that started circa 2007. In the beginning of 2016, I still sounded like a broken record: “Guys, the big Business Agility wave is coming soon!”. People would listen to me, and ask: “Where Mike, where is it?” I quickly pointed people to the known examples and case studies, but there wasn’t simply enough evidence to call it a Business Agility wave. But things changed quickly. Sometime in the beginning of 2016, all of a sudden, we started to get more and more case studies, specially in financials and insurance, but quickly expanding to pharmaceuticals, manufacturing, technology, etc.

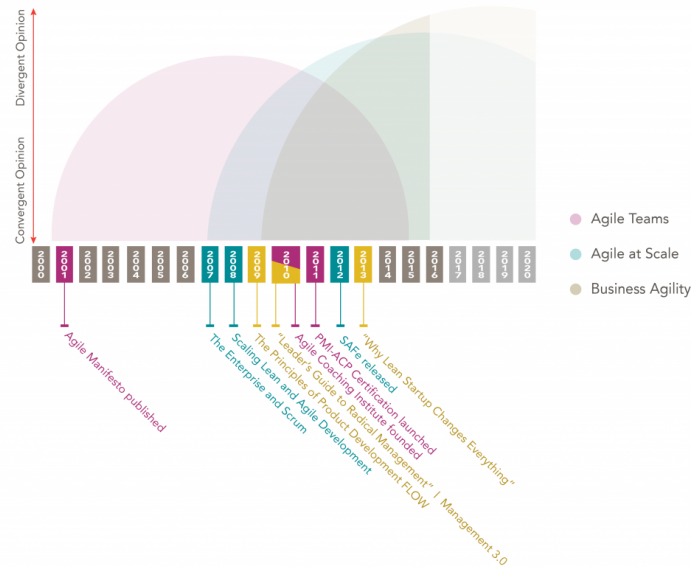


Figure 13. Agile waves: single team SW, scaled SW, Business Agility

I am now confident to say that in 2017 and the years to come, we will see Business Agility grow exponentially. In fact, we are now offering Enterprise Scrum – Business Agility classes and certifications, and the classes are filling up quickly. We are also seeing Business Agility conferences appear, for example, in 2015 there was a Business Agility conference organized by Daniel Mezick. Also, there was a recent Business Agility conference in New York city [BusinessAgility2017].

One of the recent reports from Deloitte, says that the top executive priority right now is organizational design. Executives are desperately looking for new ways to organize their companies in a more agile way [Denning-AgeOfAgile], [Deloitte2016], [Gonzalez]. Well, help is on the way, that's exactly what we can provide with the Enterprise Scrum Framework and the Enterprise Scrum Business Agility patterns.

Additional Guidance

There is also a lot of related guidance as to how to structure, manage, and transform organizations that innovate with or without Scrum – clearly there are overlapping patterns among these techniques:

- **Agile and Scrum distribution** [DistributedScrum]
- **Agile business management** - [AgileCompetitors], [RadicalManagement], [BeyondBudgeting], [LittleBets], [ConnectedCompany], [ReinventingOrganizations], [ExponentialOrganizations], [Leybourn]
- **Architecture** [Alexander], [ConnectedCompany]
- **Business models** - [BusinessModelGeneration], [ProfitZone]
- **Cognitive Science** [Snowden-Cynefin], [Snowden-ComplexActs]
- **Culture** [TribalLeadership], [SmartTribes]
- **Digitization** - [McKinsey-Digitization]

- **Good business processes** [Hammer], [Wommack2], [Wommack3]
- **Good companies** [Collins1], [Collins2], [Hamel1], [Hamel2], [Nonaka-PragmaticStrategy], [Hoshin1], [Hoshin2], [Porter1], [Porter2], [BalancedScorecard], [ToyotaWay], [Nonaka-ManagingFlow], [ScenarioPlanning]
- **Good products and services** [Coplien], [DesignThinking], [DistributedScrum]
- **Innovation** [Peters], [Pink1], [Pink2], [SchwaberSuttherland], [PowerOfScrum], [Stacey], [BlueOceanStrategy], [CEK]
- **Knowledge Management** [Nonaka-KnowledgeCreating], [Nonaka-KnowledgeCreationAndManagement], [Snowden-MultiOntology]
- **Lean Management** [Liker1], [Liker2], [Poppendieck], [Reinertsen], [Wommack1], [Wommack2], [Wommack3]
- **Scaling** [AgileSoftwareDevelopment], [Eckstein], [DAD], [Schiel], [Kniberg], [SAFe], [StoberHansmann], [Cohn3], [Schwaber], [Rawsthorne], [Schliep], [ScalingUp], [Greening], [Lefingwell], [Less1], [Less2],
- **Startup Management** [RunningLean], [LeanStartup], [Blank],
- **Teams** [Management3.0], [Atkins], [Katzenbach]
- **Transformation** [HBR-ChangeManagement], [LeadingChange], [RisingManns]

For lack of a better word I have chosen to call these techniques under the banner **Agile Management**; because whether these authors label themselves as Agile or not, they use many of the same patterns.

2. Enterprise Scrum Overview

In the introduction, we defined Enterprise Scrum as - business definition:

Enterprise Scrum is a framework to deliver the most business value in the shortest amount of time while balancing the benefits for all people involved.

Full definition:

Enterprise Scrum is a customer-centric, co-evolutionary, empirical, subsumption-based management framework to deliver the most business value in the shortest amount of time while balancing the benefits for all people involved, based on an abstraction, generalization, extension and parametrization of Scrum for generic, business, technique-pluggable, and scaled management purposes.

So how exactly do we accomplish that?

Enterprise Scrum Features and Goals

When we do Enterprise Scrum we have the following features and goals:

1. Purpose, Greatness, Passion, Ownership. We seek to ask people to do things with great purpose, and to do them as great as possible given our abilities. This ensures that people do something they believe in, and therefore something they want to do. Also, if they find themselves working with people that also believe in the same goals, it is more likely that they enjoy working with them. This makes it easier to do things with passion, with ownership, with pride. We encourage a co-creation environment, where everyone is allowed to “dream together”, and contribute to the dream.

2. Balance Management. We seek to balance the benefit of 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 business value or a better customer experience, 2) keeping employees happy, 3) having a purpose to make a better world and 4) profit. We can choose different metrics to find a different balance for different Enterprise Scrum instances.

3. Multiple Results-Oriented Metrics. We use one or more results-oriented metrics that measure people-related important things for whatever we are doing; as opposed to tracking productivity metrics, activities, effort or work in progress. For example, we may monitor: customer satisfaction, employee happiness, number of issues, profits, features built, compliance levels, etc.; whatever makes sense to optimize and balance specially with a people-centric view.

4. Greater Autonomy, Decentralization and Decoupling. One of our goals is that each

Enterprise Scrum instance is as self-directed, self-organized and self-managed as much as possible instead of working through functional, phased-based, process oriented, or product or service-oriented organizations. This allows us not only to take “smaller bets”, and “fail faster”, 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.

5. *Empirical Management and Projections.* In Enterprise Scrum, plans, forecasts and decisions are based on measurements made on our metrics, and based on 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 measurements.

6. *Decreased lifecycles for everything.* Enterprise Scrum can be used to radically improve delivery times and time-to-market for everything, from marketing campaigns, higher level of compliance, business model delivery, or increased customer experience and satisfaction. Average productivity improvements when we switch to Scrum management, is of hundreds of percent. However, pursuing “higher productivity” is pointless. It is much better to pursue a people-oriented benefit, like higher customer satisfaction, or higher employee happiness.

7. *Faster, Efficient and Effective Growth and Adaptation.* In Enterprise Scrum, we work in rapid all-at-once iterative-incremental cycles that through feedback, lead to faster and *better learning, growth, adaption, visibility, risk management, confidence, and certainty*; instead of trying to plan for everything up-front and then “execute” a plan. We manage assuming change rather than assuming a static world where long-term plans and predictions can be made without changes. For example, a startup team can quickly adjust its business model, marketing, sales, products or services, or customer service based on the feedback they may get from the customer.

8. *Better reaction time.* *Because* Enterprise Scrum works as an n-level subsumption architecture our reaction time is optimized at all levels. A subsumption architecture is one where “reality is the model”; therefore, we keep probing the reality of the specific domain in real-time, and we adapt quickly based on measurements. For example, in compliance management, we may test controls and make business adjustments as needed. This ensures that we have optimal reaction time and that we don’t disconnect from reality – which is something that large companies tend to do.

9. *Lower Risk.* We also *minimize risk* because by proving that we can do things, through results in short cycles, we can mitigate risk.

10. *ALL-at-once management.* *One of the fatal flaws of traditional managers is “aspect-oriented thinking” and focusing. While it is good to focus on an area of interest, local optimization often yields poor global results; therefore, in Enterprise Scrum, we try*

to co-evolve everything through Cycles.

11. Greater efficiency. In Enterprise Scrum, there is no phases, handoffs, departments, or approvals up and down. This leads to greater operational efficiency, better efficiency ratios and less waste. However, cost reduction, or waste elimination, are not the only reasons why we are interested in doing Enterprise Scrum. Avoiding waste is not very meaningful if we are doing the “wrong thing”.

12. Easier change management. Because work management is much simpler in Enterprise Scrum, making changes is much easier and localized.

We will now describe Enterprise Scrum in detail. We will use **bold** to denote the elements of Enterprise Scrum. For example, **Value List, Business Owner, Coach**, etc. We will use Bold Italics to point to *configuration sections* or configuration parameters e.g. ***Structure***.

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
Scrum Master – coaches Scrum Team to do Scrum for product development	Enterprise Scrum Coach – coaches Enterprise Scrum team to configure and do Enterprise Scrum
Team – does product development	Team – does any kind of work
Product Backlog – list to develop a product	Value List – list to deliver value
PBI (product backlog item) – feature or something else to be DONE in	VLI (value list item) – anything that gets DONE and delivers value

product development by passing a DOD	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, with configurable options for the cycle (Planning, Collaboration, Review, Improve), with nesting allowed in time and structure e.g. one week ICs contained in quarterly ICs contained in 1 yr. ICs.
Daily Scrums – mandatory	Daily Scrums – optional under some circumstances where subsumption is achieved anyhow
DOD, DOR – for product development	DOD, DOR – general purpose
Scrum Board – for product development	ES Scrum Board – for general purpose. Similar workflow than that of ScrumBoard, but moves VLIs from the “not selected” canvas to the “done” canvas
Velocity – single productivity metric	N Metrics – many possible metrics: customer satisfaction, employee happiness, velocity (effort), profit, etc.
	Calculations – there can be one or more calculations for different Cycles. For example, recalculation of schedules, budgets, etc.
Product Increment – an increment for for a product	Value Increment – where value is delivered for ANY purpose
PSP (potentially shippable product)	PV or AV – potential or actual value
Sprint Burndown	Charts to graph Metrics, including but not limited to Burndowns

In essence, Enterprise Scrum is an abstraction, generalization, parametrization and extension of Scrum.

Enterprise Scrum Instances

Enterprise Scrum is a *true framework* – it can be explicitly configured and instantiated, and it exists as an explicitly customizable tool through parameters in a configuration, yielding an **instance**. Once the parameters are configured, we can create Enterprise Scrum **instances** to agilely manage anything! We typically name different instances using the template:

“Instance of Enterprise Scrum for doing X”;

meaning that is no longer just the raw framework and that now includes values for the configuration, including specific techniques. 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 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
- Instance of Enterprise Scrum for Business Unit Portfolio management
- Etc.

See the **EnterpriseScrum** configuration chapter to see all possible parameters:

ESTeam – where we configure the team

customer – where we configure the customer or customer segment

informationArchitecture – where we configure the information architecture including VLI Types, surfers, canvas, etc.

knowledge – where we configure the techniques

valueList – where we configure the Initial Value List and keep track of the events in the Initial Value List

process – defines the process derived from events and techniques for initial value list and Cycles

cycles – where we configure the cycles and keep track of cycle instances

metrics – which metrics are we keeping

reports – which reports are we having

calculations – which calculations are we doing

structure – where we configure the connections to other Enterprise Scrum instances

structuralPattern – which organizational pattern are we using

collaborationMode – which collaboration mode are we using

deliveryMode – where we configure the way we deliver value

deliveryTarget – where we configure for what target we are delivery value

contractType – where we configure the contract types

Enterprise Scrum Templates

In Enterprise Scrum is it customary to use templates for different things:

- Instance of Enterprise Scrum – the whole configuration but could be incomplete
- Enterprise Scrum Canvases – just the canvas. There are currently 13 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 entries for: **ESTeam**, **Customer**, **InformationArchitecture** (including VLITypes, surfers, canvas, etc.), **ValueList**, **valueListSequence**, **Techniques**, **Cycles**, **cycleSequence**, **contributors**, **connectionMode**, **contractType**, **deliveryMode**, **deliveryTarget**.

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. You 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 works well only if the people doing Enterprise Scrum want to do Enterprise Scrum, and want to do the work they want to do, with the people that they want to do it, for a specific purpose; otherwise ... expect problems.

Liking the people you work with and loving what you do are necessary, either as a starting point, or as something you pick up along the way because this will generate the right values. These values are necessary for trust, and trust leads to a collaborative, cooperative, mutually helpful Agile culture.

Trust is the only way you can get the right culture to do Scrum or Enterprise Scrum. Once you reach trust, it is easy to share, or get to a sharing place. This is what Nonaka describes as BA in Japanese – a “sharing place”, where people can: share knowledge, collaborate, cooperate, ask for help, and provide help to others. This is the so-called Agile Mindset, which is the mental model and values that an individual has while participating in an agile environment.

Agile Culture is what happens to a group of individuals working in an agile environment, where the people in that environment have an Agile Mindset. But this Agile Mindset can't be imposed or forced onto individuals, because you can't impose or force the Scrum values if people don't like what they do, or work with people that they don't want to work with. It is pointless to say that people should “obey” or “work with” the Scrum values of commitment, courage, respect, focus, openness or honesty, if they don't like or respect their co-workers. Following values has to come from the heart.

If people are capable, love and believe in what they do, and like the people who they work with, you won't have to force them to act with the Scrum values -- they will *naturally* have the Scrum and Agile values, and the BA "sharing way".

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.

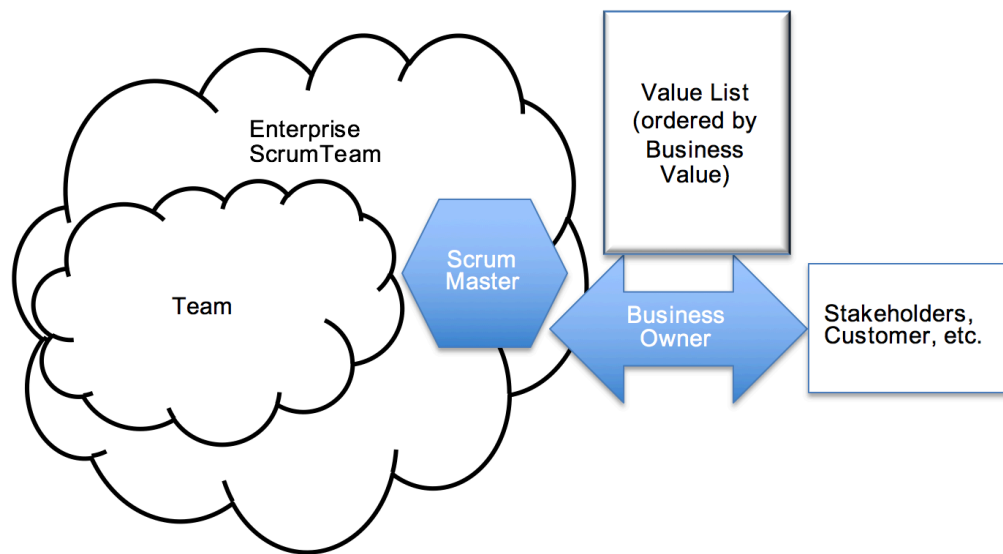


Figure 14. Enterprise Scrum Team

As opposed to Scrum, these are roles, meaning they can be distributed among 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 happiness, 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.

Therefore, 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 is ok 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.

Enterprise Scrum Teams operate at higher operational efficiency than traditional teams, reducing waste, avoiding hand-offs, functional silos or deep delegation hierarchies. This is easy to understand because they work in “reengineered” or compressed process mode. Also, from an implementation perspective, Scrum and Enterprise Scrum, use 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, sales, compliance, etc. – although we prefer that other organizational forms, a business process, a product, service, project or program. We will see in the Business Agility chapter what are better and worse ways to organize ourselves to manage a company.

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

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 **Value List** at all times by defining, prioritizing, or changing it. All of these actions are termed “refinement of the Value List”.
- Working 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**.

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 more coordination
- **Single Person Business Owner**: with one person we get expedited decisions, because there is less coordination, but 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.

When the job of the Business Owner is too large, it is also 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.)

Keep this in consideration when you implement the role.

Enterprise Scrum Coach

The **Coach**:

- helps configure the Enterprise Scrum framework for appropriate use – this is an important difference between a ScrumMaster and an Enterprise Scrum **Coach**, and the primary reason we don’t call an Enterprise Scrum **Coach** a ScrumMaster
- coaches the **Enterprise Scrum Team** to do the best Enterprise Scrum possible
- proposes initial team members, that hopefully select other team members.
- helps improve the team technically, socially, emotionally, collaboratively,
- finds gaps and provides viable strategies to improve everything: the team, the process, the configuration, etc.
- constantly empathizes, encourages, motivates, communicates, assesses, mentors, teaches, etc.
- schedules all Enterprise Scrum meetings

- 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**
- 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.
- **Distributed Coach**: it is possible that the team distributes the Coach role among all team members, the risk there is that the team can't see "beyond the forest", and could get stuck specially in social issues.

When the job of the Business Owner is too large, it is also 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.)

Team

The **Team** is one or more people that do the work.

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 **VLI**s (Value List Items) of the **Value List** DONE according to their **DOD**, and delivers **Business Value**
- chooses the amount of work into every Cycle
- always decides the scope to work that the team can get done as changes take place
- Teams accomplishes their work in the most self-organizing way possible. 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.

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 compliance management, it could a higher level of compliance or less incidents. For a software development effort, it is typically features done.

To have empirical evidence that we have obtained some business value we must measure it appropriately, and then try to improve the delivery of business value in the future through **Cycles**. It is also typical to create **charts** for the **metrics**.

Knowledge

Enterprise Scrum is a true framework, meaning that you must specify where and how specific **patterns** and **techniques** are used and for what purpose. Also, we expect to fill this knowledge repository over time as we convert “chaos into order”, chaos into complex, then complicated and then known [Cynefin].

Because Enterprise Scrum is a formal framework, we may choose to insert techniques between or within these steps, but the Enterprise Scrum definition allows to specify exactly how and where the techniques and patterns are used. This is important if we wish to re-instantiate the instance for some other team.

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

Patterns

Patterns for different domains are available as “best practices”, including those for Business Agility. Why reinvent the wheel all the time when we can use the best knowledge available in each industry?

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.

Information Architecture

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

What do we deliver and to who? What are the list of deliverables?

How do we deliver it? 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.

This also derives:

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

Canvas

Also, for convenience, we may want to build a custom Canvas for 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

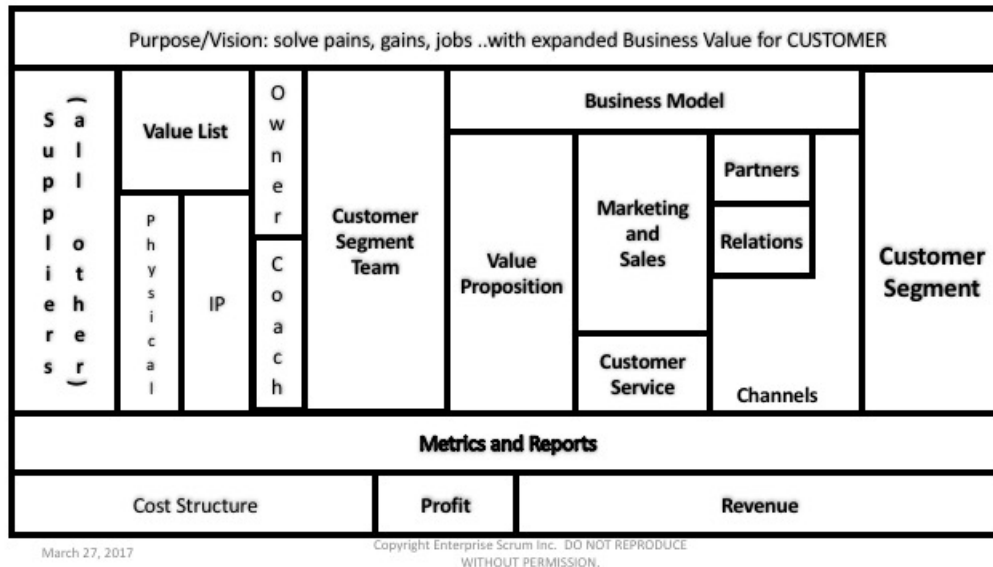


Figure 15. Enterprise Scrum Business Model Canvas

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? 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 we co-evolve customer satisfaction, requirements, architecture, the plan, the development team, etc.

The Canvas must include all of the “surfers”, attributes, for all of the VLITypes.

The Surfers map to the activities in the Initial Value List, and the PC3R Cycles.

If in addition to the above questions we ask:
 Who delivers these aspects?
 Through which activities?

Then we can use the Canvas to:

- First fill in the “big picture” through the Initial Value List
- Then, we can coevolve, including the dependencies among **VLITypes**, the different Surfers that make sense through the PC3R **Cycles**, until we reach our objectives.

Value List

The **Value List** represents the vision, and targets a purpose, through a list of **VLIs**, each of which bring **Business Value** to a customer. 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 while we coevolve several important things for that work to be done at once in each **Cycle**. And for different types of work, we have different types of VLIs, 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. When we realize a trade, all of these things must be coevolve simultaneously, even though they may have different interested parties. We call all of the important aspects or attributes of the same VLI the **Surfers**, and we coevolve them together as we get the VLI done through a wave of **Cycles**. However, if we also have to coevolve a financial compliance document, or the infrastructure required, they will require other aspects or attributes to be coevolved.

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 one of the important key aspects of Enterprise Scrum, because they generalize work for general purpose management. The attributes map to attributes of the **Surfers**, so that we can coevolve things. One of the most important attributes is the VLIttype, as

described above, because that tells us the type of work that we are doing, and therefore the Surfers we should expect in that VLIType.

For example, a VLI will eventually have information about:

- the plan to get it DONE,
- its DOR (Definition of Ready),
- its DOD (definition of done),
- 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.

Once we map the extra **VLI** attributes for a VLIType, indirectly we will 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, as in most Global 5000 companies. We must specify “when are we done”, for any business process where we want to use Enterprise Scrum.

Done is not “partially done”, “almost done”, “I think it's done” -- it's 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 signed sales contract
- 2) we must have a cashier's check that includes the total price, fees and closing costs
- 3) 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 things after each **Cycle**, and then adapting 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 a few hours 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**. We can **Plan**, 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 **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, 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 implement the solution: 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).

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. For example, 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, m-things in subsumption generalized subsumption behavior for any domain. Subsumption means “checking with the 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 integrate each one of the things coevolved into the Cycle structure. 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.

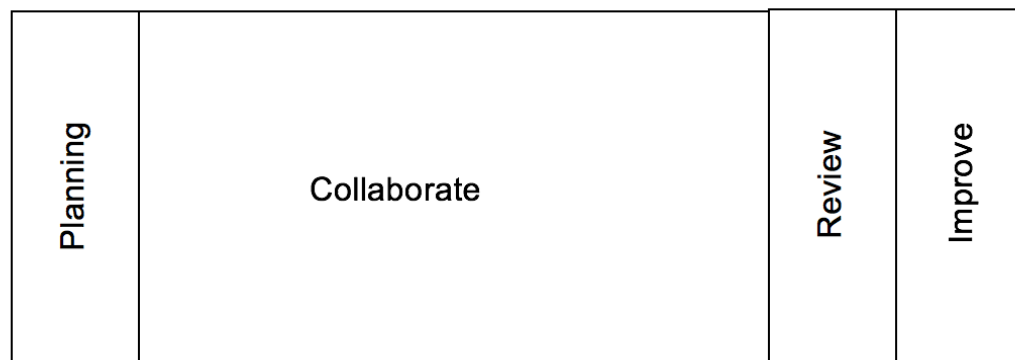


Figure 16. Cycle Structure in Enterprise Scrum

The Cycle structure in Enterprise Scrum is:

Planning – where the Team determines in **part 1**. 1) how much work they can do for all Surfers, and in **part 2**, 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 optional under certain circumstances or contexts.

Collaboration – accomplish work that passes a **DOD** through collaboration, cooperation, helping each other and sharing knowledge to get as much of the chosen work on the Canvas 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.

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.

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) overall sales, 2) sales projections, 3) sales infrastructure and 4) overall team improvements the **Surfers** of the 1st level Cycle; 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, 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
- employee happiness
- goodness for the world
- 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:

$BR \text{ (Burn Rate)} = TC / \text{Number of Yearly Cycles}$

Budgets

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

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

Even with a profit percentage X:

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

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. Even 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.

