Software Engineering Research Laboratory

Week 2: Intro to Agile SRE

409220 Software Requirements Engineering

Jim Buchan

AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

The aims for Today

- Review the challenges in SRE and importance of getting it right
- A brief History of SRE-why look back?
- What are some principles and approaches using proven practice
- Intro to Agile SRE
- Course Overview and Assessment 1

AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

What are the challenges with SRE?

- What do you remember from last week?

What are the RE processes in different SE approaches and methodologies?

(and under what circumstances might I select different ones?)

AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

What processes are ~~best~~ proven practice

What are the RE processes in different SE approaches and methodologies?

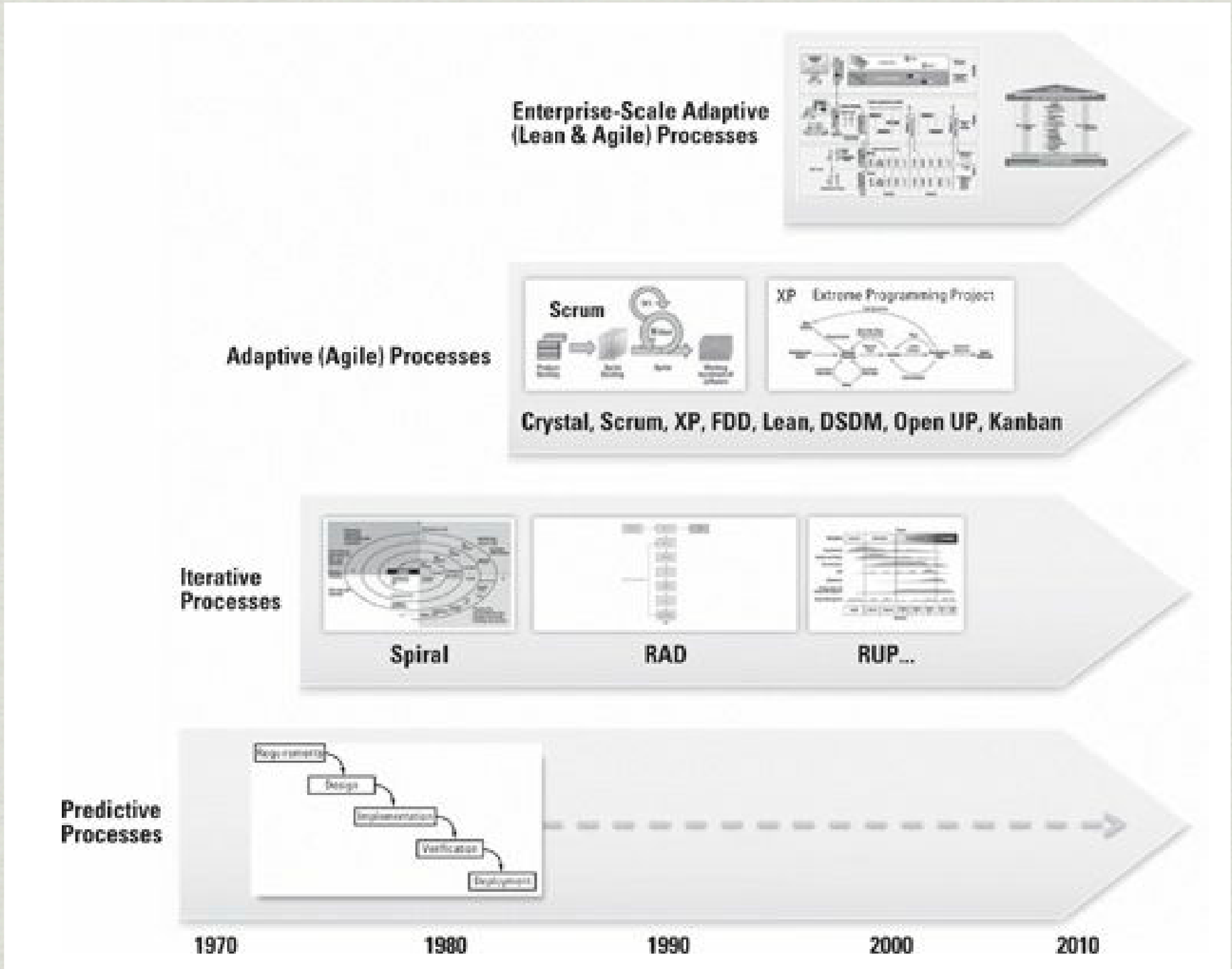
(and under what circumstances might I select different ones?)

AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

I need a volunteer

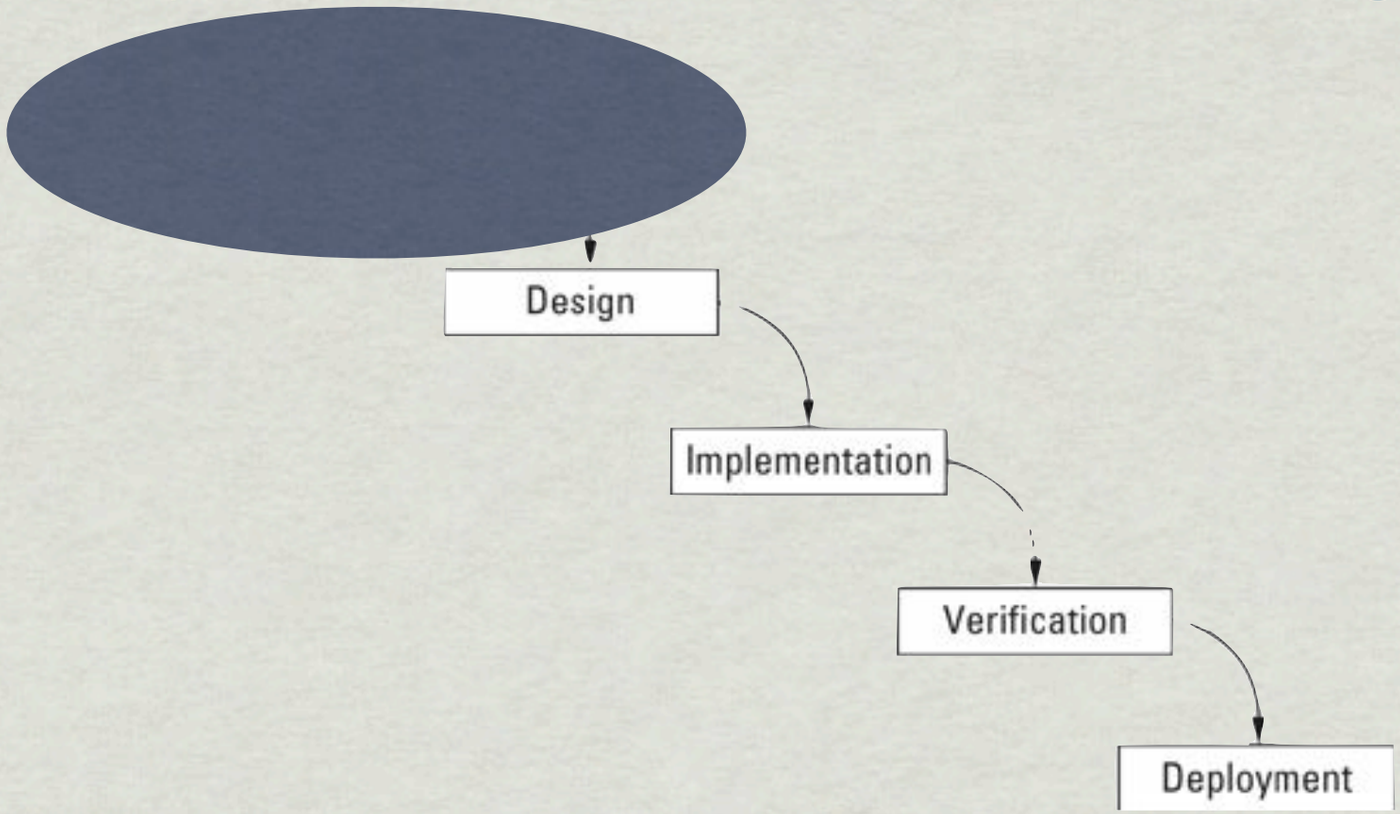
A brief history of SRE

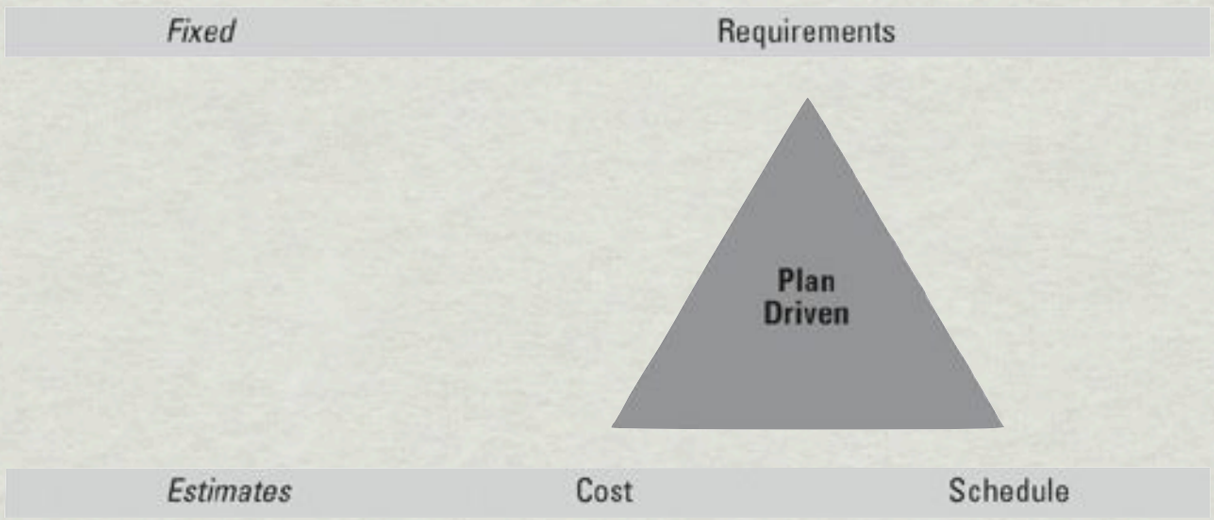
Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

