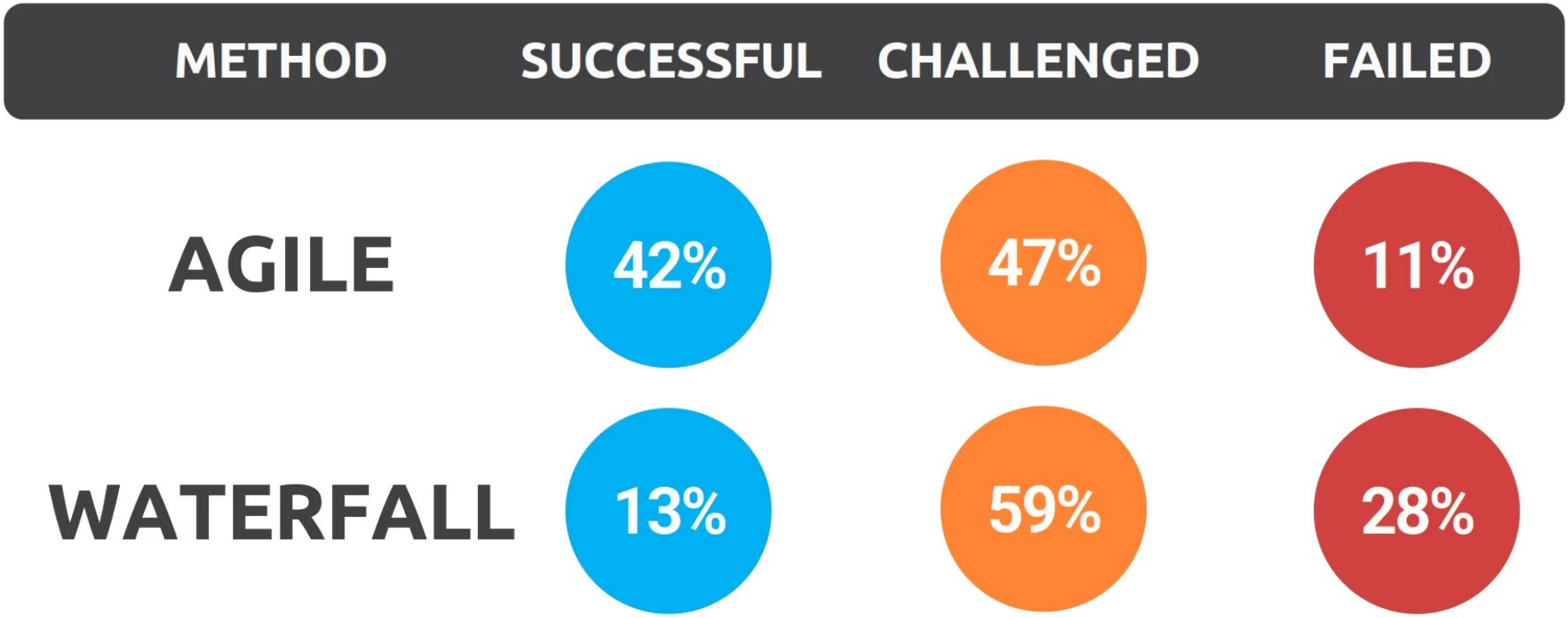
SOFTWARE ENGINEERING

Week 2_2
Agile Software Development – Part 1

Course ID 06016410,
06016321

Nont Kanungsukkasem, B.Eng., M.Sc., Ph.D.
nont@it.kmitl.ac.th

PROJECT SUCCESS RATES
AGILE VS WATERFALL



Source: Standish Group Report 2020

History

- The best way to achieve better software was through careful project planning, formalized quality assurance, use of analysis and design methods supported by software tools, and controlled and rigorous software development processes.
  - Large, long-lived software systems such as aerospace systems.

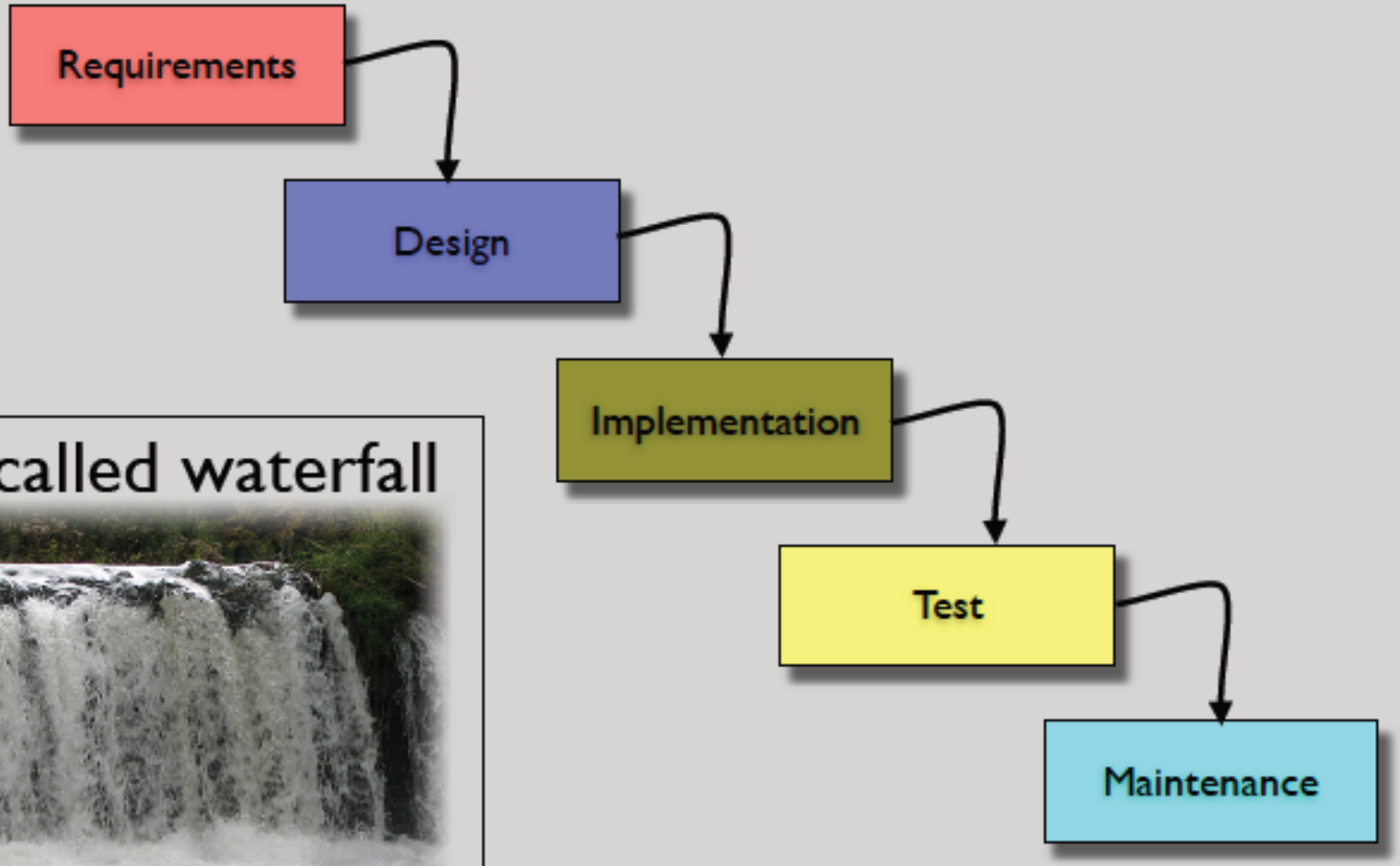
History

- Medium-sized business systems?
  - The overhead involved is so large that it dominates the software development process
- The system requirements change.
  - rework is essential and, in principle at least, the specification and design have to change with the program

What's wrong with waterfall

1. Releases take longer and longer
2. Release schedules slip
3. Stabilization at end of the release takes longer and longer
4. Planning takes too long and doesn't get it right
5. Changes are hard to introduce mid-release
6. Quality is deteriorating
7. Death marches are hurting morale

Traditional Development



Also called waterfall

Problems with sequential development

Late "real" feedback

Difficult to deal with changes

Slow - Handovers

Lack of visibility

แผนแบบนั่งเทียน ปาฏิหารย์

บรึะเจ้าเสร็จแล้ว

แผนที่วางไว้

แผนตอนทำจริง

Analogy

- You are a new night manager at FatBurger and are the only person on duty.
- A customer approaches at 11 pm. and orders a Double Fatburger with Cheese.
- You ring up the order. The price is $7.
- The customer informs you that he only has $1.
- What do you do and what do you tell the customer?
  - Turn away your only customer in the 11:00pm hour?
  - Offer something that he can afford?
  - Give it to him for $1 this time only?