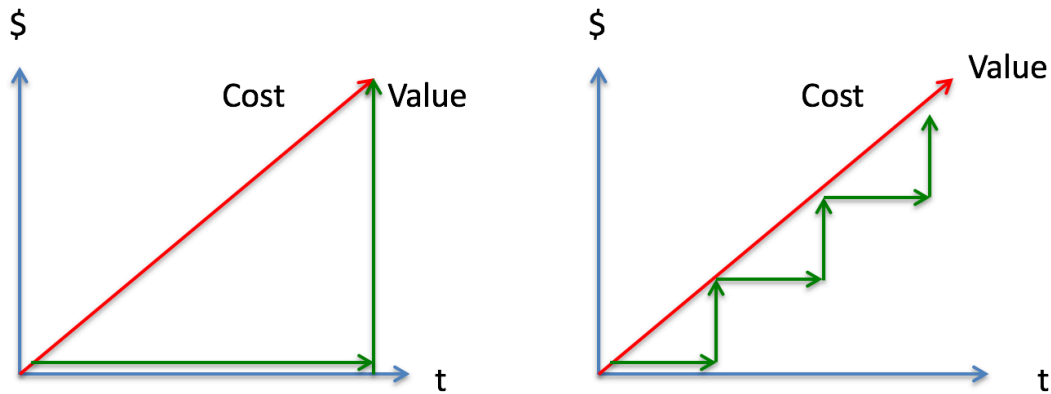


Figure 17. Comparison of business value delivery from Waterfall to Iterative Incremental

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

Iterative Incremental Growth Principle

Because this is the way we always work in Enterprise Scrum, we have formalized this as a principle:

Iterative Incremental Principle – In Enterprise Scrum everything we do is in an iterative-incremental way. We grow, learn, adapt, everything we do in an iterative and incremental way.

This is not an Enterprise Scrum principle, rather, it is a general principle that applies to everything.

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 due to the fact that 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, so we **grow** everything: sales, revenue, profit, quality, customer satisfaction, organizations, transformations, quality, etc.

Growth follows a similar curve as the S curve above, but it also has a death or termination phase:

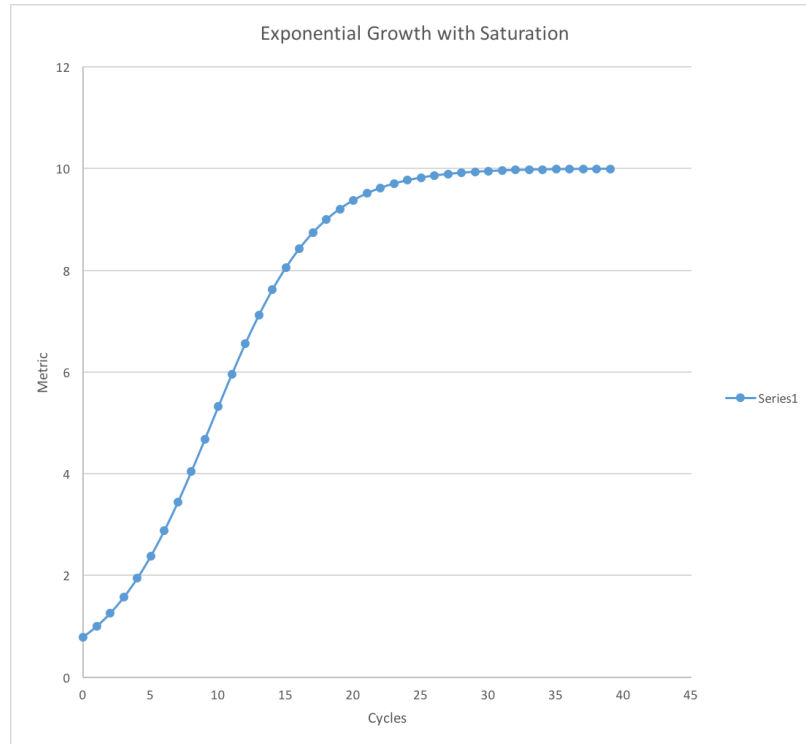


Figure 18. S curve expected growth or improvement over time.

In Scrum we grow and improve through cycles or iterations of equal length we call Sprints, by measuring the amount of work we can accomplish and then looking for ways to improve. One of Jeff Sutherland’s first Scrum articles advertised a measured 600% improvement. But of course, this improvement was not achieved in one iteration – 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 “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 “Improvement” Cycle!!!

However, in Enterprise Scrum, we are not limited to improve the “amount of work” accomplished or velocity; we can improve customer satisfaction, quality, return on investment, sales, compliance levels – just about anything we want through the introduction of metrics. In fact, we can improve, optimize or maximize not only one quantity but several of them together.

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 in order 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.

So our long-term goal is to constantly adapt, tweak or morph to achieve our goals, to survive. For example, a startup may have to endure several adaptations to find its final business model. It is rumored that Groupon went through several business models before finding its final “global coupon manager” model.

Adaptation is typically done in cycles where we test our hypothesis over a testing period and then we make adaptations, corrections or improvements to our business models through better strategies, better marketing and sales, better products and services, or even better customer feedback.

Of course adaptation is at the center of the evolution of living things, where nature constantly tests life, and applies the rules of natural selection: only animals that can adapt to their surrounding context survive. And the same is true in business. However, being somewhat “rational animals”, we can use our intelligence to evolve rather than waiting for chance to decide for us our survival.

For example, Lean Startup management tells us to implement a business idea and then test it to see if it's working according to our plans in an iterative way [LeanStartup], [RunningLean]. If it is working, persevere; or else pivot. The same is true in Beyond Budgeting where we make adjustments as we get feedback from the field to long and medium term plans [BeyondBudgeting]. And the same is true as we iterate to find a working product or service as a Blue Ocean Strategy [BlueOcean]. Finally, we could equally iterate to accomplish a different business models using the techniques described in Business Model Generation [BusinessModelGeneration]. All of these techniques are of course in agreement with concepts explained in Little Bets [LittleBets]. Other techniques like Scenario Planning, Profit Patterns, traditional Strategy, or Kano

models come handy to make better decisions as well.

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 [Brooks1], [Brooks2]; which has always been an important element in Scrum.

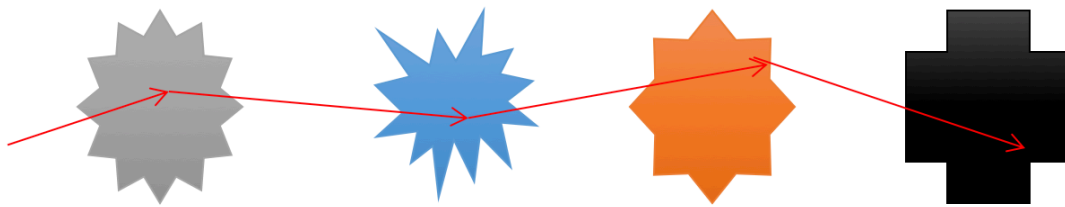


Figure 19. Change business models, products, services, features as you get feedback.

Again, for the full technical details of the Enterprise Scrum framework, see the Enterprise Scrum Definition in Appendix A.

Recursive Cycles

However, 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)

For example, for Startup management we are interested in finding a viable:

- Business Model

- Product or Service
- Effective Marketing
- Revenue/Profit

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.

We call the enclosing Cycle “the wave of the lower level cycles”, and we say that we coevolve all surfers -- all of the things that are important, through a “wave of cycles”.

Wave Principle

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 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, for example [BeyondBudgeting]. But this behavior is now generalized in Enterprise Scrum:

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 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.

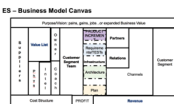
At the **Cycle Planning** we 1) move the chosen work from the “not selected” canvas or column into the selected/WIP column, and 2) we make an initial plan on how we are going to getting to **DONE**. For example, if we are doing compliance management, it could be an internal control that needs to be chosen 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. For example, give explanations of problems of what didn't get to **DONE**. For example, we show what the results of the internal controls that we tested. 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.

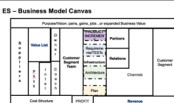
At **Improve** we improve anything we can in the entire Canvas including how to improve the team.

ES - ScrumBoard Vision



Business Owner and Stakeholders

Initial Value List



Business Owner, Stakeholders and Team

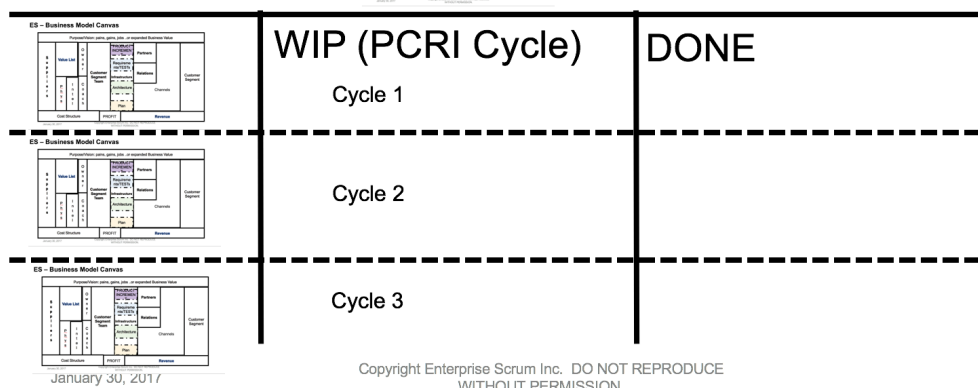


Figure 20. ES - ScrumBoard tracks how much work is getting DONE.

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”.

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.

Debts

In Scrum is customary to call the technical things not done, the “technical debt”, because in Enterprise Scrum we generalize the type of work done, we may also have:

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

These are useful concepts to understand some of the systemic issues concerning the management of a system of organization at any level, or for any purpose.

If we can identify these “debts”, we can then start mitigating them.

Enterprise Scrum Parameters

Please see the full list of parameters and questions to get the parameters in the following chapters.

3. Business Agility

What is Business Agility?

Company management means different things to different companies. There are very many ways to manage a company, and we categorize these ways by asking different questions:

- What is their industry?
- What is their focus?
- What structures do they use? Functions, processes, products, services, etc.
- Which ones are more efficient? (in cost, time, or effort to revenue ratios, etc.)
- Which ones are better for employee management or even employee happiness?
- Which ones are easier to create or transform into?
- Which ones require the lowest employee entry level?
- Which ones are easier to manage?
- Which ones are more Agile?
- Which ones are more predictable or more uncertain?
- Which ones deliver more customer satisfaction?
- Which ones are more innovative?
- Which ones were more likely to thrive and survive in the 20th century?
- **Which ones are more likely to thrive in the 21st Century?**
- Etc.

You may think that even a simple question that had a simple answer in the 20th century would have an equally simple answer in the 21st century, but sometimes that is even hard. Consider Uber, Google, Tesla or Amazon – companies get more diversified by the minute these days.

A proper scientific taxonomy of all the types of management and power structures would need to include all of these “degrees of freedom” or variables, and evidence of how effective they have been in practice could be supported by empirical evidence. However, no such taxonomy exists today to my knowledge.

One attempt of such taxonomy is given for example by Laloux in *Reinventing Organizations* [ReinventingOrganizations]. The advanced management examples he presents in his book are pointing to agile things: purpose, self-organization, decentralization, autonomy, transparency, participation, team leadership, etc. However, I disagree that organizations can be classified by a single benefit purpose; instead, they can strive for balance, like Tesla: clean energy AND profit. And therefore, I also disagree that an organization with a certain purpose, always correlates well with its organizational structure, specially if this organizational structure is new.

Nevertheless, I found Laloux work inspiring to attempt another classification.

For me, in these times of disruption, the key element is survival, adaptability, agility. So

for the purposes of competing and surviving in the 21st Century, I wanted to have a taxonomy of which organization types are most likely to thrive and survive. In other words, which ones had more Business Agility.

Enterprise Scrum Organizational Taxonomy

So we propose to classify organizational options in terms of how easy or how hard is to achieve Business Agility – manage, adapt and innovate based on customer feedback to deliver more value and better customer experience:

Organization Type	Advantages	Disadvantages	Ease of Customer/Market Feedback and Ownership	Org Adaptability based on Customer Feedback
Functional (Sloan)	Easier internal management: people work with same type of functional specialist	- No one owns the customer - Large companies tend to grow silos - Hard to coordinate among functional departments - Hard to launch NEW products and services - hard to manage portfolios	Very low	Very hard, internally everyone sees a very faint image of the customer
Process-Oriented	- Makes processes very efficient i.e. with high "operational efficiency"	- No one owns the customer - Different processes tend to grow inconsistencies - hard to launch NEW products and services - hard to manage portfolios	Low, feedback is done "process by process"	Not so good, the overall customer perspective is often lost in the process mesh
Products and Services	It is easier to manage a portfolio or products and services.	- No one owns the customer - it is very hard to manage all products and services related to a customer	Medium	Not so good because feedback is done through single product or service, and typically done through phases
Customer Segment Oriented	- high customer ownership - leads to greater customer satisfaction - "unified" management	Not necessarily easy to manage multiple business units, customer segments or value propositions	High	Not necessarily great – depends on structure underneath
Decoupled Business Units,	Easy to add or remove Business Units, Customer Segments, Products and Services	Guidance from Customers is not guaranteed	Not necessarily great	Good, but customer feedback is not guaranteed

Customer Segments, Products and Services				
Decoupled Bus, CSs, and VPs with Strong Customer Feedback	- High customer ownership - Greater customer satisfaction - Unified management - Easy to add or remove Business Units, Customer Segments, Products and Services	- some duplication or redundancy may exist on the decoupled organizations - people in different orgs may have a harder time understanding the “big picture” of the company	High	Extremely good

Table 1. Enterprise Scrum Organizational Taxonomy based on Adaptability

CEO, CFO, CIOs, CCO, CLO, CKO, CVO, CSOs, COO, CMOs, CAOs, CPOs, CROs, CXX??? ...STOP Sloan functional silos and focus on the CUSTOMER!!!!

After doing this analysis, I concluded that: ***autonomous, decentralized, decoupled, customer-centric, all-at-once managed organizations provide a much better customer adaptability*** rather than those organized by functions, processes, or single products and services. I did a similar analysis some 20 years ago, and concluded that these type of organizations were more efficient: cOOherentBPR: A pattern language to build Agile organizations [Beedle-cOOherentBPR-1997]. But they are also more likely to deliver more customer satisfaction, business value, and a better customer experience.

We could agilize functional processes, like marketing, compliance and hardware manufacturing, but really, the goal of Business Agility is all-at-once responsive management. You can in fact agilize a completely “Sloan managed” company, or all of its “functional processes” i.e. strategy, marketing, product development, customer service, sales, etc. That may have been great if we were in the middle of the 20th Century competing through “operational efficiency” and limited by “functional workers”.

However, in the 21st Century, where the customer is the center of the Universe and change is too fast to implement phase-based approaches, all-at-once customer-segment oriented organizations tend to do better.

Therefore, the management approach that I present here to manage companies – from startups to multi-nationals with several independent business units, is always 1) ALL-AT-ONCE business management, and 2) with intense customer focus by taking customer feedback quickly and creating, growing, or phasing out business units, customer segments or products and service based on the feedback.

This style of management applied to all levels results in several Business Agility

patterns.

Business Agility Patterns

One of most important document tracking business trends, the Deloitte's Human Capital Trends in 2016, which, is becoming a landmark document to establish new business directions, says that - finally, org design is the most important trend among executives worldwide [Deloitte2016]. The report also says that culture is becoming a global trend, etc.

In fact, just about everyone that pays lip service to Business Agility says org design is important, but ... how do we do it exactly? What are the patterns to build organizations with Business Agility?

From all the successful case studies analyzed, we have compiled a set of patterns to can help us grow or transform companies into Agile companies with higher Business Agility:

- 1) **All-At-Once Management** (from Enterprise Scrum) – because things change very fast, it is possible nor desirable to use phase-based approaches any more.
- 2) **Preserved Agile Growth** - don't lose agility as you grow
- 3) **Accelerated Innovation** – for startups or new business units within a large company
- 4) **Radical Decoupling**, which calls for the most decentralization, and decoupling as possible among anything we manage.
- 5) **VC Layers**. Layered funding of bets: Business -> Business Units -> Customer Segments -> Value Props
- 6) **Customer Segment Team (Mirror Pattern)**, Mirror customer segment with all-at-once organizational structures that own value delivery and customer segment experience
- 7) **Agile Unified Portfolio Management** – instead of managing a portfolio of just established or existing business (mostly Red Ocean); unify and integrate 1) new business (both Blue and Red Ocean), and 2) and the upcoming business queue; for portfolio management at all levels: business units, customer segments, value propositions.
- 8) **Triangle** - When digitization is added, use Triangle: 1-2-1-2-1 correspondence of CS - CST - App (self-serving or internal)
- 9) **Inverse Competition Pattern** – for all products and services, and in general value propositions, that customers requested from companies, have instead companies compete for the customer's business.

We have also some general principles:

- 1) Wave Principle (from Enterprise Scrum) - coevolution of multiple Surfers, generalization of Iterative-Incremental Principle (which typically people use it to evolve one thing).

2) Cumulative Principle (from Enterprise Scrum)

All-at-once Management

With a higher rate of change – as discussed in the introduction, change, feedback, innovation and adaptation can happen anywhere in the product or service lifecycle. So it is no longer possible to use the traditional phase-based “business waterfall”, (strategy, marketing, product development, or customer feedback) to deploy new business models in which every aspect may change -- we must then be willing to inspect and adapt everything at once in order to better serve our customers and achieve our goals.

In other words, the world changes too fast, so the times when we could use a phase-based or stage-based organization to deploy BPPS are gone.

The phase-based approach is simply too slow, unresponsive, error-prone and inefficient for the competitive world in which we live today because it assumes erroneously a closed-information system.

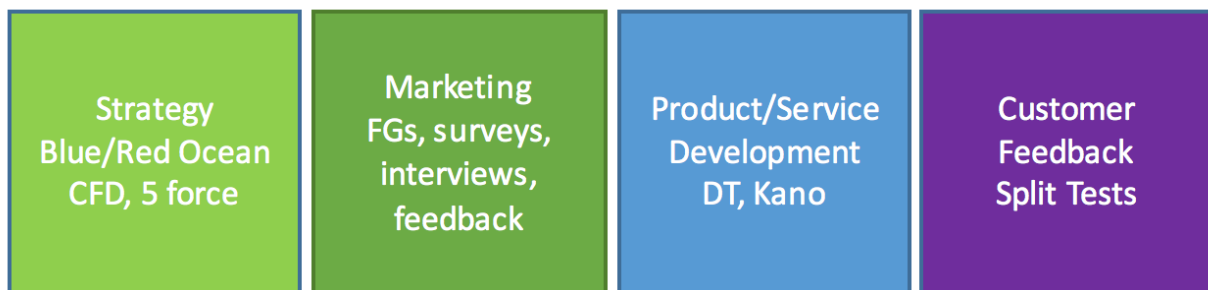
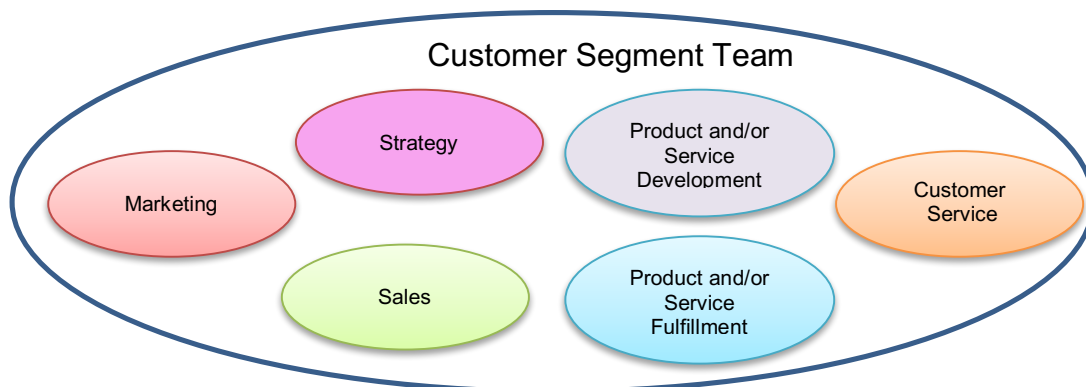


Figure 21. Business Waterfall

Therefore, in the 21st Century, we prefer all-at-once process models like Lean Startup, that include strategy, marketing, product and service development and customer feedback where adjustments can be made on everything as make progress taking advantage of all the feedback from any process area, and where we deploy entire business models – not only products or services [BusinessModelGeneration].



Preserved Agile Growth

As an interesting coincidence, I found that when a company organizes by customer segment, the style of management and structures that were used when the company was a startup, can remain now as components of the larger company.

When a company is a startup, it typically has an all-at-once business model creation, testing and evolution ala Lean Startup, because in startups, we don't do phase-based Sloan type management, we do all-at-once business model creation, evolution and testing (iterations of strategy, marketing, product/service development, feedback) ala Lean Startup. However, when a company grows, we tend to change our management style and use either functions, processes, or product/service management. But instead, if as we grow we keep the customer focus and adapt appropriately, we can keep our startup structures almost intact, and just grow them if needed. In fact, we can add more products and services to a customer segment, add more Customer Segments, or even add new Business Units and still keep the all-at-once structures as we grow with customer focus.

Following the basic premise explained above, let's follow the evolution of a company that always remains managed all-at-once agile and customer focused:

- Startup. Single product or service with a single Customer Segment in Blue or Red Ocean. The product or service is on a *trial basis*. The Customer Segment
- Startup or company with multiple products and services for a single Customer Segment.
 - Each of the products and services remain on a Lean Startup *trial basis*: create or evolve, test, and grow, change or phase out based on feedback.
- Company with multiple Customer Segments
 - If the market of a company for a product or service is segmented or diversified; we need different Scrum Teams for every customer segment or different products and service business model. For example, an investment advisor may have retail, high-net-worth, and institutional investors.
 - Each customer segment remains on a *trial basis*: create or evolve, test, and grow, change or cancel
- Company with multiple business units (each potentially with multiple Customer Segments).
 - Each business unit remains on a *trial basis*: create or evolve, test, and grow, change or cancel

The advantage of this model is that everything is managed on a *trial basis with a Customer Segment focus*, so we already acknowledged that if something is not working, we are going to phase it out, without “bleeding to death”, as so many companies end

up. Each level can operate in autonomous or nearly autonomous mode; and therefore, it is easier to add or remove different offerings as needed.

At any level, the supporting processes may benefit by digitization: enabling applications and systems, to agilize any of the core or supporting processes. It is ideal that the enabling applications are as decoupled as possible, so that in case we need to phase out a business unit, a customer segment or a product or service, we can do so with ease. On the other hand, it is also ideal not to have too much “redundant infrastructure”. We will talk more about this in the Application Portfolio chapter.

This is what we call the **Preserved Agile Growth Pattern**:

In Enterprise Scrum, at any one point in the growth of a company, the company must remain agile through the management of products and services (value propositions), customer segments, and business units.

This can only be accomplished through the **Radical Autonomy Pattern**, which says:

In Enterprise Scrum business management, we prefer to decentralization, and decouple anything we manage as much as possible.

For a company to truly be responsive, and don't waste time in phases, we need to use the **All-at-Once Business Management pattern** (as opposed to phase-based), which says that:

In Enterprise Scrum we prefer to do all-at-once management because since anything can change at any one point in time i.e. strategy, marketing, business model, regulatory, technology, customer needs/desires, etc.; it is best to take feedback from all change sources at once. This also gives us “integrated feedback” from the customer overall experience.

Since everything is “an experiment” and can change rapidly, we must be able to adapt not only what is already “existing business”, but what is experimental (disruptive Blue Ocean experiments), and what is already “standard competitive business”. All must be managed at all times; therefore we prefer to manage an **Integrated and Unified Portfolio**.

The **Integrated and Unified Portfolio Pattern** says that everything we do: 1) new upcoming in the queue (Blue and Red Ocean), 2) and existing disruptive experimental, and 3) competitive portfolio of, mostly Red Ocean, management; must be done at all levels: business units, customer segments, value propositions

When digitization is added, the Mirror Pattern can be replaced by the Triangle Pattern.

The **Triangle Pattern** says that there is a correspondence among the Customer Segment, the Customer Segment Team and the self-serving or internal application that

manages the overall customer experience.

Customer Segment Team - Mirror

Steve Denning, the author of Radical Management, and a past board member at the Scrum Alliance, says that realizing that the customer is “the center of the business Universe in the 21st Century” is a Copernican-level idea – it will revolutionize the way we do business. Once companies realize the benefits of “optimizing by Customer”, they will never go back to see the world in a different way.

So from now on, we expect more and more companies to be organized around their customers – not by functional silos, or business processes, or products and services.

We could use other management methods or processes that provide rapid feedback to implement a customer-driven all-at-once business management model; but Enterprise Scrum makes it easier to implement this type of all-at-once customer-driven management. Also, Enterprise Scrum can work in a variety of organizational forms from a layered hierarchy of Enterprise Scrum teams to a networked mesh of Enterprise Scrum teams with inter-dependencies.

An almost perfect example of this type of management are Squads at Spotify, or the way Salesforce.com manages its entire company.

"We predict that one-third of companies in the B2C space will begin changing their business structure to get closer to the customer and effectively compete on the basis of experiences. These companies will move from functional-siloed organizations that prioritize efficiency and control to customer-driven matrix structures that also leverage shared functions to protect margin." [Forrester2017]

Or better yet, organizational structure must Mirror the Customer Segments (see the Mirror pattern below). This is part of what we teach in our Enterprise Scrum classes, but "Customer-Driven" organizations are not the only thing that is needed, in addition to that, these teams must be:

- agile-managed, adapting quickly
- in subsumption with the customer's world, immediate feedback from marketing, sales, fulfillment, customer satisfaction OODA loops

- all-encompassing CX, full experience across all interactions
- all-at-once
- doing what delivers the most Business Value for a customer, out of a possible more comprehensive Value List
- keeping simultaneously employees engaged, happy if possible
- making profits (and cutting costs)

So the Customer Segment Team is an Enterprise Scrum team (scaled or not scaled), using the Enterprise Scrum - Business Model Canvas.

Based on this, we postulate the **Mirror Pattern**:

If you are in a context where you must quickly adapt to Customer demands, manage your business by organizing through Enterprise Scrum Customer Segment Teams that mirror your Customers.

