Ways of Working in SE

Week 2

Taking Stock

Class Representative, Communication Protocol and Concerns – Announcement Review

Perspectives Quiz – Implications

The schedule for the course

Where are we now?

The Assessment Schedule

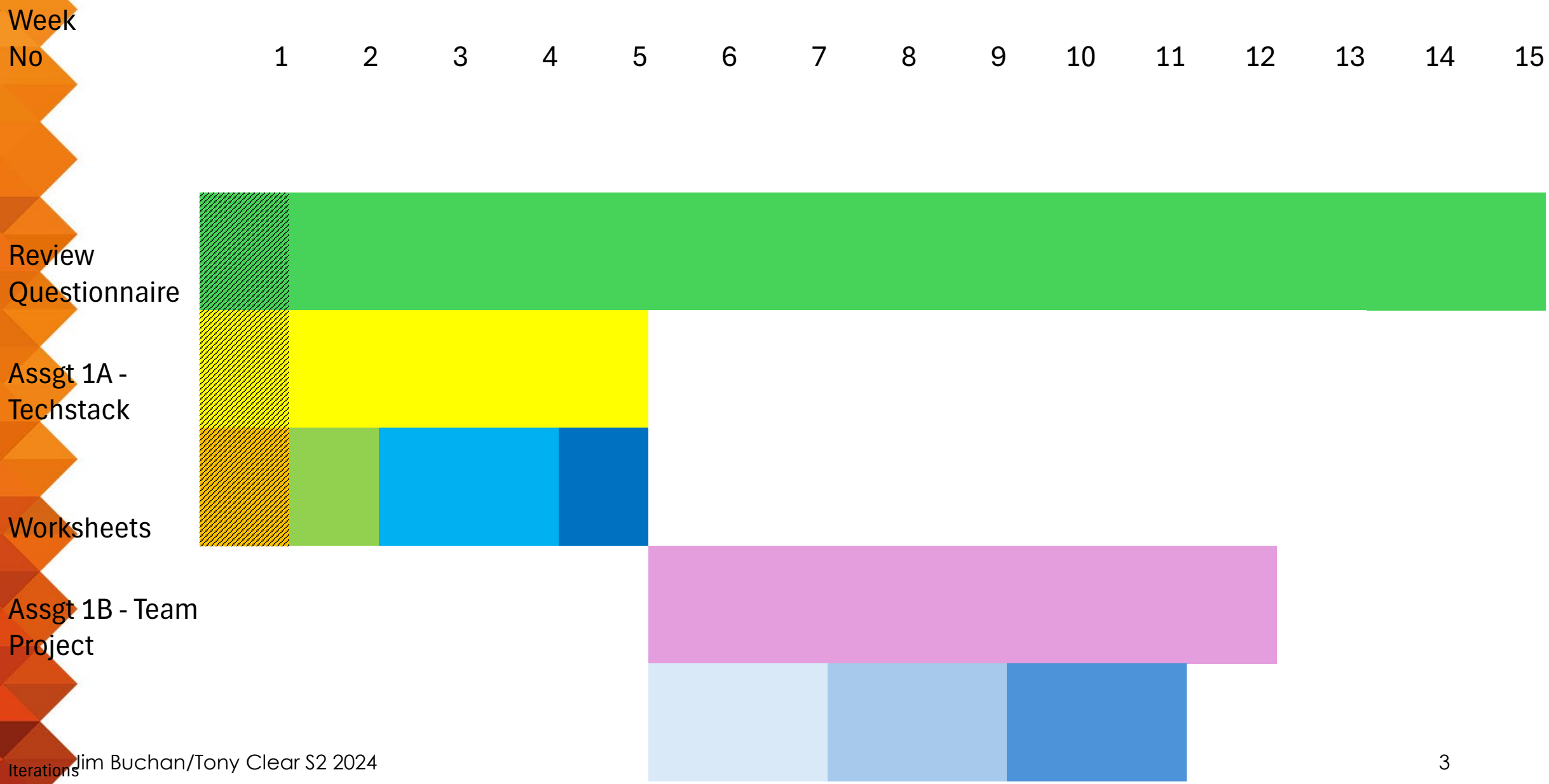
Progress – feedback, any issues?

Overview - what's coming up?

The Lecture Schedule

How does it relate to the assessment?

Taking Stock



Assignments Drive your Learning

Ass 1A preparing for Software Development (20%)
(*Individual*)

- Set up the tools an individual needs to support coding, good code craft, version control and unit testing
- Set up the tools needed to collaborate with a team to achieve product goals together

Sharing code – integrate code, review code,
Setup the tools needed to work with the selected Tech Stack (front-end/backend)
Set up tools to assure quality of product
Set up tools to deploy the product to the cloud
Set up tools to monitor and alert issues post deploy
Learn how to use the tools
Learn how to use the Tech Stack
Understand the product goals -> Product Backlog
Sprint 1 Goals -> Sprint Backlog

Submission in Tutorials weeks 1-5 (sign off by TA)
Evidence portfolio and demo

Ass1B Full SDLC full stack product Dev (50%)
(*small team - 4 Including QA*)
Capability building by Developing a Product in a small team
Apply a new tech stack and tool set
Practice DevOps and Scrum WoW
Collaborate with a Product Owner and team
Three sprints to learn fast – fast feedback
Submit – reviews weeks 7,9,11 (tutorials)
Capability and learning Portfolio with Evidence
Product increments

Sprint 1 weeks 5 and 6
Sprint 2 weeks 7 and 8
Sprint 3 weeks 9 and 10

Ass 2 Knowledge Check (30%)
(*Individual, online questions*)
A set of questions about scenarios to confirm you have understood main language and principles
Sometime in Revision weeks (Faculty schedules)

Ass1B Team Development

What do we have to **decide**, **do** and **plan** to get ready to start developing and deploying features as a team?

What types of work do we have to do and get prepared for?

Ideas?

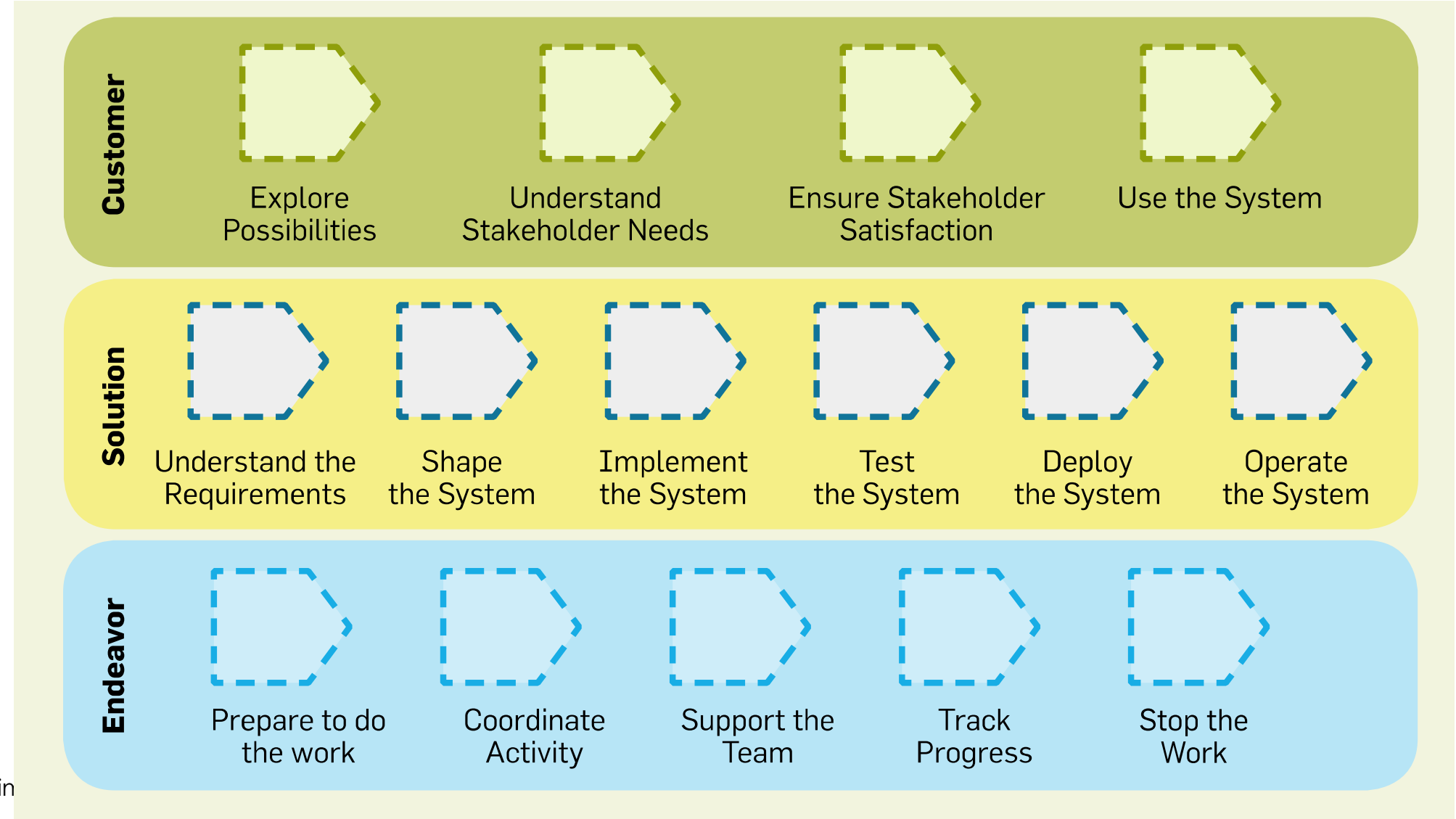
Traditional Waterfall approach

Requirements Design Build Test Deploy Maintain

Things to do – types of work

There are certain types of work that all software development includes…

SEMAT (Software Engineering Methods and Theory) (Ivor Jacobsen)



What work do software development teams have to do?

Understand the problem to be solved

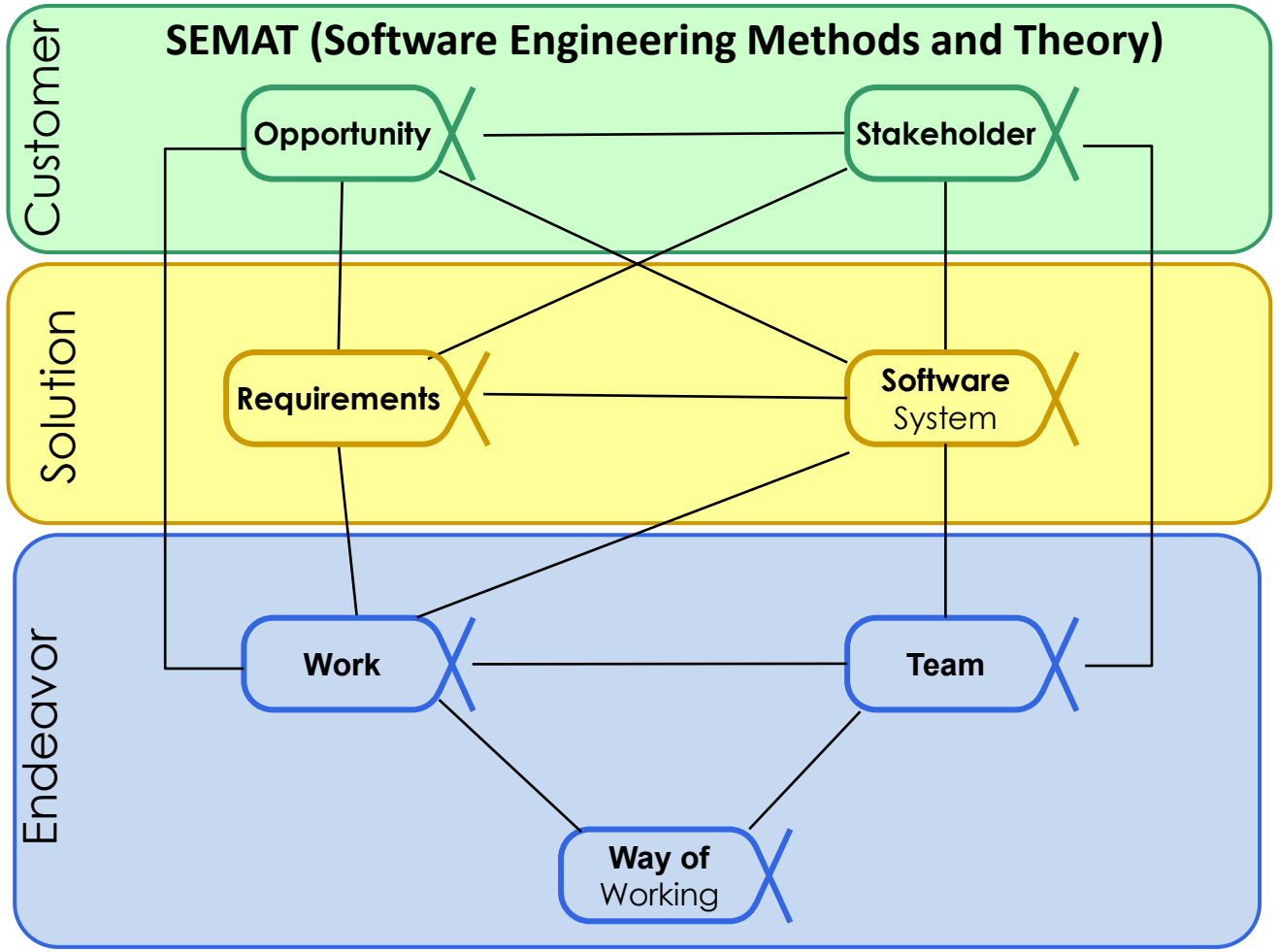
Plan the work

Design the solution

Do the work

Deliver/Deploy the solution

Maintain/update the work



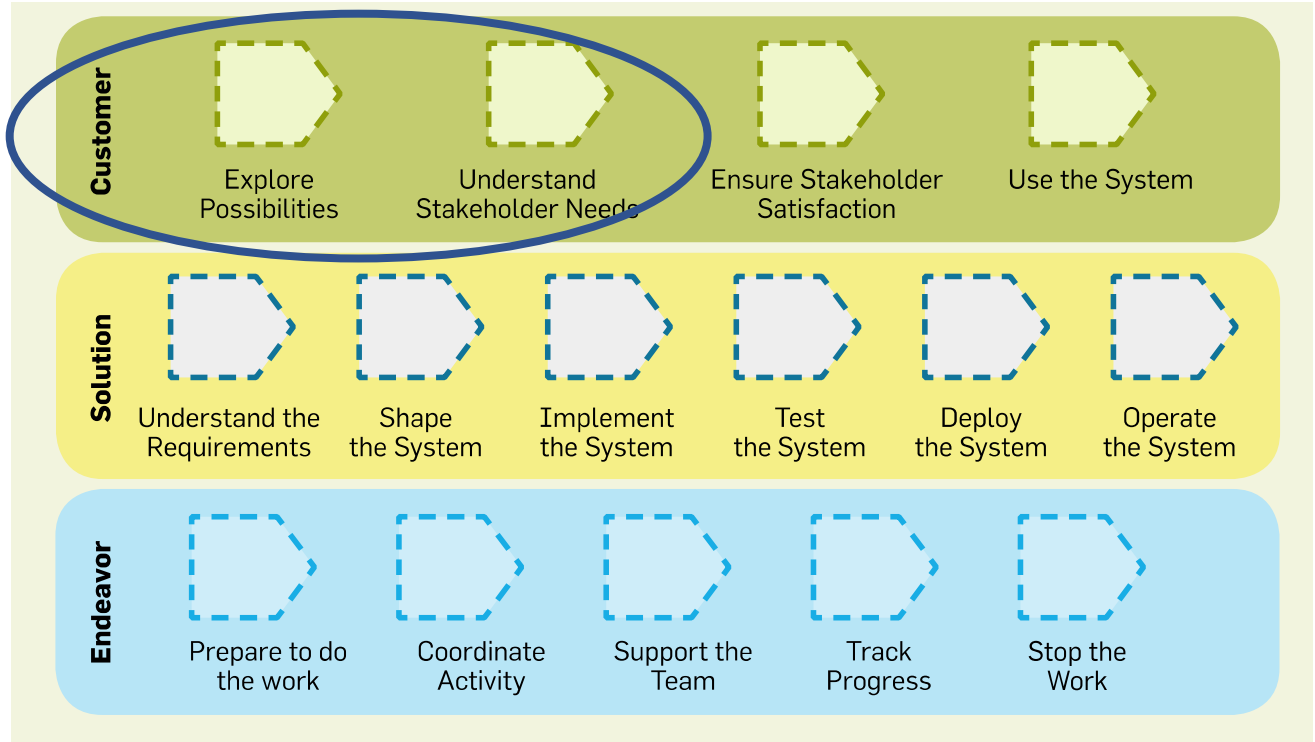
Work = ?????