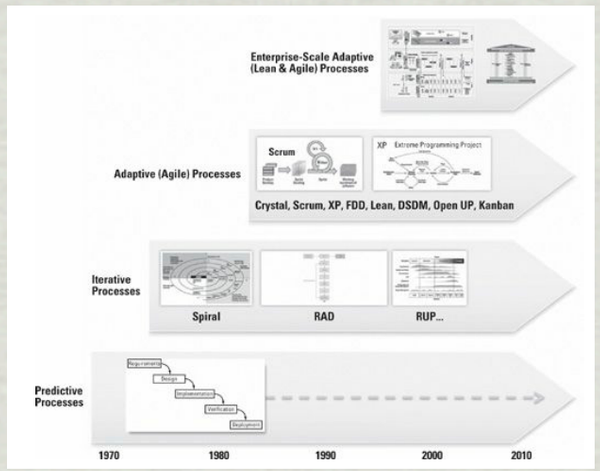


A brief history of SRE

Leffingwell, D. (2011). Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the



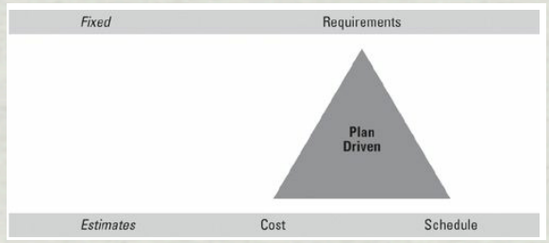


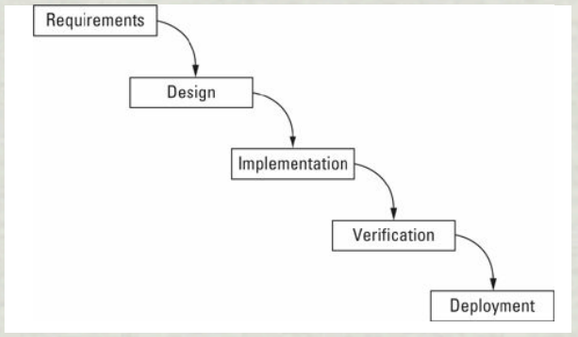


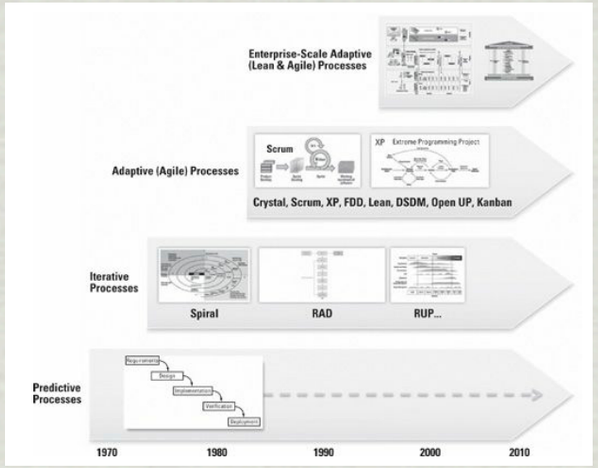
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Waterfall

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.





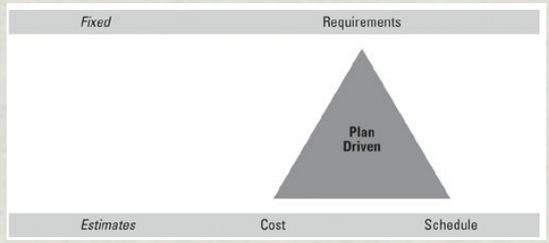


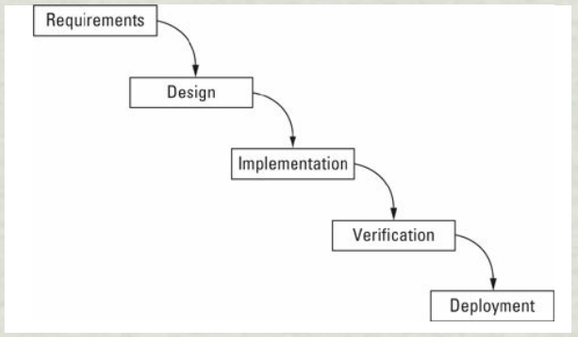
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