We define a **Customer Segment Team**, a teams that includes ALL of these areas combined: strategy, marketing, sales, operations, and customer service; and that takes customer feedback and as a whole and chooses future direction. The Customer Segment Team has a **Business Owner**, as described in the Enterprise Scrum Overview or in Enterprise Scrum Definition. The Customer Segment Team should be at the very least a proxy **Customer Segment Team** – a team that is composed by the functional area teams' Business Owners that takes the feedback and makes decisions *as a whole*.

Satisfy your customers, and good things will happen to your company:

Satisfied customers become promoters of your company.
Satisfied customers are returning customers.
Satisfied customers do not post negative social media articles.
Satisfied customers do not deter others from becoming your customers.
Satisfied customers are more likely to give your company feedback to become even better.

This allows customer-centric management, where the organization is listening to all interactions with the customer across all areas of interaction. This organization also allows the company not only to “try new things out” easier, but to manage and phase out things as needed: business units, customer segments or products and services; but always keeping an Agile all-at-once customer focus, as described above. This customer-centric Agile management is key to manage things in a more Agile way in the 21st Century as companies grow.

In other words, what we present here is a way to do company management – including middle management, while keeping customer focus and allowing accelerated innovation, testing of business ideas, and management of portfolios at all levels in an Agile customer-centric way.

With satisfied customers, companies grow, flourish, get feedback from customers easier and can improve. Companies that manage for customer satisfaction find a purpose easier, have better customer relationships, customer references, customer reviews, word of mouth recommendations, analyst reviews, website feedback, net promoter scores and social media scores. With higher customer satisfaction, companies avoid sour relationships, losing accounts, and find it harder to get beat by competitors.

In addition, companies that make their customers happier by delivering more business value to them, tend to have happier employees, that are:

- more results oriented
- more innovative
- get better at learning by accepting there is “always a way to improve”
- more service and customer oriented
- proud of who they are and what they do
- take more ownership
- etc.

Each of the customer segments then has a Business Owner, that manages a Value List for the *overall customer satisfaction of the customers*. This is important, because that’s how businesses start, grow, improve and prosper – by better satisfying customers, and using the satisfied customers’ feedback to improve and get other customers.

Supporting Processes

Supporting Processes can move to the right level as the company evolves. When we start with a startup, all the supporting processes are at the startup level; but later as we add more product or services, customer segments or even more business units, the supporting processes can move to the right level to support a larger company.

This is just an example of how the Supporting Processes can move – different companies may choose to do things differently.

We start with a Startup, a business, that has one customer segment within a business unit. The startup has to provide all of the functions that a large company does.

Startup - Product/Service development, management and evolution:

Supporting: business model development, product/service development, software development

Supporting: **Customer Segment Team**: strategy, marketing, sales, operations, and customer service

Supporting: legal, compliance

Supporting: finance, accounting, payroll, HR

Warning: avoid creating functional departments for these areas: marketing, sales,

operations, and customer service. Instead, have a single technical team, and a single **Customer Segment Team**. Some of the concepts as to who decides can be addressed with the Management 3.0 concepts [Management3.0], for example the delegation hierarchy.

As the company adds more products and services that are used by a single customer segment, then some of these services are promoted to the customer segment level.

In other words, the supporting services are now used by several products and services.

Customer Segment with Product and Service portfolio.

PROMOTED to the Customer Segment level:

Supporting: **Customer Segment Team**

Supporting: legal, compliance

Supporting: finance, accounting, payroll, HR

Warning: remember, if you can avoid “functional departments” for these areas: marketing, sales, operations, and customer service. It is less dangerous to have functional departments for the other areas.

As we add more customer segments, we end up with a customer segment portfolio, but then some of the services must move up to the business unit level – even if there is only one business unit.

Business Model with Customer Segments portfolio

PROMOTED Business Unit:

Supporting: finance, accounting, payroll, HR

BU portfolio – Venture Capital firm

PROMOTED to Business Unit portfolio:

Supporting: finance, accounting, payroll

WARNING: Notice the **Customer Segment Team** representing these areas: marketing, sales, operations, and customer service; is NEVER promoted past the customer segment level.

Enterprise Scrum Unified Portfolio and Metrics

The Enterprise Scrum Portfolio Matrix tracks the performance of products and services, customer segments or business units through carefully chosen Portfolio Metrics to expedite decisions to:

- 1) Prioritize and queue new offerings
- 2) Deliver new offerings into operations
- 3) Test new offerings
- 4) Manage and grow the existing ones
- 5) Phase out the ones that are not working any more

The Portfolio Metrics consist of key metrics originated in Blue Ocean Strategy, Exponential Organization, Beyond Budgeting, BCG Matrix, and Anasoff Matrix, Beyond Budgeting that are used into a single portfolio management system, to manage in an agile way an entire company.

The Portfolio Metrics are the following:

MTP – Is it an MTP (Massive Transformative Purpose)? Typically, this indicates that there is a very large opportunity, as MTPs typically redefine whole industries e.g. Uber.

BOS Type – Is it a Blue ocean opportunity? Is this a new niche market? This could also be interesting, as the business model may represent a “niche market”. And as we say in startup management, “there are riches in niches”.

Anasoff Type - just to know what kind of business it is existing market and product, new market, new product or service in same market, diversified (both new market and products or services)

BCG Type – what type of business is it? Not proven – question mark, proven and growing – star, making a lot of money but growing slowly – cash cow, fading out – dog.

Status- operating, being developed, queued, on hold

Revenue – Cycle and cumulative revenue

Profit - Cycle and cumulative profit

Efficiency Ratio – Cycle and cumulative percentage of sales to expenses

Gross Profit Margin – Cycle and cumulative percentage of profit over revenue

Business Model Growth – how much the business area is growing per Cycle

Market Share – what percentage it holds right now on the market (sometimes we have to use yearly data instead of monthly or quarterly data on this)

Market Share Growth – what percentage of market share growth it achieved per Cycle (sometimes we have to use yearly data instead of monthly or quarterly data on this)

Market Growth – how much is the market growing per Cycle (sometimes we have to use yearly data instead of monthly or quarterly data on this)

In a company, in a startup as well as in a VC, we are looking to optimize the following *business-oriented metrics* for products and services, customer segments and business units:

- MTP opportunities, and if possible to be the first in that opportunity, but we should not be afraid to follow others into MTPs e.g. Lyft and Uber.
- Blue Ocean opportunities, is better to find a niche, even if the opportunity is not an MTP, rather than choosing to compete in a Red Ocean market, specially if it is already saturated.
- Don't care if it is existing business or new business as long as it is profitable (Anasoff Matrix)
- We are looking for rising stars, and specially “fast rising stars” (BCG Matrix)
- High profits
- Low efficiency ratios (low investment to revenue ratios)
- High profit margin
- High business growth

- High market share if possible, but it is not mandatory
- High market share growth, if possible but it is not mandatory
- High market growth, if possible we prefer to do business in expanding markets, but not mandatory

This portfolio management matrix can also be used for VC firms by adding a percentage of ownership/interest for the VC firm.

Enterprise Scrum Unified Portfolio Management

The **ES – UP (Enterprise Scrum Unified Portfolio)** gives visibility to ALL areas of interest simultaneously; therefore, only ONE matrix or unified portfolio is needed to manage an entire company with multiple business units, which may in turn have different customer segments with multiple products and services for each one of them.

Also, the same matrix can be used to track startups, medium size companies, or large diversified multi-nationals.

BU	Customer Segment	Product or Service	Status	CS	NPS	MTP BCG Anasoff BOS	Revenue	Cost	Profit	Profit Margin	Growth	Market Share	Market Share Growth	Market Growth
Bank	Commercial Bank	Check 1 Check 2 Check 3	Ongoing		9	Cow RED Existing	298M 12B	287.3 11.57B	10.7M 427M	3.6% 3.56%	-.1%	.03%	-.0005%	.9%
	Retail Bank	Check 1 Savings 1	Ongoing		9.2	Cow Red Existing	97M 5.5 B	93.61 5.35B	3.39M 150M	3.49% 2.7%	4.24 %	.02%	.001%	3.5%
CC	Retail Credit Card	Cc a Cc b Cc c	Ongoing		8.9	Star Red Diversify	11M 78M	5.5M 40M	5.5 M 38M	51.9% 49 %	2.8%	.0003%	- .00002%	3%
	Mutual Fund		Develop		8.5	Star Red Diversify	0	2M 15M	0	0	0	0	0	2%
Investment Adviser	Investment Advisor		Develop			Red Diversify	0	1.3M 8M	0	0	0	0	0	2.7% yearly
BD	Broker Dealer		ON HOLD			Star Red Diversify	0	0 0	0	0	0	0		14%
	Virtual Hedge Fund Trader		Queued 2016 Q3			MTP Star BLUE Diversify	0	0 0	0	0	0	0		> 100%

Business Unit of Customer Segment Portfolio

We can also zoom at an entire business unit. We provided the ES – BMC to manage this level in the Company Management chapter.

So this view provides a way to look at the performance or status of a single business unit. For example, the following view is highlighting the bank business unit:

BU	Customer Segment	Product or Service	Status	CS	NPS	MTP BCG Anasoff BOS	Revenue	Cost	Profit	Profit Margin	Growth	Market Share	Market Share Growth	Market Growth
----	------------------	--------------------	--------	----	-----	---------------------	---------	------	--------	---------------	--------	--------------	---------------------	---------------

Bank	Commercial Bank	Check 1 Check 2 Check 3	Ongoing		9	Cow RED Existing	298M 12B	287.3 11.57B	10.7M 427M	3.6% 3.56%	-.1%	.03%	-.0005%	.9%
	Retail Bank	Check 1 Savings 1	Ongoing		9.2	Cow Red Existing	97M 5.5 B	93.61 5.35B	3.39M 150M	3.49% 2.7%	4.24 %	.02%	.001%	3.5%
CC	Retail Credit Card	Cc a Cc b Cc c	Ongoing		8.9	Star Red Diversify	11M 78M	5.5M 40M	5.5 M 38M	51.9% 49 %	2.8%	.0003%	- .00002%	3%
	Mutual Fund		Develop		8.5	Star Red Diversify	0	2M 15M	0	0	0	0	0	2%
Investment Adviser	Investment Adviser		Develop			Red Diversify	0	1.3M 8M	0	0	0	0	0	2.7% yearly
BD	Broker Dealer		ON HOLD			Star Red Diversify	0	0 0	0	0	0	0		14%
	Virtual Hedge Fund Trader		Queued 2016 Q3			MTP Star BLUE Diversify	0	0 0	0	0	0	0		> 100%

Figure 23. Business Unit View

Customer Segment or Product and Service Portfolio

We will assume that we are using the customer-centric organizational constructs as discussed in the Company Management chapter. As discussed before, the only way to do this is by having integrated Customer Management across all customer-facing functions through a **Customer Segment Team**. However, since we may have more than one business unit, we must take a partial view of the ES – UP to manage Customer Segment Portfolios which are, by definition inside a business unit.

We can also zoom to the Product and Service Portfolio level for a single Customer Segment:

BU	Customer Segment	Product or Service	Status	CS	NPS	MTP BCG Anasoff BOS	Revenue	Cost	Profit	Profit Margin	Growth	Market Share	Market Share Growth	Market Growth
Bank	Commercial Bank	Check 1 Check 2 Check 3	Ongoing		9	Cow RED Existing	298M 12B	287.3 11.57B	10.7M 427M	3.6% 3.56%	-.1%	.03%	-.0005%	.9%
	Retail Bank	Check 1 Savings 1	Ongoing		9.2	Cow Red Existing	97M 5.5 B	93.61 5.35B	3.39M 150M	3.49% 2.7%	4.24 %	.02%	.001%	3.5%
CC	Retail Credit Card	Cc a Cc b Cc c	Ongoing		8.9	Star Red Diversify	11M 78M	5.5M 40M	5.5 M 38M	51.9% 49 %	2.8%	.0003%	- .00002%	3%
	Mutual Fund		Develop		8.5	Star Red Diversify	0	2M 15M	0	0	0	0	0	2%
Investment Adviser	Investment Adviser		Develop			Red Diversify	0	1.3M 8M	0	0	0	0	0	2.7% yearly
BD	Broker Dealer		ON HOLD			Star Red Diversify	0	0 0	0	0	0	0		14%
	Virtual Hedge Fund Trader		Queued 2016 Q3			MTP Star BLUE Diversify	0	0 0	0	0	0	0		> 100%

Figure 24. Customer Segment or Product and Service Portfolio

Single Value Proposition (Product or Service)

If we only wanted to take a look at one value proposition, we can zoom in to the product or service level.

Accelerated Innovation

There are striking differences on what is thought to be the “best management” in the 20th Century, and what we now think is “the best management” for our times. There are new emerging Business Agility patterns and techniques.

In the 20th Century innovation was important, but we wanted to create long-term plateaus where we competed primarily on operational efficiency with competitors in a particular industry. Also, once a company gained “market leadership” and a “large market share”, companies did not need to change that fast – they could sit for years and years exploiting their cash cows. Therefore, companies in leadership positions, could afford much waste, and although arguably they were often organized for “bottom line” management – preventing “higher costs, in reality they were very inefficient as they tend to be managed through deep functional hierarchies. So long were these cycles, that some companies plans assumed their business models were “permanent” and did not need to adapt ever. This of course, cause the “cultural lock-in” effect, and with that the inability to change, innovate, and improve despite threats.

However, in the 21st century, we can’t make the same assumptions. Fast business cycles, product cycles, and industry and company transformations are the norm, and therefore companies must adapt and innovate fast to customer needs. They can’t afford to be organized in slower, less reactive ways; and therefore must be organized for extreme efficiency in all-at-once collaborative, cross-functional teams. They must also assume that anything they do must be adapted and tested continuously, and is on a “trial basis” – everything must be tested, get feedback on, and adapted accordingly, including letting it go!

As innovation crosses over the business model space, the opportunities are immense. Literally, there is an Uber for every industry and for every area [ExponentialOrganizations]. Enterprise Scrum, the framework, has that as a goal, to not only transform in the most agile way, but to deliver the most agile architecture through an agile framework, Enterprise Scrum, through the use of agile patterns and techniques.

4. Enterprise Scrum configuration

Notation:

// comment

// assignment
=

// function calling with parameters
//
// all parameters are by reference
//
function calling and list of parameters = foo[a, b, c, ...]

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

// list of
list of a's = (a)

// value of
value of parameter a = |a|

// relationships
// extends
A extends B = A inherits from B

// class B
// association, uses or contains A
//
// A is an attribute of B
//
// all associations are by reference.
B
{
 A
}

Enterprise Scrum parameters

EnterpriseScrum
{
 // -----
 // INSTANCE
 // -----
}

```

instanceName,
instanceType = <development, customization, operations, etc.>,

// -----
// TEMPLATE
// -----
// IF using a template COPY the template
// fields that are filled into THIS instance
// -----
template[template name],

// -----
// ETeam or teams
// -----
ETeam,

// -----
// CUSTOMER
// -----
customer,

// -----
// IA for DOMAIN
// -----
// information architecture → Cycles
informationArchitecture[customer],

// canvas must include VLITypes and surfers “aspects” of the VLIType
canvas = canvasFromIA[informationArchitecture],

// all VLITypes from canvas
VLITypes (VLIType) = VLITypesFromIA[canvas],

// ALL surfers for ALL VLITypes
// ALL IMPORTANT THINGS – determines extra VLI attributes!
surfers = surfers[(VLIType)],
vliExtraAttributes = getVLIExtraAtrributes[VLIType],

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

// -----

```

```

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

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

cycleTypes = getCycleTypes[process],
metricsForCycles = getMetricsForCycles[cycleTypes],
reportsForCycles = getReportsForCycles[cycleTypes],
calculationForCycles = getCalculationsForCycles[cycleTypes],

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

// -----
// DEBTS
// -----
debt
{
actionNotDone[
type =
<
technical,
refinement,
decision,
research,
monitoring,
change,
cultural maturity,
motivation,
learning
>,

```

```

description
}
debts(debt)

}

```

instanceName,
instanceType = <development, customization, operations, etc.>,

instance – describes the instance of Enterprise Scrum that we are either developing, using or describing by one or more teams. The hope that is that while people work on some specific domain and develop instances of Enterprise Scrum, that they reuse of some of these instances, either within their company, or even throughout an industry.

instance. instanceName - instanceName is the name of the Scrum instance. A unique identifier that defines what the teams that do this instance do, and is well aligned with the **Principal VLI-type** below. It uses the preformatted template: “The instance of Enterprise Scrum for <activity>”.

For example, there can be Enterprise Scrum instances for:

The instance of Enterprise Scrum for Startup Management
The instance of Enterprise Scrum for Marketing and Sales
The instance of Enterprise Scrum for Compliance Management
Etc.

One or more teams can implement the same or similar instances of Enterprise Scrum.

Q: What is the name of this instance of Enterprise Scrum?

instance. instanceType -- development, customization, growth, management i.e. operations, phase out, etc.

Sometimes is useful to know what is the type of work we are doing because this might tell us more about what to expect from our instance evolution as a general behavior, specially in terms of what the Cycles do.

Q: – what is the purpose of this instance?

template[template name] – allows you to use a template that may have different types of information pre-configured.

See the detailed description below under **Template Parameters**.

Q: – What is the name of the template we are using?

ESTeam – describes the Enterprise Scrum team and all the roles, members, skills, gaps trainings, states. Each Enterprise Scrum instance has ONE team only. The ESTeam configuration parameter also allows one to configure relationships, so a company could have very many teams working through relationships.

See the detailed description below under **ESTeam**.

Q: – What is the Enterprise Scrum working on this instance?

customer – describes who the customer is (are), industry, area in the industry, segment, and how to deliver Business Value.

See the detailed description below under **Customer**.

Q: – Who is the Customer in this instance?

```
// -----  
// IA for DOMAIN  
// -----  
// information architecture → Cycles  
informationArchitecture[customer],  
  
// canvas must include VLITypes and surfers “aspects” of the VLIType  
canvas = canvasFromIA[informationArchitecture],  
  
// all VLITypes from canvas  
VLITypes (VLIType) = VLITypesFromIA[canvas],  
  
// ALL surfers for ALL VLITypes  
// ALL IMPORTANT THINGS – determines extra VLI attributes!  
surfers = surfers[(VLIType)],  
vliExtraAttributes = getVLIExtraAttributes[VLIType],
```

Q: – What is the Information Architecture for this domain?

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

Q: – What is the knowledge for this instance?

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

Q: – What is the Value List for this instance?

```
// -----  
// PROCESS  
/ -----  
process [informationArchitecture],  
  
cycleTypes = getCycleTypes[process],  
metricsForCycles = getMetricsForCycles[cycleTypes],  
reportsForCycles = getReportsForCycles[cycleTypes],  
calculationForCycles = getCalculationsForCycles[cycleTypes],
```

Q: – What is the process for this instance?

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

Q: – What is the structure for this instance?

```
// -----  
// DEBT  
// -----  
debt  
{  
actionNotDone[  
type =  
<  
technical,
```

```
refinement,  
decision,  
research,  
monitoring,  
change,  
cultural maturity,  
motivation,  
learning  
>  
description  
}  
debts(debt)
```

Q: – What are the debts for this instance?

Template parameters

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

templates (template)

template – this parameter describes the template if any that we are using. If we are not using a template, this field should be blank or say “none”. The template could have one or more of the following:

- a configuration template,
- a canvas template,
- an Initial Value List process template,
- an Initial Value List template,
- a complete process including Initial Value List and Cycle(s) structure template,
- a cycle content template (through a repeatable VLI)

configurationTemplate – provides a configuration template for an instance, with the Information Architecture information already filled out:

- a pre-configured information architecture, that includes a canvas, VLITypes, surfers

(aspects of the , and extra attributes for the surfers.

- a pre-configured Initial Value List sequence – an initial set of a dependencies among surfers
- Cycles with parameters
- etc.

Q: what Enterprise Scrum configuration are we using?

canvasTemplate – provides the canvas used by this instance. This is the most important template, technically, the template would give enough information to run an instance, because from the canvas all of the other things could be derived. Nevertheless, it is also helpful to have other templates.

For example, we present throughout the book, and available online, canvases for:

SW

HW

Company Management

Startup Management

Marketing and Sales

Compliance

Q: Which canvas was used if any? i.e. Marketing and sales, HW, SW, compliance, etc.

initialValueListProcessTemplate Specify which Enterprise Scrum Instance Value List process template was used.

Another common template is one that already has an Initial Value List. This is useful for projects, or processes that are similar.

For example, Value Lists to do specific PeopleSoft, Oracle Financials or SAP installations may be available. This is typically done at the project or process level. These templates are called Enterprise Scrum Project/Process Template.

Q: which initial Value List process template was used if any?

initialValueListTemplate{name} Specify which Scrum Instance Value List template was used, or leave it blank.

For example, Value Lists to do specific PeopleSoft, Oracle Financials or SAP installations may be available. This is typically done at the project or process level. These templates are called Enterprise Scrum Project/Process Template.

Q: which initial Value List was used if any?

completeProcessTemplate{name} - Specify which Enterprise Scrum Process Template was used, or leave it blank. For example, there is a complete process template when we use single team, User Stories and Release Planning. By convention, the template values would be used but a highlight is used to indicate changes in the values of the parameters.

For example, templates for a specific type of Software Development, or for Company Management, or for Cancer Research may be available.

A database of such templates will be available in the future through participating Enterprise Scrum practitioners.

Q: which complete process template was used if any?

cycleTemplate{name} - Specify the Cycle template used or leave it blank. This is typically used for Scrum management with almost identical Cycles like Sales Management, or Compliance Management, where there might not be much new in the Cycle Value List, but of course it is still important to collaborate, share knowledge and improve.

Q: which cycle Value List was used if any?

templates (template) - an instance can use one or more templates. A complete instance could use all of the possible templates: process, valueList and Cycle types.

Q: which list of templates is available?

Enterprise Scrum Team Parameters

ESTeam

```
{  
name,
```

```
// business owner  
businessOwnerType=<single, many, distributed>,  
businessOwner[name],  
businessOwners(businessOwner),
```

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

```

// team
team,
skillMatrix,

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

// stakeholders
stakeHolders(stakeholder),

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

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

relationships(relationship),

}

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

member
{
name
skills(skill)
happiness
emotional maturity
participation = <full-time, part-time>
collaboration = <independent, friction diplomatic, jelled>
}

```

```

training
{
  name
  description
  level
}

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

  // technical skills
  // -> add ALL
  teamCurrentTechSkills[team]
  // -> add ALL
  teamDesiredTechSkills[team]
  technicalSkillsGAP[team]

  // social skills
  // -> add ALL
  teamCurrentSocialSkills[team]
  // -> add ALL
  teamSocialGaps[team]
  teamDesiredSocialSkills[team]

  // Trainings required from GAPs
  trainingCoachingRequired[team]

  // Trainings performed
  trainings(training)
}

```

ESTeam – the **ESTeam** parameter describes the Enterprise Scrum team:

ESTeam.businessOwner – name and contact info of the **businessOwner**

ESTeam.stakeHolders – name and contact info for each of the stakeholders

ESTeam.coach – the name and contact info of the Scrum Master

ESTeam.team – list of team members: names, contact info

ESTeam.xFunctionalSkillMatrix – it is useful to track and inventory the overall skills of the team. The overall skills needed are typically known but they can change over time, and therefore, this matrix may be dynamic.

The Vision that we created earlier should help us find a good **instanceName**, to recruit the right members for our **ESTeam (businessOwner, stakeHolders, coach and team)**, to better define or understand our **businessPurpose**, and most importantly to define what **businessValue** is.

Because Enterprise Scrum teams can be composed of multiple Scrum Teams, the team members at one level can be – but don't have to be, Business Owners for a lower level team. We will see that Scrum instances are related through other parameters as well like Structure Parameters, and the attributes of the **valueListItem**.

dependsOn – this is the list of all organization we depend on.

dependOnUs – this is the list of all organization that depend on us.

This means that there are one or more **valueListItems** in our team that are dependent on other **valueListItems** from other teams or one or more of their **valueListItems** are dependent on us. We can keep track of these dependencies through the **dependsOnVLI**s parameter in the **valueListItem** See Value List parameters.

In some cases, these dependencies can be almost static or can change on a daily basis. For example, advertising agencies sometimes produce websites that always depend on compliance and marketing approval, and where these interactions are well-established.

However, in large-scale software development, the relationships of the development teams can vary rapidly – some times daily. See also the **dependsOnVLI**s parameter in the Value List **valueListItem** parameter.

It is useful to use a dependency matrix to track these dependencies – specially if they vary rapidly.

	Team 1	Team 2	Team 3	Team 4
Team 1	N/A	X		X
Team 2	X	N/A		
Team 3	X		N/A	
Team 4			X	N/A

parent -- name of the Scrum instance that is the parent, or higher level management for the process. This only applies for the delegation and centralized options where there is a hierarchy or either power or knowledge.

A **parent ESTeam**, in delegation or centralization modes, see **structure** below, can add

one or more Scrum Team as **contributors** reporting

This indicates that there are one or more **valueListItems** in the Value List for this team that have been delegated by a different team. In other words, there are **valueListItems** that have a **parentVLI**. See the attributes in **valueListItem**.

For example, in a large software project where there is one business team delegating responsibility and making business decisions to the contributor delegated teams.

Typically, a parent aggregates a common metric from the children. For example, in large-scale software development, we typically aggregate the velocity of individual but cooperating teams as a “Global Velocity”. Or in sales, we aggregate in the parent the combined sales of independent teams, products, or services.

ESTeam. subsumption

Subsumption

```
{  
subsumptionLayer – defines the subsumption layer:  
cycleName. the name of the Cycle to be used.  
reality - the “reality” to be used for the layer. For example, in software is the “integrated and tested code” and the “customer feedback”.
```

Customer parameters

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

customer – describes the customer, customers or customer segment with some important attributes.

customer. multiplicity – tells us how many customer there are and if they are known. This corresponds to the following values:
single
manyDefined
customerSegment

Q: how many customers are we trying to serve from this instance of Enterprise Scrum? One? More than one but we know which ones? A customer segment?

customer. industry – this parameter describes the industry in which the Scrum team works. For example, insurance, financial transactions, DNA research, audit, compliance, rocket development, etc.