https://leewaterman.com/dont-make-squirrel-burgers/

Predictive vs. Adaptive

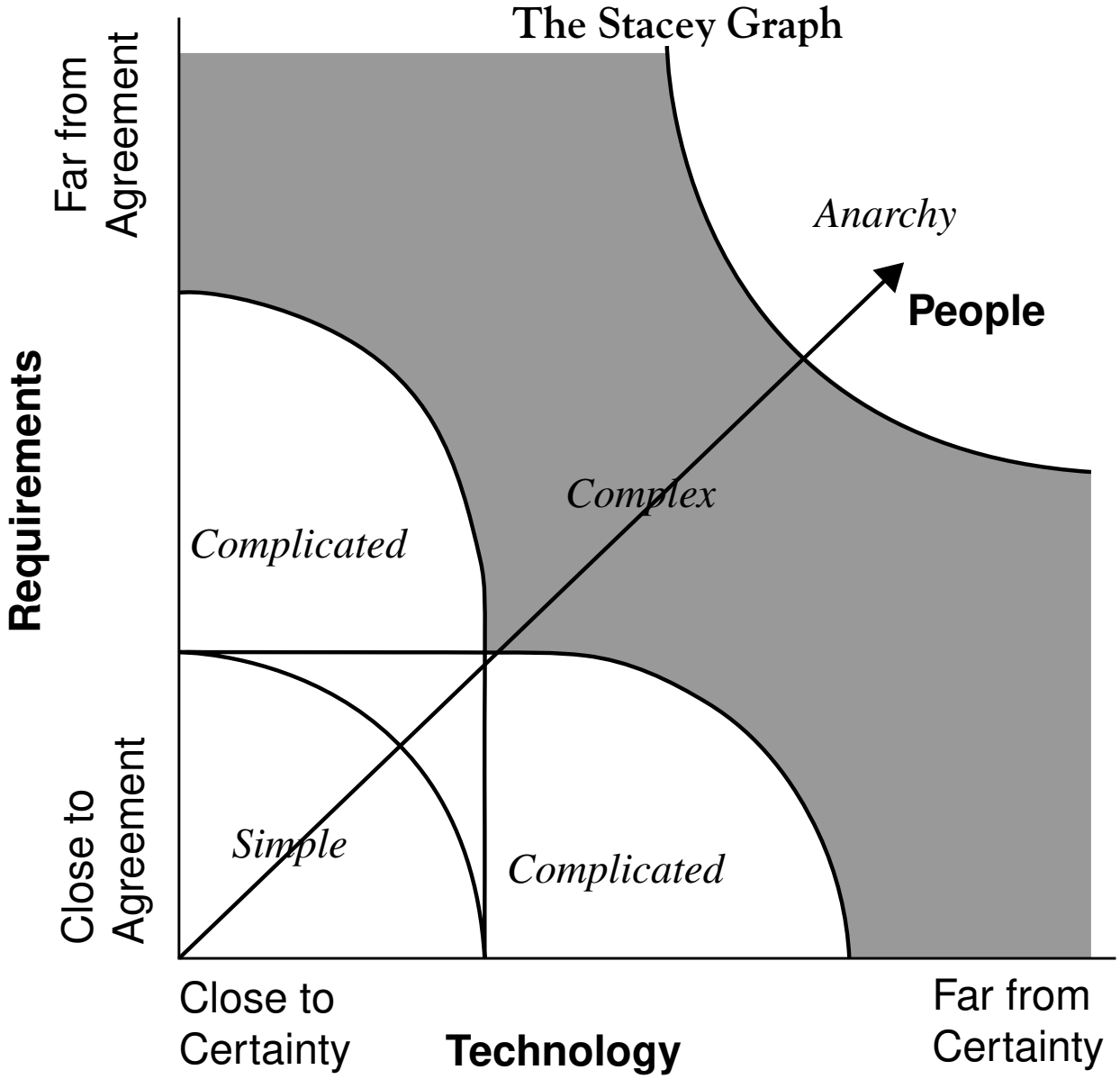
Predictive

Start with Plan and all requirements → End with all requirements completed

Adaptive

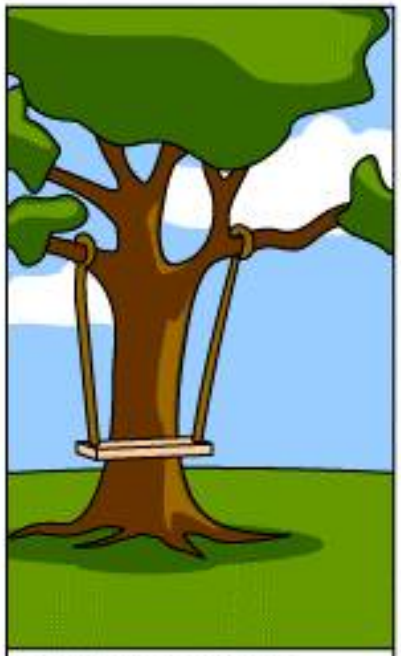
What's changed

- Requirements
- Technologies
- People



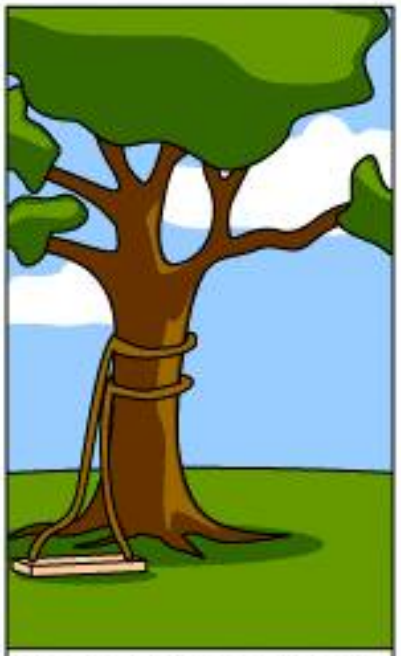
R. Stacey, *Complexity and Emergence in Organizations* (London: Routledge, 2001).

How the customer explained it



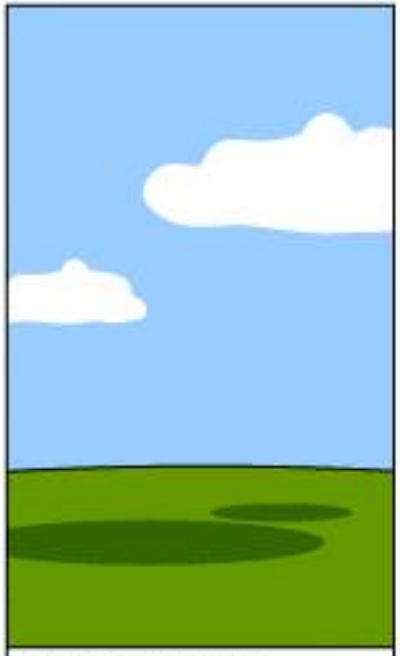
How the Project Leader understood it

How the Analyst designed it

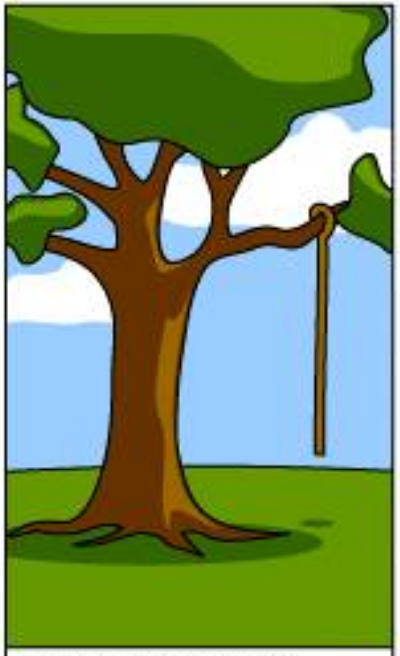


How the Programmer wrote it

How the Business Consultant described it

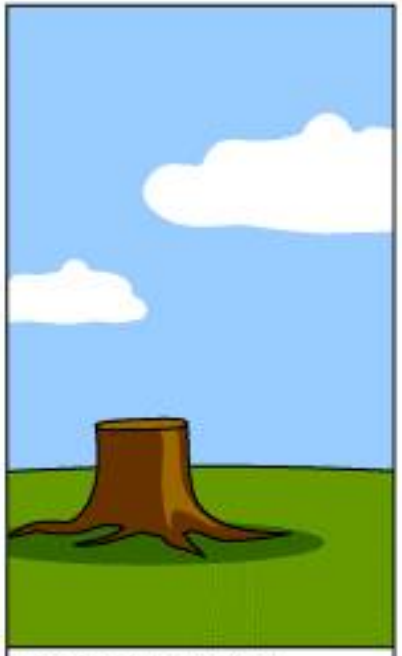


How the project was documented



What operations installed

How the customer was billed



How it was supported

What the customer really needed

Dramas

- วิกฤตการณ์โปรแกรมเมอร์ไทย
  - http://pantip.com/topic/30186572
- Agile Thailand 2013 : วิกฤติโปรแกรมเมอร์ไทย แก้ได้ด้วยอไจล์
  - https://www.youtube.com/watch?v=gVaRj5GgOPg
- วิกฤตการณ์นี้เกิดจากใคร?
  - http://pantip.com/topic/32323444
- เด็กไอทีตกงาน จะโทษใคร
  - http://www.youtube.com/watch?v=7lGrHVh_Tu4&feature=youtu.be

- Introduction to Agile and Scrum
  - https://www.youtube.com/watch?v=XU0IIRltyFM&t=3s

The Agile Manifesto

https://agilemanifesto.org/history.html

- On February 11-13, 2001, seventeen software developers, writers, and consultants, called The Agile Alliance, met at the Snowbird resort in Utah to discuss lightweight development methods and published the **Manifesto for Agile Software Development.**
  - Representatives from Extreme Programming, SCRUM, DSDM, Adaptive Software Development (ASD), Crystal, Feature-Driven Development, Pragmatic Programming, and others sympathetic to the need for an alternative to documentation driven, heavyweight software development processes convened.