IS IT LINEAR?

How do we know quality is good enough?

Who is involved and when and role?

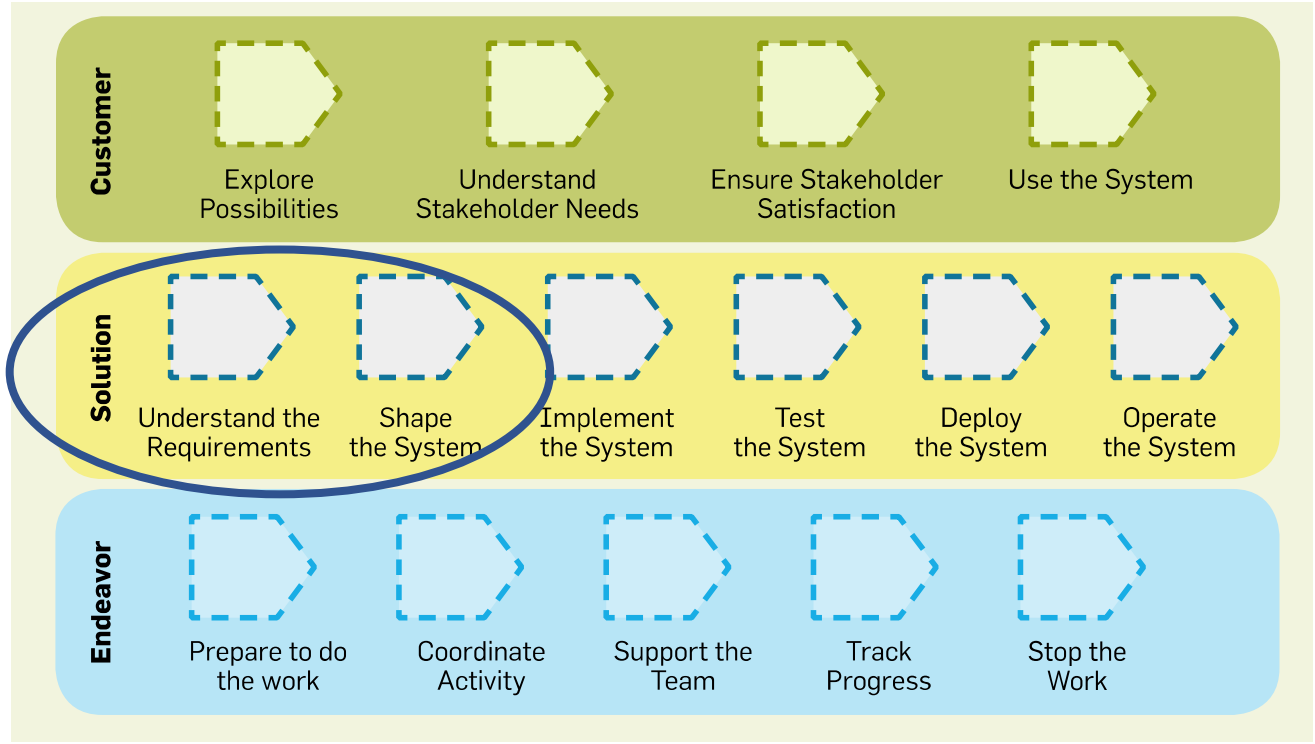
Agree on a system for working together to get this work done - time and cost and tech and skill constraints



Work closely with the users and client – frequent interactions and coordination and feedback

Need a product goal /vision we share

Iterative incremental development
-so we can learn and discover by doing and getting feedback in rapid learning loops
-so we can deliver small bits of value to the client frequently
-break down the problem and solution



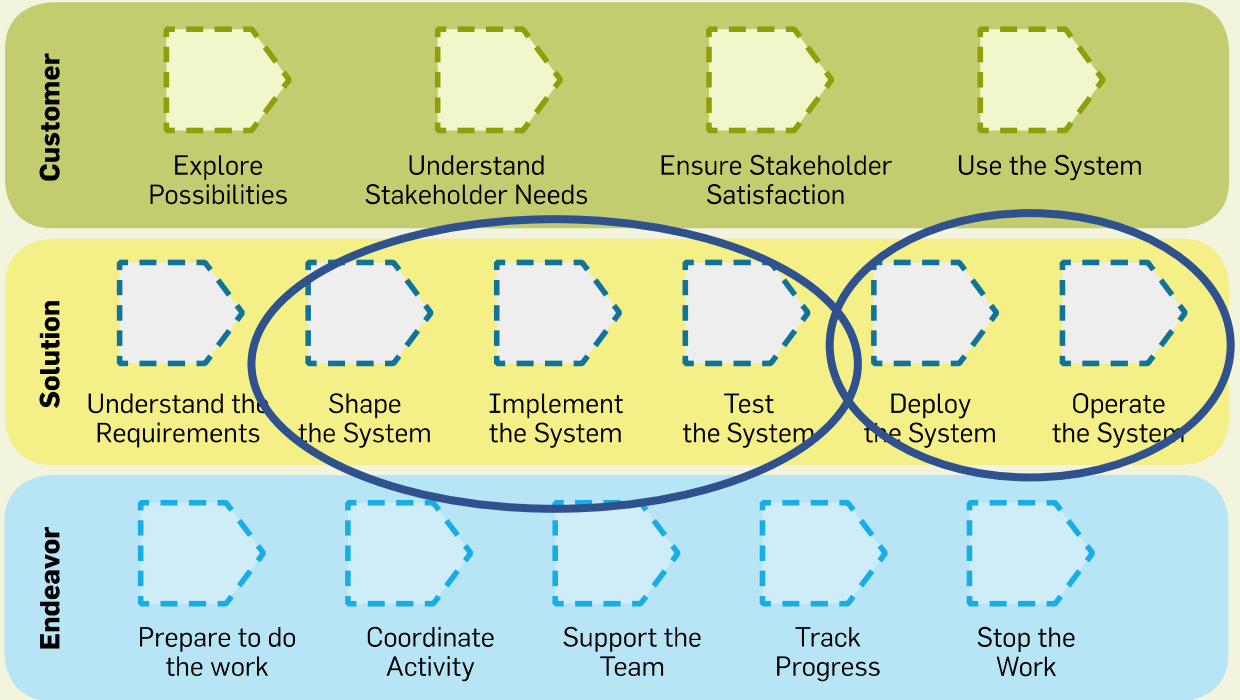
User stories

User Story Maps of product

Maintained list of what user requirements still needs to be done
-Have Iteration goals

List of user requirements for the next Iteration

Product Design incrementally



Overall architecture and tech stack

Web App
Layered architecture

React.js/NEXTjs for the UI (front-end)
MongoDB for the Document Database
Nest/Express.js and Node.js for back end

Iterative and incremental

Git and GiHub for version control and sharing and merging code

Continuous Integration (GitHub)

Continuous Deployment

Test-driven development

Pair and mob programming

Unit tests

Acceptance tests (BDD)

Other tests?

Cloud based deploy - Heroku

Let's talk to the client (Product Owner PO) Peter

Goal – get a vision of what the problem is and why it is a problem and what needs to change if our product is used.

End with a Product Goal statement

Start to break the problem into EPIC User stories

Principles

Ask OPEN question

Active Listening

Stay in Problem space

Write things down – these will become user stories

Engaging with the client (Product Owner PO) Peter

Active Listening

Reimold, C. (1991). 'Hey, don't listen to me like that!' How the way you listen can make or break communication. IPCC 91 Proceedings The Engineered Communication, https://ieeexplore.ieee.org/abstract/document/172722

Active listening—

Showing you have thoroughly understood the information imparted

Active listening has two parts to it:

1. You listen closely to get the essence of what the speaker is saying.

2. You restate what you think he said to make sure you've understood. ("Let me see if I've understood up to here." "Do you mean...?")

Creative listening—

Building on the speaker's ideas to form something new: a new idea, a new application, a new solution to a problem

Supportive listening—

Giving the speaker emotional sustenance by showing you empathize with his or her feelings

Continuous Integration in a nutshell

Local development machine with Git.
Local files with code, tests, configurations etc
Modify
(Use VS Code to change Code and Test)
Commit
(Snapshot with description of change and why and add to local repo)

GitHub with linked repo

With code from all developers and previously developed code merged and tested.

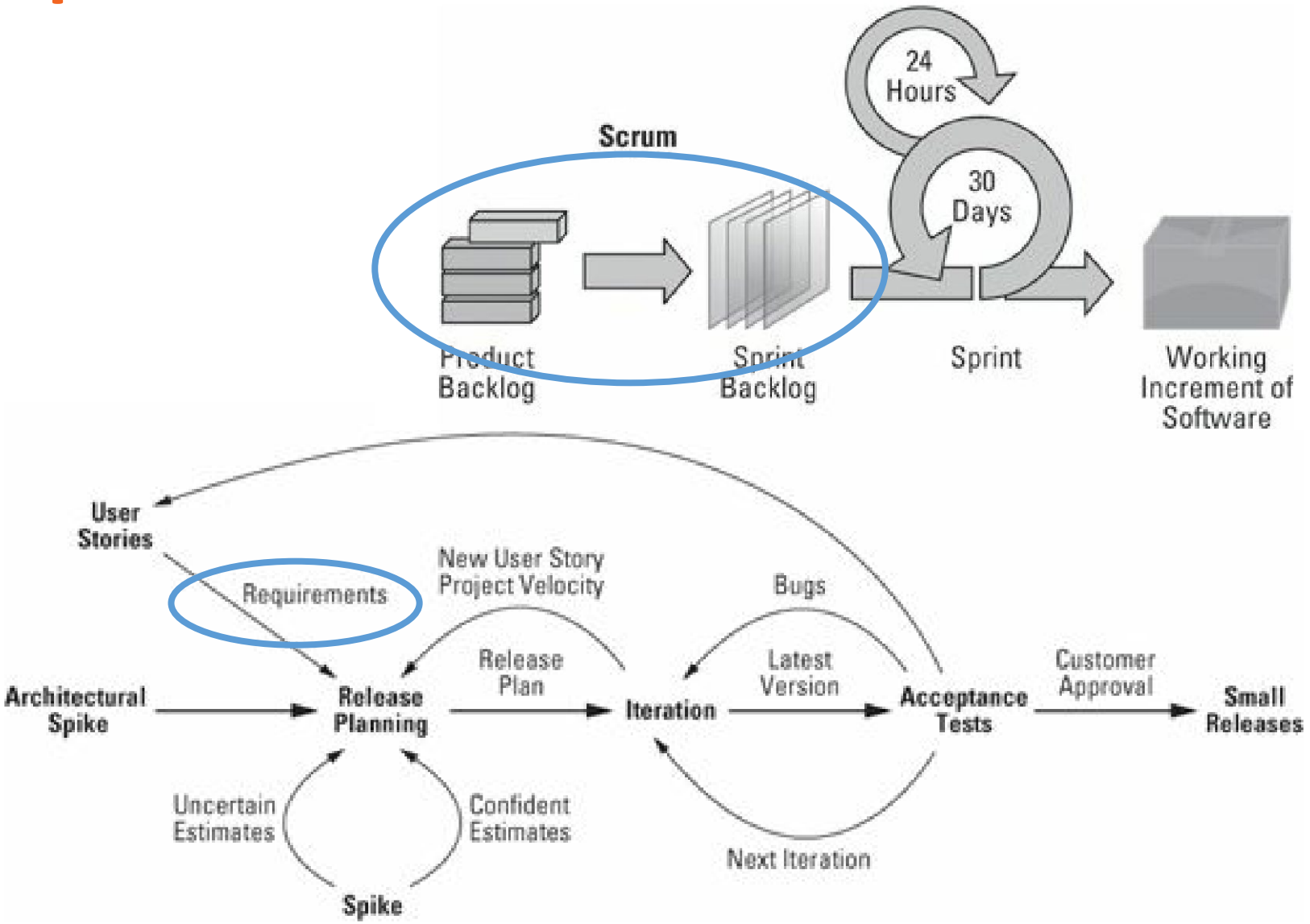
Modify
Commit
Etc
(can roll back to previous version of code, create and work in branches etc)

Ready to merge with pre-production branch codebase (integrate)
Push to team shared GitHub Repository
(Automatically) run tests on whole code based before merging – CI server?

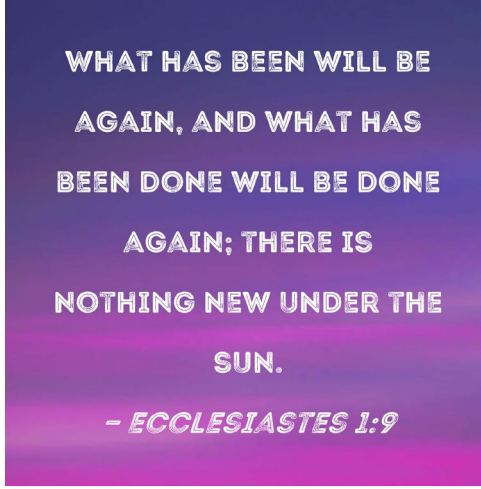
Automatically run code standard checks with Linter or SonarQube or similar
Automatically merge if pass tests

Continuous = every few hours (before feature finished)

RE in Agile is iterative – not big upfront



Iterative Enhancement – Not New !



"battle lines between proponents of

agile software development ecosystems (ASDE's) and

rigorous system development methodologies (RSM's),

based upon fundamentally different assumptions about how the world and organizations work".

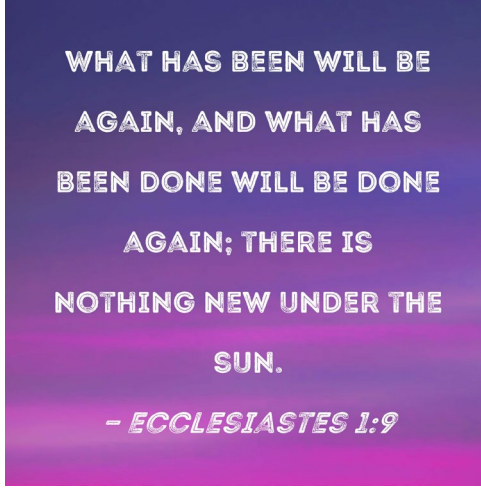
BUT

As agile methods become more popular, some view

Basil, V. R., & Turner, A. J. **(1975). Iterative enhancement: A practical technique for software development.** *IEEE Transactions on Software Engineering*(4), 390-396.

Larman, C., & Basili, V. R. (2003). **Iterative and incremental developments. a brief history**. *Computer, 36*(6), 47-56. doi:10.1109/MC.2003.1204375

Iterative Enhancement – 1975 and earlier !



iterative, evolutionary, and incremental software development—a cornerstone of these {agile} methods — as the "modern" replacement of the waterfall model,

but its practiced and published roots **go back decades**.[Larman & Basili]

The **first step** in the application of the iterative enhancement technique to a software development project consists of a **simple initial implementation of a skeletal subproblem** of the project. (Basili & Turner)

Basil, V. R., & Turner, A. J. **(1975). Iterative enhancement: A practical technique for software development.** *IEEE Transactions on Software Engineering*(4), 390-396.
Larman, C., & Basili, V. R. (2003). **Iterative and incremental developments. a brief history**. *Computer, 36*(6), 47-56. doi:10.1109/MC.2003.1204375

Iterative Enhancement – Early Backlogs?

This skeletal implementation **acts as an initial guess** in the process of developing a final implementation which meets the complete set of project specifications.

A ***project control list*** is created that **contains all the tasks that need to be performed** in order to achieve the desired final implementation.

At (1ny given point in the process, the project control list acts as a measure of the "distance" between the current and final implementations.

Basil, V. R., & Turner, A. J. **(1975). Iterative enhancement: A practical technique for software development.** *IEEE Transactions on Software Engineering*(4), 390-396.

Iterative Enhancement – Updating the control list?

In the **remaining steps** of the technique the **current implementation is iteratively enhanced** until the final implementation is achieved.

Each iterative step consists of selecting and removing **the next task from the list**, **designing** the implementation for the selected task (the *design phase),* **coding and debugging** the implementation of the task (the *implementation phase),* performing an **analysis of the existing partial implementation** developed at this step of the iteration (the *analysis phase),* and **updating the project control list** as a result of this analysis.

The **process is iterated** until the project control list is empty, i.e., until a final implementation is developed that meets the project specifications.

Basil, V. R., & Turner, A. J. **(1975). Iterative enhancement: A practical technique for software development.** *IEEE Transactions on Software Engineering*(4), 390-396.