Q: What industry or industries does it involve?

customer. area – this parameter describes the area of the industry, for example, “SEC Financial Compliance” in compliance, or “mobile app development” in software development.

Q: What is the area of that industry above?

Customer. segment – this attribute describes the customer segment.

Q: what is the customer segment we are doing work for in our instance?

Customer. customerBusinessOwner – describes the customer’s business owner. The person that represents the customer.

Q: who is the person that represents the customer if any?

Business Value

businessValue

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

businessValue – describes what is business value for this instance.

businessValue. description – Scrum is a system to deliver the most business value in the shortest amount of time. Therefore, knowing what business value means to you is one of the most important things when using Scrum. This parameter defines what business value is for our Scrum instance and gives a list of variables to be optimized.

Q: what is the description of business value including the different “high level” important variables – just a list?

businessValue. Whys – these are the pains, gains and jobs of why this is a useful instance for the customer.

Q: what are the pains, gains and jobs that these instance offers the customer?

businessValue. optimizationVariables – the list of business variables to optimize together. The optimizationVariables should be associated with a **metric** if at all possible, so that we can truly track quantitatively if at all possible. It is not always possible, to track everything quantitatively, unfortunately.

For example, an optimization variable could be Customer Satisfaction, which then could have a NPS **metric** associated with it.

Q: what are the detail optimization variables that we will choose to optimize for this instance?

businessValue. optimizationVariableToMetricMap – this is the entire map of **optimizationVariables** to **metrics**.

For example, for software development, one could have:

Optimization Variable	Metrics	Unit
Customer Satisfaction	Customer Satisfaction evaluation results	Scale from 1-10
Productivity	Velocity	User story points
Employee Happiness	eNPS	1-10

Q: what is the mapping between the optimization variables and metrics to measure them?

businessValue. optimizationGoals – describe what type or range of equilibriums we want to achieve.

For example, we could be optimizing future sales of a commercial software product; therefore, **businessValue**, would simply to maximize profit. This is where Scrum stands: one single productivity metric to optimize, velocity (officially), and not much else. Although, some of us in the trenches have always looked to optimize other things, like customer satisfaction, employee happiness, etc.

However, in Enterprise Scrum it could be more complex than that. For example, **businessValue** in software development could be a balance or compromise of many variables like:

- maximizing profit (or keeping it above a certain value),
- minimizing cost (or keeping it below a certain value)
- maximizing customer satisfaction (or keeping it above a certain value)
- maximizing employee happiness (or keeping it above a certain value)

Where all values would come from a metric.

The optimizationGoals can link to a **technique** or a **techniqueList**, to optimize our **businessValue**. A technique tells us exactly how to complement Scrum to optimize businessValue or some other helpful thing.

Q: how do we want to balance the optimization variables through their metrics for this instance? It could be dynamic depending on context.

Information Architecture parameters

informationArchitecture

{

// -----

// **DELIVERABLES**

// -----

//

// **what is delivered by the ETeam?**

//

deliverable,

deliverables(derivable),

// -----

// **VLI Types**

// -----

//

// **get VLITypes from deliverables 1-to-1**

//

VLITypes = VLITypesFromDeliverables[deliverables],

// **what VLIType delivers most of the Business Value**

//

principalVLIType = [VLI-type1]

// -----

// **SURFERS**

// -----

// **what are the aspects of a VLI?**

```

VLI surfers = surferfromVLIs [VLIType],

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

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

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

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

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

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

// -----
// C8W Technique
// -----
deliverables = deliverablesFromC8W[]
VLITypes = VLITypesFromC8W[]
Surfers = surfersFromC8W[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(architectureModel)

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

}
```

The Value List (or Backlog), is formed by a list of VLIs (**valueListItems** or PBIs, product backlog items). Each VLI (PBI), is supposed to add a granule or quantum of business value by definition, or have another VLI require a particular VLI to add business value.

However, each VLI represents several important views or aspects of work, and the information of carried by the VLIs has relationships and meanings that define the type of work that we are doing.

Value List. deliverable

deliverable – an outcome of the activity. For example, in software development, a product increment for every release. In compliance management, the data of an internal control, a policy and procedure. In executive management, it could be a documented decision. Etc.

Value List. deliverables

deliverables – a list of all the deliverables for the activity. For example, for software development, the list of everything delivered:

- infrastructure
- product increment
- documents
- support service
- etc.

Value List. principalVLIType

principalVLIType == |VLI-type 1|

For example, in software development, the principal type of work is a feature, including all the things needed to get it DONE at once:

- functional requirements,
- plan
- design or architecture

- the actual product

because that's how we primarily deliver Business Value to our users.

Another example, is a Real-State property, it includes:

- the property itself
- the plan to sell the property
- the legal contracts
- the inspection
- the valuation
- the website or advertising related to it
- etc.

VLI cut across multiple information architecture views. This is what we call the VLI to surfer mapping above.

Typically, we have a principal activity in anything we want to agilize, but we may want to get the values for all the VLI types before we start any work.

architectureType – this parameter describes the type of architecture or the team is working on. This reflects which concepts we work on and what relationships may have. The architecture is related to the **industry**, **area** and the **instanceName**, but here we describe the kind of abstractions that we deal in that domain from the perspective of what we are building: a system, a building, a business, etc. So the for example, the architecture can be: <“business architecture”, “system architecture”, “product architecture”, “compliance architecture”, “executive decision architecture”, “asset management architecture”, “portfolio management architecture”, etc.>

This does not imply that we should use “architectural models”. In fact, we prefer not to use any architecturalModels, because when doing Scrum, according to the subsumption concepts, “reality is the model”.

surfer – an important view of the information architecture for an interested party.

For example, in software development these are the views that are interesting for the corresponding parties (one or more):

- Purpose, interested party: everyone
- Requirements, interested party: Customer, Business Owner, Team
- Software Architecture, interested party: Team
- Plan, interested party: everyone
- Infrastructure, interested party: Team, Business Owner, Customer
- Team, interested party: Coach, Business Owner
- Etc.

The interested parties are important to state because later when we do Cycles, these are the parties that will Plan, Collaborate, Review and Improve things on them.

surfers(surfer) – the list of all of all important things for the instance.

You can ask the question: what are all the important things that all people care about and work on in this activity? That would be the list of surfers.

But to be clear, a surfer is not a deliverable, is just an aspect or view of a deliverable. For example, requirements or architecture for software development.

C8WtoSurfersMap – this is a tool to derive the surfers – the important aspects of what we deliver (if you can't immediately tell them).

The C8W model is the customer plus 8 questions:

- customer (for who)
- why
- what
- how to deliver
- how to implement
- who
- when
- how much
- where

Each C8W question maps to one or more surfers. For example, in software development the different views could be:

- customer (for who): one or more customers, or customer segments e.g. travelers by airplane
- why: purpose, e.g. travel agency with cost savings by daily monitoring of price changes
- what: requirements e.g. the list of features for the travel agency
- how to deliver: cloud, enterprise software, packaged software, etc.
- how to implement: infrastructure, development environment with specific technologies, logical architecture, physical architecture.
- who: who will do the work, which teams, or teams e.g. software development, infrastructure team, etc.
- when: plan with schedule, plan, effort per VLI
- how much: plan with budget, cost per VLI
- where: trivial, what locations e.g. same location for all teams

surferDepenenceMap – a dependency graph of the surfers among the VLITypes.

If -> == depends on

For example, in software development:

requirements -> customer

architecture -> requirements
infrastructure -> requirements

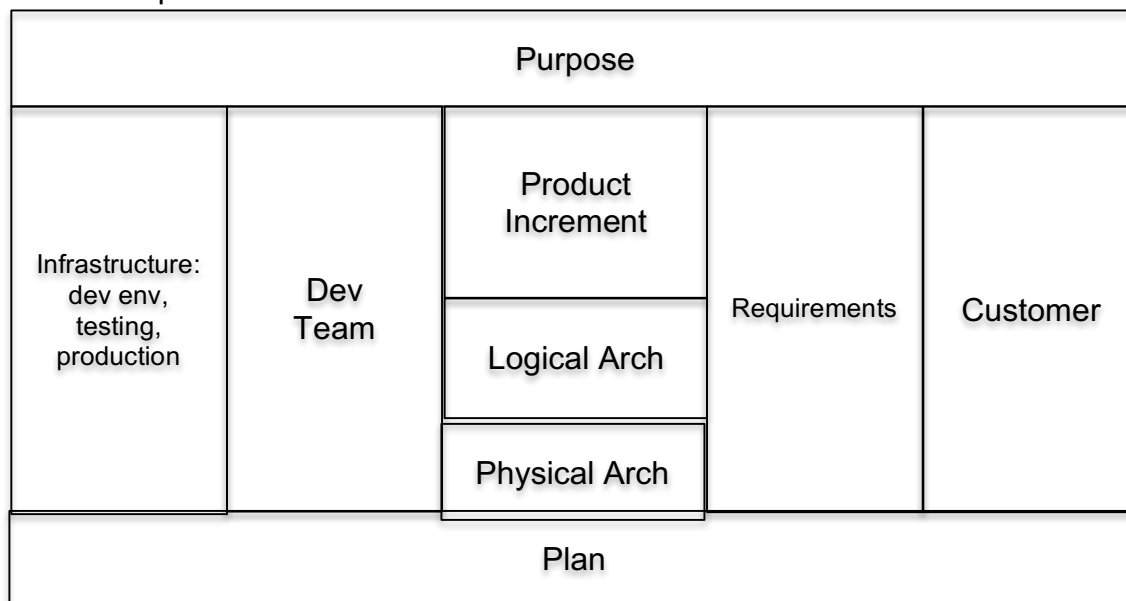
database -> business logic
business logic -> UX Design
UX Design -> architecture

team -> requirements AND infrastructure AND architecture (if team is variable)

plan (cost and schedule) -> requirements AND infrastructure AND architecture AND team

canvas – a picture that includes all VLITypes as boxes in the canvas, and that can account for all the surfers for each VLIType. The canvas is a diagram of the flatten dependency graph starting with the highest or root dependency on the right (typically the customer).

For example:



Basically this is a graphic of how ALL the important things for different viewers “travel together” coevolving with each other. We will coevolve all of this views together through **cycles**.

VLtoSurfersMap (surfers) – a map showing how to map the different important aspects or views of all-at-once VLIs, value list items, that deliver business value to a customer. This is what tells us what attributes each of the different VLIs should have – because they depend on the surfer mapping.

Essentially, we derived the VLIs and the Surfers independently:

- 1) what are the things we deliver incrementally? i.e. the VLIs, and
- 2) what are the important aspects we are interested from different perspectives? i.e. the Surfers

So this maps addresses which VLIs deliver which important aspects or let do certain people do work.

Different VLI types will have different attributes. For example, in software development a feature VLI will have different attributes than an infrastructure VLI, or a development environment VLI.

Here is an example of the **VLIToSurfersMap (surfers)** map:

product increment VLI – plan, requirements, physical and logical architecture
infrastructure VLI – plan, physical architecture
document VLI – plan, requirements

Some surfers are shared among all VLITypes like “plan or requirements”, but some are local to a VLIType like “logical architecture” or “physical architecture”.

For example, description, priority, size (effort), size (cost), are shared among different VLITypes, but domain-specific attributes like “components” would only apply within ONE VLIType.

VLIAAttributesToSurferMap – finally, we map of which surfer attributes should be carried as VLI attributes for all VLITypes, or for a VLIType.

The typical maps, for example, for software development, always start with the common fields among all VLIs:

- Priority - plan
- Definition - requirements
- DOR -
- DOD - requirements
- Effort
- Architectural Pattern
- Etc.

This is a good start to think about the extra attributes needed for each VLIType (see below).

This map is optional, because we may want to wait until we start defining the VLIs to add more attributes, and therefore, this is not as important to do up-front.

InitialValueListprocess – now that we have the dependency graph of the surfers,

which gave us the **canvas**, it is also now possible to derive the Initial Value List process, and the contents of the **cycle's review** and **improve**, because the Initial Value List process should follow this dependency graph.

So for example, for software development, reading the canvas
<HERE>

architectureModel – describes an architecture model. For example, a set of interconnected patterns, a set of UML diagrams, information dependency mapping, etc.; for whatever activity we may be interested in doing. Architecture models are useful some times, but they are optional.

architecturalModels(architectureModel) – this is a list of all architectural models

architectureManagementType – this parameter describes how we evolve the architecture: <self-organizing, Agile Architect, Traditional Architect, External Architecture Team, etc.> Some of these options are more “agile” than others of course. In doing Scrum, we much prefer that the architecture is managed by the entire team.

architect – this could be “none” or it can name a person that is doing the **architecturalManagementType** above. In doing Scrum we much prefer to do self-organizing architectural management but sometimes that is not possible.

VLITypes

PBI

```
{  
  name,  
  description,  
  priority,  
  size  
}
```

VLIType extends PBI

```
{  
  prioritizable  
  orderingTechnique  
  sizeable  
  sizingUnits(sizingUnit[metric])  
  sizeUncertainty  
  dependentVLIs  
  DORStandard  
  DODStandard  
  DeliverableToVLIsMap
```

cumulativeDOD

surfers(surfer)
}

VLITypes (VLIType)

VLIType

VLIType – an increment of a deliverable. For example, a feature in software development. A document or internal control in compliance management. A decision in executive management. A property sold in real-state management.

VLIType. prioritizable.

Prioritizable - Yes or No.

There are some situations where the Value List may not be possible or desirable to prioritize in terms of business value. For example, in a sales process, some real state properties may not have a higher priority than some other ones, or we may choose not to give them a priority. The same is true for a new car sales process or to a customer service process.

VLIType. orderingTechnique.

orderingTechnique - Name and description of the technique that we use to order by priority our Value List.

In Scrum the default prioritization technique is a comparison in the relative Business Value to be delivered by each PBI.

However, in Enterprise Scrum we have specified what business value is and the variables to optimize in our Scrum instance. Therefore, these **orderingTechnique** tells us exactly how to use these variables in our Value List to order by priority.

VLIType. sizeable

sizeable - specify if the principal Value List Item can be sized.

It is much preferred to do Scrum when we can measure things, but it is also possible to do Scrum without even making a single measurement. If this is the case the benefits of doing Scrum will not be quantified, and we say that we are doing “**non-quantifiable Scrum**”. Notice there is also a **quantitative** attribute on a Cycle – both are needed because even if the VLI is **sizeable**, we may choose to do a **quantitative** Cycle.

sizingUnits

VLIType. sizingUnit{metric}

sizing unit - specify what units to use to size

This sizing unit must be related to a **metric** to the **principalVLIType** within the **Cycle**. The **Cycle** can be used to improve one or more **metrics** such as Customer Satisfaction, Productivity, Quality, Performance, Learning, Accomplishment, Building, Profit, Adaptation, Transformation, Improvement or Growth **CSPQ- PLABPATIG**.

For example, a Sales process may size its backlog by revenue or profit not necessarily by effort.

VLIType. sizingUnits(sizingUnit)

sizingUnits – is the list of all sizingUnits. This applies to both 1) different VLIs that require different sizing units, or 2) a complex VLI that requires 2 sizing units; for example, an activity that requires both effort and cost.

VLIType. sizeUncertainty.

size uncertainty - estimated average percentage of uncertainty in a VLI type.

Every VLI is uncertain but some VLIs are more uncertain than others. If we can determine where most uncertainty concentrates among our VLIs, we can plan more carefully.

For example, in research and development, if we chose “effort” in some unit of time as our sizing unit, we may determine that our effort estimates could have a standard deviation of 100%. Ideally, we would measure this uncertainty from previous experience, but this uncertainty could be estimated. In the case where we really don’t have any idea of the uncertainty to be expected we could leave this value as UNDETERMINED, and then measure the uncertainty later as work is performed.

VLIType. dependentVLIs

dependentVLIs - Yes or No

Value List Items are sometimes dependent on each other. Therefore, we must specify if the components are dependent or independent for each VLI or VLI type.

For example, if we are monitoring internal controls and each control represents a VLI, the controls are typically independent from each other. In hardware development, there are typically many dependencies among components, but some components do not interact with each other.

The Daily Scrums are still very useful even when we have independent VLIs, because team members can still help each other, collaborating, picking up work from other team members, and exchanging knowledge and experience.

VLIType. DORStandard

DORStandard - this is the DOR standard for a VLI type but it can be overridden with a more specific DOR for each VLI.

We should provide a DORStandard for every VLI type if possible, because this determines a standard by which all VLIs of that type are ready to be worked on.

VLIType. DODStandard

DODStandard - list is the DODStandard for a VLI type but it can be overridden with a more specific DOD for a specific VLI.

Sometimes, we can assign DODStandards to VLI types. This is useful. But we need to be careful – each VLI must satisfy its own DOD which may have to be unique.

VLIType. DeliverableToVLIsMap.

DeliverableToVLIsMap -- define what type of components will the VLI types may result into.

To be clear, we make a distinction between the goal – VLIs, and the thing that was delivered to fulfill the goal Increment Component(s).

For example, in software development our goal is to develop a feature for a user; but we deliver a product increment or component – the software component of the Increment that fulfills that goal. Therefore, in software development the **DeliverableToVLIsMap** is a table that maps the Product Increments to functional requirements.

In software development, we are so used to deliver either a Product Increment, or a set of components that sometimes we forget this mapping could be important.

Let's give another example -- a financial company that needs to be compliant with the Investment Company's Act. Its goals for a Compliance Cycle, may be to be compliant with some or all of the appropriate SEC's statues in the Investment Company Act.

As part of this goal, it writes policies and procedures, defines and executes internal controls to prove that it is compliant with the policies and procedures, and provides activity, testing and other compliance reports.

The VLI is: Monthly Compliance with the SEC Investment Company Act Iteration
The Increment Components to achieve this goal together with its acceptance criteria are:

- **the compliant processes** in the organization as measured by the set of internal controls, which represent its acceptance criteria
- **each of the internal controls** with its acceptance criteria
- **the internal document titled “Investment Company Act Policies and Procedures”** that outlines the compliance policies and procedures to be enacted

by the operations' processes, which map to the internal controls described above. The acceptance criteria of the internal documents is that they allow the organization to be compliant with the Investment Company Act.

- **the Internal Controls Monthly report**, with the acceptance criteria of having passed a certain percentage or list of mandatory regulatory controls.
- etc.

Surfers

Surfers – list of **surfers** for the VLI i.e. aspects of the VLI.

cumulativeDOD – **cumulativeDOD** is a parameter that let's us track whether the DOD for that VLType needs to be passed in a cumulative way or not.

For example, in software development is customary to pass DOD for a PI (product increment) *cumulatively in regression for the all of the software product that was accumulated through product increments before.*

However, in Enterprise Scrum we have made it optional depending on context.

VLITypes – **VLITypes** is the list of VLI types that the Scrum instance supports.

VLIs are always delivered through “things you can touch” – and they may have one or more interesting views i.e. the surfers, as we discussed above.

These are the different types of work that we are planning to do with our instance of Scrum through Cycles and deliver them as Business Value. For example, in an instance for software development, we could do infrastructure VLIs, architecture components VLIs, implementation of functional requirements VLIs, etc.

Enterprise Scrum can handle any type of mix of work, but we prefer to concentrate on a specific kind of work. We usually refer to this as the “color mix”, or “colorization” of the Value List (Backlog) i.e. the types of allowed VLIs.

Examples of VLIs:

Business Process	VLIs
Software Development	Software Requirements, Infrastructure (Testing, Integration, etc.), Technical Components.
Hardware Development	Hardware Features, Acquisition, Infrastructure (Testing, Integration), functional requirements

Compliance Management	Deployed Business Processes with new Compliance Standards, Compliance Documents, Policies and Procedures, Internal Control definitions, Board-level reports, etc.
Sales	Net Profits of: Properties, Automobiles, retail goods, etc., Partnerships, etc.
Strategy	Strategic Key Initiatives, implementation
Customer Service	Customer Service Standards, customer service processes, customer satisfaction, etc.
	Etc.

As we described above, each one of these **VLITypes** may include different **surfers**, which in turn will mean to have different **VLI attributes**.

Knowledge Parameters

knowledge

```
{
  techniques(technique),
  patterns(pattern)
}
```

technique

```
{
  name,
  purpose,
  processStep,
  howItIsUsed,
  interfaces,
  mappings,
  references
}
```

pattern

```
{
  name,
  context,
  problem,
  solution,
  rationale,
  resultingContext,
}
```

These parameters tell us about what techniques we are inserting into the Scrum framework to specialize our Scrum instances for whatever purposes we need.

Techniques. technique

technique - technique describes a technique that has been inserted into our Enterprise Scrum instance to help us in a particular way. The subparameters to describe our technique is:

name – describes what the technique does

purpose – what purpose does the technique have

processStep – the process step where the technique is introduced

howItIsUsed – a description of how the technique is used in our Scrum instance

interfaces - what inputs do the technique need from Scrum, what outputs to Scrum does it produce

mappingsToScrum –and how exactly the outcomes are mapped to Scrum and its artifacts, for example, the Value List

references – provides where we can find more information about it

For example, we may introduce a Blue Ocean strategy's Strategy Canvas as business technique to help us find a better company, portfolio or product direction. Let's assume that is a product.

Therefore, our Strategy Canvas **technique** is configured:

name – Strategy Canvas

purpose – identify strategic factor values so that we can better create and prioritize our Value List in order to position our product in a unique niche

processStep – the technique should be used before the creation of Value Lists – both initially and after every Refinement

howItIsUsed – the product factors are initially analyzed and then re-analyzed at the beginning or every Refinement so that we can create a better Value List

interfaces – the Value List can be used as input to create the current state of the Strategy Canvas. After analysis is market, competitors, suppliers, and other factors; modifications can be made to the Strategy Canvas, and its output can feed into the Value List.

mappingsToScrum –the outcome of the Strategy Canvas helps us as high level guidance for the creation or modification (reprioritization) or our Value List. Typically a Strategy Canvas factor maps into multiple **valueListItems** (VLIs).

references – Blue Ocean Strategy books, websites, internal documents, previous analysis documents, etc.

Other techniques from Lean Startup, Profit Zone, Design Thinking, Scenario Planning, Blue Ocean Strategy, Red Ocean Strategy can be inserted that way as well.

Techniques. techniqueList (technique)

techniqueList -- techniqueList is simply a list of all the techniques used within our Scrum instance.

Value List Item

valueListItem extends **VLIType**

```
{
VLIType,
DOR,
DOD,
planForVLI,
volunteers,
parentVLI,
dependsOnVLIs,
selectable,
time = <cycle, unselectable, scheduled, repeatable [cycleName] >,
structure = <singleton, collection [order], workflow[orderedPairs{singleton,
actor}]>,
actionMods = <work, refine, decide, test, research, answer, monitor>,

// domain specific from VLITypes – DOMAIN SPECIFIC MODEL
Revenue,
Profit, Etc.

// admin
Created by,
Added in Date,
Budget,

}
```

valueList(valueListItem),

In Scrum this is called the Product Backlog but because we build products, but in Enterprise Scrum we are interested in delivering the most business value as possible.

Value List Item (Type). valueListItem.

valueListItem -- defines a value list item – a granule or quantum of business value. It can be a direct contribution or an indirect contribution of business value e.g. an integration server to be able to deliver functions in software development, or software to support a sales process, etc.

The **valueListItem** has a mandatory list of VLI attributes as well as optional ones:

name – name of the VLI

description – description of the VLI

priority – priority of the VLI (typically a number 1-N) if it is **prioritizable**

size – size of the VLI in the **sizingUnits** chosen

DOR – definition of ready. It may be inherited for each VLI from the **VLIType**. It's a list that we call **entryCriteria** (to match the DOD "acceptance criteria").

DOD – definition of done for the VLI. It may be inherited from **VLIType**. We call the elements of this list **acceptanceCriteria**.

We also may have not only the **acceptanceCriteria** but require a list of data that constitutes proof we passed DOD, **acceptanceCriteriaProof**:

acceptanceCriteria(acceptanceCriteriaProof)

Notice, this is a "definition of done", not of "partially done". The different conditions within a DOD must specify when there is "nothing else left to do".

In strategy, the definition of done for a VLI that has the description "Working with a partner in China", may have a DOD that includes:

- Signing the partnership contract
- Having at least 3 customers through that partner in China by end of Q1 2014

That is, when we have achieved these things and we have **acceptanceCriteriaProof** one or more for each **acceptanceCriteria**, an only then, we would consider our VLI "done".

planForVLI -- a plan for VLI or a VLI type. A plan can be a list of tasks, a list of components, a list of sub-goals, workflow or state machine, typically annotated with owners, and may include an estimated time to completion or effort (or date to be finished). The plan needs to satisfy the DOD for the particular Scrum instance.

parentVLI – the parent VLI if there is one. Because in Enterprise Scrum, we can do work with multiple teams, some times is useful to have a **parentVLI** parameter, so that summary VLIs can be tracked and traced to more defined VLIs and vice-versa. When there is one or more VLIs that have a common

The **parentVLI** summarizes the work in a coordination, delegation or subsumption network.

dependsOnVLIs – the list of dependent VLIs if there are any. It can prove useful to have a list of VLIs that each of the VLIs depend on, if this list is more or less static. However, if these relationships change fast, it could be hard to track them. When there is one of more VLIs in our Scrum instance that depends on the VLIs from another Scrum instance, we say that our Scrum instance **dependsOnVLIs** that instance. See above in Structure Parameters.

selectable – yes, if the VLI is selectable into an Cycle, no otherwise

volunteers – the people that work on the VLI
time {sched, repeatable {cycleName}}
structure <singleton, collection {order}, workflow{orderPairs{singleton, actor}}>
actionMods = <work, refine, decide, test, answer, monitor>

Other optional attributes, which may include attributes specific to a **VLIType** are:

volunteers – the list of volunteers for a VLI

Created by,

Added in Date,

Budget,

Revenue,

Profit,

Etc.