Kent Beck, James Grenning, Robert C. Martin, Mike Beedle, **Jim Highsmith**, Steve Mellor, Arie van Bennekum, Andrew Hunt, **Ken Schwaber**, **Alistair Cockburn**, Ron Jeffries, **Jeff Sutherland**, Ward Cunningham, Jon Kern, **Dave Thomas**, Martin Fowler, Brian Marick

The Manifesto for Agile Software Development

(คำแถลงอุดมการณ์แห่งอไจล์)

https://agilemanifesto.org/

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

(เราค้นพบวิธีที่ดีกว่าในการพัฒนาซอฟท์แวร์จากการลงมือทำจริงและช่วยเหลือผู้อื่น นั่นคือ เราให้ความสำคัญกับ:)

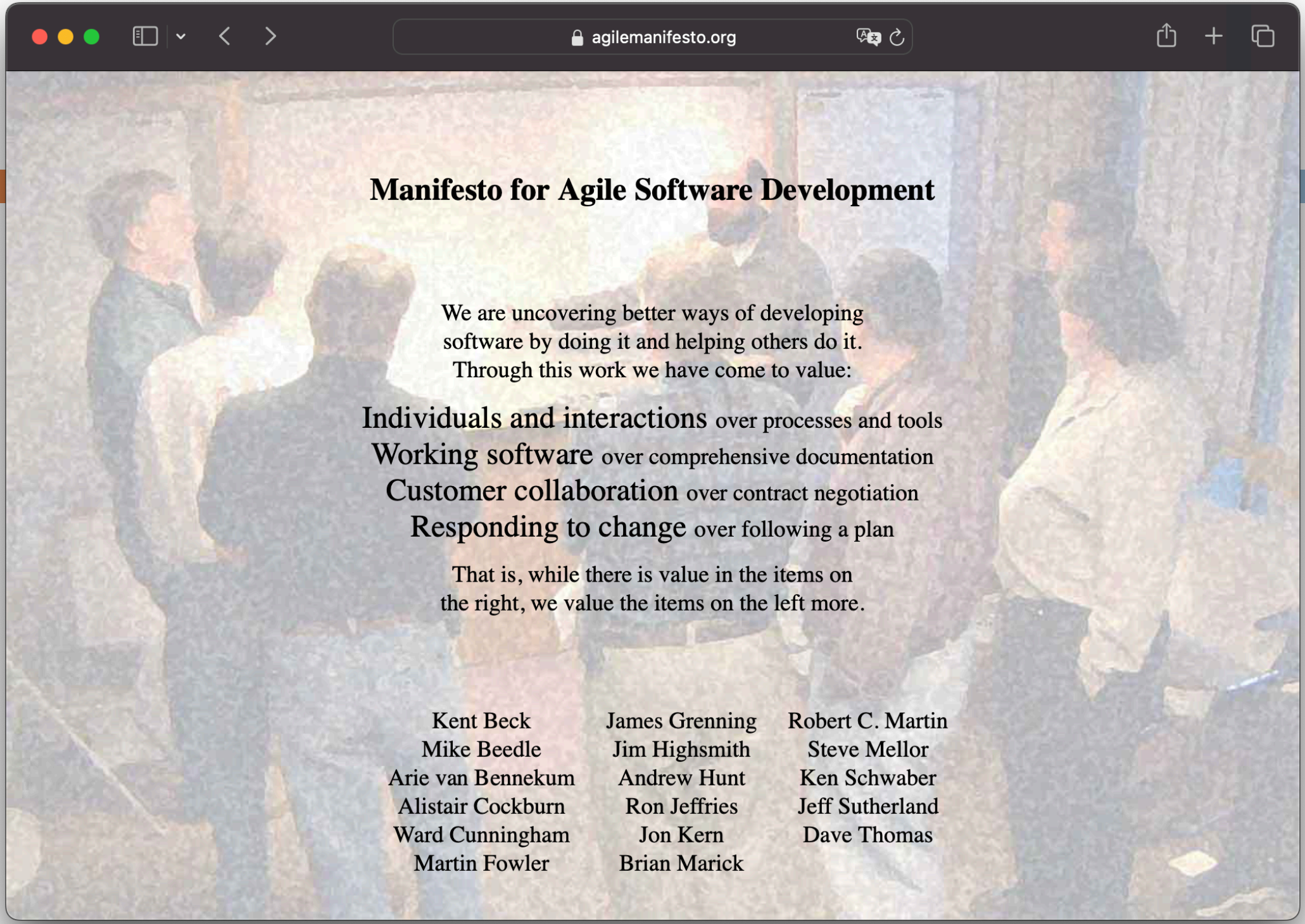
- ***Individuals and interactions*** over processes and tools
  (**คนและการมีปฏิสัมพันธ์กัน** มากกว่าการทำตามขั้นตอนและเครื่องมือ)
- ***Working software*** over comprehensive documentation
  (**ซอฟต์แวร์ที่นำไปใช้งานได้จริง** มากกว่าเอกสารที่ครบถ้วนสมบูรณ์)
- ***Customer collaboration*** over contract negotiation
  (**ร่วมมือทำงานกับลูกค้า** มากกว่าการต่อรองให้เป็นไปตามสัญญา)
- ***Responding to change*** over following a plan
  (**การตอบรับกับการเปลี่ยนแปลง** มากกว่าการทำตามแผนที่วางไว้)

That is, while there is value in the items on the right, we value the items on the left more."

(ทั้งนี้ แม้เราจะเห็นความสำคัญในสิ่งที่กล่าวไว้ทางด้านขวาแต่เราให้ความสำคัญกับสิ่งที่กล่าวไว้ทางด้านซ้ายมากกว่า)

(https://agilemanifesto.org/iso/th/manifesto.html)

Kent Beck et al.



https://agilemanifesto.org/

Principles behind the Agile Manifesto

หลักการเบื้องหลังคำแถลงอุดมการณ์แห่งอไจล์

https://agilemanifesto.org/principles.html

We follow these principles:

(พวกเราทำตามหลักการเหล่านี้:)

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
   (ความสำคัญสูงสุดของพวกเราคือความพึงพอใจของลูกค้าที่มีต่อการส่งมอบซอฟท์แวร์ที่มีคุณค่าต่อลูกค้าตั้งแต่ต้นอย่างต่อเนื่อง)

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
   (ยอมรับการเปลี่ยนแปลงความต้องการของลูกค้าแม้ในช่วงท้ายของการพัฒนาเพราะอไจล์สามารถแปรเอาความเปลี่ยนแปลง มาเป็นความได้เปรียบในการแข่งขันของลูกค้า)

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
   (ส่งมอบซอฟท์แวร์ที่ใช้งานได้จริงอย่างสม่ำเสมอ อาจเป็นทุกสองถึงสามสัปดาห์หรือทุกสองถึงสามเดือน โดยควรทำให้ระยะเวลาระหว่างการส่งมอบนั้นสั้นที่สุดเท่าที่เป็นไปได้)

https://agilemanifesto.org/iso/th/principles.html

Principles behind the Agile Manifesto

หลักการเบื้องหลังคำแถลงอุดมการณ์แห่งอไจล์

4. Business people and developers must work together daily throughout the project.
(ตัวแทนจากฝ่ายธุรกิจและนักพัฒนาจะต้องทำงานร่วมกันเป็นประจำทุกวันตลอดโครงการ)

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
(ทำให้แน่ใจว่าสมาชิกโครงการเข้าใจและมีจุดมุ่งหมายของโครงการร่วมกัน สร้างสภาวะแวดล้อมและให้การสนับสนุนในสิ่งที่พวกเขาต้องการและให้ความไว้วางใจแก่พวกเขาในการที่จะทำงานให้บรรลุเป้าหมายนั้น)

6. The most efficient and effective method of conveying information to and within a development team is face–to–face conversation.
(วิธีที่มีประสิทธิภาพและประสิทธิผลสูงสุดในการถ่ายทอดข้อมูลต่างๆไปสู่ทีมพัฒนาและภายในทีมพัฒนาเองคือการพูดคุยแบบซึ่งหน้า)

7. Working software is the primary measure of progress.
(ซอฟท์แวร์ที่ใช้งานได้จริงเป็นตัวหลักในการวัดความก้าวหน้าของโครงการ)

Principles behind the Agile Manifesto

หลักการเบื้องหลังคำแถลงอุดมการณ์แห่งอไจล์

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
(กระบวนการอไจล์สนับสนุนให้เกิดการพัฒนาแบบยั่งยืน กล่าวคือผู้สนับสนุนนักพัฒนา และตัวแทนผู้ใช้ควรจะสามารถรักษาอัตราเร็วในการทำงานร่วมกันให้คงที่ได้ตลอดไป)

9. Continuous attention to technical excellence and good design enhances agility.
(การใส่ใจในความเป็นเลิศทางเทคนิคและงานออกแบบที่ดีอย่างต่อเนื่องจะช่วยเพิ่มความเป็นอไจล์).