A Requirements Refinery for Agile Software Product management

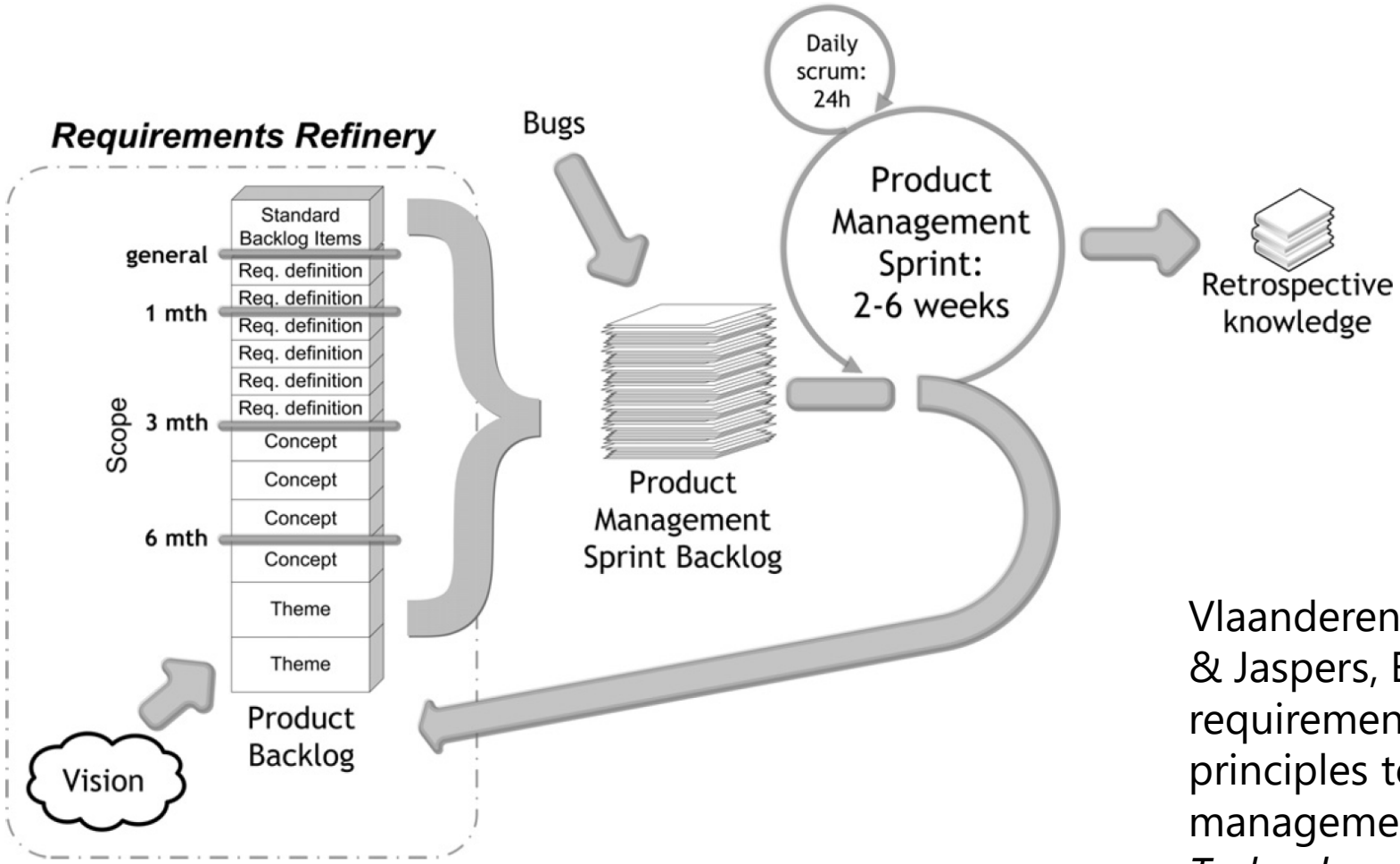


Fig. 1. Agile SPM knowledge flow.

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011, January). The agile requirements refinery: applying SCRUM principles to software product management. *Information and Software Technology, 53*(1), 58-70. doi:10.1016/j.infsof.2010.08.004

Software Product management and Software Development

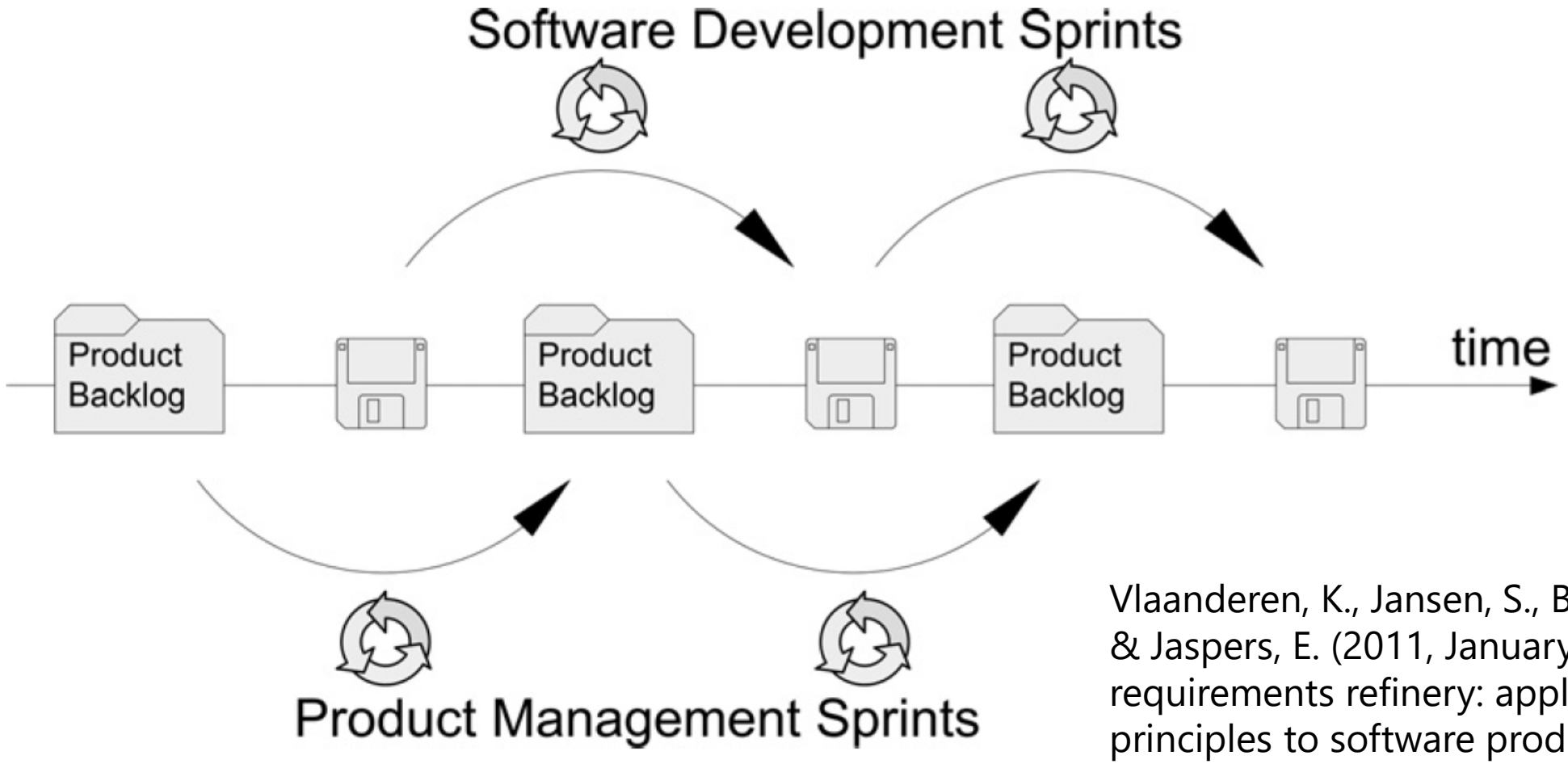


Fig. 3. Alternating sprints.

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011, January). The agile requirements refinery: applying SCRUM principles to software product management. *Information and Software Technology, 53*(1), 58-70. doi:10.1016/j.infsof.2010.08.004

Software Product management Process

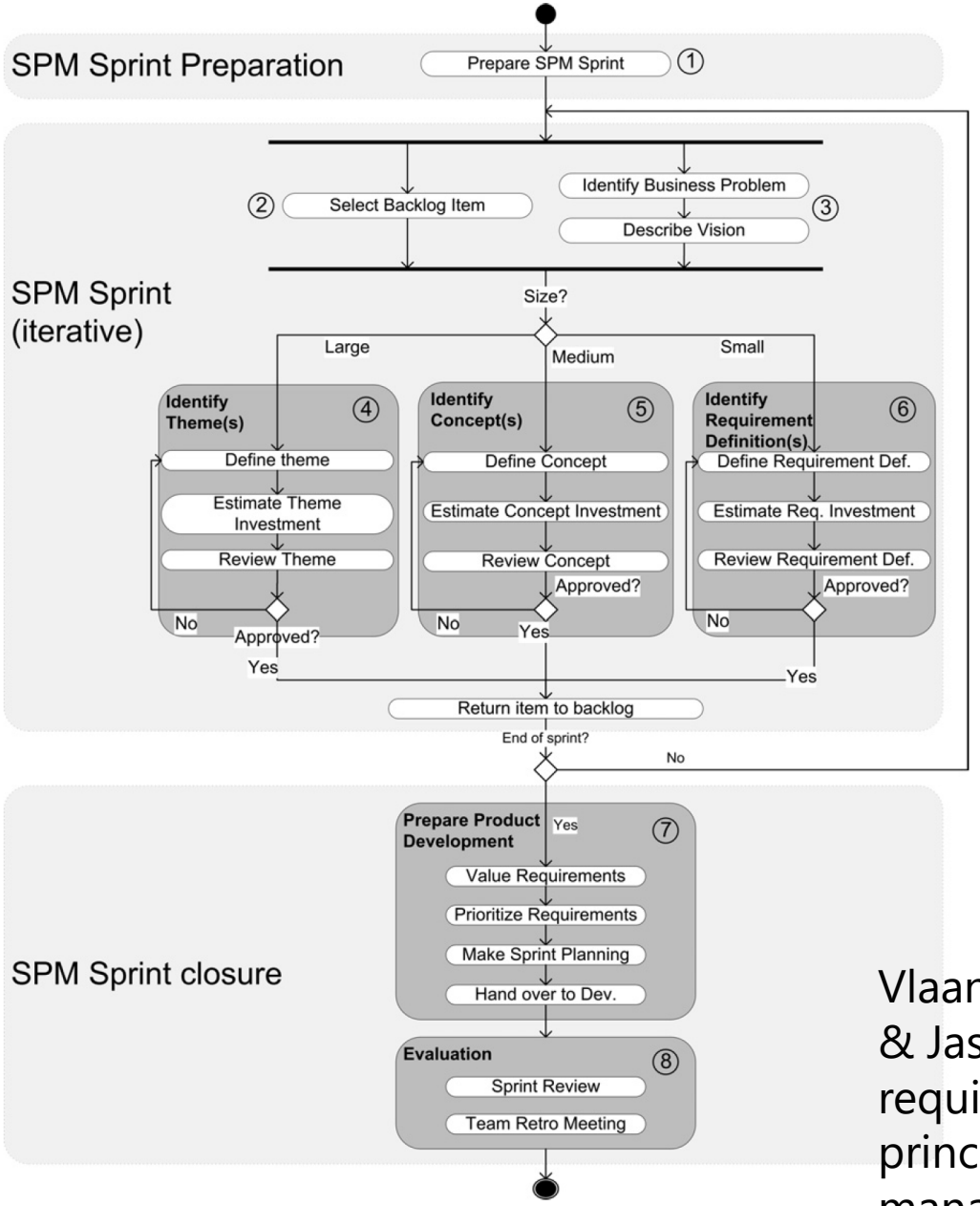
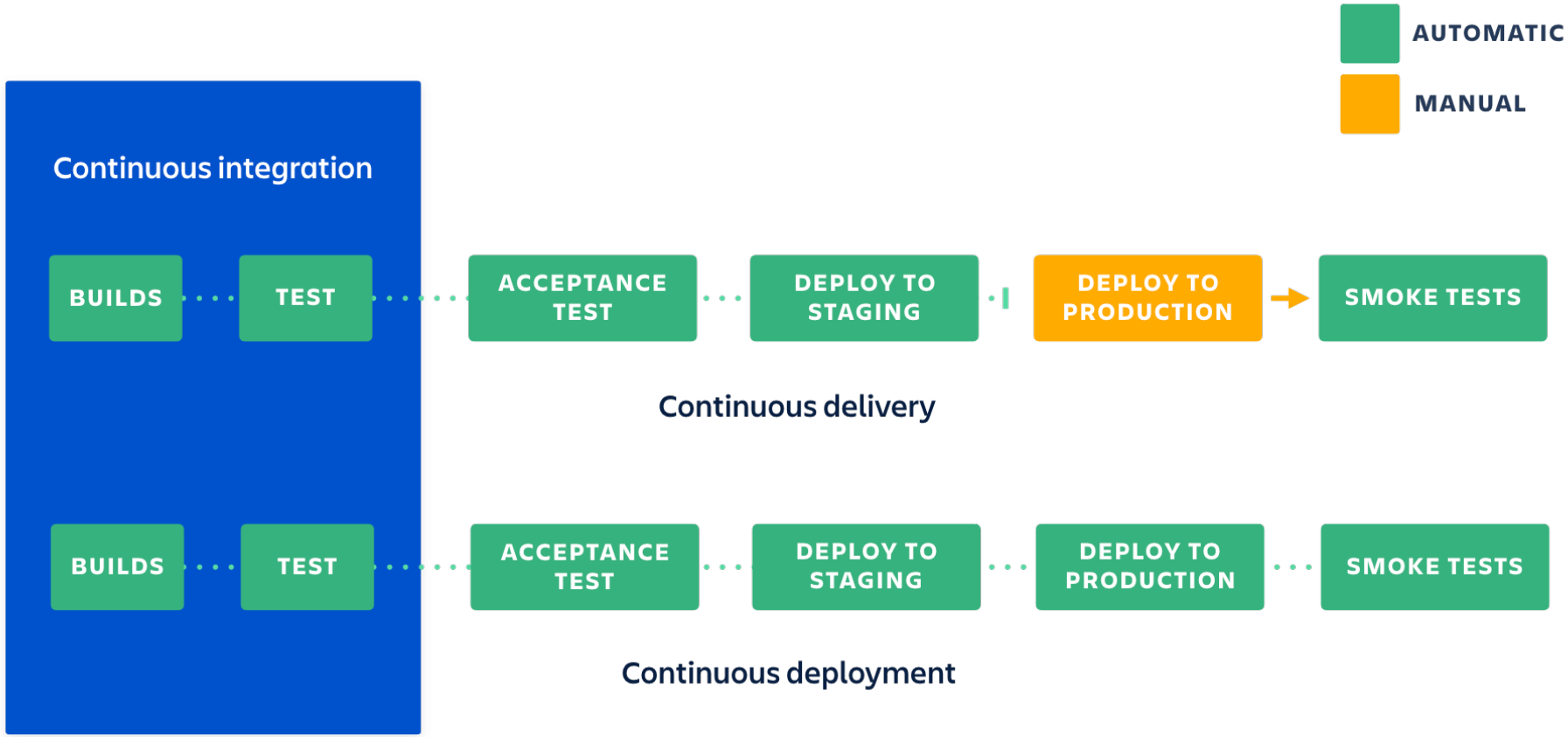


Fig. 2. SCRUM software product management process.