Other attributes can provide different metrics that may make sense in our situation. We will describe and give examples of these metrics later when we talk about Cycles (Sprints). Some of those metrics may define alternate velocities. For example, they may define target profits in a sales process per Cycle, etc.

Process Parameters

```
process
{
// -----
// 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
{
name,
agenda[listOfActivities],
typicalDuration[hours],
techniquesUsed[techniques]
},
eventTypes (eventType),

event
{
eventType,
timeDate,
duration,
attendees[ESteam]
```

```

},

sequenceStep
{
  surfers,
  event
},

InitialValueList
{
  initialValueListSequence(sequenceStep),

  initialValueListSequence = initialValueListSequenceFromIA[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(events),
  initialValueListevents = initialValueListEvents [initialValueListSequence],
}

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

  // Cycle structure
  cycleSequence(sequenceStep),

  cycleSequence = cycleSequenceFromIA[surfers],

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

  // Cycle instances with ALL the data they hold

```

```

cycleInstances(cycleEvents),
}

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

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

```

Cycle Parameters

listOfCycles(Cycle)

```

Cycle
{
name,
level,
length,
qualitative,

// subsumption
surfersForCycle(surfers),

// PC3R
style = <PCRI, PC3R>
recommendedPE3RtimeFrames,
planningType = <WBS, other>,
collaborationType = <independent, collaborative, other>,
reviewType[ surfers, <cumulative, incrementOnly >]
improveType[ surfers, <wellImprove, other>],

// Cycle DOR/DOD
cycleDOR,
cycleDOD,
passedCumulativeDOD

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

```

```
// reports
scrumBoard,

// team dependencies for Cycle
dependencyMatrix,

// value
valueIncrementName,
usableValue
}
```

A Scrum Team can have one or more Cycles. For example, a development team may have 3 Cycles:

- 2 week Development Cycles
- 3 months Release Cycles (6 Development Cycles)
- 1 year Business Validation Cycles (4 Release Cycles, 24 Development Cycles and some time off)

It is a good idea that all the **contributors** of a Scrum Team automatically inherit all of their Cycles of the **parent**.

Cycle. Cycle

Cycle – defines a Cycle type: how we are planning, accomplishing, measuring, and what kind of work we are improving.

The subparameters of an Cycle are:

```
name,
level,
length,
qualitative,
accountingType <granule, percentage>
recommendedPE3RtimeFrames,
surfersForCycle(surfers)
planningType <none, WBS, other>,
collaborationType <single, collaborative, other>,
reviewType <cumulative, incrementOnly>,
improveType <wellImprove, other>,
cycleDOD,
metrics (metric{name, description, unit, frequency}),
charts(metricChart {name, description, metric, type}),
scrumBoard,
dependencyMatrix,
valueIncrementName,
usableValue,
```

calculations (calculation {description, formula})

For each of the Cycles listed above we need the following parameters:

name -- This is the name for our Cycle. We typically use the name of our Work Type == <VLI-type 1> and a frequency. For example, “Weekly Sales Cycle”.

length – it is the length of the Cycle. First layer Cycles typically are 1-4 weeks of duration. Second layer vary from 1 month through a year. Third layer are typically at least six month long and are the business validation level.

qualitative – describes whether the Cycle is quantitative or qualitative. We prefer to do quantitative cycles that use good metrics, but sometimes that’s not possible

recommendedPE3RtimeFrames, the times recommended for each of the PE3R activities, typically in hours per Cycle.

surfersForCycle(surfers) – this is the list of **surfers** for which we are doing subsumption in that cycle, and which we are coevolving in that cycle.

That means that we are doing Planning, Collaboration, Review, Improve,

planningType <WBS, other>, If it is WBS, the typical “different levels of breakdown of work” will be used; otherwise, you must specify how are plans for VLIs in that Cycle will be done. This should constraint the **planForVLI** in the **valueListItem**.

executionType <single, collaborative, other>, at the very list we should know how the execution to get things done is accomplished by either one, or more people and in what mode.

reviewType { <cumulativeTime, incrementOnly >},

Review must be done for all **surfers** in the **Cycle**.

the valid options are “cumulative” or “incrementOnly”. For example, when we do portfolio reviews, it’s nice to include the existing products and services, not only the ones being developed in the queue. On the other hand, when we do software development, it is nice to do just “increment only” reviews, because there is little value in reviewing something that we already know it works.

improveType

Improve must be done for all **surfers** in the **Cycle**.

Some surfers are always included by default: Team, Customer, plan, purpose,

team - <wellImprove, other>, the valid options are “default”, with just a list of what we are doing well, and how to improve, or to specify a different type. See the Agile Retrospectives book for other valid options [AgileRetrospectives]. This option can be overridden in the **cycleInstance**.

cycleDOD - DOD for the Cycle. For example, in software development, a Sprint DOD includes all of the PBIs (VLIs) DODs, but it can add things like versioning, labeling, deployment tasks, notifications, etc.

passedCumulativeDOD – is Boolean variable that confirms that the DOD for VLITypes that were required to pass cumulative DOD was passed.

metrics -- list of metrics to be used in the Cycle.

In regular Scrum we typically just have one metric velocity, which is measure of productivity, and is always measured in units of effort. Because of that, we tend to optimize velocity, and with that productivity. However, in Enterprise Scrum we are allowed to have multiple metrics or measurements. It is convenient to know all the metrics that we are using so that we can balance the optimization of our Cycle accordingly.

Metrics can be of many different types. For example:

- customer satisfaction
- employee happiness
- goodness for the world
- profit
- cost savings
- productivity
- quality
- performance
- learning
- accomplishment
- building
- adaptation
- transformation
- improvement or
- growth
- etc.

For each important measurable thing to us, we must define a **metric**.

metric(name, description, unit, frequency) - define a generalized measurement to track progress or status in an Cycle. Typically an **optimizationVariable** is used in a cumulative or non-cumulative way, either as either growing towards a goal, or as the “left to the goal”.

It is important to choose relevant business-value like metrics that are important to the customer – not just arbitrary things that are convenient to measure. For example, in a sales process, rather than measuring an internally important measurement such as “effort per sales”, it is better to measure “customer satisfaction”.

This allows Enterprise Scrum to define one or more useful metrics, as opposed to just have the canonical “velocity” measured in units of effort. Metrics have the purpose to measure how close we are to our goals.

For example, some metrics could be revenue, profit or customer satisfaction in an Cycle.

Each metric has the following attributes:

name – the name of the metric. For example, “net profit”

description – tells us what it measures and how to measure it

unit – it tells us the units we are using to measure the metric

frequency – suggested frequency of measurement

charts – this is just the list of charts for the Cycle

metricChart (name, description, metric, type) – this parameter defines a chart where the metric is used.

The attributes of a chart are:

name – this is the name of the chart

description – it tells you what it is and how to build it

metric – it tells us what metric to use (see description above)

type – burnup, burndown, other

Typically, we graph the metrics on the Y-axis and time on the X-axis. The resulting graphs can be both burndowns or burnups, or just a graph of the metrics like “average daily customer satisfaction” when customers are taking surveys after a service appointment, for example.

scrumBoard -- specify which type of Scrum Board we will use to manage the Cycle workflow

Having a DOD (definition of done) allows us to workflow our VLIs across the Scrum Board in the same way that we do in Scrum:

- Not selected – VLIs that have not been selected into an Cycle.
- Selected – VLIs have been selected into an Cycle. In the case VLIs are not

selectable, all VLIs are assumed to be selected (see the Selectable parameter below), or in WIP if there is some work done on them already.

- WIP (work in progress), for VLIs that have been selected for this Cycle and that some work has been done on them. The WIP column can be further customized to match a specific workflow, or leave it as “free flow” until the DOD is achieved.
- DONE – for VLIs that satisfy the DOD – hopefully with nothing else left to be done and the product owner, or someone delegated from the product owner has approved of the VLI.

This is the same workflow as defined with Scrum but just with a couple of variations as to what can be given order or priority, what VLIs can be selectable, and respective DOR and DODs.

dependencyMatrix – is a chart to map dependencies for a Cycle of either VLIs, persona or teams. This defines the format to be used by the Cycle instances.

valueIncrementName – This is the name of our increment. We typically call the “VLI-type1 Increment”. For example, “Profit Increment” or “Sales Increment” or “Product Increment”, etc.

usableValue -- define what is usable value.

In regular Scrum this Business Value comes as Product Increment that has the quality to be Potentially Shippable Product. Potentially Shippable Product means that this is a product that has nothing left to be done to be shipped. However, when doing Enterprise Scrum to manage other business processes we may not necessarily be “shipping a product”. Moreover, this is only “potential value”.

In Enterprise Scrum, we are more interested in tracking the **usableValue** if possible – not the “potential value”.

calculation (description, formula) – a **calculation** is a derived quantity that is useful to know for an Cycle

calculations (calculation) – **calculation** is the list of calculations that are useful for an Cycle. For example, budget, schedule, customer satisfaction, compliance level, etc.

Cycle Instance Parameters

cycleInstances(cycleInstance)

cycleInstance extends Cycle

```
{  
  icID,  
  initialValueListForCycle(valueListItem),
```

```

doneValueListForCycle(valueListItem),
retrospective[(wells), (improvements)],
measurements(measurement),
chartInstances(chartInstance),
calculationsInstances(calculationInstance),
scrumBoardInstance,
dependencyMatrixInstance
}

```

Cycle Instances. CycleInstances(CycleInstance)

improvementCycleInstances – this is the list of Cycle Instances saved with all of their data

Cycle. cycleInstance

For each instance of an Cycle we capture the following information:

```

icID,
initialValueListForCycle(VLI), doneValueListForCycle(VLI)
retrospective(well, improvements),

```

```

measurement
{
metric,
timestamp
}

```

```

measurements (measurement),
chartInstances (chartInstance),
calculationsInstances,
scrumBoardInstance,
dependencyMatrixInstance,

```

icID – this are the name or value identifying a particular Cycle. For example, “Sprint 23, Release 2.1” in a software development.

initialValueListForCycle(VLI) – this is the initial **valueList**

doneValueListForCycle(VLI) – this is the **valueList** that got done (passed the DOD and was approved by the Business Owner).

retrospective(well, improvements) – this captures the results of the retrospective. It is desirable to bring these results to the next retrospective as a starting point.

measurements (metrics) – these are measurements for the different metrics (generated with a particular frequency). For example, in compliance management we could define one metric as “number of controls tested”, and another one as “number of incidents”. The measurements are the actual numbers.

chartInstances (chartInstance) – these are the instantiated charts for that particular Cycle instance (including the data)

calculationsInstances – this is the list of calculations for the Cycle. These are the

actual calculations (including the numbers)

ScrumBoardInstance - this is the snapshot of the instance of the ScrumBoard used.
dependencyMatrixInstance – this is the ongoing dependency matrix tracking internal and external dependencies.

Structure Parameters

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

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

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

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

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

structure

In Enterprise Scrum we can accomplish the work with one or more teams. When we start a project it is easier to start with one team and then add more teams as needed. However, once you master scaling, it is possible to start with multiple teams at once but it is not the recommended way to get started.

The structure parameters allow Enterprise Scrum to scale into larger organizations by connecting multiple Enterprise Scrum instances in different ways. The structure parameters in the ESTeam tell us specifically how Scrum instances are linked to other:

- 1) which ones are connected to each other – this can vary in time, through the **dependsOn(ESTeam)**, **dependOnUs(ESTeam)** parameters,
- 2) In what mode are the different instances connected to each other.

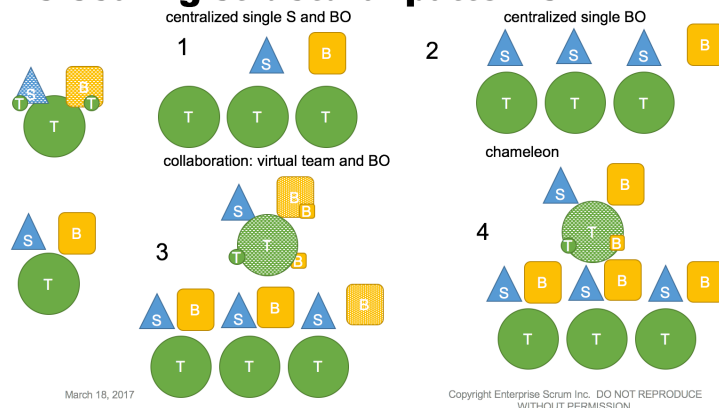
In Enterprise Scrum we have 4 ways to connect instances: subsumption, collaboration, delegation or centralized.

Even if the scaled organization works in mixed mode, it is important to define these ways of working to understand how scaling works.

scaling

scaling. scalingPatterns

ES scaling structural patterns



scaling. contributors -- Enterprise Scrum instances that contribute together.

The **contributors** parameter is a list of all of the Scrum instances that work together.

This also implies that some of the VLIs among different Scrum Team instances may

have dependencies. For example, that one or more of our **valueListItems** are the **parentVLI** or have **dependsOn** or **dependOnUs** relationships with **valueListItems** in other (contributor) teams.

The **contributors** typically have a common metric, so that the parent can aggregate it. For example, individual sales teams can have a Revenue or Profit metric that can be aggregated by the parent so that the parent can report on overall performance.

structure. connectionMode

connectionMode – can take different values: **<subsumption, collaboration, delegation, centralized>**, and it defines the connection mode used by the Enterprise Scrum Team instances.

Here are a few patterns that we use to build organizations.

Feature Teams - Scrum Team per Business Area

We typically form new helper Scrum Teams around business areas because then there is a one-to-one correspondence among:

Customer benefit or feature ->
 Business Area ->
 Subsystem (if it a by product exists) ->
 Scrum Team

For example, in a benefits management system we would strongly suggest forming teams for Pension, Defined Contribution (401K), and Health and Welfare plans. It also work on other domains, for example, real state sales can be split in Retail, Mansion, Vacation or Commercial. Larman and Vodde call these “feature teams” – a good name for software or product development [LeSS1], [LeSS2].

Be aware that these relationships may change over time, so we could rearrange the team’s structure as needed.

These “feature teams” can operate under any one of the modes.

In SAFe, for example, there is a centralized delegated control for business value. But in LeSS, there is only centralized control of business value.

In Enterprise Scrum we allow:

4 contracts types:	fixed date, fixed price, time and materials, cost-plus
4 delivery mechanisms:	continuous deployment, continuous delivery, Cycle delivery, Release delivery
4 connection modes:	subsumption, collaboration, delegation, centralization
4 delivery targets:	large project, program, portfolio, Enterprise Architecture

Add a parent Scrum Team to represent Big Picture

Suppose that the opposite happened teams were added as needed to offer more

services to the customer but now no controls the overall picture. When this is the case, it is a good idea to add a **parent** Scrum Team to coordinate the “face to the customer”. The Business Owners of each of the Scrum Teams may form themselves a **parent** Scrum Team that has a **constrained autonomy** since it depends on its contributors.

Delegation/Centralized

The relationship of parent to contributors may vary. In some cases is more delegation: **contributors** work for the **parent** with feedback from the trenches as to why they can do but trying to achieve a “grand vision” – unity of purpose, from a Chief Product Owner. This is needed sometimes in some environments but there are other options.

Collaboration

It is also possible to have a softer cooperating self-organizing relationship: the **contributors** just contribute to their success self-organizing to achieve their individual goals, and resolving conflicts and dependencies as needed with other Scrum Teams – there is no **parent** or there is only a very high-level **parent**. Every team leads itself communicating, co-adapting and flocking with other teams.

As a general rule is that the **contributors**, should operate as autonomously as possible. This is a critical pattern in Scrum-managed organizations: direction or new information may come from any other layer, and therefore the entire organization needs to quickly adapt to this new direction generated by the new information.

Subsumption

It is also possible to operate through a set of subsumed behaviors. This can be done in:

- 1) a one to one relationship, each one with a **subsumptionLayer**, or
- 2) through a set of **contributors** just contribute to the success of the **parent's** main behavior, choosing the best behavior among the Scrum Teams through a defined set of behaviors that subsume each other. The **parent** may give a direction to the **contributors** but then as the **contributors** teams find market information, they may send direction (information), that affects the **parents** overall direction.

contractType

deliveryMode

deliveryTarget

5. Enterprise Scrum Example – Startup or Innovative Business Unit

NAME

The instance of Enterprise Scrum for Startup Management or Innovative Business Unit

...in a NEW Blue Ocean market

We can rewrite this instance name this way:

The instance of Enterprise Scrum for a NEW Single Product/Service Management in a company in a NEW Blue Ocean market

to indicate that it is possible for companies of any size to manage startup-like business units, as long as they understand this type of management.

ADVANTAGES

Motivation

A startup (or new business unit), in a new Blue Ocean market, as defined by Steve Blank is [Blank]:

A startup is a temporary organization in search of a scalable, repeatable, profitable business model.

We can best manage all phases of a startup (or new business unit) in an Agile way with Enterprise Scrum to achieve this scalable, repeatable and profitable business model:

- 1) from the time is searching for its business model,
- 2) through initial growth,
- 3) scalability and profitability, or
- 4) even as it is phased out if things are not working out.

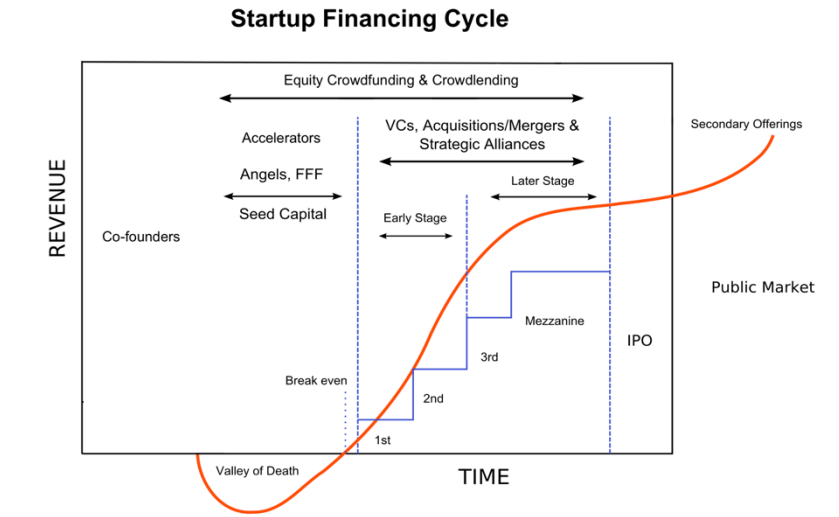


Figure 25. Startup Timeline

Enterprise Scrum is a good way to manage a startup because it allows the evolution of the customer, the business model, the solution (product or service), and the startup organization and processes simultaneously.

Examples

New Governance Inc.
Spotify new business units

Framework - techniques, patterns

There are very many business management techniques that we can use for startup management:

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

Enterprise Scrum Industry Impact

Enterprise Scrum allows startups and companies of any size to start and manage a new business unit as if they were startups *in an accelerated and more accurate way*. In that

sense, Enterprise Scrum serves as an accelerator program for the startup (innovative business units).

Why is this important? Because of disruption.

As we enter the Innovation Revolution (4th Industrial Revolution), the number of successful startups (or new company innovative divisions), will accelerate not only in numbers but in revenue and profit. Some of them will become MTPs (massive transformative purposes), like Uber, Tesla, Slack, Oculus Rift, etc. Disruption is the mark of MTPs.

Wealth has never been created, transformed or destroyed this fast ever.

Here is a chart that shows that “Time to Unicorn” is rapidly decreasing in time:

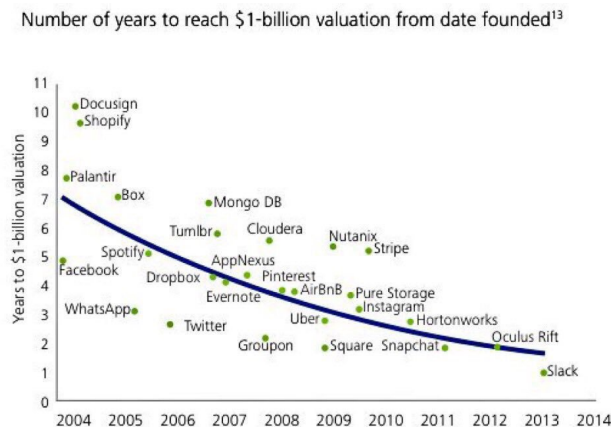


Figure 26. Time to Unicorn vs time

Being more pleasing or transformative to the customer and delivering faster and better solutions to the market has never yielded more profits and revenue than today. Remember, roughly 50% of the companies that were here in the year 2000 are not here any more. So this instance, and other instances like this, have the potential to determine who are the winners, or the losers in the 21st century.

TWEET: Wake up before DISRUPTION UBERizes your company. Innovate, delight customers, make employees happy, find an MTP [#ES](#)

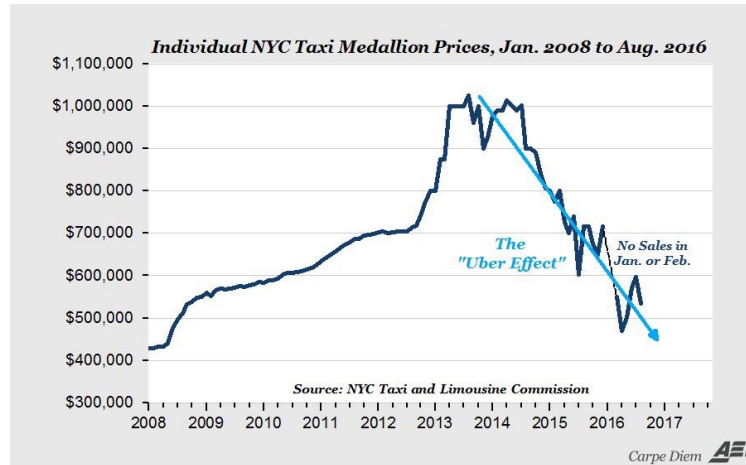


Figure 27. NYC Taxi Medallion Prices vs Time

As startups, unicorns and traditional companies in any industries compete for the current GWP (Growth World Product) of 75 trillion (Geary-Khamis dollars), the more customer-delighting and agile-managed companies are, the more likely they will be to attract and retain customers and beat competitors for an existing or new industry.

Rank	Name	Country	Industry
1	TESLA MOTORS INC.	USA	Automobile Manufacturers
2	SALESFORCE.COM INC.	USA	Application Software
3	REGENERON PHARMACEUTICALS	USA	Biotechnology
4	INCYTE CORP	USA	Biotechnology
5	ALEXION PHARMACEUTICALS INC.	USA	Biotechnology
6	UNDER ARMOUR INC.	USA	Apparel, Accessories & Luxury Goods
7	MONSTER BEVERAGE CORP	USA	Soft Drinks
8	UNILEVER INDONESIA TBK PT	IDN	Household Products
9	VERTEX PHARMACEUTICALS INC.	USA	Biotechnology
10	BIOMARIN PHARMACEUTICAL INC.	USA	Biotechnology

Figure 28. Most innovative companies

This is a special time in the history of humanity – we are just entering the Innovation Revolution, the landscape of exponential organizations [ExponentialOrgs], what the WEF calls the “4th Industrial Revolution” [4thIndustrialRevolution] where: companies reach 1 Billion dollar valuations in record times each year, innovation is rampant in every industry, and customers are more powerful than ever.

TWEET: 🦄 #EnterpriseScrum accelerates the success of startups through: search, growth, and execution (scalable, repeatable, profitable businesses)

CONFIG

In this section is where we specify the parameters to configure our instance of Enterprise Scrum. We won't talk about all the parameters because they are too many. Instead, we will pick up the ones that are most relevant to this instance.

Enterprise Scrum Team

Enterprise Scrum Team: Business Owner

The Business Owner for Startup Management is the person managing the entire startup. This may or may not be the Business Owner of the Customer Segment Team, which is in charge of the business direction of the startup based on customer feedback and technical constraints.

However, the Business Owner owns the final decision on what gets done in the startup, and what gets delivered as a business solution. The Business Owner therefore must know what is MVP (minimal viable product), or even better MLP (minimum likeable product).

Enterprise Scrum Team: Coaches

If we are doing Scrum or Enterprise Scrum we need a ScrumMaster to guide us through the Scrum process. However, in this instance the ScrumMaster is overseeing the process that the entire startup uses to get things done.

Enterprise Scrum Team: Team

Startup team, which may be composed with one or more teams, or a new Business Unit team or teams.

Business Team

We proposed in the Company Management chapter to call this team the Customer Segment Team, that mirrors the customer. When companies are startups they must do “all at once” business management, that is a single group of people:

- decides business strategy
- does marketing
- sells the product or service
- gets customer feedback and adjusts things as needed
- etc.

We also proposed in the Company Management chapter, that even established companies should use this pattern: this is what allows companies to keep the agility of a startup, and avoid the tar pits of functional management. In the 21st Century, everything should be treated on a “trial basis”, whether they are new or existing: business units, customer segments, and products and services.

Product Team

Startups may or may not have a separate product development team. Also, and depending on the nature and size of the technical work, there could be several technical teams.

Customer

Here is the key idea: when you are in a NEW Blue Ocean market, you may not know exactly who your customers are! This is what Steve Blank calls Customer Development – the finding or identification of who your customers really are. So in startups or innovative business units within Blue Ocean markets, we are trying to find the customers.

Startups typically have a single product or service but they may have one or more customer segments interacting and/or served in the business model. For example, a financial institution has retail, high-net work, and institutional investors, so each one of these classes of investors is a Customer Segment. A social network or a search engine has typically two sets of customers: the users and the advertising companies.