10. Simplicity – the art of maximizing the amount of work not done – is essential.
(ความเรียบง่าย หรือศิลปะในการทำงานอย่างพอเพียง นั้นสำคัญยิ่ง)

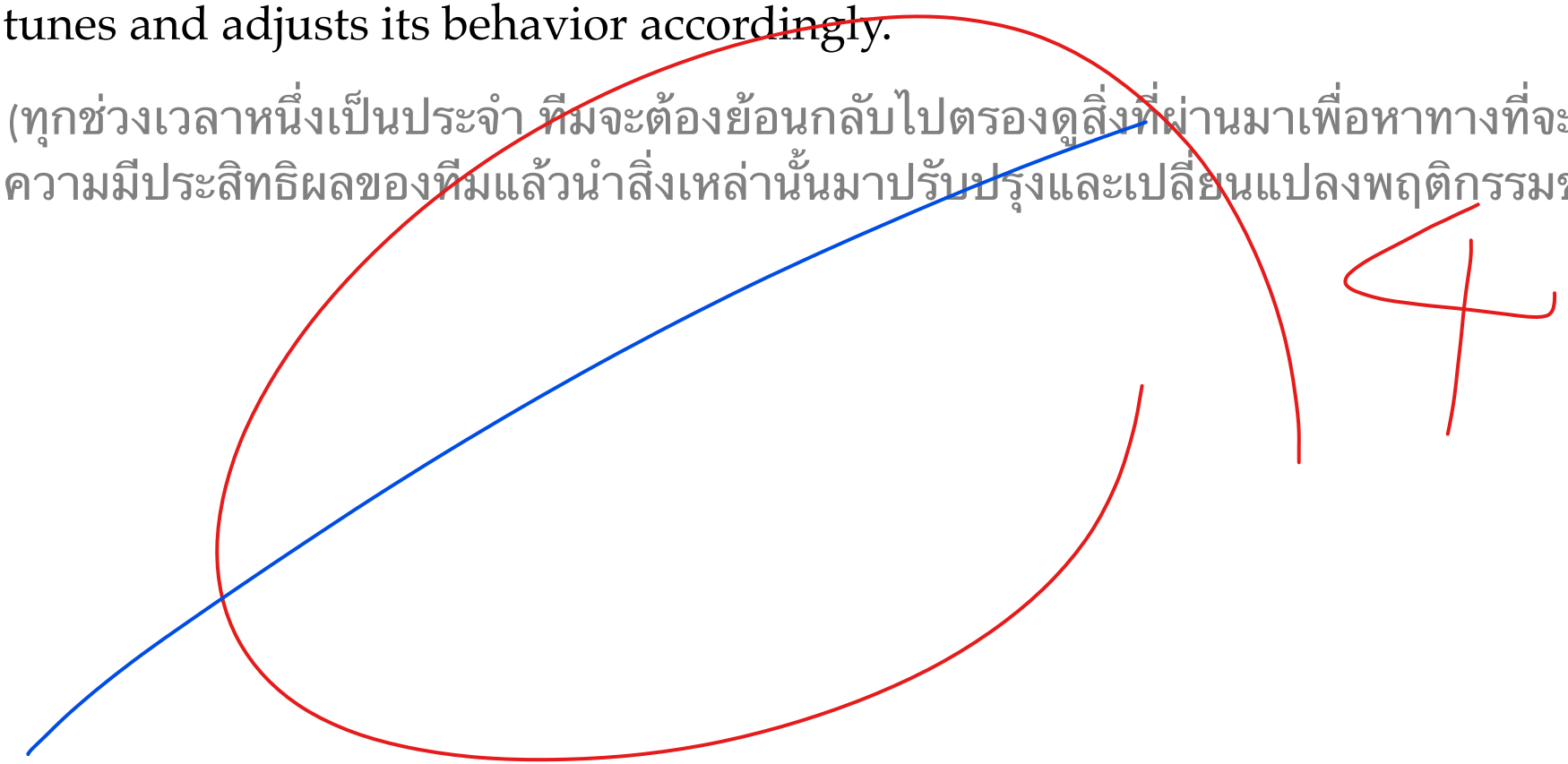
11. The best architectures, requirements, and designs emerge from self–organizing teams.
(สถาปัตยกรรมซอฟต์แวร์ ความต้องการของลูกค้า และงานออกแบบที่ดีที่สุด เกิดจากทีมที่บริหารจัดการตัวเองได้)

Principles behind the Agile Manifesto

หลักการเบื้องหลังคำแถลงอุดมการณ์แห่งอไจล์

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

(ทุกช่วงเวลาหนึ่งเป็นประจำ ทีมจะต้องย้อนกลับไปตรองดูสิ่งที่ผ่านมาเพื่อหาทางที่จะพัฒนาความมีประสิทธิผลของทีมแล้วนำสิ่งเหล่านั้นมาปรับปรุงและเปลี่ยนแปลงพฤติกรรมของทีม)



The principles of agile methods

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Embrace change	Expect the system requirements to change, and so design the system to accommodate these changes.
Incremental delivery	The software is developed in increments, with the customer specifying the requirements to be included in each increment.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.
People, not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.

An Agile Team

- Competence
- Common focus
- Collaboration
- Decision-making ability
- Fuzzy problem-solving ability
- Mutual trust and respect
- Self-organization

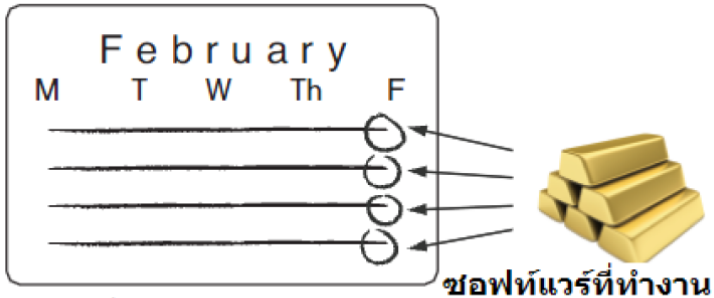
Agility in the context of software engineering work

"A set of Values and Principles"
rather than a particular methodology

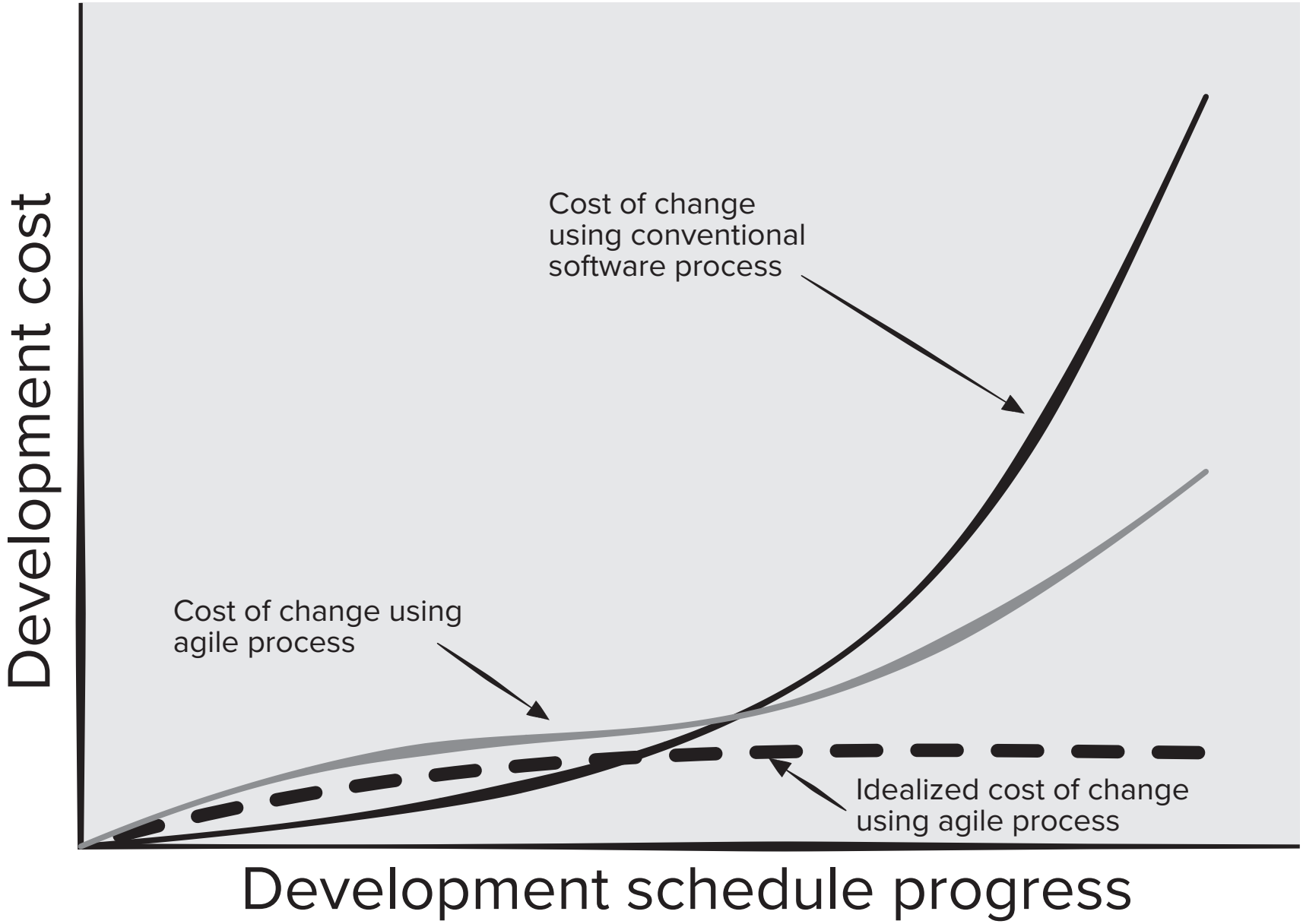
- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed