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011, January). The agile requirements refinery: applying SCRUM principles to software product management. *Information and Software Technology, 53*(1), 58-70. doi:10.1016/j.infsof.2010.08.004

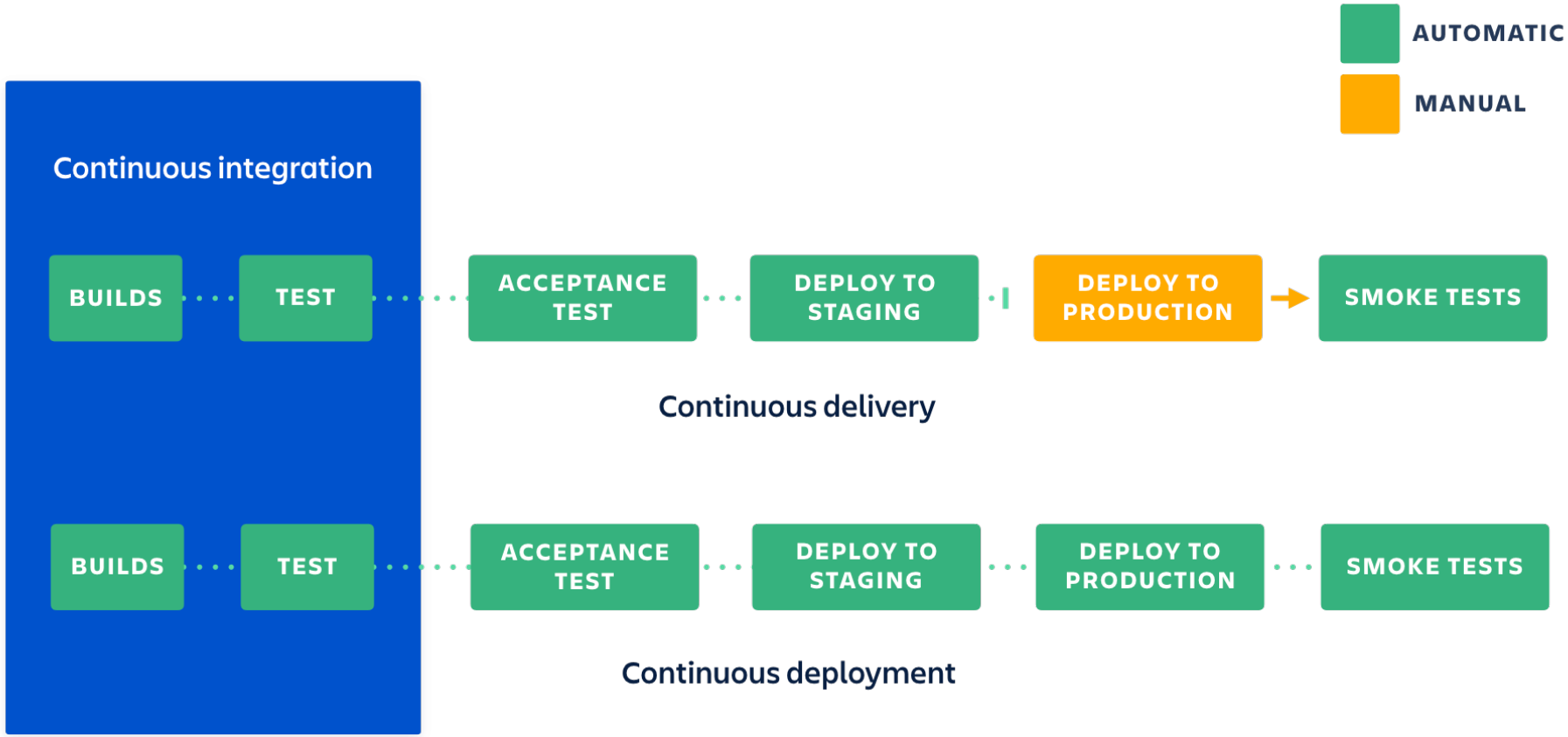
CI/Continuous Deployment (or Delivery) Overview

https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment



CI/Continuous Integration in a Nutshell

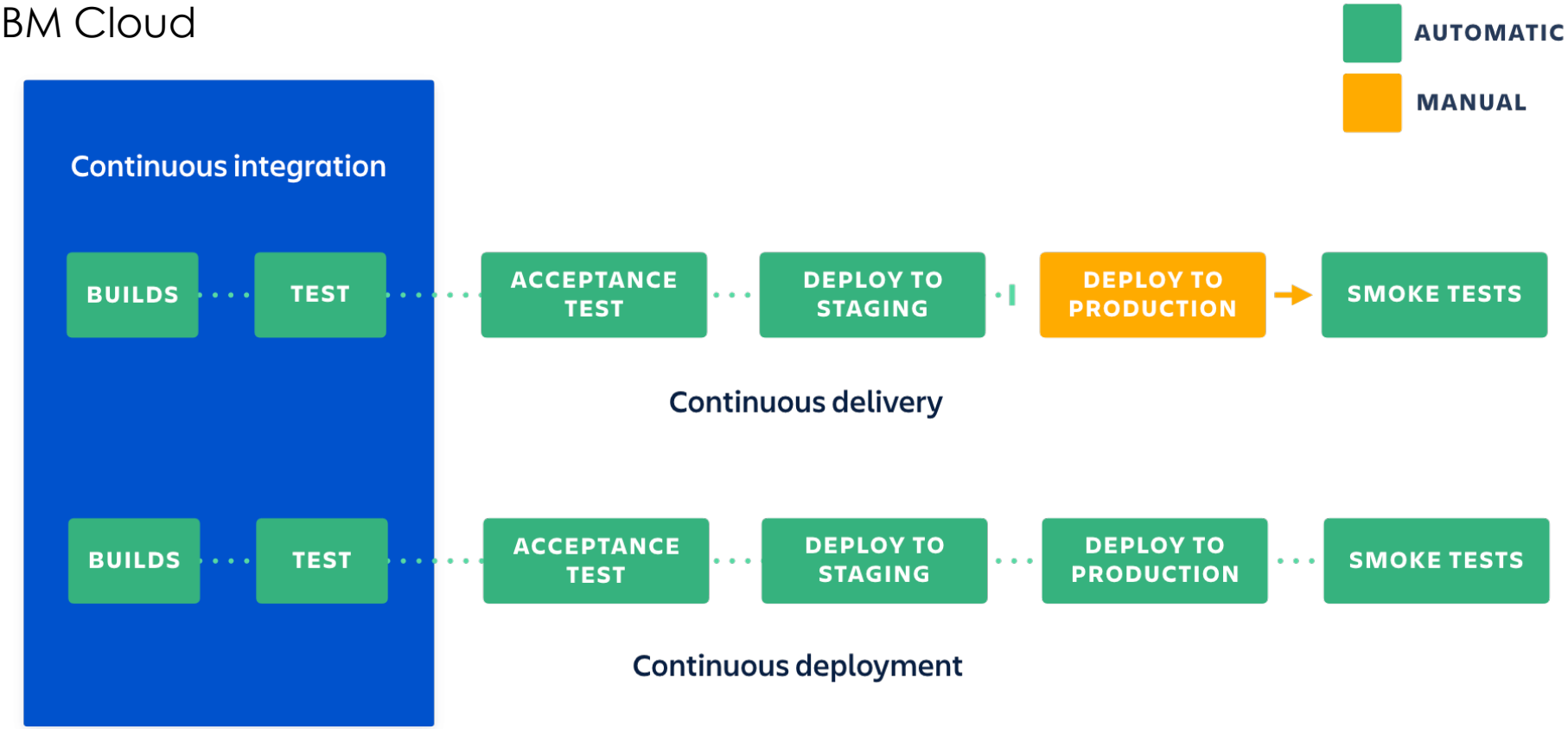
https://www.youtube.com/watch?v=1er2cjUq1UI
Eric Minick IBM Cloud



CI/CD Continuous Deployment (or Delivery) in a Nutshell

https://www.youtube.com/watch?v=2TTU5BB-k9U

https://www.youtube.com/watch?v=LNLKZ4Rvk8w
Eric Minick IBM Cloud

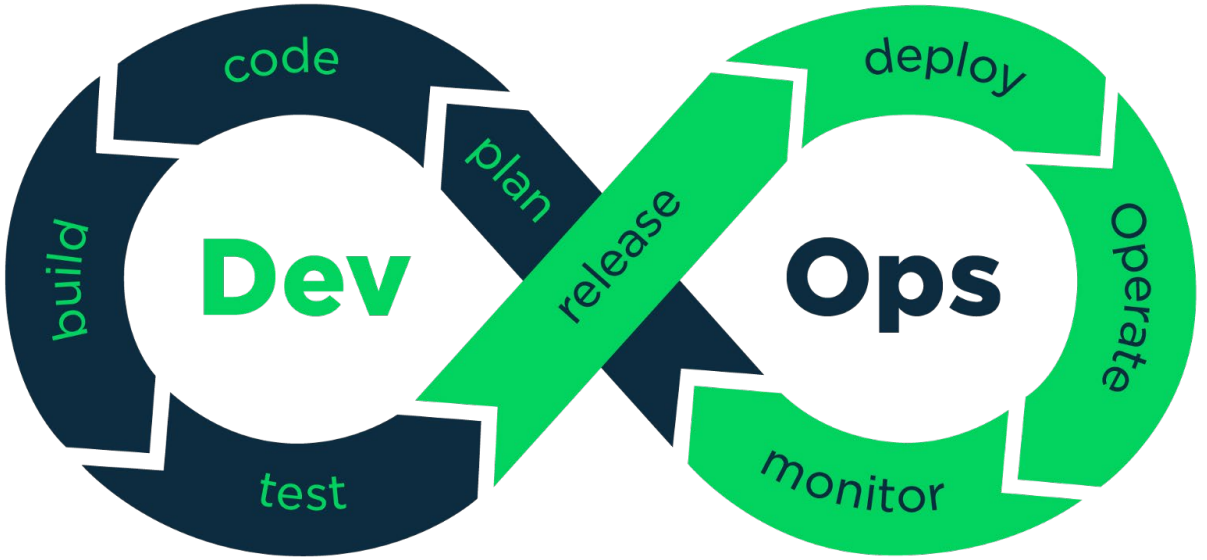


DevOps as a WoW and values – we will come back to this

Values

- Deploy/release frequently (several time a day?)
- Own the product
- Team accountability extends to deployment/release and post-release

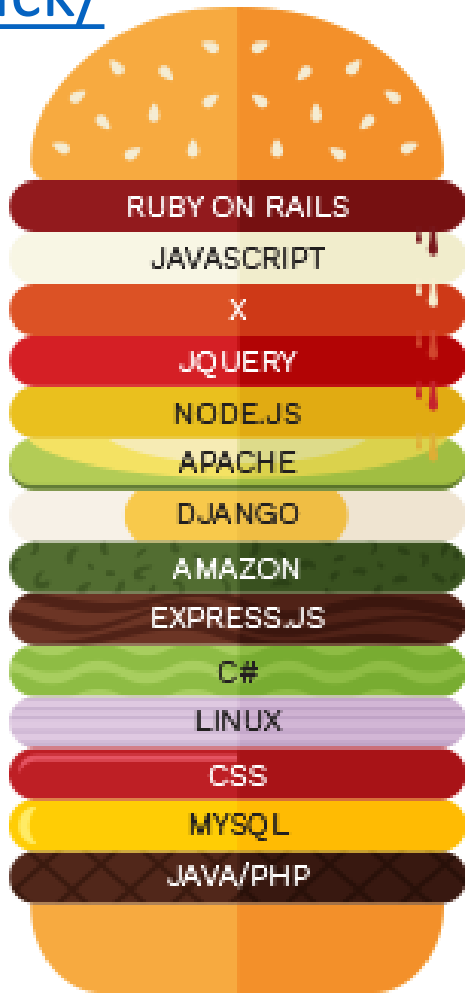
Enablers



- test automation, (CI/CD automation)
- deploy automation,
- infrastructure as code
- rollback/fix forward,
- A/B and canary testing,
- feature flags,
- microservices,
- increased observability -monitoring, alerting and logging post release
- Psychological safety

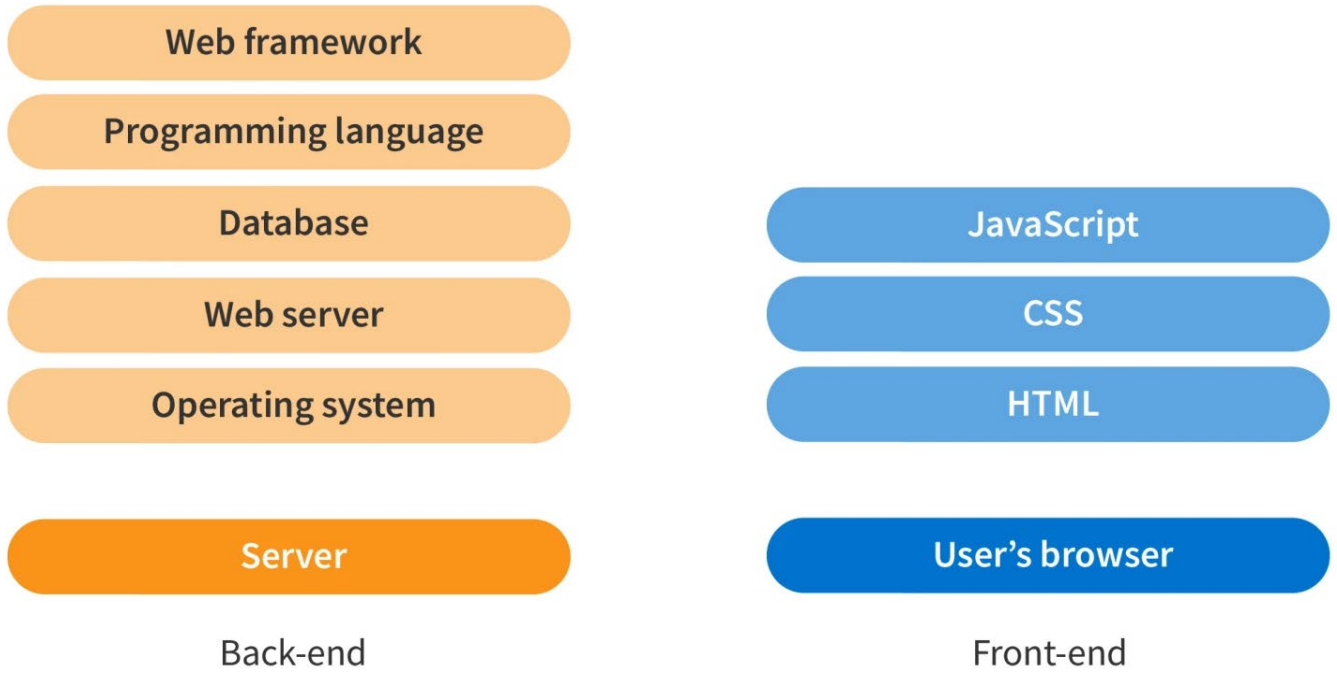
Technology stacks and full-stack developers

https://www.thesoftwareguild.com/blog/build-your-own-technology-stack/



Technology stacks and strategy

https://www.aha.io/roadmapping/guide/it-strategy/technology-stack

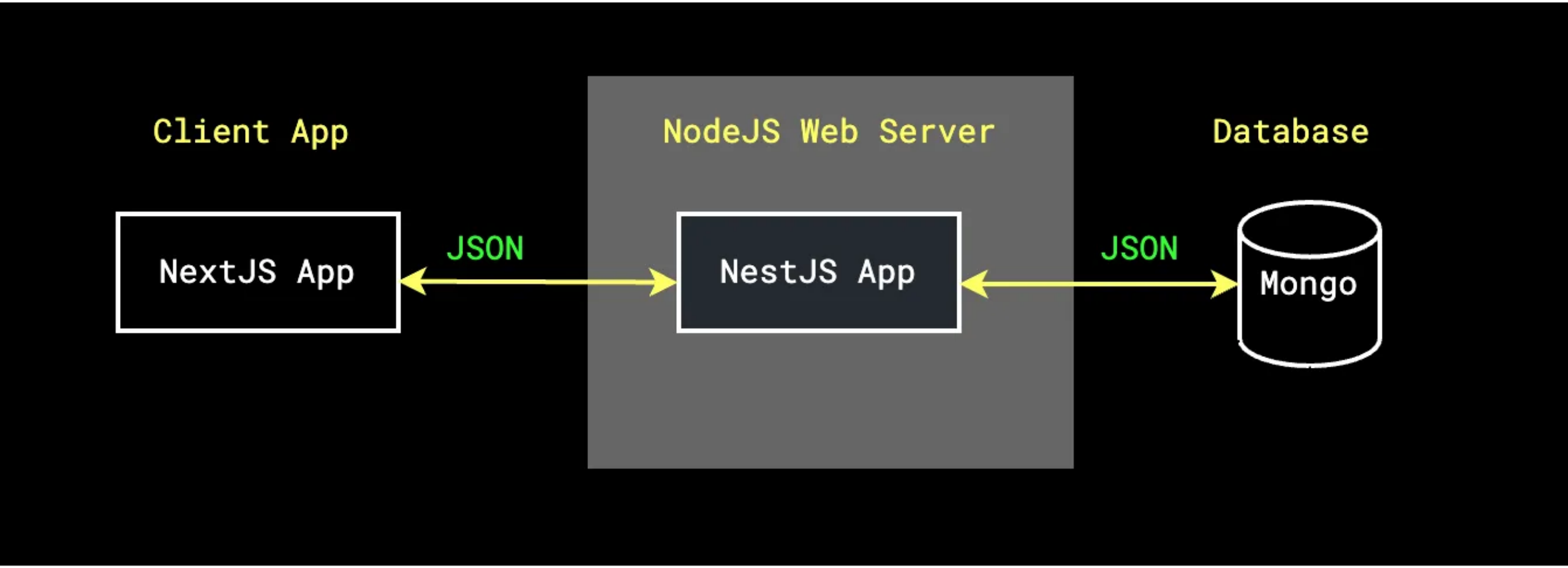


MNNN Stack in a nutshell

Readings

https://medium.com/aws-tip/the-backend-part-of-mnnn-stack-mongodb-nestjs-nextjs-and-nodejs-6bde9adfedd9
https://medium.com/aws-tip/understanding-nestjs-architecture-f257d054211d