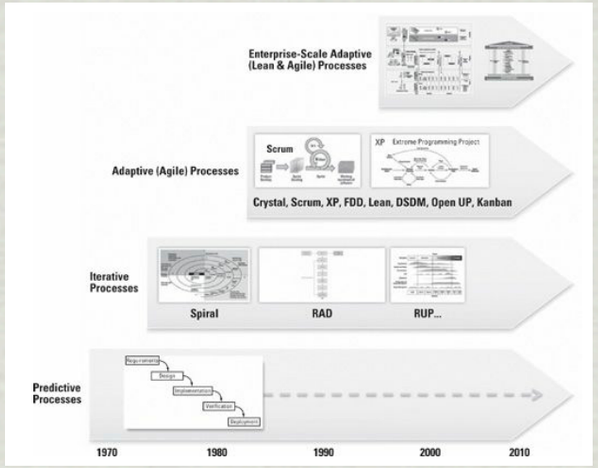
Waterfall

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- The model was itself born as a fix to an earlier problem, which was the “code it, fix-it, code-it-some-more-until-it’s-quickly-not-maintainable” tendency of prior practices.







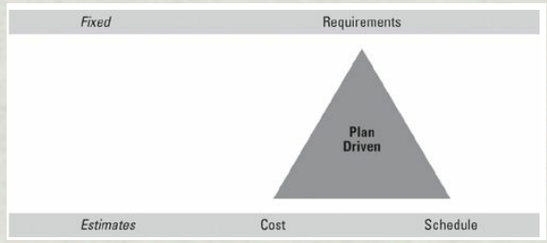
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

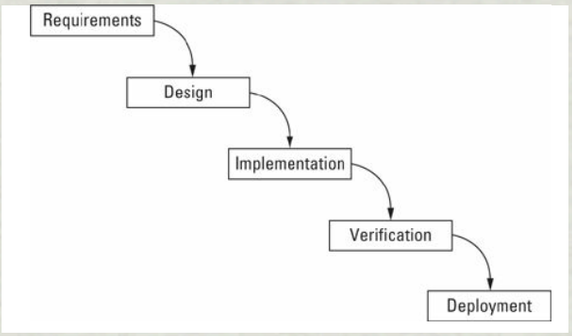
Waterfall

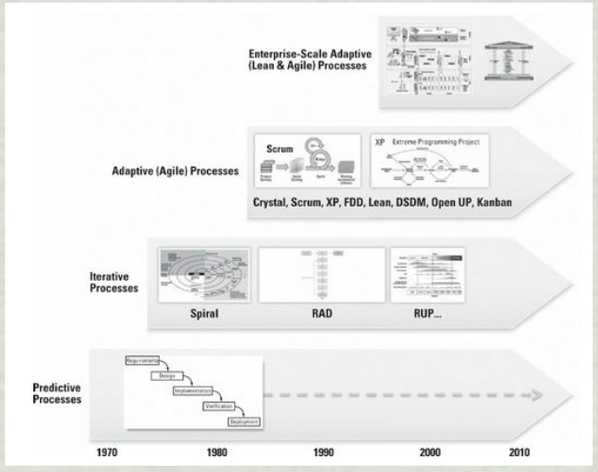
Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- The model was itself born as a fix to an earlier problem, which was the “code it, fix-it, code-it-some-more-until-it’s-quickly-not-maintainable” tendency of prior practices.

- It appears to be extremely logical and prescriptive. Understand requirements. Design a system that conforms. Code it. Test it. What could be more sensible and logical than that?





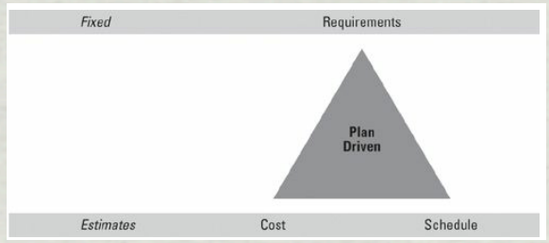


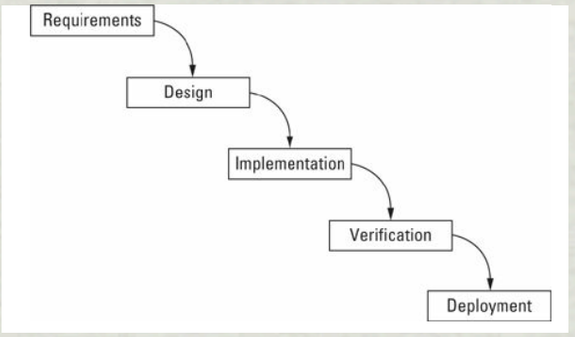
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

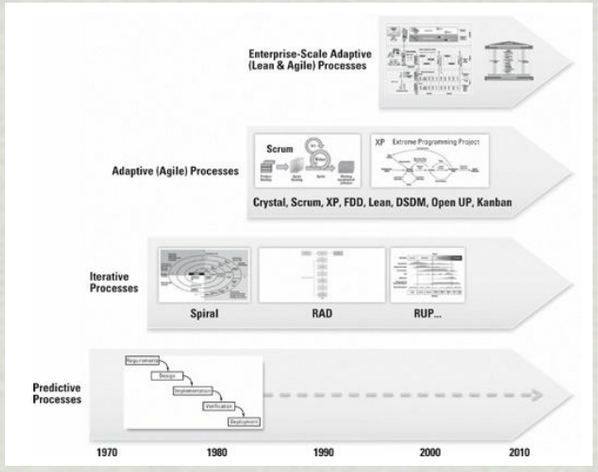
Waterfall

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- The model was itself born as a fix to an earlier problem, which was the “code it, fix-it, code-it-some-more-until-it’s-quickly-not-maintainable” tendency of prior practices.
- It appears to be extremely logical and prescriptive. Understand requirements. Design a system that conforms. Code it. Test it. What could be more sensible and logical than that?
- It worked to a point (we did and still do ship a lot of software using the model). As a result, companies built their project and program governance models, including business case and investment approvals, project review and quality assurance milestones, and the like, around its flawed software life cycle.







AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Waterfall

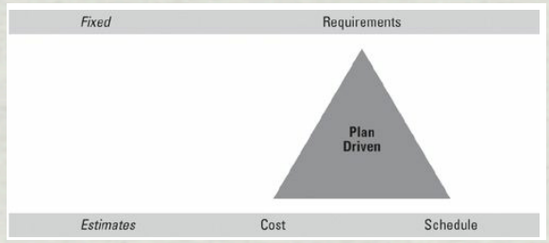
Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

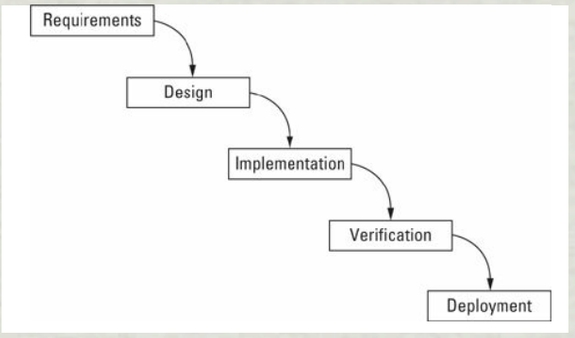
- The model was itself born as a fix to an earlier problem, which was the “code it, fix-it, code-it-some-more-until-it’s-quickly-not-maintainable” tendency of prior practices.

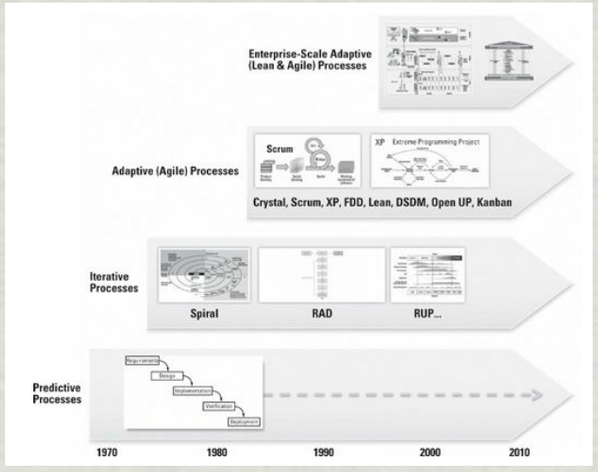
- It appears to be extremely logical and prescriptive. Understand requirements. Design a system that conforms. Code it. Test it. What could be more sensible and logical than that?

- It worked to a point (we did and still do ship a lot of software using the model). As a result, companies built their project and program governance models, including business case and investment approvals, project review and quality assurance milestones, and the like, around its flawed software life cycle.

- It reflects a continuing market reality—customers still do impose fixed-date/fixed- requirements agreements on suppliers, and they will likely continue to do so for years to come. (And, yes, sometimes we impose them on ourselves.)