Yielding …

- Rapid, incremental delivery of software



Agility and the Cost of Change



Empirical Process Control

- Uses Inspection and subsequent adaptation to optimize realization of goals
- Transparency is required for inspection and adaptation
- Transparency requires courage and change in reward systems

Relegion

The Church Of Scrum

Why does Scrum so often feel like organized religion? And how can we make things better?

ธรรมไจล์

เหนือสิ่งอื่นใดคือการเติมเต็มความพอใจให้ลูกค้าด้วยการส่งงานที่มีคุณภาพเปี่ยมล้นอย่างต่อเนื่องสม่ำเสมอ

agile culture
because agile isnt process

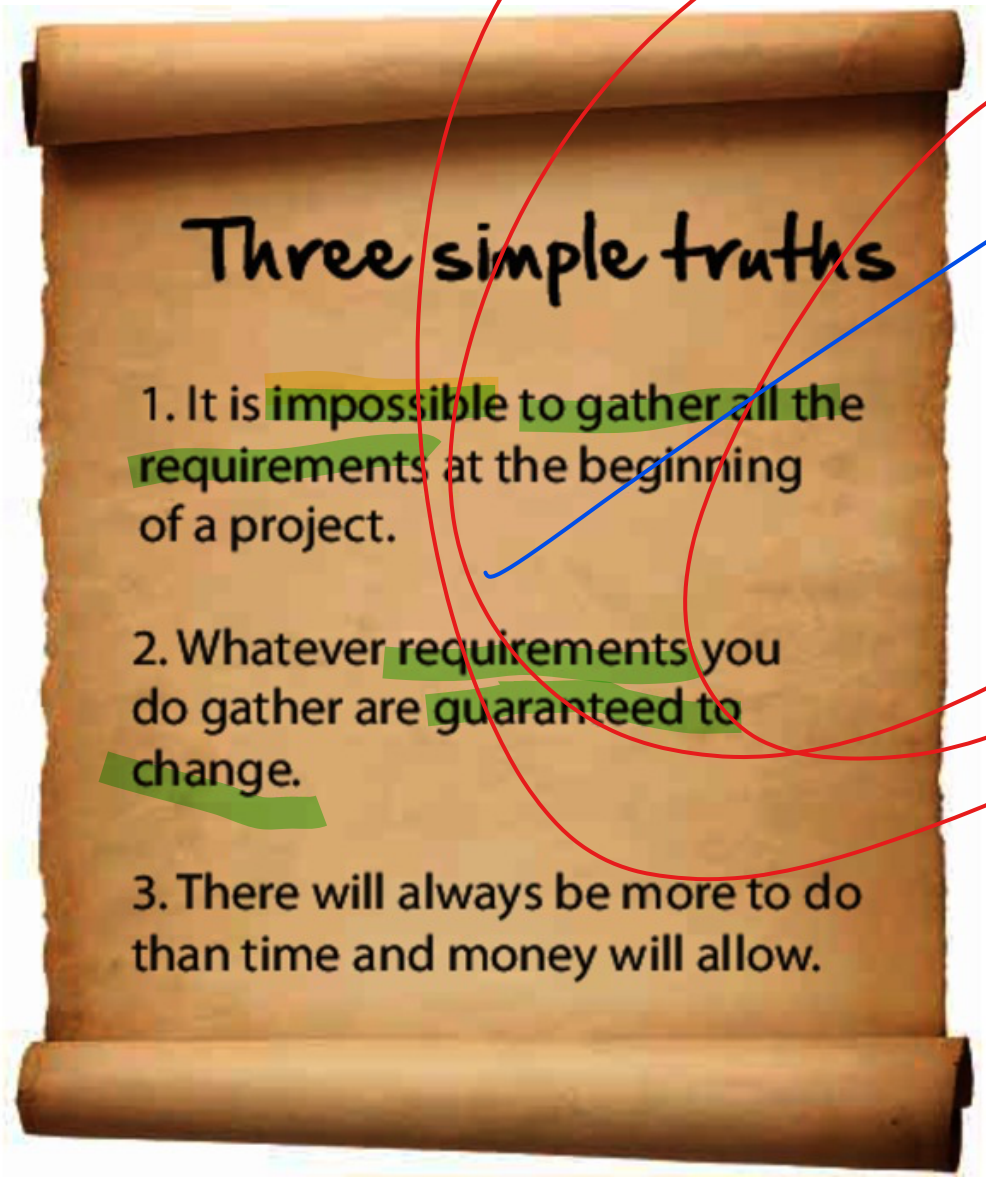
ธรรมไจล์

ความสำเร็จทั้งหลายของเจ้าวัดได้ด้วยซอฟท์แวร์ที่ทำงานได้จริงเท่านั้น

Three Simple Truths

believe to this 3 truths to avoid drama

https://www.oreilly.com/library/view/the-agile-samurai/9781680500066/f_0015.html

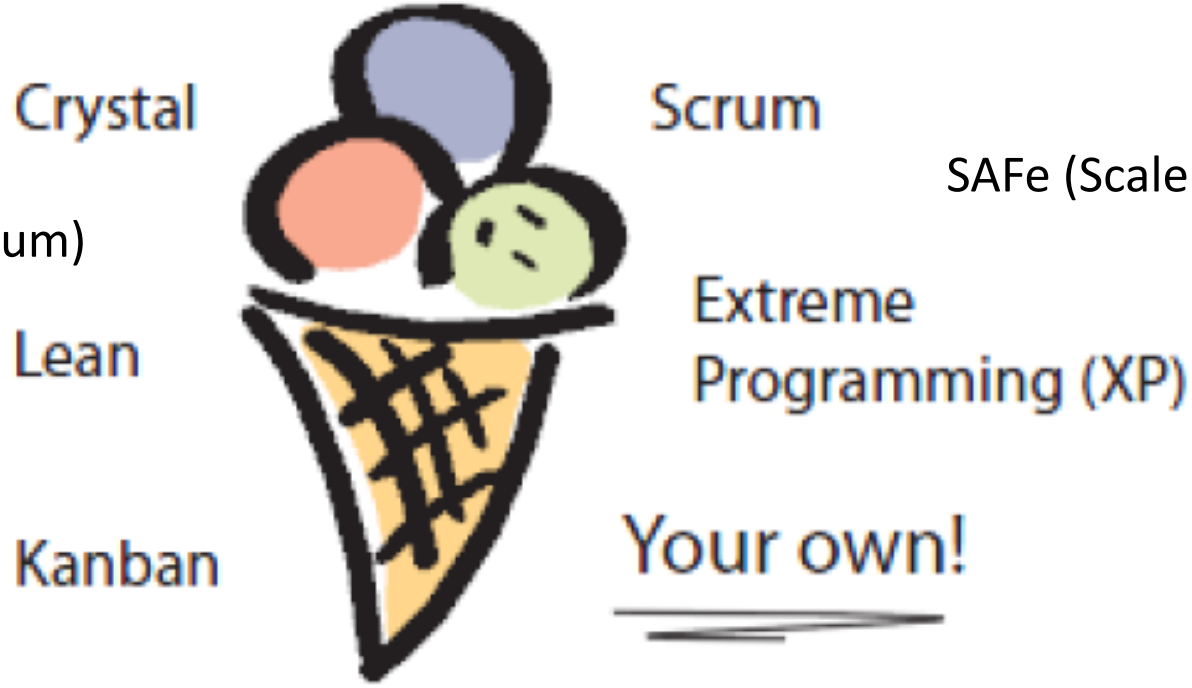


three simple truth
1. It is impossible to gather all the requirements at the beginning of a project
2. whatever requirements you do gather are garanteed to change
3. there will always be more to do than time and money will allow

- Once you accept these three simple project truths, much of the stress and anxiety traditionally associated with software delivery disappears.
- You are then able to think and innovate with a level of focus and clarity that escapes most in our industry.