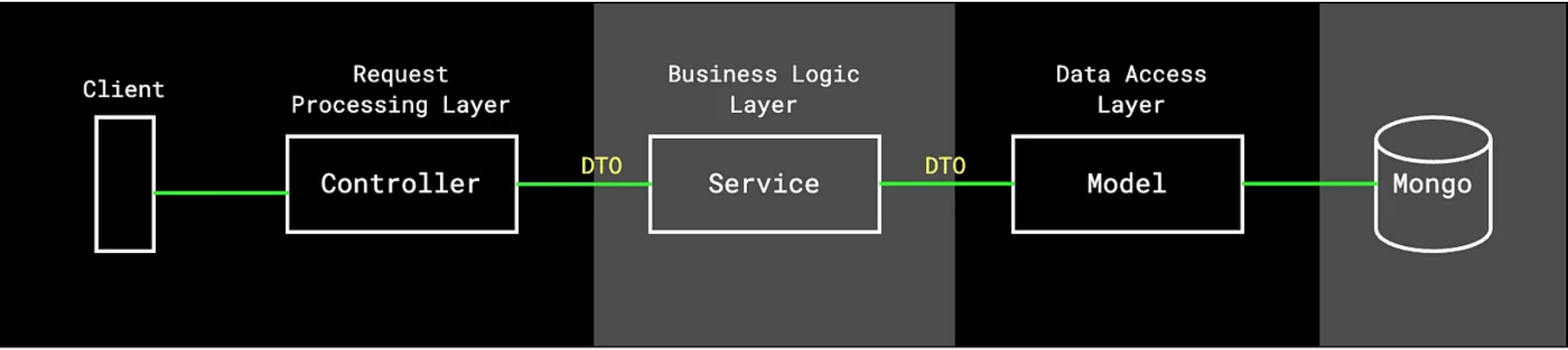
MNNN Components



Read this

https://medium.com/aws-tip/the-backend-part-of-mnnn-stack-mongodb-nestjs-nextjs-and-nodejs-6bde9adfedd9

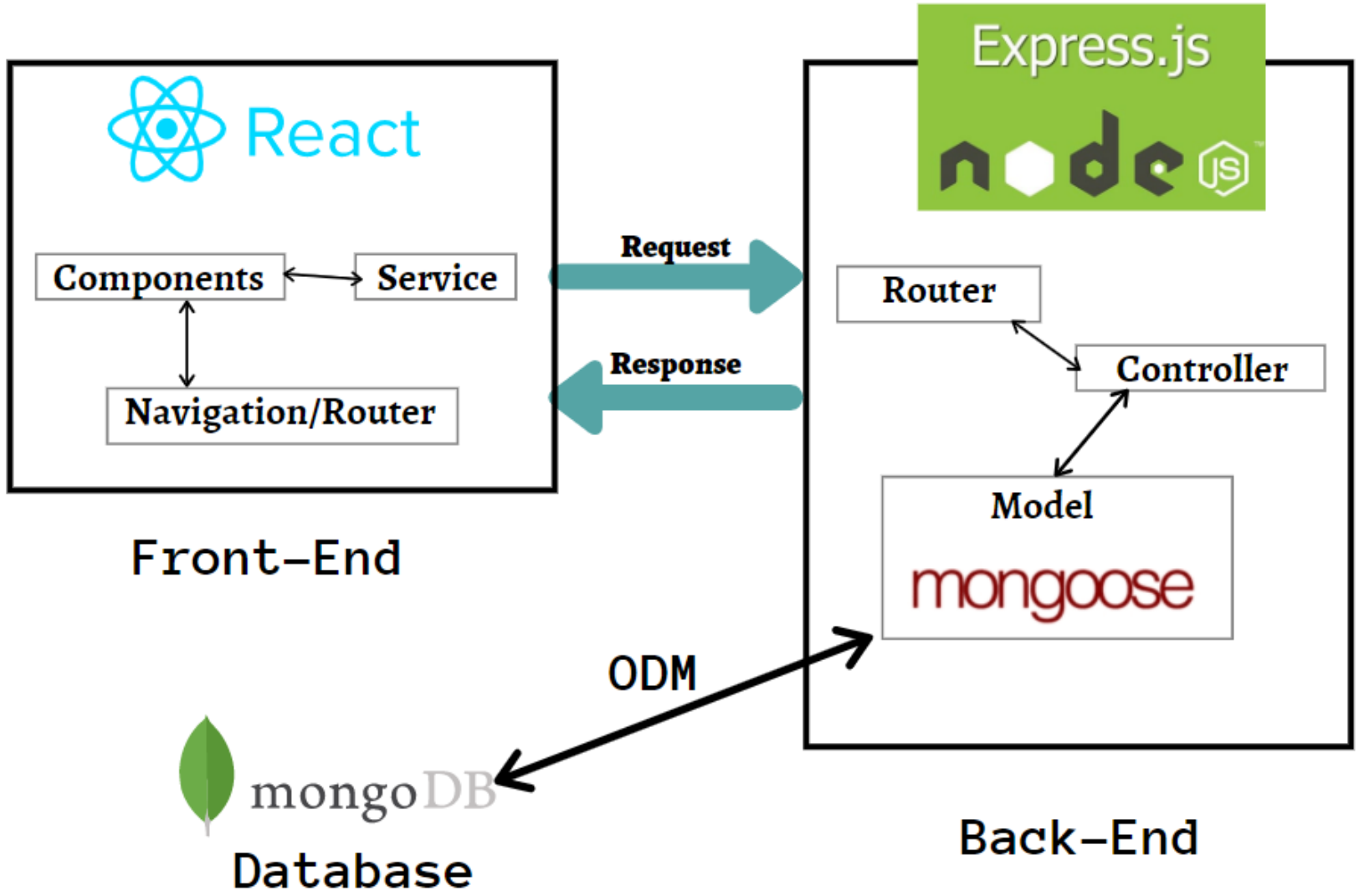
MNNN Layered Architecture



Read this

https://medium.com/aws-tip/the-backend-part-of-mnnn-stack-mongodb-nestjs-nextjs-and-nodejs-6bde9adfedd9

An Alternative - MERN Stack in a nutshell



Read this

https://medium.com/techiepedia/what-exactly-a-mern-stack-is-60c304bffbe4

MongoDB

Document based noSQL database
Cloud based service – Mongo Atlas

Mongodb.com

```json
{
  "_id": "5cf0029caff5056591b0ce7d",
  "firstname": "Jane",
  "lastname": "Wu",
  "address": {
    "street": "1 Circle Rd",
    "city": "Los Angeles",
    "state": "CA",
    "zip": "90404"
  },
  "hobbies": ["surfing", "coding"]
}
```

Express.js and Node.js

Express - routing and server (middleware)

Node.js is a javascript runtime so javascript applications can be run outside the browser

Expressjs.com

Nodejs.org

Nest.js

Nest.js Crash Course 2023: A Comprehensive Step-by-Step Tutorial

Sakura Dev

But like all random resources off the internet they need to be viewed critically :-)

Video

https://www.youtube.com/watch?v=Hv70fn8xTL4

Agile values

Agile Manifesto – 4 key Values

12 key Principles

Modern Agile – 4 key values

Heart of Agile – 4 key actions

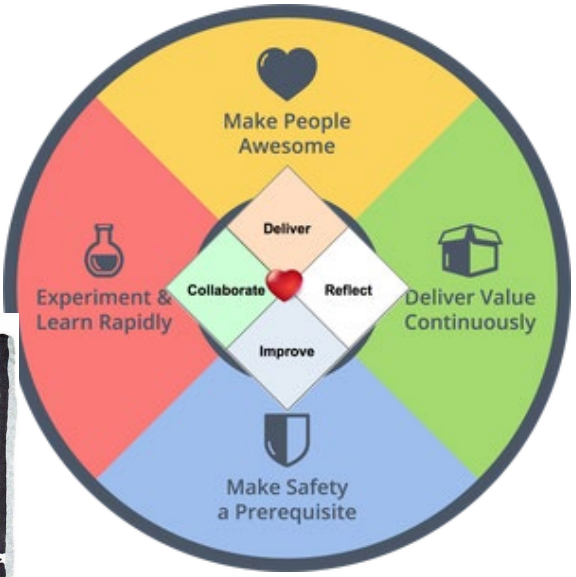
Transparency
Inspect and Adapt

Scrum WoW

Guidelines for working based on empiricism– iterative and incremental workflow. Values and 3 pillars.

DevOps WoW

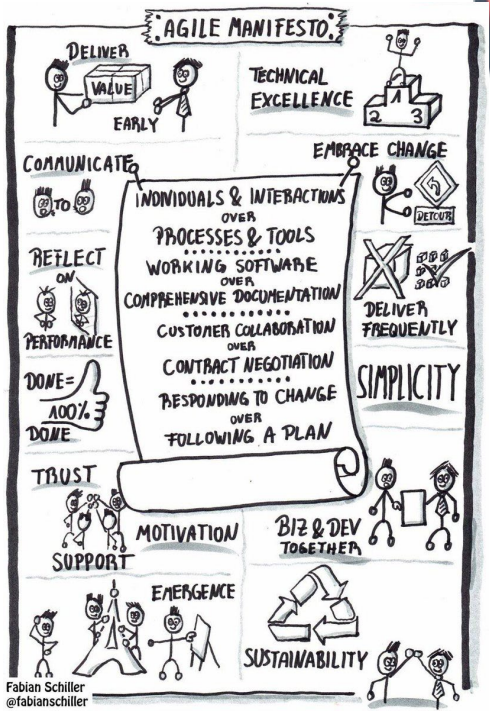
Continuous Integration
Continuous Delivery and Deployment
Automation of testing, deploy, infrastructure
Team Takes responsibility of Deploying
Team takes responsibility post deployment



Kanban WoW

Lean WoW

Focus on – visibility of work, limit WIP in work queues, reduce waste, improve cycle time



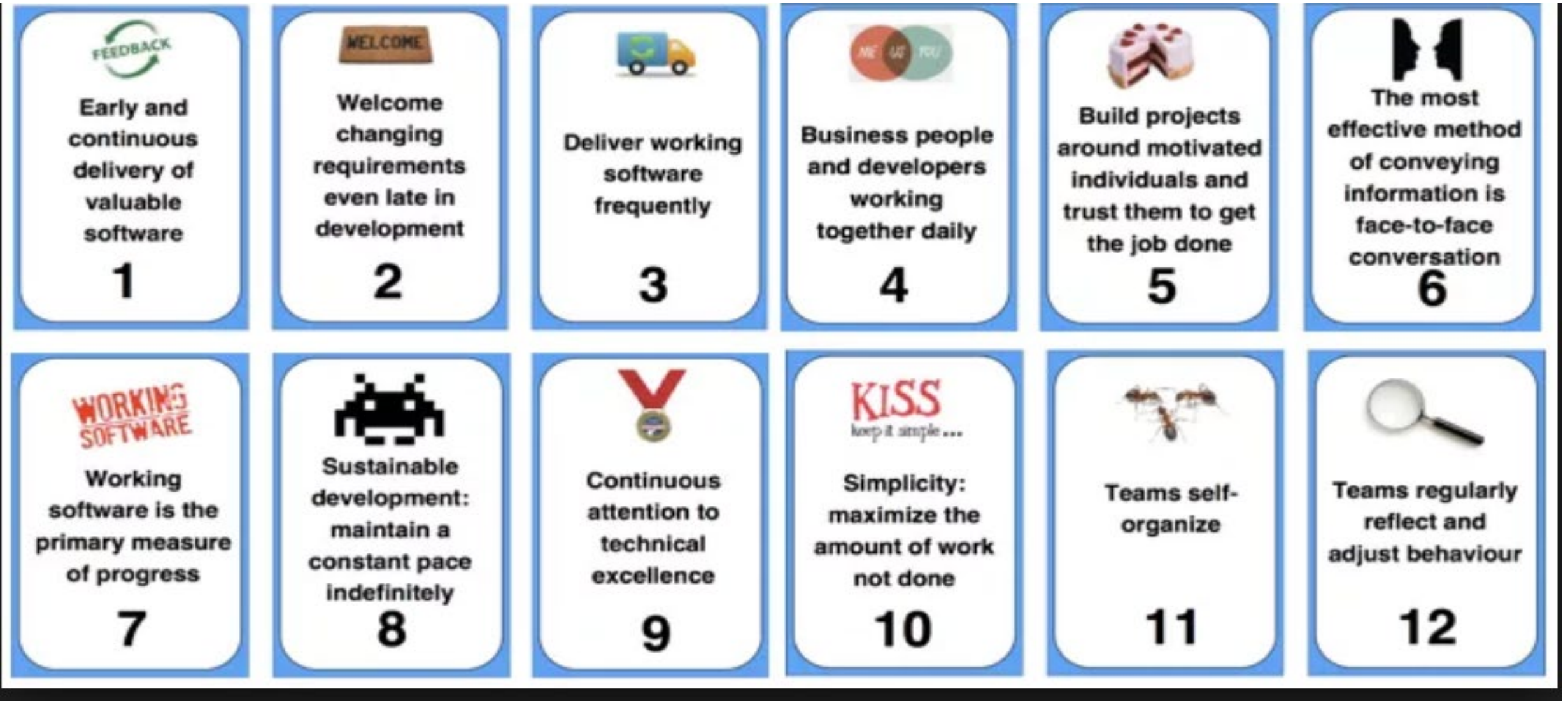
Continuous = frequent & small

Scrum is Agile….what does that mean?