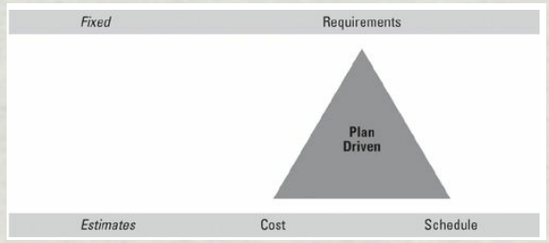


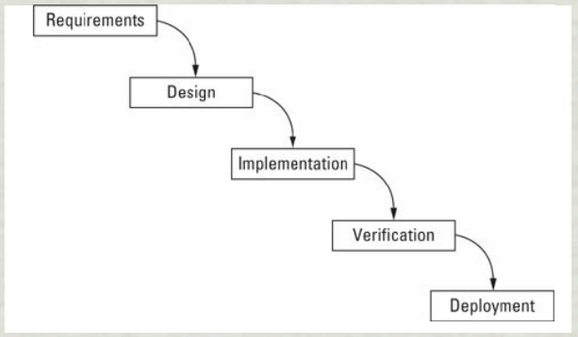


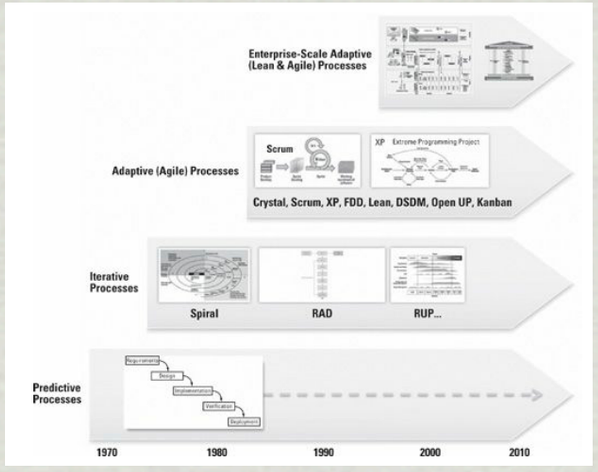


AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

- The model was itself born as a fix to an earlier problem, which was the “code it, fix-it, code-it-some-more-until-it’s-quickly-not-maintainable” tendency of prior practices.
- It appears to be extremely logical and prescriptive. Understand requirements. Design a system that conforms. Code it. Test it. What could be more sensible and logical than that?
- It worked to a point (we did and still do ship a lot of software using the model). As a result, companies built their project and program governance models, including business case and investment approvals, project review and quality assurance milestones, and the like, around its flawed software life cycle.
- It reflects a continuing market reality—customers still do impose fixed-date/fixed- requirements agreements on suppliers, and they will likely continue to do so for years to come. (And, yes, sometimes we impose them on ourselves.)