ไอติมหลากรส (Agile SW process models)

Different agile methodologies have emerged over the years

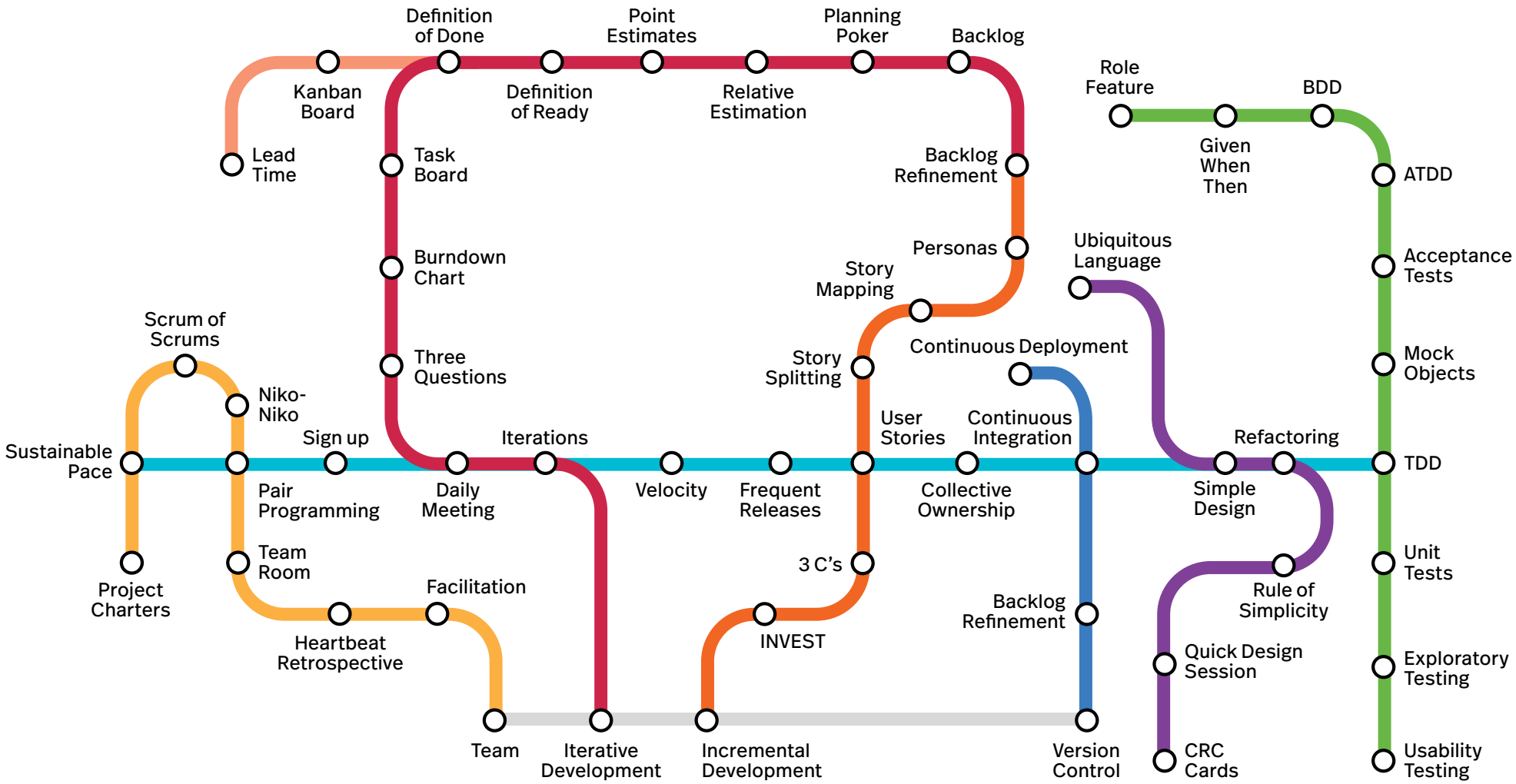


FDD (Feature-driven development)

DSDM (formally known as Dynamic System Development Method)

https://www.scrum.org/resources/what-scrumbut

https://www.agilealliance.org/wp-content/uploads/2023/11/agile-alliance-subway-map-to-agile-practices.pdf



The colored "subway" lines represent practices from the various Agile approaches or areas of concern.

Extreme Programming	Scrum	Design
Teams	Product Management	Testing
Lean	DevOps	Fundamentals

https://www.agilealliance.org/agile101/subway-map-to-agile-practices/

Common Employment

- Small, close-knit teams
- Regular, frequent, disciplined customer requirements meetings
- A **code-centric** approach, **documentation on an as-needed** basis (e.g., **high-level** requirements statements only)
  - The use of user stories as the basis for requirements-end-to-end accounts of how users need to accomplish individual tasks
- Customer representatives working within the team

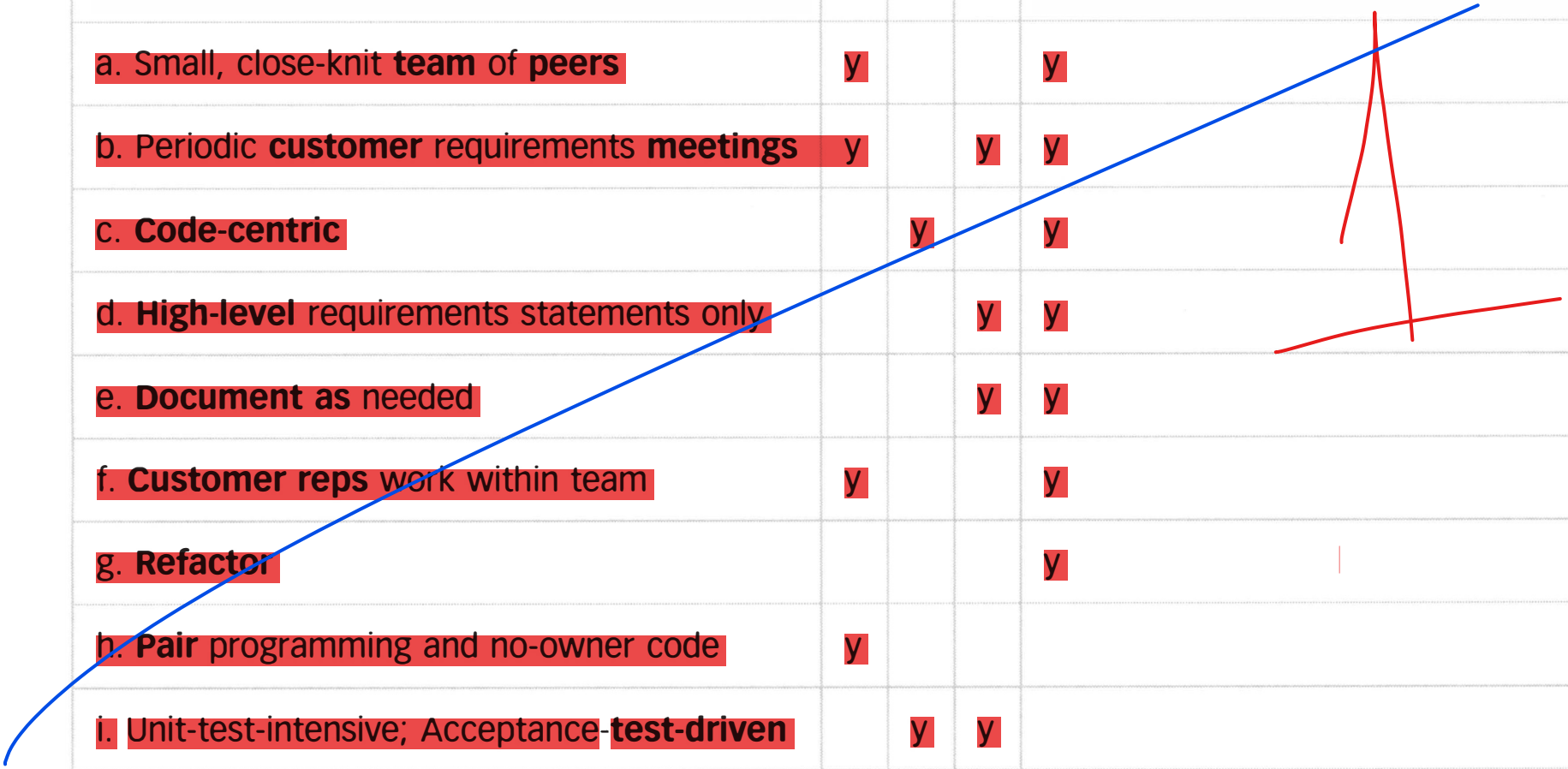
Common Employment

- Refactoring, a kind of disciplined code improvement
- Pair programming, in which two programmers work at a single workstation
- Continual unit-testing, and acceptance tests as means of setting customer expectations