2001

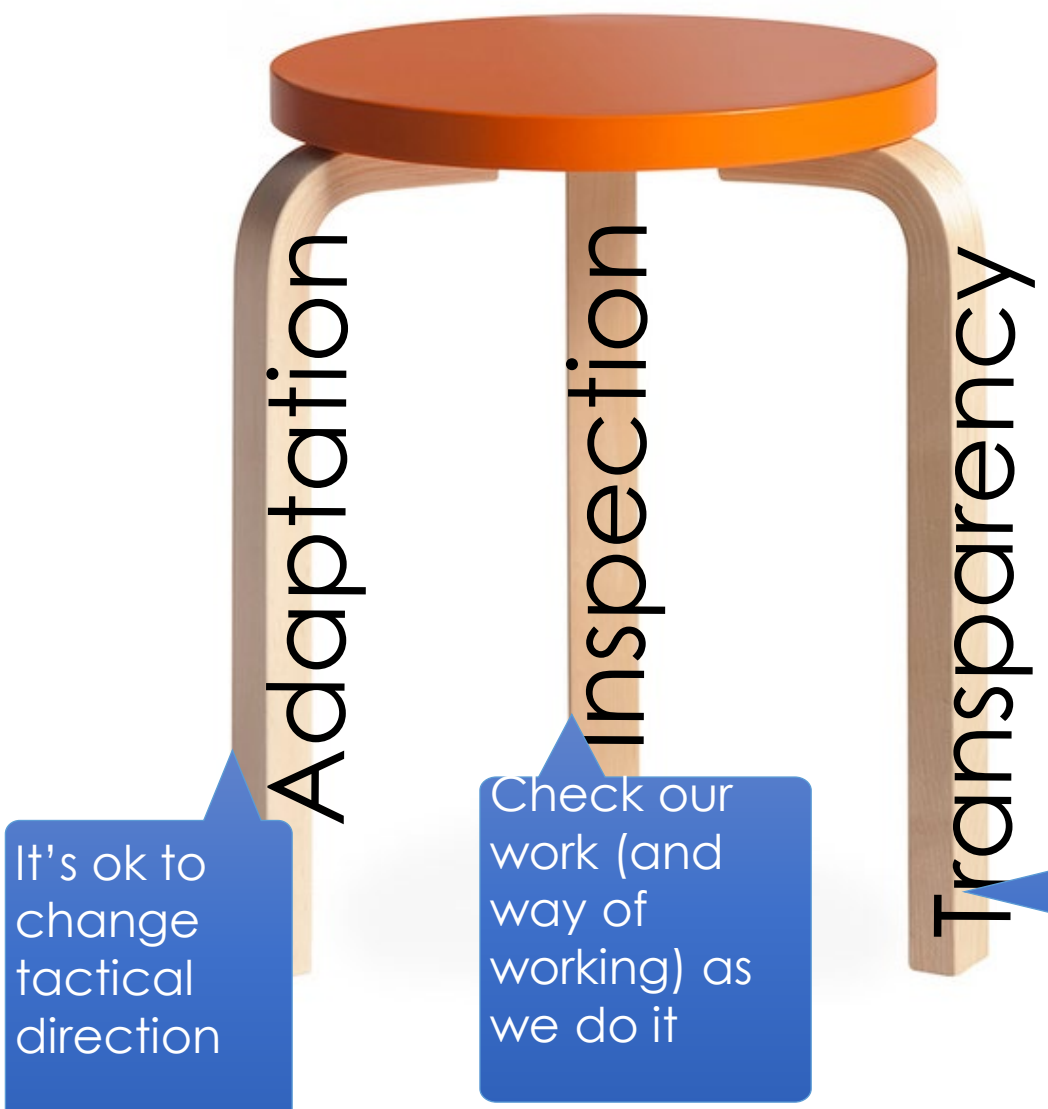
http://agilemanifesto.org

The Agile Manifesto – guides decisions, behaviour



Webinar by Ahmed Sidky – CEO of ICAgile course
https://www.softed.com/assets/Uploads/Resources/Agile/The-Agile-Mindset-Ahmed-Sidky.pdf

The 3 foundational ideas of Agile



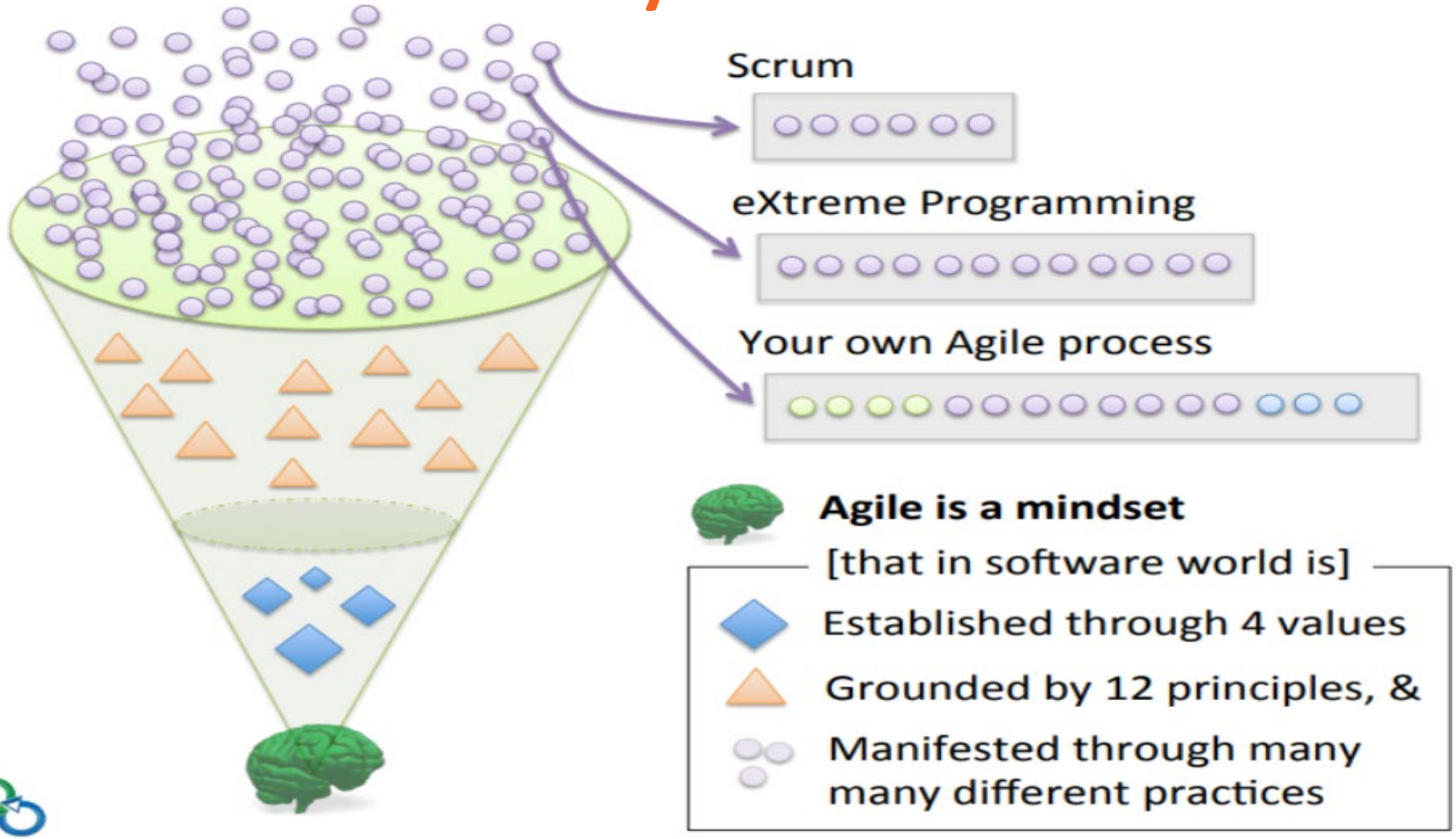
- Adaptation
  - willingness to change if it makes sense
  - understand implications of change and adapt
  - willingness to experiment to check alternatives
  - Desire to learn and improve

- Inspection
  - willingness to reflect on what has happened
  - curious and questioning
  - Proactive about improvement (not content with status quo)
  - Plan action (adaptation) based on evidence
  - Evaluation of action based on evidence
  - willing to put in effort

Transparency

- Workflows and progress are very visible to everyone
- Decisions are collaborative where it makes sense
- Different points of view are sought proactively
- Honesty and openness are normal (tempered with empathy)
- The truth is is visible
- Mistakes and questions are ok

The Agile Manifesto – many WoW



Webinar by Ahmed Sidky – CEO of ICAgile course
https://www.softed.com/assets/Uploads/Resources/Agile/The-Agile-Mindset-Ahmed-Sidky.pdf

Approach to uncertainty with an Agile mindset

Fixed Mindset
approach to
managing
uncertainty

Reducing uncertainty by "nailing things down."

Looking to fix and confirm things.

Agile Mindset
approach to
managing
uncertainty

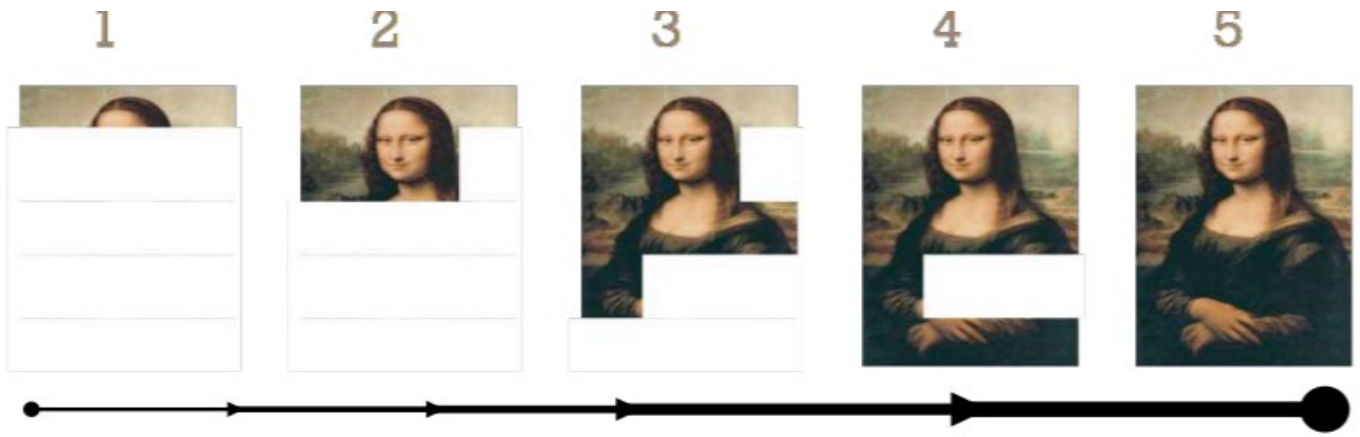
Reducing uncertainty by discovering and learning.

Looking to learn and discover in the most efficient way possible.

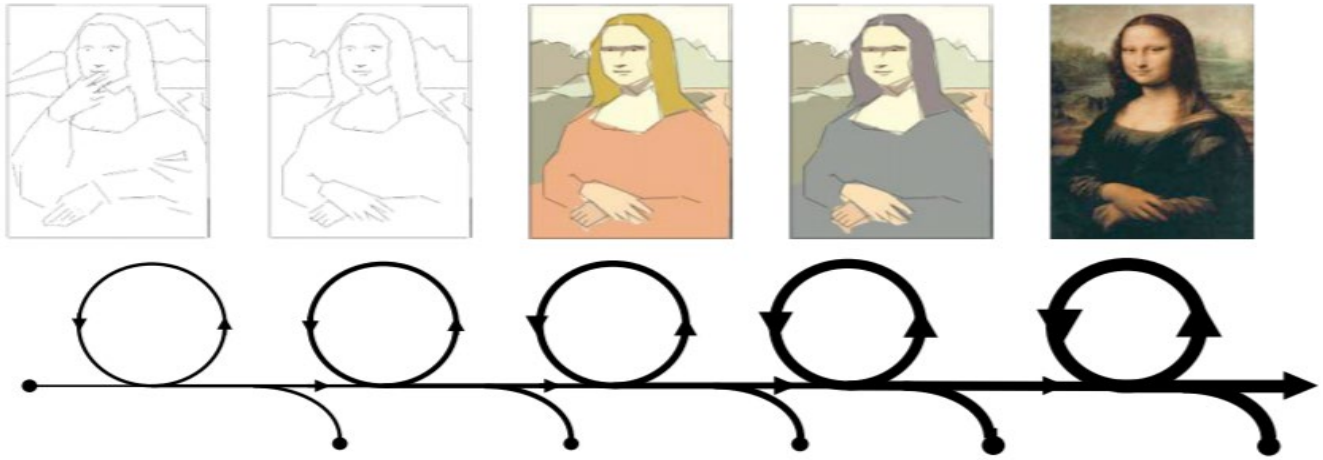
Webinar by Ahmed Sidky – CEO of ICAgile course
https://www.softed.com/assets/Uploads/Resources/Agile/The-Agile-Mindset-Ahmed-Sidky.pdf

Incremental delivery with Agile mindset

Must "nail down" the output in order to start delivery (Linear Thinking)



Discover and learn through valuable output and welcoming change (Circular Thinking)



Webinar by Ahmed Sidky – CEO of ICAgile course
https://www.softed.com/assets/Uploads/Resources/Agile/The-Agile-Mindset-Ahmed-Sidky.pdf

The Growth mindset and the Agile Mindset

Carol Dweck

I believe that my **[Intelligence, Personality, Character]** is inherent and static. Locked-down or fixed. My potential is determined at birth. It doesn't change.

Avoid failure

Desire to Look smart

Avoids challenges

Stick to what they know

Feedback and criticism is personal

They don't change or improve

I believe that my **[Intelligence, Personality, Character]** can be continuously developed. My true potential is unknown and unknowable.

Desire continuous learning

Confront uncertainties.

Embracing challenges

Not afraid to fail

Put lots of effort to learn

Feedback is about current capabilities

Webinar by Ahmed Sidky - https://www.softed.com/c

These practices are often associated with Agile WoW

Product visioning
Project chartering
Affinity (relative) estimation
Size-based (point) estimation
Planning poker
Group estimation
Value-based documentation
Prioritized product backlog
User stories
Progressive elaboration
Personas
Story maps / MMF
Story slicing
Acceptance tests as requirements
Short iterations
WIP Limits
Early and frequent releases
Roadmapping
Velocity-based planning and commitment
Iteration planning / Iteration backlog
Release planning / Release backlog
Time boxed iterations
Adaptive (multi-level) planning
Risk backlog
Team structure of VT / DT
Pull-based systems
Slack
Sustainable pace

Frequent face-to-face
Team chartering
Cross-silo collaborative teams
Self-organizing teams
Cross-functional teams
Servant leadership
Task volunteering
Generalizing specialist
Tracking progress via velocity
Burn-up/burn-down charts
Refactoring
Automated unit tests
Coding standards
Incremental/evolutionary design
Automated builds
Ten-minute build
Monitoring technical debt
Version control
Configuration management
Test driven development
Pair programming
Spike solutions
Continuous integration
Incremental deployment
Simple design
End-of-iteration hands-on UAT
Automated functional tests
Automated developer tests (unit tests)
Exploratory testing
Software metrics

Modern Agile

Joshua Kerievsky
Keynote Agile conference 2016

Rather than "customer collaboration over contract negotiation," Modern Agile's "Make people awesome" is more important for our focus. We want our entire ecosystem to be awesome.

Though "working software over comprehensive documentation" was great in 2001, today we want to "deliver value continuously," he says. "The bar has been raised."

"Responding to change over following a plan" was incredibly important when we wanted to defeat waterfall with agile, he explains. Now we want to "experiment and learn rapidly" because "sometimes we don't even know what the problem is."

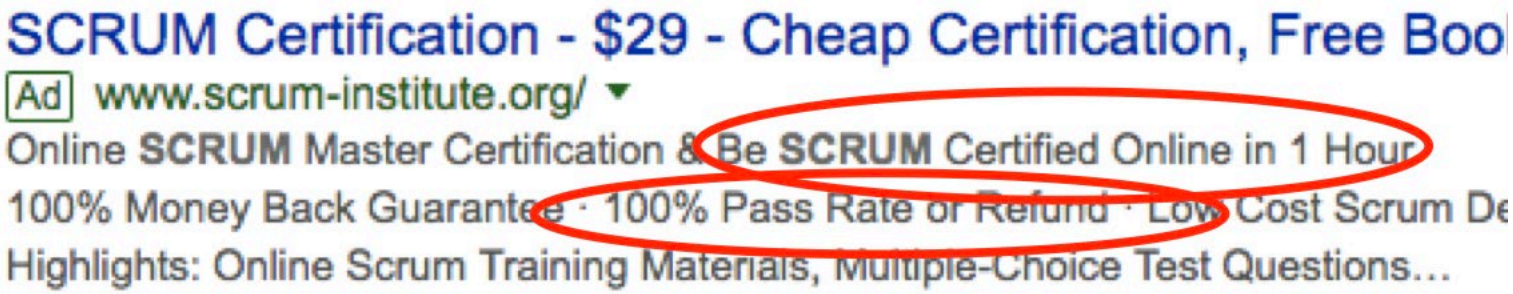
"Individuals and interactions" could be replaced by "make safety a prerequisite." We want to provide psychological safety in our interactions.