Customer: Business Value

Business Value changes as the startup (or new business unit) evolves.

Within the search phase Business Value is:

- finding a customer
- finding a product or service is a value proposition for a customer
- finding a business model
- visits
- users
- cohorts
- employee happiness
- customer satisfaction
- finding an MTP
- fail faster: fail as fast as you can. it is better to fail with 1-million-dollar investment than with a 100-million-dollar investment!
- etc.

In the growth phase Business Value changes - a balance of:

- profitability (profit, revenue)
- growing product
- faster time to market: better time-to-market means larger market share
- market share (100% if it is a truly a Blue Ocean)
- faster validation: not every idea is a “good idea”
- cheaper development: the NPE of startups and brand teams is super high, compared to what it costs in a traditional Sloan-managed organization
- faster decision making – instead of passing things from one functional

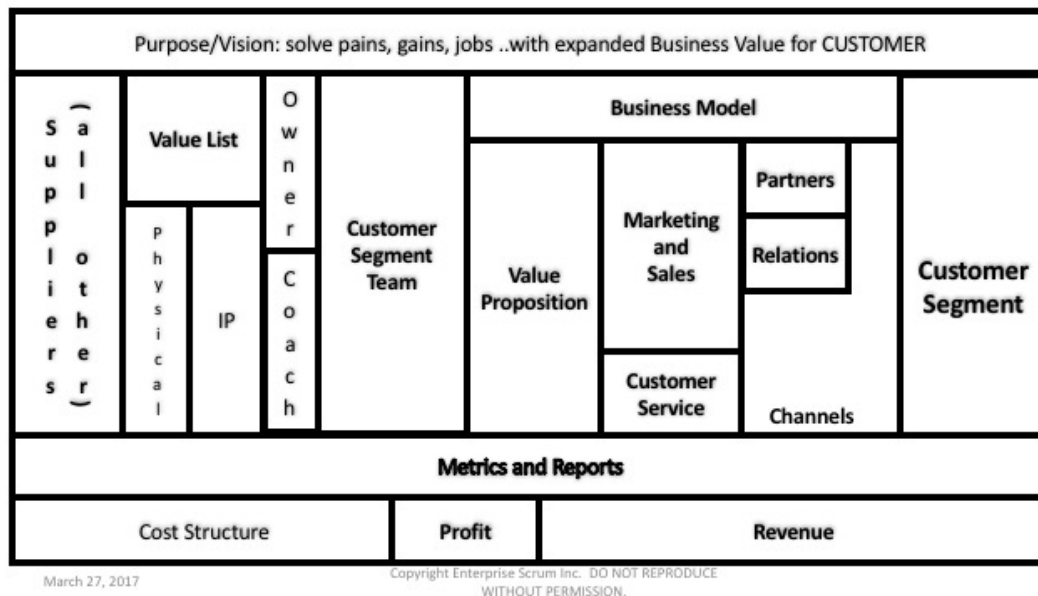
department to the next

In the execution phase Business Value is:

- stability
- repeatability
- scalability
- profitability

Information Architecture: Canvas

ES – Business Model Canvas



Information Architecture: Principal VLI

These have to be features for the customer, either as part of an application or the business process. For example, one of the features is “Trip Registration”.

Information Architecture: DORStandard

Conditions to start implementing VLIs. For example, these could be the defined, approved and sized (in effort, dollars, etc.):

- features of an application
- features of a business process,
- new hires
- etc.

Information Architecture: DODStandard

Conditions to get VLIs DONE. For example:

- acceptance tests for features of the application

- completion of a hiring requirement
- completion of the requirements of a business process
- etc.

Process: Cycles

Short-term Cycles: 1-4 week cycles

Business Cycles: monthly, quarterly

Process: Optional Activities

We could do one or more of these activities:

- find a Blue Ocean opportunity based on the analysis of factors in the market using the Strategy Canvas. Remember what we say in the startup community: there are “riches in niches” [BlueOcean]. This is typically done before the mandatory activities but can be adjusted at Refinement in a Cycle.
- If possible, find an MTP [ExponentialOrganizations]
- Observe the customer to determine pains, gains or jobs using Design Thinking techniques [DesignThinking]. This is typically done before the mandatory activities but can be adjusted at Refinement in a Cycle.
- Design our value proposition VPD (value proposition design). This is typically done before the mandatory activities but can be adjusted at Refinement in a Cycle.
- Build an initial business model using the BMC (business model canvas) from BMG (business model generation), or use our own ESBMC (Enterprise Scrum Business Model Canvas). This is typically done before the mandatory activities but can be adjusted at Refinement in a Cycle.
- Release Planning – forecast budgets and schedules for multiple smaller Cycles. This is typically done after the mandatory activities but can be adjusted at Refinement in a Cycle.

INSTANCE

To use an instance of Enterprise Scrum is very easy: simply use the appropriate canvas for the activity, in this case Startup Management, and then evolve it through the ES – ScrumBoard.

ES - ScrumBoard

Vision

Business Owner and Stakeholders

Initial Value List

Business Owner, Stakeholders and Team

	WIP (PCRI Cycle)	DONE
	Cycle 1	
	Cycle 2	
	Cycle 3	

March 27, 2017

Copyright Enterprise Scrum Inc. DO NOT REPRODUCE
WITHOUT PERMISSION.

C
o
a
c
h

Vision

To grow a successful startup or business unit in a new Blue Ocean market through all phases: search, growth, execution.

For example, a new travel agency that allows you to register your trips in advance to compare flight prices over time before you buy to optimize your purchases, which either buys a ticket or alerts you based on rules that depend on pricing.

Initial Value List

The list of things to do to get to MVP.

Define

Here we define that we need to do to start our startup. This is our Value List, and we may have to do:

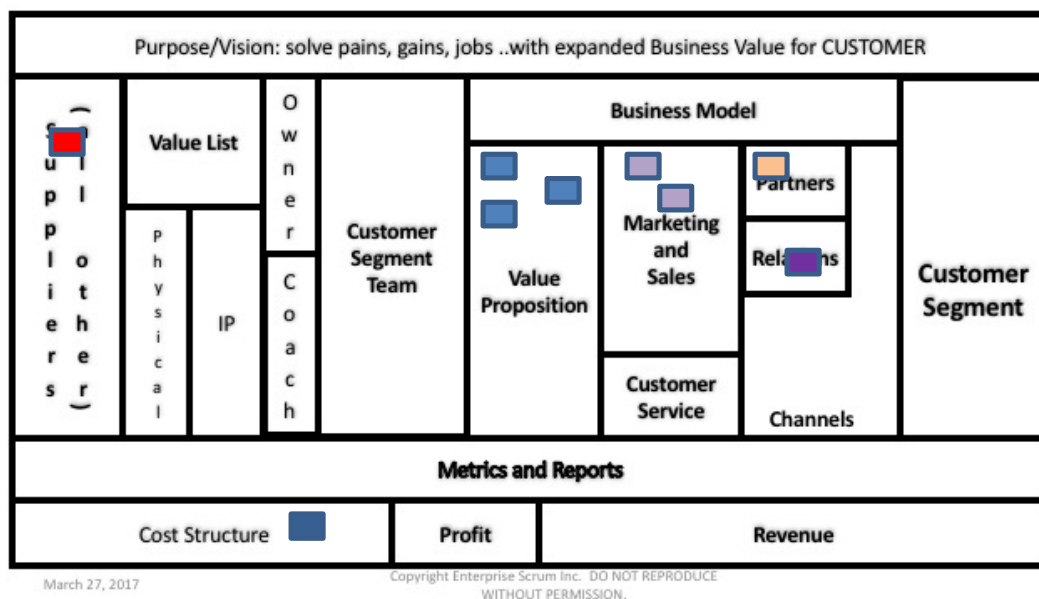
- experiments to see if there is interest
 - observing potential customers to see if there are pains, gains or jobs
 - seeking for funding
 - finding partners
 - developing relationships
 - identifying suppliers
- etc.

Each of these things that we do is a VLI, or value list item.

If you notice, all of these things above, have a corresponding box in the Business Model Canvas. See the figure below. The different color squares represent post-its -- “things to do” in each VLIType – in each of the boxes in the canvas, which represent the categories listed above.

When do we know when to stop: when the Business Owner determines we have enough features to go “live” with the startup.

ES – Business Model Canvas



Prioritize

This is where the business-related features get prioritized. All of them must be prioritized since we need to know which ones need to be implemented first.

We prioritize, or order by priority, all of the post-its in the canvas.

Also, it is a good idea to categorize the Value List as “must haves”, “want to haves”, and “nice to haves” to ensure we work on the ones absolutely needed to go live, and later on the ones that are just added as optional ones (want to haves and nice to haves).

Size

For a startup, both time and money are important, so in as much as possible, we should give size to both effort and cost.

CYCLES

Each and every cycle we do work that has Business Value. First, we do work so that we can launch our startup with an MVP.

But after that, we cycle so that we can:

Validate the customer

Validate the problem

Validate the product features

Test and improve the MVP

Better identify the customer

Validate the hypothesis of the value proposition,

Validate the hypothesis of the customer segments,

Validate the hypothesis of the channels

Customer-validation questions might include: cohorts (conversions from browsing to buying, for example), average order size, customer lifetime value, average time to first order, rate of sales pipeline growth, improvement in close rate, and revenue per salesperson.

In addition to the startup metrics above, the relatively few financial metrics that a startup board should be tracking are:

cash-burn rate,

number of months' worth of cash left,

short-term hiring plans

Each Cycle builds more of the business model, but the business model gets adapted in each Cycle based on the feedback from:

- 1) potential or better actual users or customers
- 2) technology constraints or findings
- 3) market changes
- 4) team limitations or capabilities
- 5) etc.

Iterate in Cycles with Enterprise Scrum until we release an MVP, or better an MLP (most likeable product).

Test our MVP in the market, and based on the results determine to do either small tweaks, major tweaks or stop the project

In parallel, build and manage our team with the best and brightest and keep them happy [Management3.0], [ManagingForHappiness]

Cycle

Planning

Plan which sections of the business model will be DONE in that Cycle. The team(s), 1) choose which VLIs to implement, 2) makes a tentative plan to accomplish this.

From a workflow perspective, here we are moving the post-its from the “not selected” canvas, to the

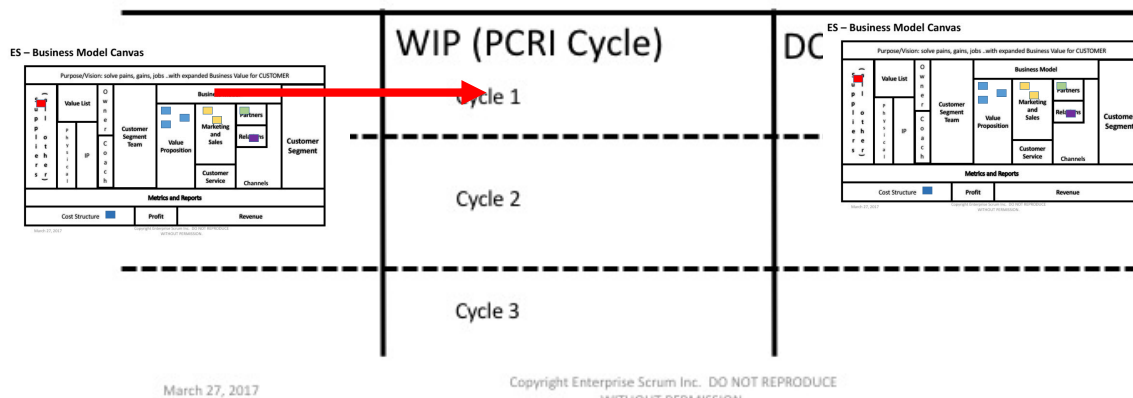
ES - ScrumBoard

Business Owner and Stakeholders

Vision

Initial Value List

Business Owner, Stakeholders and Team



C
o
a
c
h

Collaboration

At collaboration, we are working together to do everything that we chose in the planning. Therefore, we are getting things DONE according to their DOD.

From a workflow perspective, work remains in the WIP column until we finish the Cycle.

Review

At Review, we are going to review what we accomplished in the Cycle, and where we are cumulatively. However, a noticeable difference is that when we “grow startups”, we

don't review, improve and refine the product or service only, we review, improve and refine EVERYTHING: the product and service, the business model and even the growth hack strategy!

From a workflow perspective, we are now moving what we got DONE to the DONE canvas on the right.

We are then going to review everything that makes sense.

ES - ScrumBoard

Vision

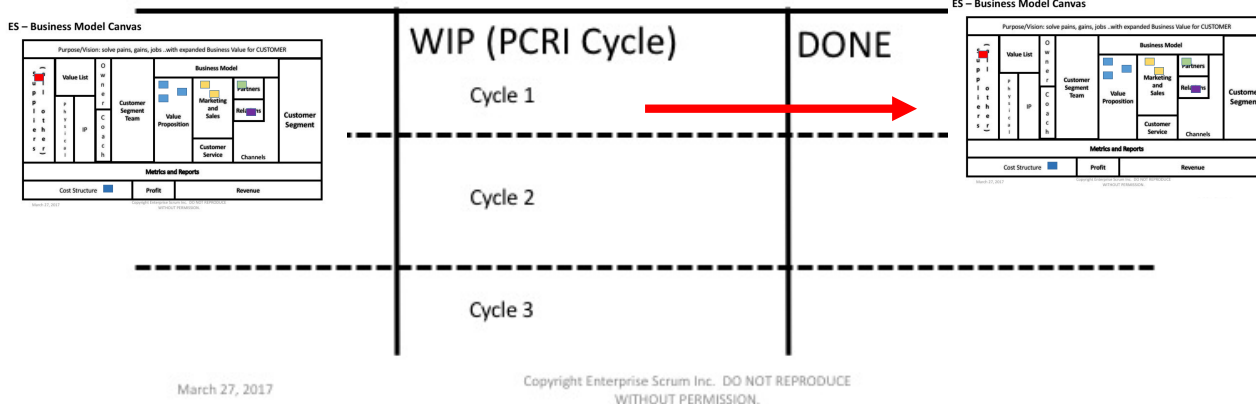
Business Owner and Stakeholders

C

Initial Value List

Business Owner, Stakeholders and Team

O



Improve

When we review things, we are also at liberty to choose everything that we deem important to improve. For example, we could improve the Value Proposition or the Team.

Metrics

We can track “effort DONE”, as in software development, but also track cost to track expenses, and therefore measure “how much cash we have”. This is an important measure for startups: because it determines how long you can survive until you get either another round of funding or start being profitable.

These metrics eventually will help us make decisions about the feasibility and success of our startup by producing income and profitability statements.

If the startup grows, and later there are multiple products and services for a customer segment, these metrics can be then fed into the portfolio management for that customer segment.

Reports

We can have a variety of reports. For example, customer satisfaction reports, number of users, cohorts (e.g. conversions from visits into paid users), and later as we grow, income reports, profitability reports, expense reports, effort DONE reports, activities DONE reports can be produced to help us manage our startup.

Plans and Forecasts

With the size, and the velocity, in effort and money (cost/profit or funding), it is possible to make projections on schedule, budget and risk management.

The following formula applies to both time and money:

$$C = VL/V$$

For example, if we have 100 weeks of work, and we know we can accomplish 10 weeks of work in a Cycle, we can produce a schedule:

$$C = 100 \text{ weeks of work} / 10 \text{ weeks/Cycle} = 10 \text{ Cycles}$$

However, we should check if the money is compatible with this schedule. Assuming we have \$1M left after some fixed costs, and that we spend \$80K per Cycle, we can afford:

$$C = \$1M / \$80K = 12 \text{ Cycles.}$$

We should be ok, as long as we get more funding or profits.

6. Conclusions – a call to GREATNESS!

We have seen in previous chapters how to use Enterprise Scrum in detail. Similarly, we can agilize anything else, including an entire company, with Enterprise Scrum.

Enterprise Scrum – Business Agility for Companies of ANY size!!!

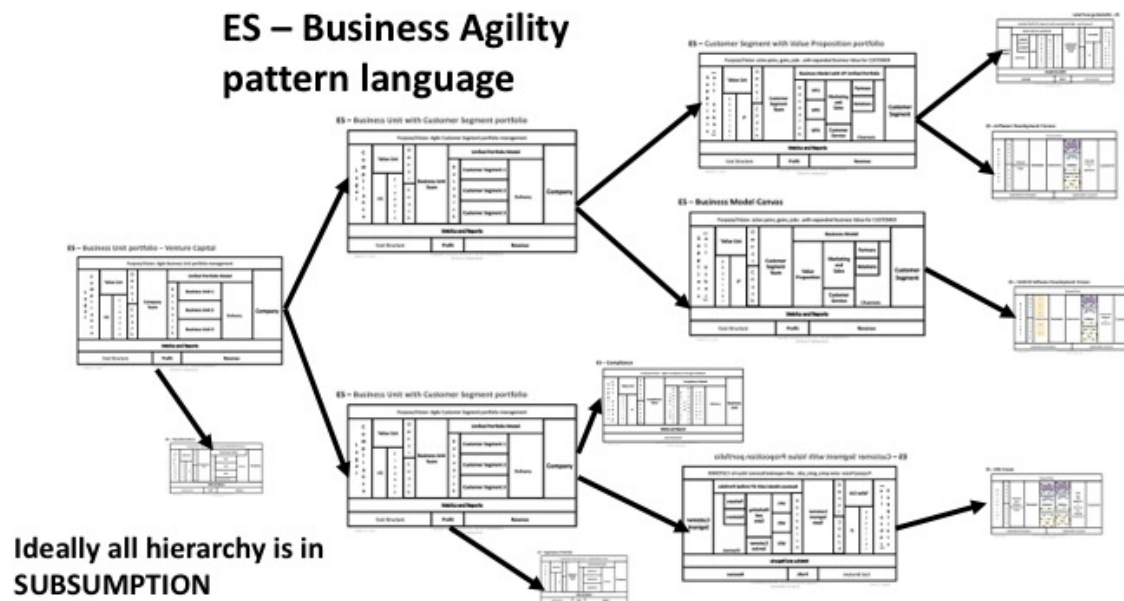


Figure 29. Business Agility for Entire Company

Innovation Revolution

Like it or not, we live in the age of the Innovation Revolution, and it is easy to predict who the winners will be in the 21st Century:

The winners in the 21st century will be companies that can innovate and adapt better and faster, to find and delight their customers.

Invitation – a call to Greatness

Ask your companies to do something GREAT, to serve a purpose in the world your employees can believe in. Ask your companies to innovate more at all levels of scale, to delight your customers more, to approach work with more passion, ownership, and happiness.

Ask your companies to reach Business Agility and lead the Innovation Revolution!

I hope you find Enterprise Scrum useful to do so.

References

1. [4thIndustrialRevolution] Schwab, Klaus; The Fourth Industrial Revolution (p. 1); World Economic Forum; Kindle Edition; 2016.
<http://www.weforum.org/agenda/archive/fourth-industrial-revolution/>
2. [Agile-3rdWave] The third wave of Agile
3. http://www.solutionsiq.com/the-third-wave-of-agile/?utm_source=twitterfeed&utm_medium=twitter
4. [AgileAtlas], <http://agileatlas.org/atlas/scrum>
5. [AgileBusinessManifesto] <http://www.AgileBusinessManifesto.org>, 1/5/2015.
6. [AgileCompetitors] Steven L. Goldman, Roger N. Nagel, Kenneth Preiss; Agile Competitors and Virtual Organizations: Strategies for Enriching the Customer (Kindle Locations 205-206). Kindle Edition.
7. [AgileManifesto] www.AgileManifesto.org ; 2/11/2001.
8. [BeedleSchwaber] Beedle M., Schwaber K., Agile Software Development with Scrum, Prentice Hall, 2001.
9. [AgileRetrospectives] Esther Derby, Diana Larsen, Ken Schwaber; Agile Retrospectives: making good teams great; Pragmatic Programmers Series, 2006.
10. [Alexander] Alexander, Christopher (1979). *The Timeless Way of Building*. Oxford University Press. [ISBN 978-0-19-502402-9](https://www.amazon.com/dp/0195024029).
11. [ArthurDLittle] Arthur D. Little, Innovation Excellence Study, 2012.
12. [Atkins] Lyssa Atkins, Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition, Addison and Wesley, 2010.
13. [BalancedScoreCard], Kaplan, Robert S.; Norton, David P. (1996-08-02). The Balanced Scorecard: Translating Strategy into Action (Kindle Location 3). Perseus Books Group. Kindle Edition.
14. [BCG-DieAnotherDay] by Martin Reeves and Lisanne Pueschel, Die Another Day: What Leaders Can Do About the Shrinking Life Expectancy of Corporations <https://www.bcgperspectives.com/content/articles/strategic-planning-growth-die-another-day/>
15. [Beedle-cOOherentBPR-1997] cOOherentBPR: A pattern language to build Agile organizations, Michael A. Beedle, PLoP '97 Proceedings, Tech. Report #wucs-97-34, Washington University (1997).
16. [Beedle-EnterpriseArchitecturePatterns-1998] Enterprise Architecture Patterns: Building Blocks of the Agile Company, Michael A. Beedle, SIGS, New York, (1998).
17. [BeyondBudgeting] Jeremy Hope; Robin Fraser; Beyond Budgeting: How Managers Can Break Free from the Annual Performance Trap . Harvard Business Review Press. Kindle Edition (2003-02-25).

18. [Blank] Blank, Steve; Dorf, Bob (2014-01-12). The Startup Owner's Manual: The Step-by-Step Guide for Building a Great Company (pp. 35-36). K&S Ranch. Kindle Edition.
19. [BlueOcean], K. Chan and R. Mauborgne, Blue Ocean Strategy – how to create uncontested Market Space and Make a the Competition Irrelevant
20. [BotsmanRogers] Rachel Botsman, Roo Rogers, What's Mine Is Yours: The Rise of Collaborative Consumption, HarperCollins, Kindle Edition.
21. [Brooks1], R. A. Brooks (1987). "[Planning is just a way of avoiding figuring out what to do next](#)", Technical report, MIT Artificial Intelligence Laboratory.
22. [Brooks2] R. A Brooks (1991). "[Intelligence Without Representation](#)", Artificial Intelligence 47 (1991) 139-159.
23. [BusinessAgility2007] Business Agility 2017, <http://www.businessagility2017.com>
24. [BusinessModelGeneration] Osterwalder, Alexander; Pigneur, Yves (2013-02-01). Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers (Kindle Location 361). Wiley. Kindle Edition.
25. [BusinessModelYou] Business Model You: A One-Page Method For Reinventing Your Career (Kindle Locations 411-412). John Wiley and Sons. Kindle Edition.
26. [CEK] Cooper, Edgett and Kleinschmidt, *Best Practices in Product Innovation: What Distinguishes Top Performers* by Cooper, Edgett and Kleinschmidt, 2011.
27. [Christensen] Christensen C., The Innovator's Dilemma: when New Technologies Cause Great Firms to Fail, Harvard Business Press, 1997.
28. [CitiesLive] Henrik Mårtensson, Why Cities Live and Companies Die, <http://kallokain.blogspot.se/2012/11/why-cities-live-and-companies-die.html?m=1>
29. [Cohn1] Cohn, Mike (2004-03-01). User Stories Applied: For Agile Software Development (Kindle Locations 269-270). Pearson Education (USA). Kindle Edition.
30. [Cohn2], Cohn M., Agile Estimating and Planning, Prentice Hall, 2006.
31. [Cohn3], Cohn M., Succeeding with Agile: software development using Scrum, Addison-Wesley, Upper Saddle River NJ, 2010.
32. [ConsciousCapitalism] John Mackey, Raj Sisodia, Bill George, Grover Gardner; Conscious Capitalism: Liberating the Heroic Spirit of Business, Harvard Business Review Press; 2012.
33. [CollaborativeIntelligence] Dawna Markova, Angie Mcarthur, Collaborative Intelligence: Thinking with People Who Think Differently, Random House Digital, Inc., Kindle Edition.
34. [Collins1], Collins J. Porras, J., Built to Last, Harper Collins, Collins Business Essentials, 1994
35. [Collins2], Collins J., Good to Great: why some companies make the Leap... and other don't?, Harper Collins, 2001

36. [ConnecedCompany] The connected Gray, Dave; Wal, Thomas Vander (2012-08-30). The Connected Company (p. 3). O'Reilly Media. Kindle Edition.
37. [Coplien] James O. Coplien, Borland Software Craftsmanship: A New Look at Process, Quality and Productivity, Software Production Research Department, AT&T Bell Laboratories, Proceedings of the 5th Annual Borland International Conference, Orlando, Florida, 5 June 1994.
38. [CreativeDestruction] Richard Foster and Sarah Kaplan; Creative Destruction: Why Companies That Are Built to Last Underperform the Market--And How to Success fully Transform; Kindle eBook; The Crown Publishing Group; 2011.
39. [CreativeEconomy] Howkins, John (2002-06-27). The Creative Economy: How People Make Money from Ideas (Kindle Location 309). Penguin Books Ltd. Kindle Edition.
40. [Culture-18Stats] 18 Statistics About Company Culture You Can't Afford To Ignore
41. <http://www.stackhands.com/blog/company-culture/company-culture-statistics/>
42. [DAD] Scott Ambler, Scott Lines, Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise, IBM Press, 2012.
43. [Deloitte2016] **Global Human Capital Trends 2016**,
<https://www2.deloitte.com/us/en/pages/human-capital/articles/introduction-human-capital-trends.html>
44. [Denning-AgeOfAgile] Steve Denning, Age of Agile,
<http://www.forbes.com/sites/stevedenning/2016/12/09/the-age-of-agile-what-every-ceo-needs-to-know/#7cb1d0ea621d>
45. [Denning-AgileInnovationEngine] Steve Denning, Agile: The World's Most Popular Innovation Engine,
<http://www.forbes.com/sites/stevedenning/2015/07/23/the-worlds-most-popular-innovation-engine/>
46. [Denning-AgileMainstream] Coding, Agile & Scrum Go Mainstream,
<http://www.forbes.com/sites/stevedenning/2015/06/14/coding-agile-scrum-go-mainstream/>
47. [Denning-AgileObjections] Steve Denning, The Case Against Agile: Ten Perennial Management Objections,
<http://www.forbes.com/sites/stevedenning/2012/04/17/the-case-against-agile-ten-perennial-management-objections/> [Denning-BestKeptSecret] Steve Denning, The Best-Kept Management Secret On The Planet: Agile,
<http://www.forbes.com/sites/stevedenning/2012/04/09/the-best-kept-management-secret-on-the-planet-agile/>
48. [Denning-EmbraceAgile] Forbes - Steve Denning
49. <http://www.forbes.com/sites/stevedenning/2016/04/21/hbrs-embrace-of-agile/#617d7e2027fe>

50. [Denning-Explaining Agile] Steve Denning, Explaining Agile,
<http://www.forbes.com/sites/stevedenning/2016/09/08/explaining-agile/#4b7f875d2ef7>
51. [Denning-FixingCapitalism] Steve Denning, Is the problem with capitalism that people are trying to fix it?
http://stevedenning.typepad.com/steve_denning/2011/01/is-the-problem-with-capitalism-that-people-try-to-fix-it.html
52. [Denning-ManagementFad] Steve Denning, Is Agile Just Another Management Fad? <http://www.forbes.com/sites/stevedenning/2015/06/22/is-agile-just-another-management-fad/>
53. [Denning-RadicalManagement], Stephen Denning, The leader's guide to radical management : reinventing the workplace for the 21st century, John Wiley & Sons, Inc. All rights reserved. Published by Jossey-Bass A Wiley Imprint 989 Market Street, San Francisco, CA 94103-1741. [Denning-ScrumMajorDiscovery] Steve Denning, Scrum Is A Major Management Discovery, <http://www.forbes.com/sites/stevedenning/2011/04/29/scrum-is-a-major-management-discovery/>
54. [DesignPatterns] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides; Design Patterns: Elements of Reusable Object-Oriented Software 1st Edition; Pearson Education (USA); 1994.
55. [DesignThinking], Kelley T. Littman Jonathan, The Art of Innovation, DoubleDay, 2000.
56. [DevTools] DevAcademy, 9 Game Changing Tools That Every Remote Developer Can't Live Without, <https://stories.devacademy.la/9-game-changing-tools-that-every-remote-developer-cant-live-without-57c6867cefb0#.36hkc7cw2>
57. [Drucker1], Drucker, Peter (1957). Landmarks of Tomorrow, New York: Harper & Row. pp. 122. ISBN 978-1-56000-622-0.
58. [DesignThinking], Brown T, Change by Design: how design thinking transforms organizations and inspires innovation, Harper Collins, 2009.
59. [DistributedScrum] Woodward, Elizabeth; Surdek, Steffan; Ganis, Matthew (2010-06-21). A Practical Guide to Distributed Scrum (Kindle Location 286). Pearson Education (USA). Kindle Edition.
60. [Eckstein] Eckstein, Jutta; *Agile Software Development in the Large: Diving Into the Deep (Dorset House eBooks) (Kindle Locations 11-12). Pearson Education. Kindle Edition.*
61. [EighthSkills] Stéphanie Thomson, 8 skills Google looks for in its managers, <https://agenda.weforum.org/2015/11/8-skills-google-looks-for-in-its-managers/>
62. [ExponentialOrganizations] Salim Ismail, Michael S. Malone, Yuri van Geest, Peter H. Diamandis, Exponential Organizations: Why new organizations are ten

times better, faster, and cheaper than yours (and what to do about it), Amazon Digital Services, Inc., Kindle Edition.