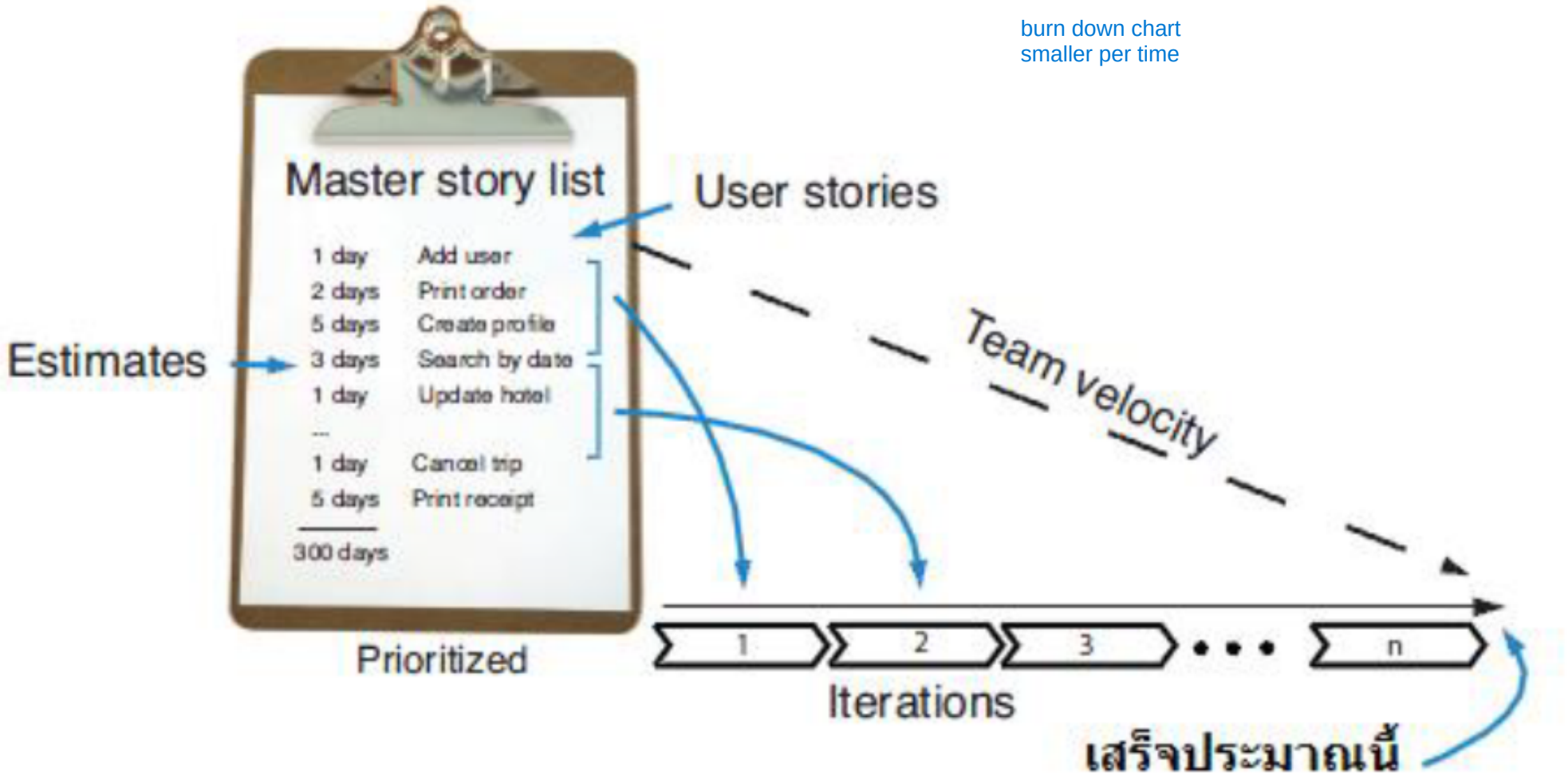
Ways to address

the principles of the Agile Manifesto

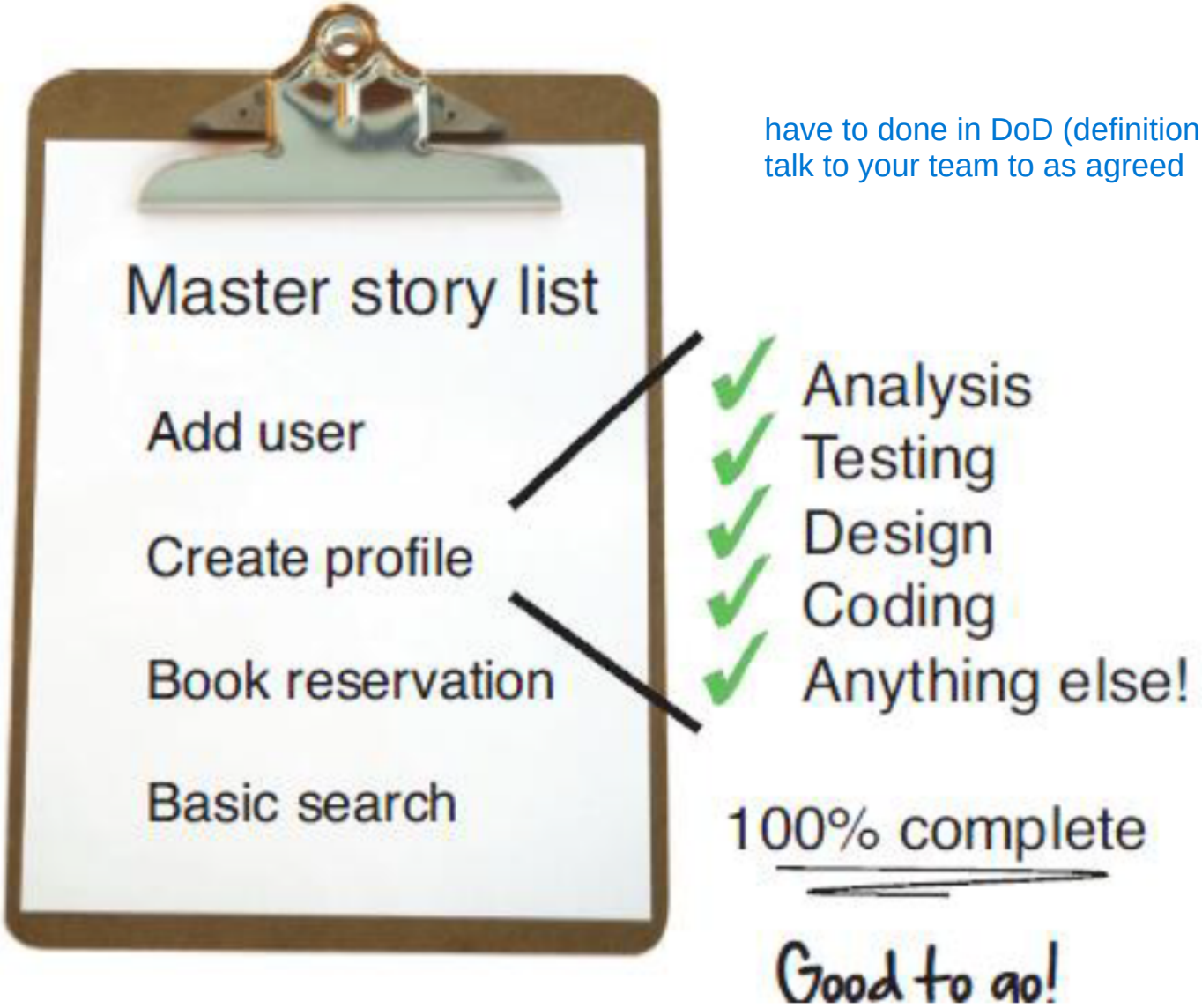
MANIFESTO →	1. Individuals and interactions over processes and tools	2. Working software over comprehensive documentation	3. Customer collaboration over contract negotiation	4. Responding to change over following a plan
RESPONSES:				
a. Small, close-knit **team** of **peers**	y			y
b. Periodic **customer** requirements **meetings**	y		y	y
c. **Code-centric**		y		y
d. **High-level** requirements statements only			y	y
e. **Document as** needed			y	y
f. **Customer reps** work within team	y			y
g. **Refactor**				y
h. **Pair** programming and no-owner code	y			
i. Unit-test-intensive; Acceptance-**test-driven**		y	y	
j. **Automate** testing		y	y	