Heart of Agile

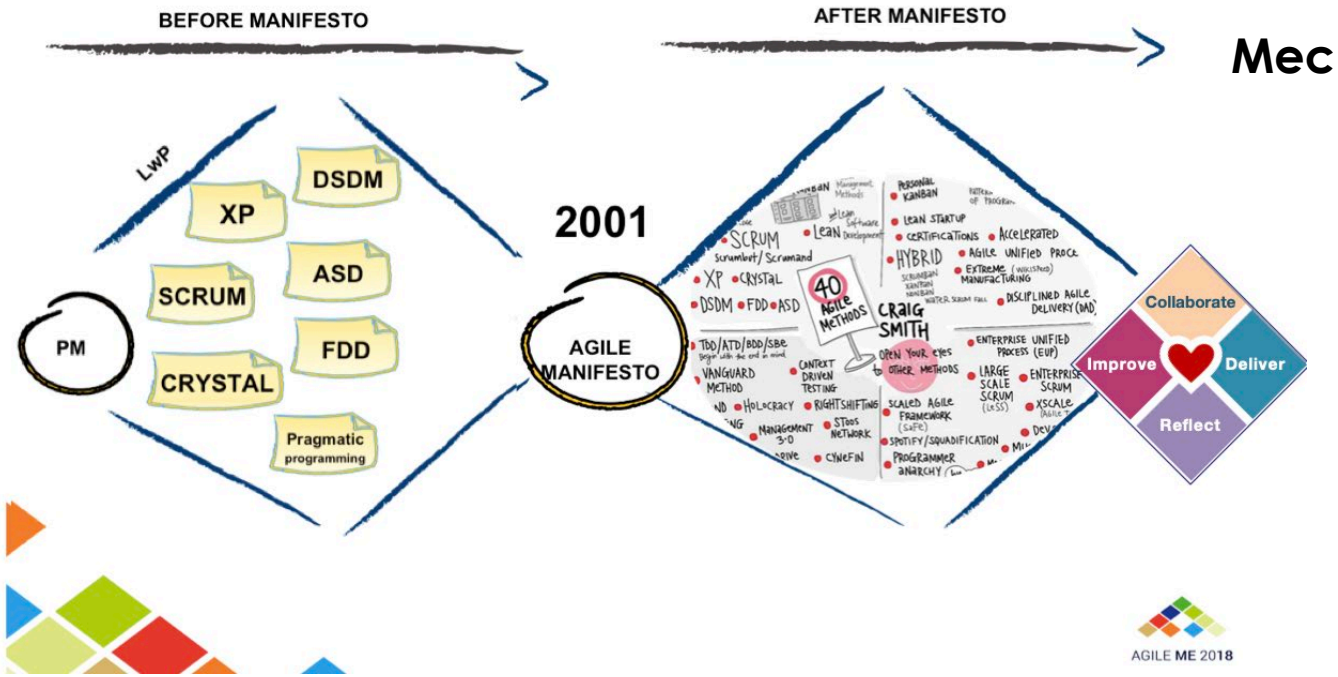
https://heartofagile.com/lets-begin/

144 LOs in Scrum training

Oversimplified:

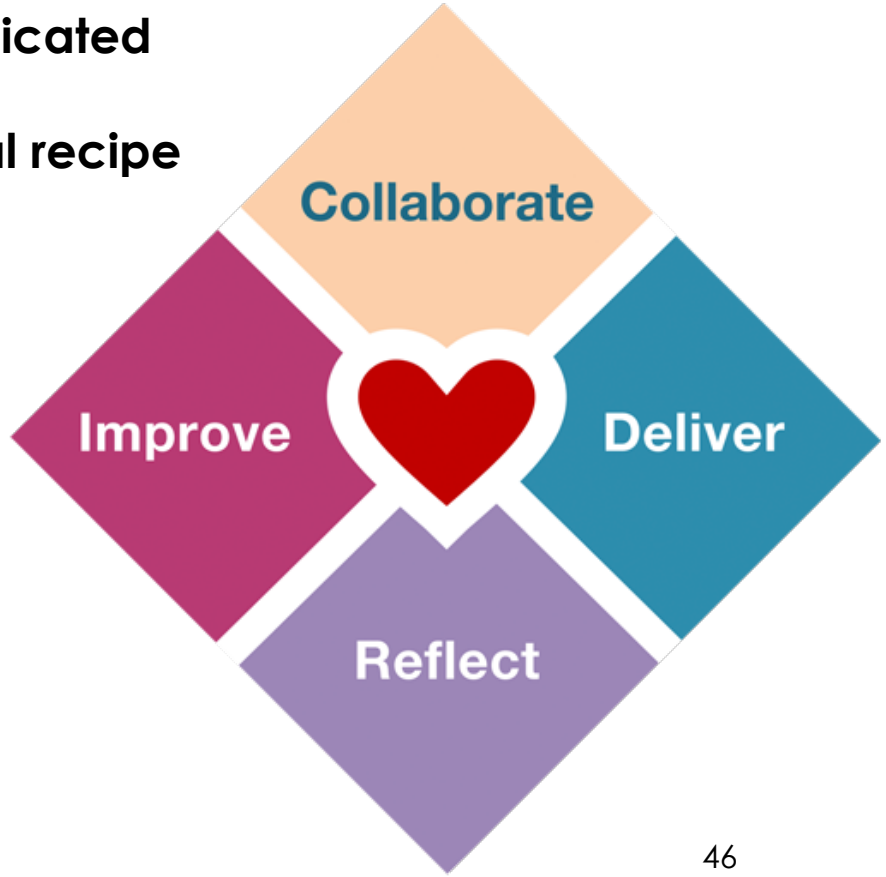


Pierre Hervouet's recap of history:



Overcomplicated

Mechanical recipe



“Heart of Agile” - Talk

https://heartofagile.com/video-of-the-latest-talk-on-heart-of-agile-by-alistair-cockburn-denmark-october-2

https://www.itu.dk/om-itu/presse/nyheder/2018/video-agil-ophavsmand-besoegte-danmark

Last October [2018] Dr. Alistair Cockburn talked to more than 700 people at the IT University of Copenhagen about the lastest ideas and experiments on Heart of Agile.

- **“Heart of Agile” Article from Crosstalk**
https://heartofagile.com/the-heart-of-agile-crosstalk-magazine/

Well Before "Heart of Agile"

https://alistair.cockburn.us/coming-soon/

And reading for the really dedicated…

Cockburn, A. (2003). *People and Methodologies in Software Development* [Doctoral Dissertation, University of Oslo]. Oslo. Retrieved 8/03/2022 from https://www.researchgate.net/profile/Alistair-Cockburn/publication/253582591_People_and_Methodologies_in_Software_Development/links/56d434b208ae2ea08cf8e076/People-and-Methodologies-in-Software-Development.pdf

Or just read the abstract!!

Agile Modelling and Agile Ways of Working

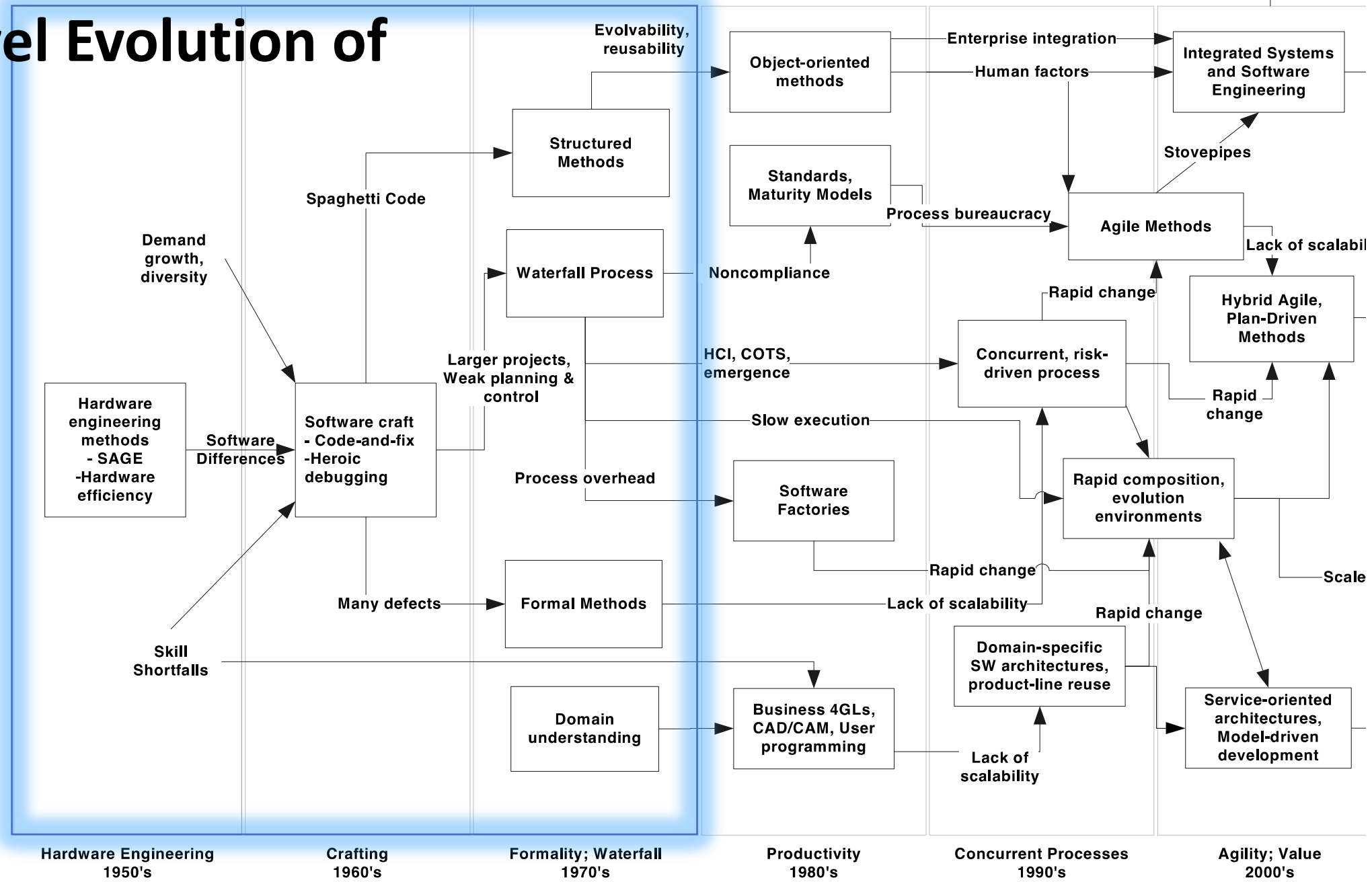
"The Criteria for Determining Whether a Team is Agile"

Scott Ambler

https://agilemodeling.com/essays/agilecriteria.htm

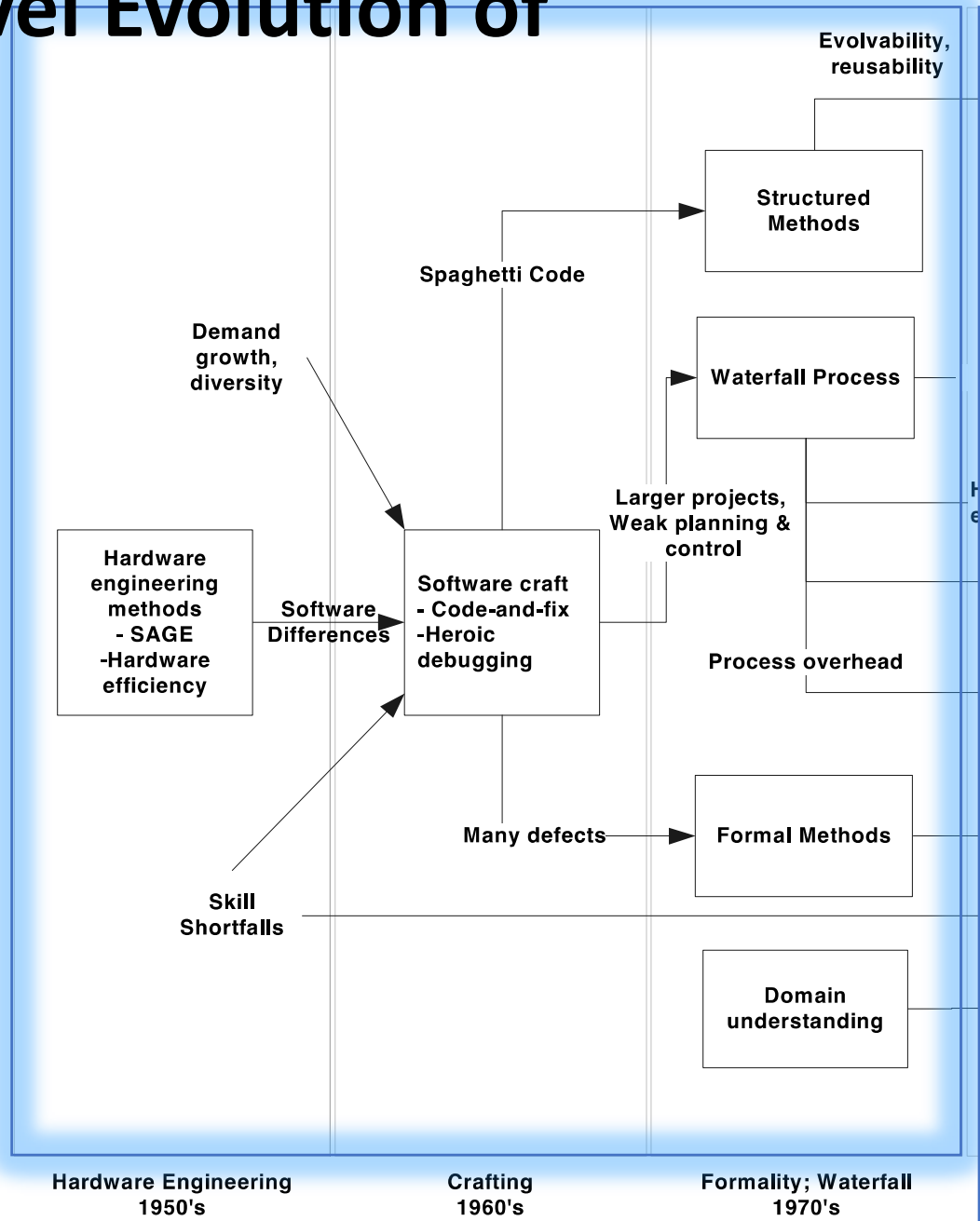
https://agilemodeling.com/essays/introductiontoam.htm