63. [Evo] Tom Gilb, Evo, <https://www.gilb.com/>
64. [FifthDiscipline] Peter M. Senge; The Fifth Discipline: The Art & Practice of The Learning Organization; The Crown Publishing Group; Kindle eBook; 2010.
65. [FixingTheGame] Roger L. Martin; Fixing the Game: Bubbles, Crashes, and What Capitalism Can Learn from the NFL; Harvard Business Review Press; Kindle Edition, 2011.
66. [FlatArmy] Dan Ponefract; Flat Army: Creating a Connected and Engaged Organization Kindle Edition.
67. [Ford1], Ford, Henry; Crowther, Samuel (1930). Edison as I Know Him. Cosmopolitan Book Company. p. 15 (on line edition).
68. [Forrester2017] Forrester - 2017 Predictions: Dynamics That Will Shape The Future In The Age Of The Customer, 2017 <https://go.forrester.com/wp-content/uploads/Forrester-2017-Predictions.pdf>
69. [GameStorming] Gray, Dave; Sunni Brown; James Macanufo (2010-07-21). Gamestorming: A Playbook for Innovators, Rulebreakers, and Changemakers (p. 1). OReilly Media - A. Kindle Edition.
70. [GoogleEffectiveTeams] Stéphanie Thomson, Google's surprising discovery about effective teams, <https://agenda.weforum.org/2015/12/googles-surprising-discovery-about-effective-teams/>
71. [GreatReset] Florida, Richard; The Great Reset: How the Post-Crash Economy Will Change the Way We Live and Work; HarperCollins; 2010; Kindle Edition.
72. [Greening] Enterprise Scrum: Scaling Scrum to the Executive Level <https://www.computer.org/csdl/proceedings/hicss/2010/3869/00/10-01-01.pdf>
73. [GrowthSummit] Growth Summit, <https://www.gazelles.com/summits/growth-2015/#sthash.x7i7PKVh.dpbs>
74. [IntrinsicMotivation] Christian Erburu, WHY INTRINSIC MOTIVATION IS THE KEY TO CHANGE MANAGEMENT, <http://www.happymelly.com/why-intrinsic-motivation-is-the-key-to-change-management/>
75. [Gonzalez] Agile and Scrum are bigger than IT, <http://ceoworld.biz/2016/12/20/agile-scrum-far-bigger>
76. [GWP] Gross World Product, https://en.wikipedia.org/wiki/Gross_world_product
77. [Hamel1] Hamel, Gary; Prahalad, C. K. (1996-03-21). Competing for the Future, Perseus Books Group.
78. [Hamel2], Hamel G., What matters now: how to win in a world of relentless change, ferocious competition, and unstoppable innovation, Jossey-Bass (Wiley Imprint), San Francisco CA, 2012
79. [Hammer], Michael Hammer and James Champy, Reengineering the Corporation, Harper-Collins, New York, 1993.

80. [Happiness-NEWROI] Happiness the NEW ROI
81. <https://thefutureorganization.com/is-happiness-the-new-roi/>
82. [HappyPeople] Happy People make Good Companies
83. https://hbr.org/2016/01/manage-your-emotional-culture?utm_campaign=harvardbiz&utm_source=twitter&utm_medium=social
84. [HBR-AgileStory] Darrell K. Rigby, Jeff Sutherland, Hirotaka Takeuchi, HBR - The real story of Agile <https://hbr.org/2016/04/the-secret-history-of-agile-innovation>
85. [HBR-ChangeManagement] Harvard Business Review (2011-02-24). HBR's 10 Must Reads on Change Management (including featured article 'Leading Change,' by John P. Kotter) (Kindle Location 977). Perseus Books Group. Kindle Edition.
86. [HBR-EmbracingAgile] Darrell K. Rigby, Jeff Sutherland and Hiraakata Takeuchi, HBR - Embracing Agile <https://hbr.org/2016/05/embracing-agile>
87. [HeartOfAgile] Alistair Cockburn, Expanding the Diagram, <http://heartofagile.com/expanding-the-diagram/>, 2016.
88. [Holocracy] Brian J. Robertson, Holacracy: The New Management System for a Rapidly Changing World, Macmillan, Kindle Edition
89. [Hoshin1], Hoshin kanri for the lean enterprise : developing competitive capabilities and managing profit / Thomas L. Jackson, New York : Productivity Press, c2006.
90. [Hoshin2],, Hutchins, David (2012-09-01). Hoshin Kanri (Kindle Locations 4-6). Ashgate Publishing. Kindle Edition.
91. [InsideApple] Adam Lashinsky; Inside Apple: How America's Most Admired--and Secretive--Company Really Works; Grand Central Publishing; Kindle Edition; 2012.
92. [Kanter], Kanter, Rosabeth Moss (14 June 2011). "Innovation: the classic traps". *Harvard Business Review on Inspiring and Executing Innovation*. Harvard Business Press. pp. 149–181. ISBN 978-1-4221-6261-3.
93. [Kaplan] Kaplan, Saul, The Business Model Innovation Factory: How to Stay Relevant When The World is Changing. John Wiley and Sons. Kindle Edition.
94. [Katzenbach] Jon Katzenbach, Douglas K. Smith, The Wisdom of Teams: Creating the High-Performance Organization, Harper Collins, 2006.
95. [Kniberg] Kniberg, Henrik, Lean from the Trenches: Managing Large-Scale Projects with Kanban (Kindle Location 185). Pragmatic Bookshelf. Kindle Edition.
96. [Kniberg-AgileEverywhere] Henrik Kniberg, Agile Everywhere, <http://blog.crisp.se/2016/11/21/henrikkniberg/agile-everywhere-slides-from-my-keynote-at-agile-tour-montreal>
97. [Kniberg-ETUL] MVP or ETUL <http://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp>

98. [KotterForbes] Can you handle an exponential rate of change?
<http://www.forbes.com/sites/johnkotter/2011/07/19/can-you-handle-an-exponential-rate-of-change/>
99. [KPMG] KPMG, Business Transformation and the Corporate Agenda, 2014,
<https://www.kpmg.com/US/en/IssuesAndInsights/ArticlesPublications/Documents/business-transformation-corporate-agenda.pdf>
100. [LeadersDilemma] Jeremy Hope, Peter Bunce, Franz Rösli; The Leader's Dilemma: How to Build an Empowered and Adaptive Organization Without Losing Control; Wiley; Kindle Edition; 2011.
101. [LeadershipAgility] William B. Joiner and Stephen A. Josephs; Leadership Agility: Five Levels of Mastery for Anticipating and Initiating Change (J-B US non-Franchise Leadership); Wiley; Kindle eBook; 2007.
102. [LeadingChange] Kotter, John P. (1996-08-07). Leading Change (Kindle Location 83). Perseus Books Group. Kindle Edition.
103. [Leybourn] Evan Leybourn, Directing the Agile Organisation: A lean approach to business management, Kindle eBook, 2016.
104. [LeanStartup], Ries E., The Lean Startup: how to make Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Crown Business, New York, 2011.
105. [LearningConsortium] <http://www.scrumalliance.org/learningconsortium>
106. [Lefingwell] Leffingwell, Dean (2007-02-26). Scaling Software Agility: Best Practices for Large Enterprises (Kindle Location 10). Pearson Education (USA). Kindle Edition.
107. [LeSS1], Larman C. Vodde B., Scaling Lean and Agile Development: thinking and organizational tools for large-scale Scrum, Addison and Wesley, Upper Saddle River NJ, 2009.
108. [LeSS2] Larman, Craig; Vodde, Bas (2010-01-26). Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum (Kindle Location 10). Pearson Education (USA). Kindle Edition.
109. [Liker1], Liker J., The Toyota Way – 14 management principles from the world's Greatest Manufacturer, McGraw-Hill, New York, 2004
110. [Liker2], Liker J. Morgan J., The Toyota Product Development System: integrating people, process and technology, Productivity Press, New York, 2006.
111. [LittleBets] Sims, Peter, Little Bets: How Breakthrough Ideas Emerge from Small Discoveries. Free Press. Kindle Edition. (2011-04-19).
112. [Nagel1], Nagel R., –21st Century Manufacturing Enterprise Strategy, Roger Nagel, Iacocca Institute, Lehigh University, 1991
113. [Management3.0] Jurgen Appelo, Management 3.0: Leading Agile Developers, Developing Agile Leaders, Addison and Wesley Professional, 2011.

114. [ManagingForHappiness] Jurgen Appelo, Managing For Happiness,
115. [McKinsey-Digitization] Digitization,
https://www.slideshare.net/mobile/McK_CMSOForum/haves-and-havemores-the-accelerating-digitization-of-the-us-economy
116. [McKinsey-OperatingModel] Santiago Comella-Dorda, Swati Lohiya, and Gerard Speksnijder, An operating model for company-wide agile development,
<http://www.mckinsey.com/business-functions/business-technology/our-insights/an-operating-model-for-company-wide-agile-development>
117. [MMM], Brooks F., The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley, Boston, 1974, 1995.
118. [MultiAgent] Wooldridge, Michael (2002). *An Introduction to MultiAgent Systems*. John Wiley & Sons.
119. [NetworkImperative] Libert, Barry; Beck, Megan; Wind, Jerry, The Network Imperative: How to Survive and Grow in the Age of Digital Business Models (p. 2), Harvard Business Review Press, Kindle Edition, 2016.
120. [Nexus] NEXUS <https://www.scrum.org/Resources/The-Nexus-Guide>
121. [Nonaka-PragmaticStrategy] Nonaka, Ikujiro; Zhu, Zhichang. Pragmatic Strategy (Kindle Location 188). Cambridge University Press. Kindle Edition.
122. [Nonaka-ManagingFlow], Nonaka, I., Toyama, R. and Hirata, T. 2008. Managing Flow: A Process Theory of the Knowledge-Based Firm. New York: Palgrave Macmillan.
123. [Nonaka-KnowledgeCreating] Ikujiro Nonaka; Hirotaka Takeuchi. The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation (Kindle Location 91). Kindle Edition.
124. [NonakaTakeuchi], Takeuchi, Hirotaka; Nonaka, Ikujiro (January–February 1986). "The New Product Development Game" (PDF). Harvard Business Review.
125. [Nonaka-KnowledgeCreationAndManagement] Ichijo, Kazuo; Nonaka, Ikujiro (2006-11-08). Knowledge Creation and Management: New Challenges for Managers (Kindle Locations 20-21). Oxford University Press. Kindle Edition.
126. [Ohno], Ohno, Takechi, Toyota Production System: Beyond Large-Scale Production, Productivity Press, Portland OR, 1978.
127. [OneClick] Richard L. Brandt; One Click: Jeff Bezos and the Rise of Amazon.com; Penguin Publishing Group; Kindle eBook; 2011.
128. [OrgPatterns] James Coplien, Neil Harrison, Organizational patterns, Prentice Hall, 2004.
129. [Osterwalder] Alexander Osterwalder (2004). The Business Model Ontology - A Proposition In A Design Science Approach. PhD thesis University of Lausanne.
http://www.hec.unil.ch/aosterwa/PhD/Osterwalder_PhD_BM_Ontology.pdf

130. [OperatingModel] Aaron Dignan, The Operating Model That Is Eating The World, <https://medium.com/@aarondignan/the-operating-model-that-is-eating-the-world-d9a3b82a5885>
131. [Owen-PowerOfSpirit] Harrison Owen, The Power of Spirit: How Organizations Transform (Kindle Locations 43-45). Berrett-Koehler Publishers. Kindle Edition.
132. [PassionateCompany] Passionate Company <https://hbr.org/2016/02/how-to-build-a-passionate-company>
133. [PEAK] Chip Conley and Tony Hsieh; Peak: How Great Companies Get Their Mojo from Maslow (J-B US non-Franchise Leadership); Wiley; Kindle eBook; 2009.
134. [Peers] Robin Chas, Peers Inc: How People and Platforms Are Inventing the Collaborative Economy and Reinventing Capitalism, Amazon Digital Services, Inc., Kindle Edition.
135. [PersonalityType] Travis Bradberry, Is this personality type more successful in the workplace? <https://agenda.weforum.org/2015/11/is-this-personality-type-more-successful-in-the-workplace/>
136. [Peters] Peters, Tom (2010-09-08). The Circle of Innovation: You Can't Shrink Your Way to Greatness (Vintage) (Kindle Location 80). Random House, Inc.. Kindle Edition.
137. [Pink1], Pink, Daniel H. (2011-04-05). Drive: The Surprising Truth About What Motivates Us (Kindle Locations 77-78). Riverhead Books. Kindle Edition.
138. [Pink2], Pink, Daniel H. (2006-03-07). A Whole New Mind: Why Right-Brainers Will Rule the Future . Penguin Group. Kindle Edition.
139. [Poppendieck], Poppendieck M. Poppendieck T., Lean Software Development: an Agile Toolkit, Addison and Wesley, 2003
140. [Porter1], Porter M., Competitive Advantage: creating and sustaining superior performance, The Free Press, 1985 1998.
141. [Porter2], Porter M., Competitive Strategy: Techniques for Analyzing Industries ad Competitors, The Free Press, 1980, 1998.
142. [PEAA] Martin Fowler, Patterns for Enterprise Application Architecture, Addison-Wesley Professional, 2002.
143. [PLOPD1] James O. Coplien, Douglas Schmidt, Pattern Languages of Program Design, Addison-Wesley Professional, 1995.
144. [PLOPD2] John Vlissides, James O. Coplien, Norman L. Kerth, Pattern Languages of Program Design 2, Addison-Wesley Professional, 1996.
145. [PLOPD3] Robert C. Martin, Dirk Riehle, Frank Buschmann, Pattern Languages of Program Design 3, Addison-Wesley Professional, 1997.
146. [PLOPD4] Brian Foote, Neil Harrison, Hans Rohnert, Pattern Languages of Program Design 4, Addison-Wesley Professional, 1999.

147. [PLOPD5] Dragos Manolescu, Markus Voelter, James Noble, Pattern Languages of Program Design 5, Addison-Wesley Professional, 2006.
148. [POSA1] Frank Buschmann and Regine Meunier, Pattern-Oriented Software Architecture, A System of Patterns: Volume 1 (Wiley Software Patterns Series), Dec 7, 2000 | Kindle eBook.
149. [POSA2] Douglas C. Schmidt and Michael Stal, Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects: Volume 2 (Wiley Software), Apr 19, 2013 | Kindle eBook.
150. [Peopleware], Tome DeMarco and Tim Lister, Peopleware: Productive Projects and Teams, Dorset House, 1987.
151. [PWC], Price Waterhouse Coopers, "15th Annual Global CEO Survey 2012," available at <http://www.pwc.com/gx/en/ceo-survey/pdf/15th-global-pwc-ceo-survey.pdf>.
152. [PowerofScrum] Sutherland, Jeff; van Solingen, Rini; Rustenberg, Eelco (2012-01-31). The Power of Scrum, Kindle Edition.
153. [ProcessControlTheory], Ogunnaike Babatunde A. and Harmon Ray W., Process Dynamics, Modeling and Control, Oxford University Press, 1994.
154. [ProfitPatterns] Ted Moser, Kevin Mundt, James A. Quella, Adrian J. Slywotzky; Profit Patterns: 30 Ways to Anticipate and Profit from Strategic Forces Reshaping Your Business, Crown, 1999.
155. [ProfitZone], Slywotzky, Adrian J.; Morrison, David J.; Andelman, Bob (2007-12-18). The Profit Zone: How Strategic Business Design Will Lead You to Tomorrow's Profits (Kindle Location 127). Random House, Inc.. Kindle Edition.
156. [Rawsthorne] Dan Rawsthorne, Scaling Scrum with Scrum, <https://leanpub.com/PPSAD>
157. [Reinertsen] Donald G Reinertsen, The Principles of Product Development Flow: Second Generation Lean Product Development; Celeritas Publishing; Kindle Edition, 2014.
158. [ReinventingGiants] Bill Fischer, Umberto Lago, Fang Liu; Reinventing Giants: How Chinese Global Competitor Haier Has Changed the Way Big Companies Transform, Amazon Digital Services, Inc., Kindle Edition
159. [ReinventingOrganizations], Laloux, Frederic, Reinventing Organizations: A Guide to Creating Organizations Inspired by the Next Stage of Human Consciousness, Nelson Parker, Kindle Edition, 2014.
160. [RisingManns] Rising, Linda Ph.D.; Manns, Mary Lynn Ph.D. (2004-10-04). Fearless Change: Patterns for Introducing New Ideas (Kindle Location 165). Pearson Education (US). Kindle Edition.
161. [RunningLean] Maurya, Ash (2012-02-24). Running Lean: Iterate from Plan A to a Plan That Works (Lean (O'Reilly)) (Kindle Location 2). O'Reilly Media. Kindle Edition.

162. [SAFe] Scaled Agile Framework, <http://scaledagileframework.com/>
163. [ScalingUp] Verne Harnish, Scaling Up: How a Few Companies Make It...and Why the Rest Don't (Rockefeller Habits 2.0), Amazon Digital Services, Inc., Kindle Edition.
164. [ScenarioPlanning], Wade, Woody (2012-03-14). Scenario Planning: A Field Guide to the Future (Kindle Location 59). John Wiley and Sons. Kindle Edition.
165. [Schiel] James; Schiel (2012-05-14). Enterprise-Scale Agile Software Development (Applied Software Engineering Series) (Page 19). CRC Press. Kindle Edition.
166. [Schliep] Andreas Schliep, Scaled Principles, <http://www.scaledprinciples.org>
167. [Schwaber] Ken Schwaber, The Enterprise and Scrum, Microsoft Press, 2007.
168. [SchwaberSutherland] Schwaber, Ken; Sutherland, Jeff (2012-03-23). Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, And Leave Competitors In the Dust (p. 4). John Wiley and Sons. Kindle Edition.
169. [Sutherland] Jeff Sutherland, The Art of Doing Twice the Work in Half the Time, Crown Business, New York, 2014.
170. [ScrumGuide] Jeff Sutherland and Ken Schwaber, Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game, <http://scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf#zoom=100>, July 2013.
171. [ScrumHBR] Lucia Miree, John Galletly, Scrums, Sprints, Spikes and Poker: Agility in a Bulgarian Software Company, <http://hbr.org/product/scrums-sprints-spikes-and-poker-agility-in-a-bulga/an/W12802-PDF-ENG?Ntt=Scrum>
172. [ScrumPLOP] <http://www.scrumplop.org>
173. [Stacey], Ralph D. Stacey, Complexity and Creativity in Organizations, Berrett-Koehler Publishers; 1 edition (January 15, 1996).
174. [SmartTribes] Comaford, Christine (2013-05-30). SmartTribes: How Teams Become Brilliant Together (p. 233). Penguin Group US. Kindle Edition.
175. [Snowden–Cynefin] Snowden, David J. (July 2000). "Cynefin, A Sense of Time and Place: an Ecological Approach to Sense Making and Learning in Formal and Informal Communities". Conference proceedings of KMAC at the University of Aston.
176. [Snowden-ComplexActs] Snowden, David J. (2002). "[Complex acts of knowing: paradox and descriptive self-awareness](#)". *Journal of Knowledge Management* **6** (2): 100–111.

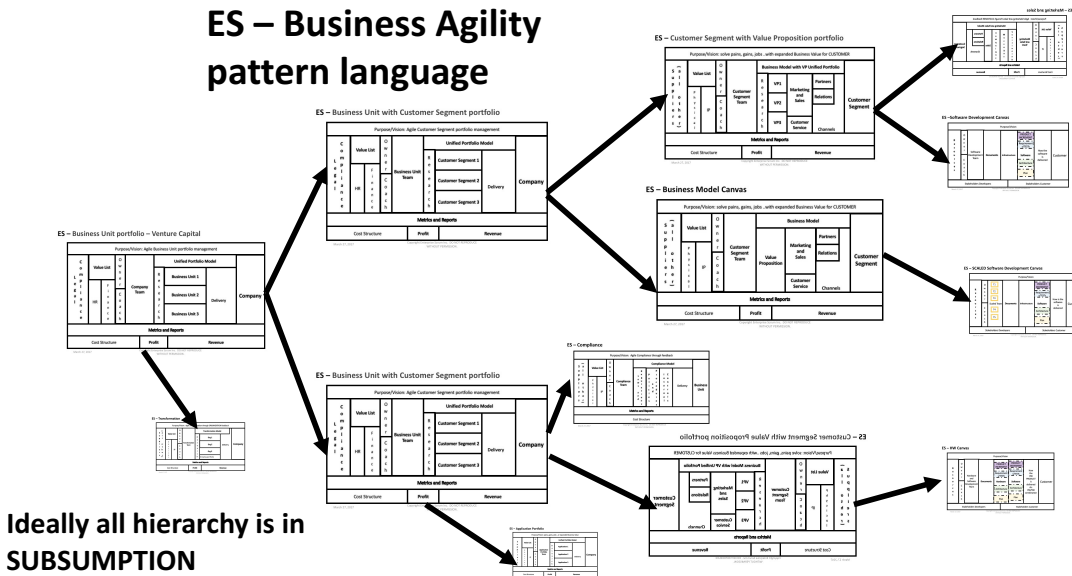
177. [Snowden-MultiOntology] Snowden, David (2005). "[Multi-ontology sense making – a new simplicity in decision making](#)". *Informatics in Primary Health Care* **13** (1): 45–53. Retrieved 6 January 2013.
178. [StandishGroup] Standish Group, The Chaos Manifesto, 2012.
179. [StandishGroup–ChaosReport] Standish Group, The Chaos Report, 2016.
180. [StoberHansmann] Stober, Thomas; Hansmann, Uwe (2009-11-19). Agile Software Development: Best Practices for Large Software Development Projects (Kindle Location 5). Springer. Kindle Edition.
181. [Sutherland] Jeff Sutherland, Scrum: The Art of Doing Twice the Work in Half the Time, Random House, 2014.
182. [Swarm] Swarm Behavior, http://en.wikipedia.org/wiki/Swarm_behaviour
183. [ThirdWave] Steve Case, The Third Wave: An Entrepreneur's Vision of the Future, Simon and Schuster Digital Sales Inc., Kindle Edition, 2016.
184. [TWI] Training Within Industry, http://en.wikipedia.org/wiki/Training_Within_Industry.
185. [TribalLeadership] Logan, Dave; King, John; Fischer-Wright, Halee (2012-01-03). Tribal Leadership: Leveraging Natural Groups to Build a Thriving Organization (Kindle Locations 395-398). HarperCollins. Kindle Edition. [Wicked], Peter DeGrace, Leslie Hulet Stahl, "Wicked Problems, Righteous Solutions: A Catalog of Modern Engineering", Prentice Hall, 1990.
186. [UltimateQuestion] The Ultimate Question 2.0 (Revised and Expanded Edition): How Net Promoter Companies Thrive in a Customer-Driven World, Harvard Business Review Press; Kindle Edition; 2011.
187. [VPD] Alexander Osterwalder, Yves Pigneur, Gregory Bernarda, Alan Smith; Value Proposition Design: How to Create Products and Services Customers Want, Strategyzer, Amazon Digital Services, Inc., Kindle Edition.
188. [Weinberg] Gerald Weinberg, Quality Software Management (Vols 1-4), Dorset House Publishing Company, Incorporated (January 1997).
- 189.
190. [Wommack1], Womack J. Jones D. Roos D., The Machine that Changed the World, Free Press, New York, 1990, 2007.
191. [Wommack2], Womack J. Jones D., Lean Thinking: Banish Waste and Create Wealth in your Corporation, 2nd Edition, Free Press, New York, 1996, 2003.
192. [Wommack3], Womack J. Jones D., Lean Solutions, The Free Press, 2005
193. [X] Brian Solis; X: The Experience When Business Meets Design; Wiley; Kindle Edition; 2015.
194. [Yahoo] Rodd Wagner, The Seven Lessons Of Marissa Mayer's Loss Of Command At Yahoo, <http://www.forbes.com/sites/roddwagner/2016/03/08/the-seven-lessons-of-marissa-mayers-loss-of-command-at-yahoo/#68f97bf736c3>

195. [Zhexembayeva] Nadya Zhexembayeva, To Hold on, Let go, TEDx talk, <https://youtu.be/f4kySpcdvFg?list=PL96Jlk4GPOhWQSp9hH9yrmoFHN42F8qtp>
196. [ZoneToWin] Geoffrey Moore, A.; Zone to Win: Organizing to Compete in an Age of Disruption (Kindle Locations 58-62). Diversion Books. Kindle Edition.