Agile


burn down chart
smaller per time
Master story list
1 day Add user
2 days Print order
5 days Create profile
3 days Search by date
1 day Update hotel
...
1 day Cancel trip
5 days Print receipt
300 days
User stories
Estimates
Team velocity
Prioritized
1
2
3
n
Iterations
เสร็จประมาณนี้

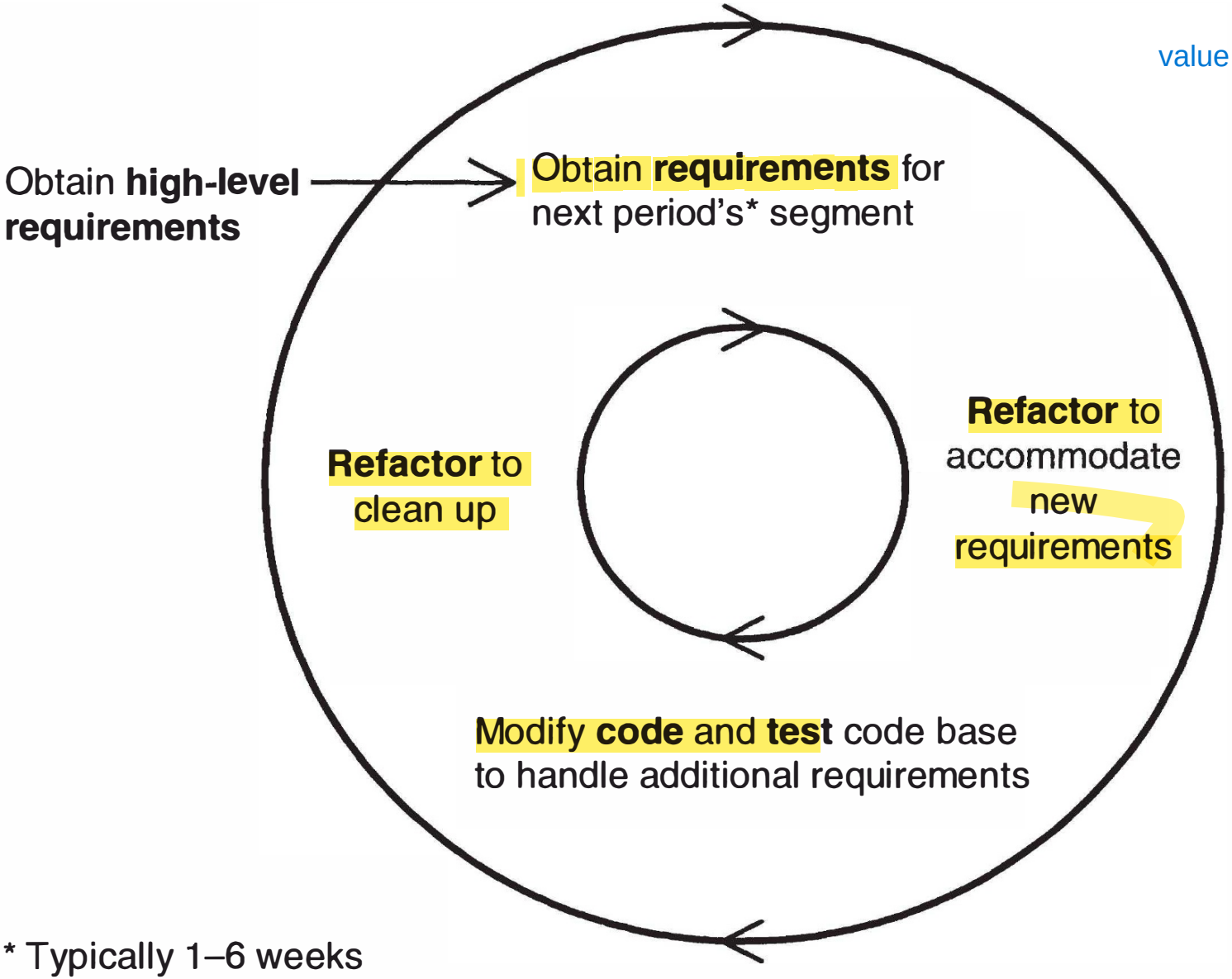
เสร็จแปลว่าเสร็จ



have to done in DoD (definition of done)
talk to your team to as agreed

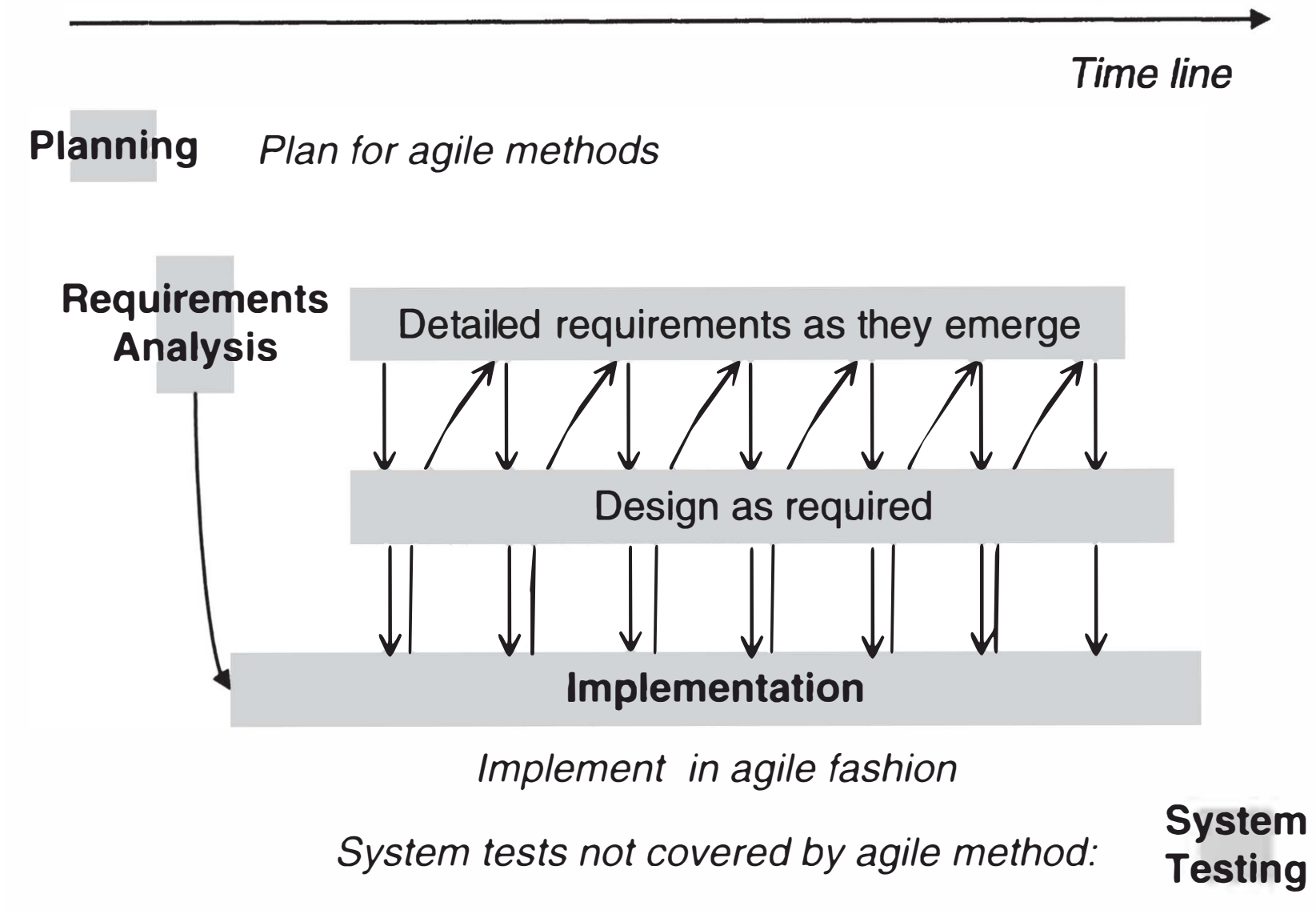
Typical Agile Development Iteration

value is on coding



* Typically 1–6 weeks

The Agile Schedule



Common Characteristics

- The processes of specification, design and implementation are interleaved.
- There is no detailed system specification, and design documentation is minimized or generated automatically by the programming environment used to implement the system.
- The user requirements document is an outline definition of the most important characteristics of the system.
- The system is developed in a series of increments.
- End-users and other system stakeholders are involved in specifying and evaluating each increment.
- Extensive tool support is used to support the development process.

Agile

- Pros
  - The project always has demonstrable results: The end product of each iteration is working software.
  - Developers tend to be more motivated: Developers prefer to produce working artifacts and tend not to like creating documentation.
  - Customers are able to provide better requirements because they can see the evolving product.
- Cons
  - Problematical for large application: Agile methods are more readily used for smaller projects. There is debate about their utility for large projects.
  - Documentation output is questionable: Since documentation takes second place, there is a question as to whether necessary documentation will ever be produced.