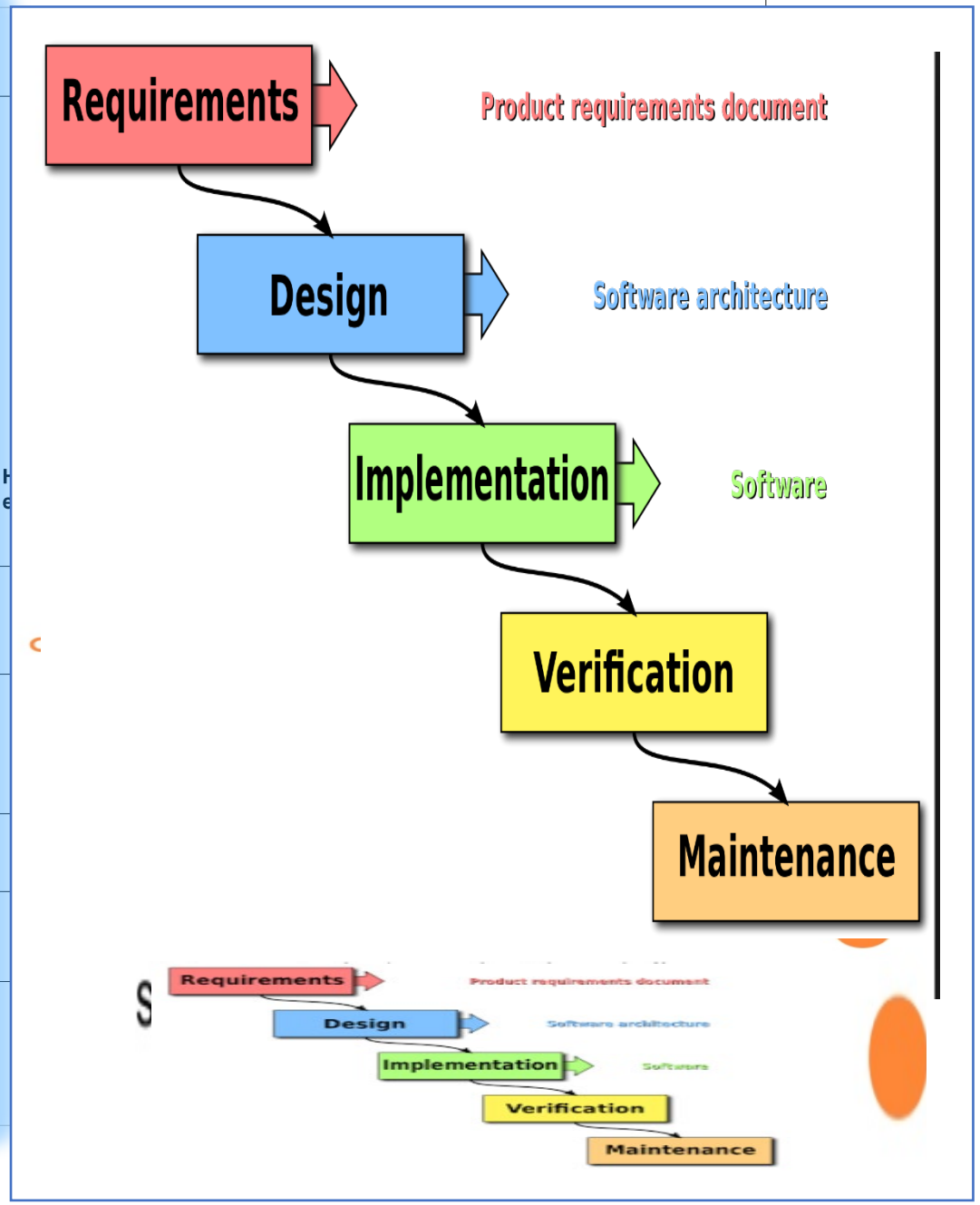
High-level Evolution of SE



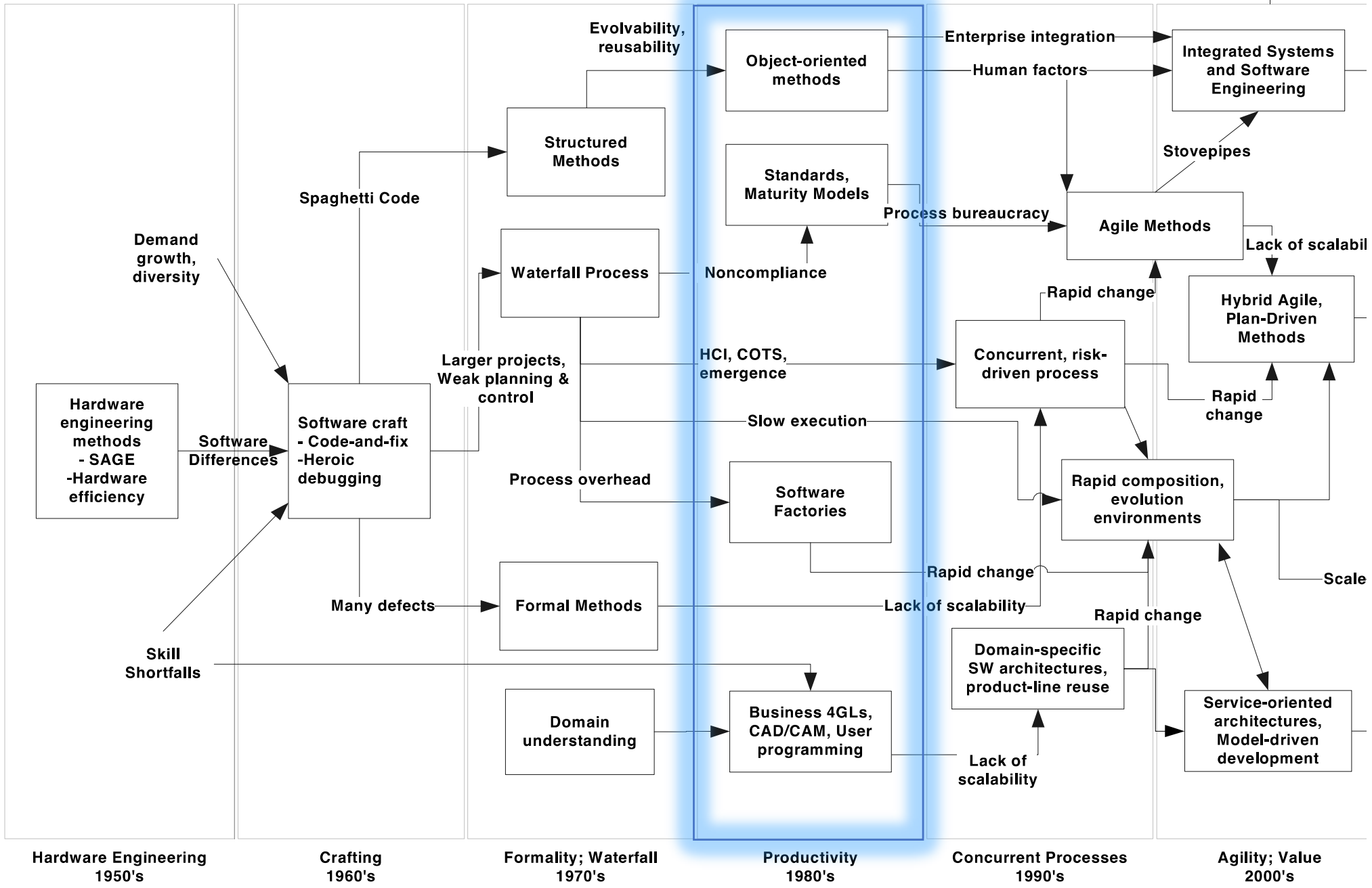
Boehm, B. (2006) “A view of 20th and 21st century software engineering,” presented at the Proceeding of the 28th international conference on Software engineering - ICSE '06, New York, New York, USA, p. 12.

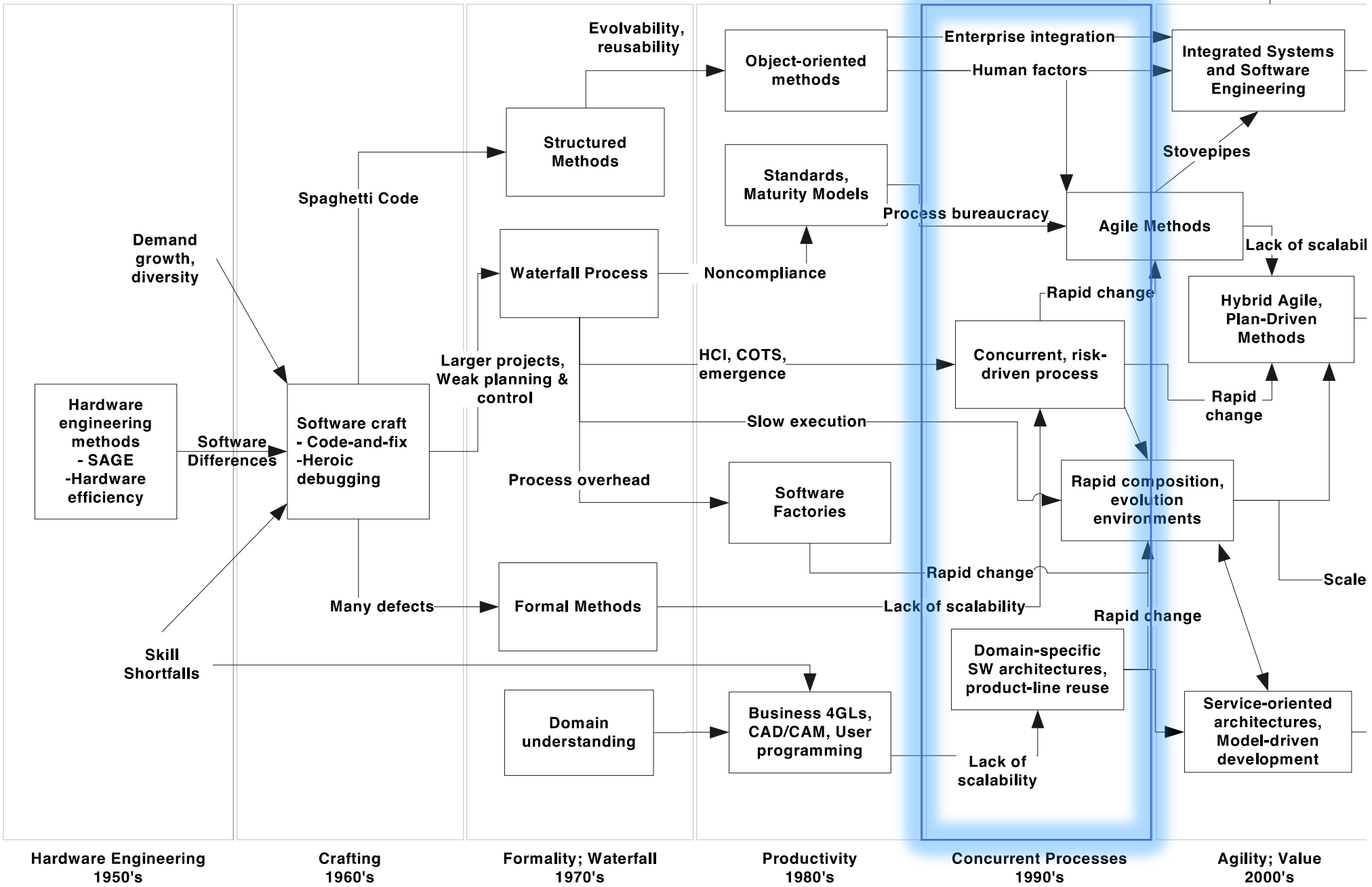
High-level Evolution of SE

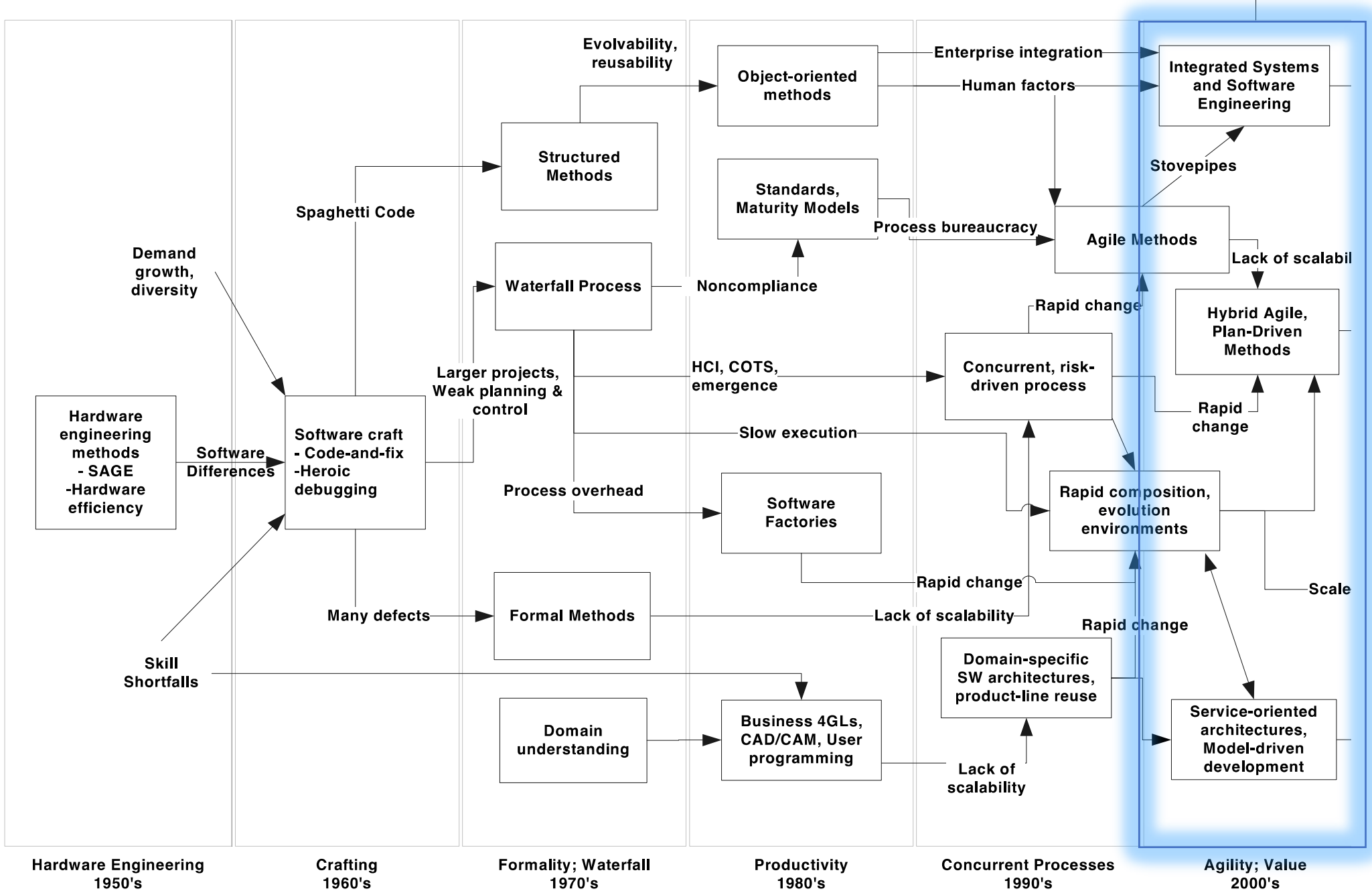




Boehm, B. (2006) "A view of 20th and 21st century software engineering," presented at the Proceeding of the 28th international conference on Software engineering - ICSE '06, New York, New York, USA, p. 12.


Evolvability, reusability
Object-oriented methods
Enterprise integration
Integrated Systems and Software Engineering
Human factors
Structured Methods
Stovepipes
Standards, Maturity Models
Spaghetti Code
Process bureaucracy
Agile Methods
Demand growth, diversity
Waterfall Process
Noncompliance
Lack of scalabil
Rapid change
Hybrid Agile, Plan-Driven Methods
HCI, COTS, emergence
Concurrent, risk-driven process
Larger projects, Weak planning & control
Hardware engineering methods - SAGE -Hardware efficiency
Software Differences
Software craft - Code-and-fix -Heroic debugging
Slow execution
Rapid change
Process overhead
Software Factories
Rapid composition, evolution environments
Rapid change
Scale
Many defects
Formal Methods
Lack of scalability
Rapid change
Skill Shortfalls
Domain-specific SW architectures, product-line reuse
Domain understanding
Business 4GLs, CAD/CAM, User programming
Service-oriented architectures, Model-driven development
Lack of scalability
Hardware Engineering 1950's
Crafting 1960's
Formality; Waterfall 1970's
Productivity 1980's
Concurrent Processes 1990's
Agility; Value 2000's


Evolvability, reusability
Object-oriented methods
Enterprise integration
Integrated Systems and Software Engineering
Human factors
Structured Methods
Stovepipes
Spaghetti Code
Standards, Maturity Models
Process bureaucracy
Agile Methods
Lack of scalabil
Demand growth, diversity
Waterfall Process
Noncompliance
Rapid change
Hybrid Agile, Plan-Driven Methods
Larger projects, Weak planning & control
HCI, COTS, emergence
Concurrent, risk-driven process
Hardware engineering methods - SAGE -Hardware efficiency
Software Differences
Software craft - Code-and-fix -Heroic debugging
Slow execution
Rapid change
Process overhead
Software Factories
Rapid composition, evolution environments
Rapid change
Scale
Many defects
Formal Methods
Lack of scalability
Rapid change
Skill Shortfalls
Domain-specific SW architectures, product-line reuse
Domain understanding
Business 4GLs, CAD/CAM, User programming
Service-oriented architectures, Model-driven development
Lack of scalability
Hardware Engineering 1950's
Crafting 1960's
Formality; Waterfall 1970's
Productivity 1980's
Concurrent Processes 1990's
Agility; Value 2000's


Hardware engineering methods - SAGE -Hardware efficiency
Software Differences
Demand growth, diversity
Skill Shortfalls
Software craft - Code-and-fix -Heroic debugging
Spaghetti Code
Many defects
Larger projects, Weak planning & control
Structured Methods
Waterfall Process
Formal Methods
Evolvability, reusability
Object-oriented methods
Standards, Maturity Models
Noncompliance
Process overhead
Software Factories
HCI, COTS, emergence
Slow execution
Domain understanding
Business 4GLs, CAD/CAM, User programming
Enterprise integration
Human factors
Process bureaucracy
Rapid change
Concurrent, risk-driven process
Rapid change
Lack of scalability
Domain-specific SW architectures, product-line reuse
Lack of scalability
Integrated Systems and Software Engineering
Stovepipes
Agile Methods
Lack of scalabil
Hybrid Agile, Plan-Driven Methods
Rapid change
Rapid composition, evolution environments
Scale
Rapid change
Service-oriented architectures, Model-driven development
Hardware Engineering 1950's
Crafting 1960's
Formality; Waterfall 1970's
Productivity 1980's
Concurrent Processes 1990's
Agility; Value 2000's

Diverse sources of knowledge/learning….

- **Online Tutorials** - Freecode.com, YouTube
- **Online documentation** - tools, libraries eg Docker, GitHub
- **Podcasts** Engineering Culture by InfoQ
- **Newsletters** - InfoQ
- **Meetup groups** Agile Auckland, DevOps, Ministry of testing,
- **Blogs** - Medium, Freecode.com,
- **GitHub Repositories** - Open source projects,
- **Company websites** -Google, Xero, Microsoft, Facebook, Netflix, Basecamp
- **Published research**– ACM, IEEExplore, SpringerLink, ScienceDirect
- **Online Q&A** -Stackoverflow
- **Guest speakers from industry**
- **Work in Industry or Open source projects**
- **Each other –** teaching and listening
- **GenAI services? More later…**
- **Your lecturer**

Takeaways from today…

Iterative and incremental ways of working established and valuable

Scrum is not prescriptive about many aspects of SE INTENTIONALLY

CI/CD focus on automating the integration and then deployment (release)

So small features can be released frequently

CI is a workflow for building, integrating and QA of shared code when working in a team

Practice and experiment with the WoW and techniques and tools!!

Modern Agile and The Heart of Agile are modernizing and simplifying the values behind the goal of Agile

Questions and Comments….