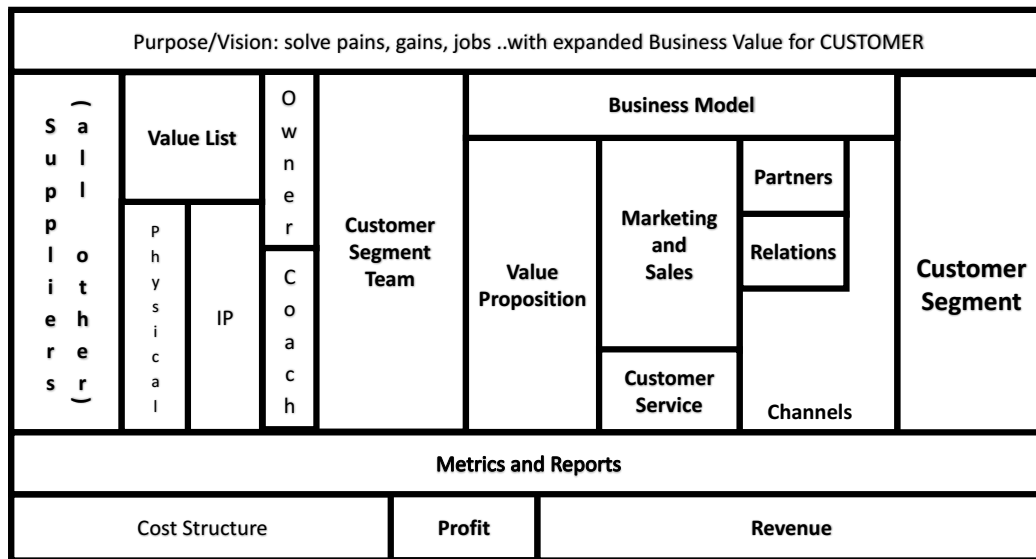
Appendix A. Other Canvases

Enterprise Scrum – Business Agility for Companies of ANY size!!!

ES – Business Agility pattern language



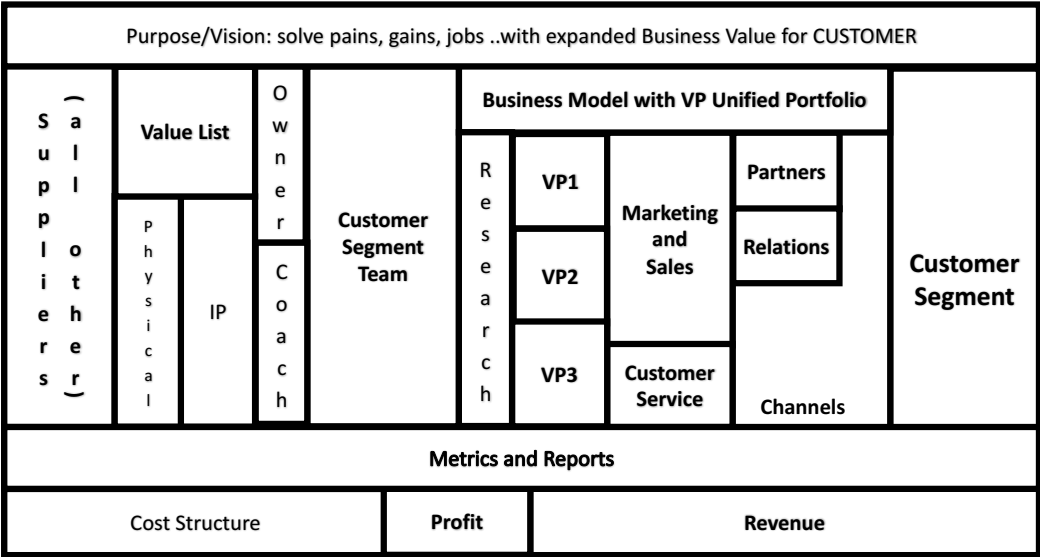
ES – Business Model Canvas



May 13, 2017

Copyright Enterprise Scrum Inc. DO NOT REPRODUCE WITHOUT PERMISSION.

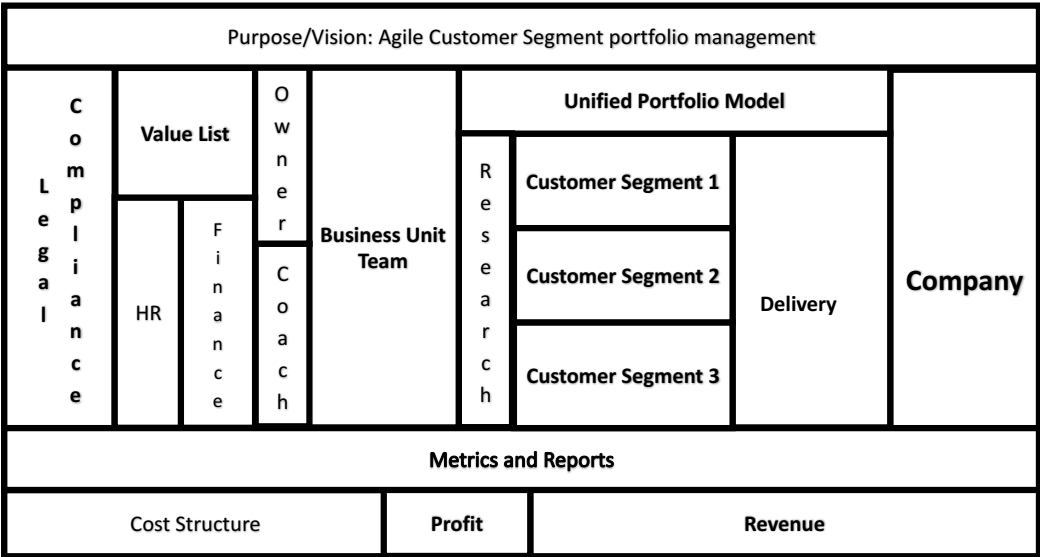
ES – Customer Segment with Value Proposition portfolio



May 13, 2017

Copyright Enterprise Scrum Inc. DO NOT REPRODUCE WITHOUT PERMISSION.

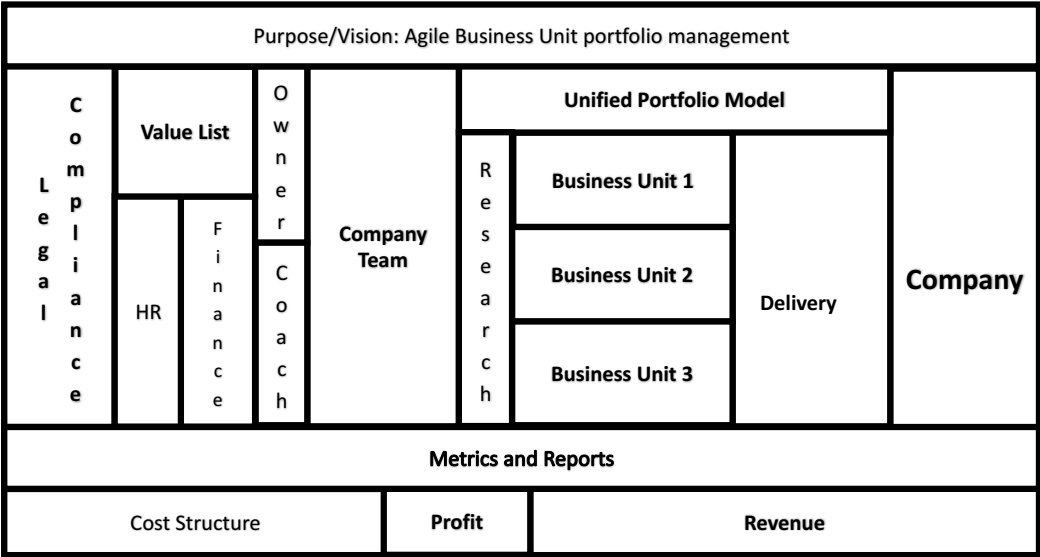
ES – Business Unit with Customer Segment portfolio



May 13, 2017

Copyright Enterprise Scrum Inc. DO NOT REPRODUCE WITHOUT PERMISSION.

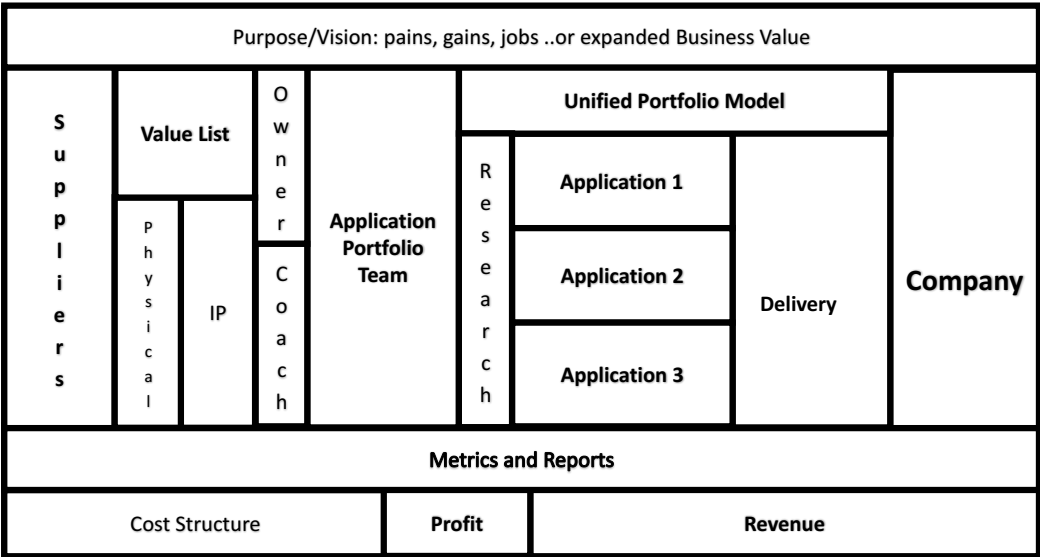
ES – Business Unit portfolio – Venture Capital



May 13, 2017

Copyright Enterprise Scrum Inc. DO NOT REPRODUCE
WITHOUT PERMISSION.

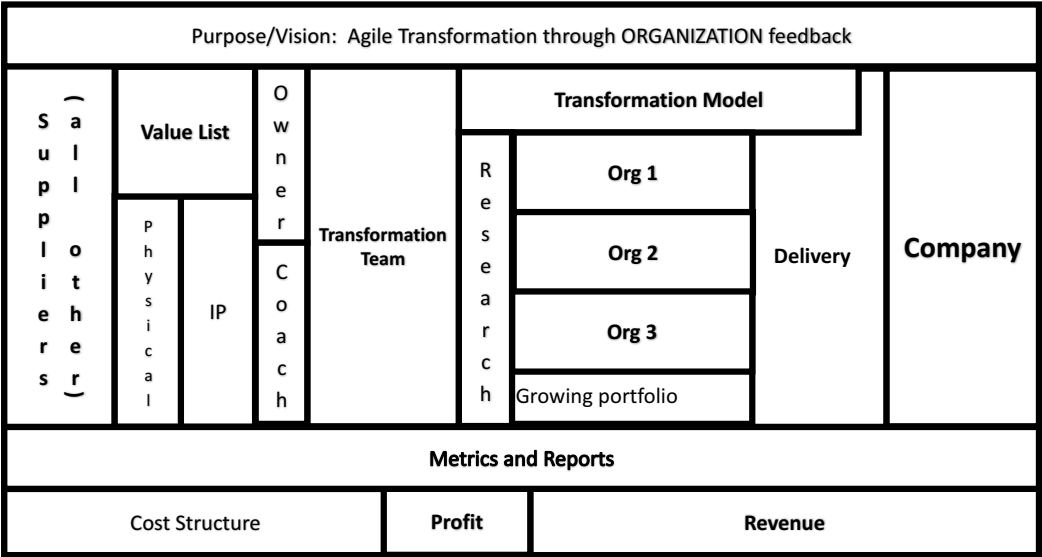
ES – Application Portfolio



May 13, 2017

Copyright Enterprise Scrum Inc. DO NOT REPRODUCE
WITHOUT PERMISSION.

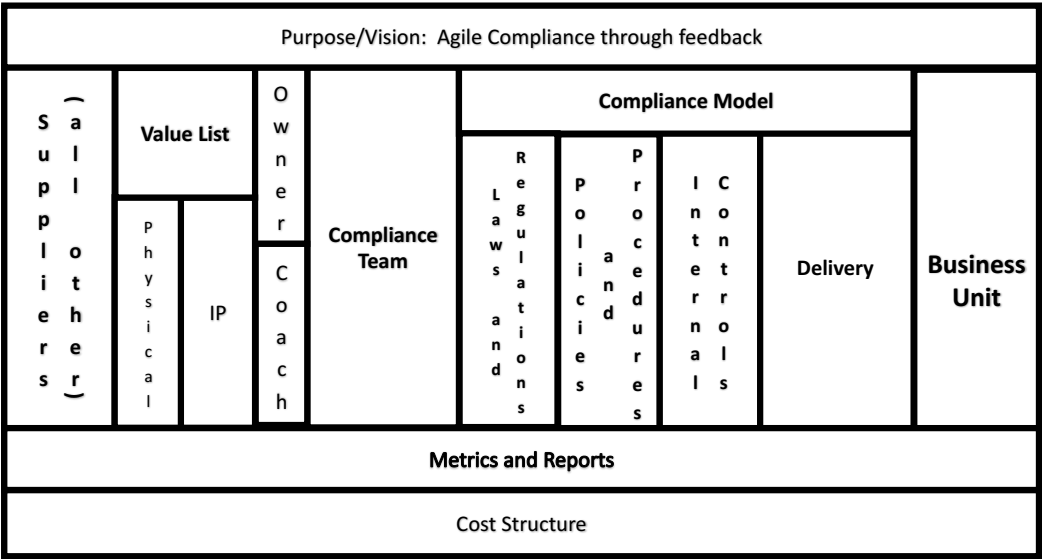
ES – Transformation



May 13, 2017

Copyright Enterprise Scrum Inc. DO NOT REPRODUCE
WITHOUT PERMISSION.

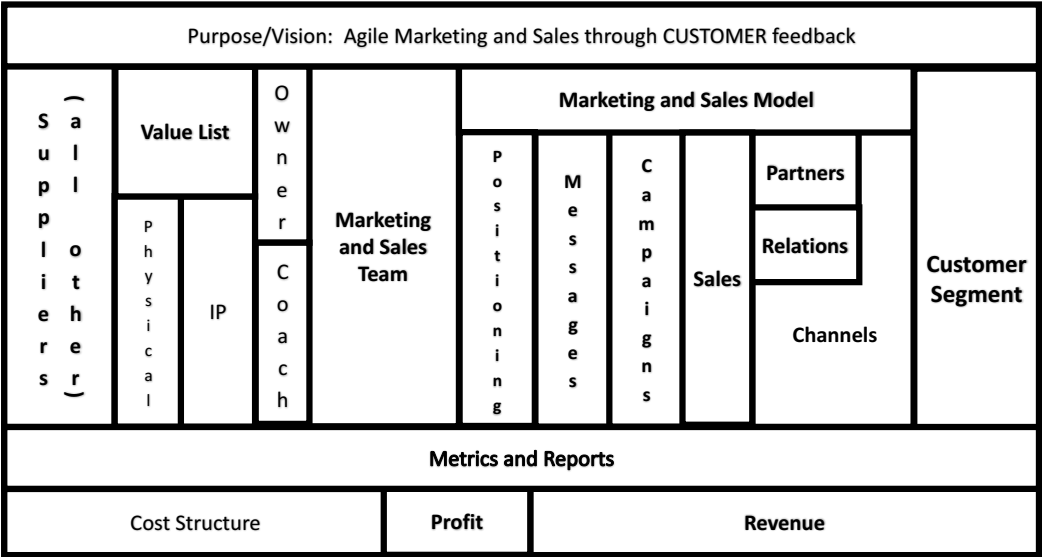
ES – Compliance



May 13, 2017

Copyright Enterprise Scrum Inc. DO NOT REPRODUCE
WITHOUT PERMISSION.

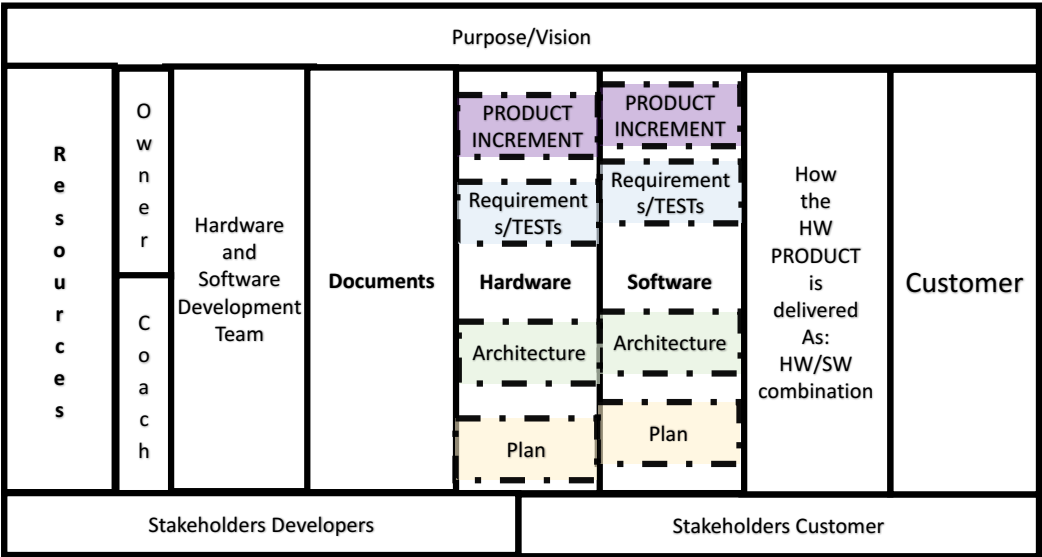
ES – Marketing and Sales



May 13, 2017

Copyright Enterprise Scrum Inc. DO NOT REPRODUCE
WITHOUT PERMISSION.

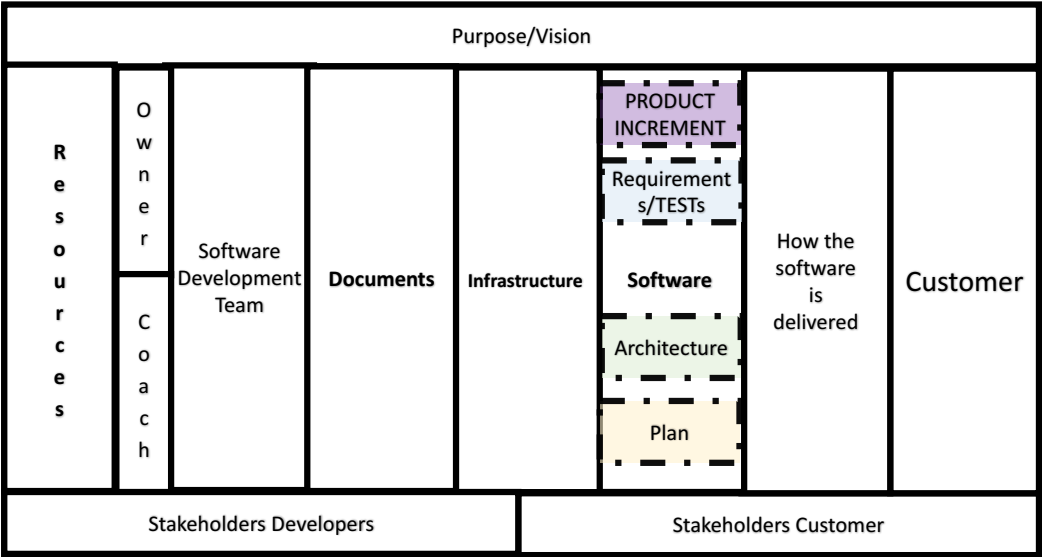
ES – HW Canvas



May 13, 2017

Copyright Enterprise Scrum Inc. DO NOT REPRODUCE
WITHOUT PERMISSION.

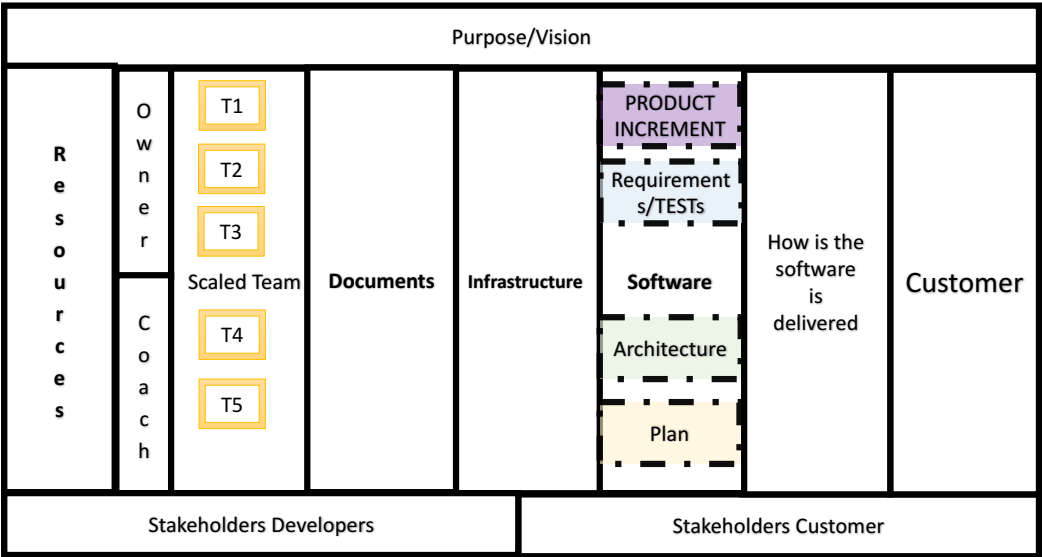
ES –Software Development Canvas



May 13, 2017

Copyright Enterprise Scrum Inc. DO NOT REPRODUCE WITHOUT PERMISSION.

ES – SCALED Software Development Canvas



May 13, 2017

Copyright Enterprise Scrum Inc. DO NOT REPRODUCE WITHOUT PERMISSION.

TEMPLATE Canvas

Purpose/Vision					
Value List	O w n e r	Team	What gets delivered? = VLITypes What are the aspects of each of the VLITypes = surfers = aspects What attributes are needed to get them DONE? = extra VLIType attributes Shared attributes Unique attributes	How to deliver, communicate	Customer
R e s o u r c e s	C o a c h				
Stakeholders Developers			Stakeholders Customer		

May 13, 2017

Copyright Enterprise Scrum Inc. DO NOT REPRODUCE WITHOUT PERMISSION.

Generic Canvas

Purpose/Vision							
Resources	Owner	Team	VLI 3 of VP	VLI 2 of VP	VLI 1 of VP	How to deliver, communicate	Customer
	Coach						
Stakeholders Developers				Stakeholders Customer			

May 13, 2017

Copyright Enterprise Scrum Inc. DO NOT REPRODUCE WITHOUT PERMISSION.

ES - ScrumBoard

Business Owner and Stakeholders

Vision

C
o
a
c
h

Initial Value List

Business Owner, Stakeholders and Team

	WIP (PCRI Cycle)	DONE
	Cycle 1	
	Cycle 2	
	Cycle 3	

May 13, 2017

Copyright Enterprise Scrum Inc. DO NOT REPRODUCE
WITHOUT PERMISSION.