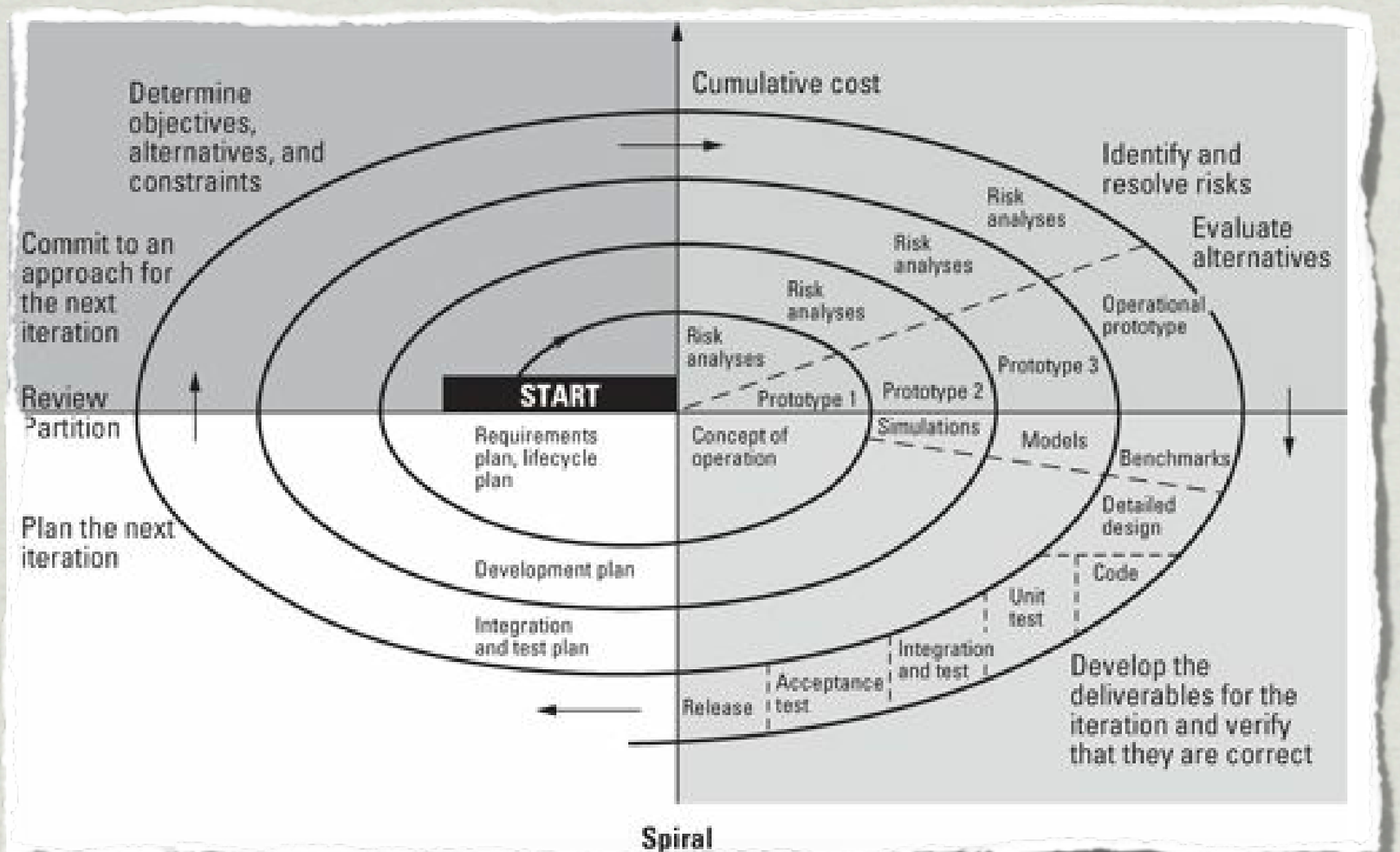


AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

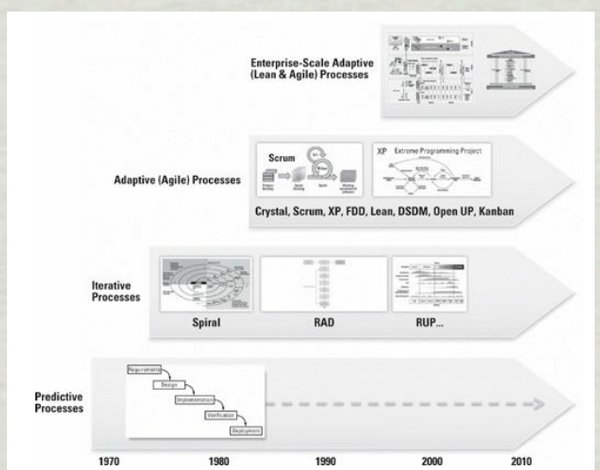
Spiral

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

Discovery Based Processes Iterative and Incremental



Spiral

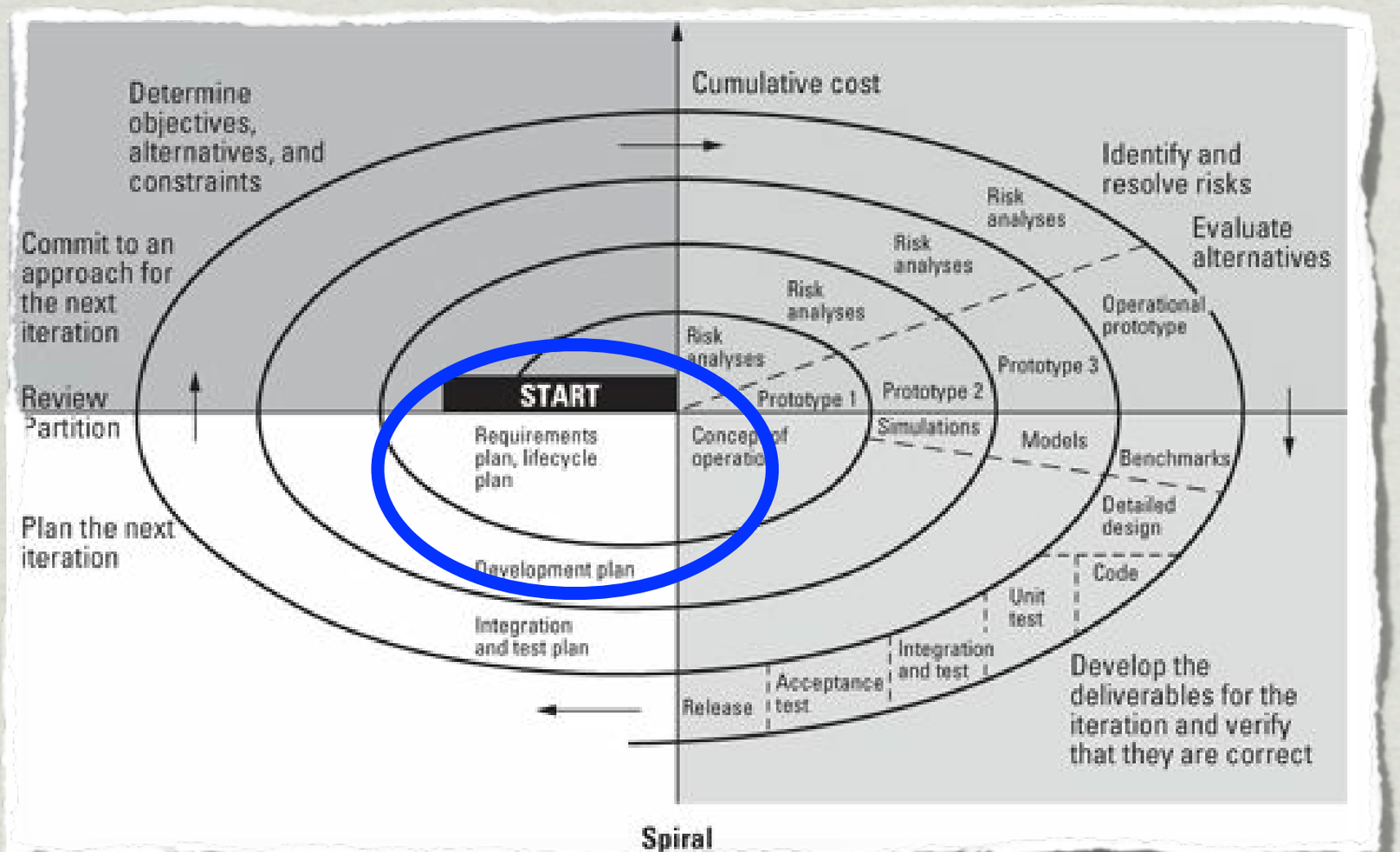


AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

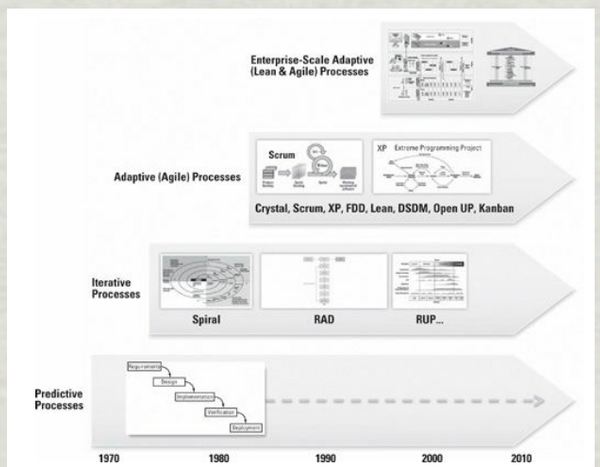
Spiral

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

Discovery Based Processes Iterative and Incremental



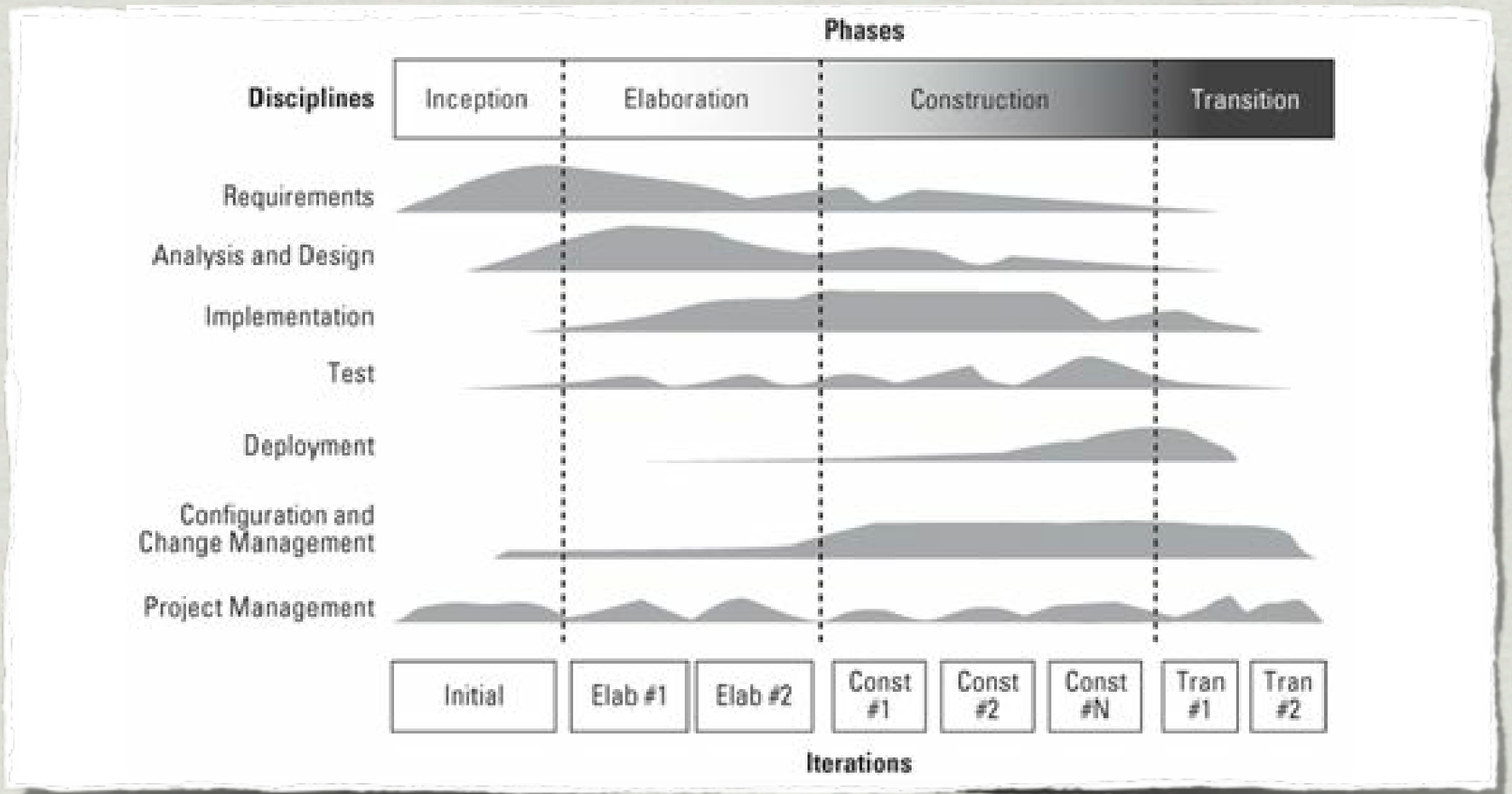
Spiral

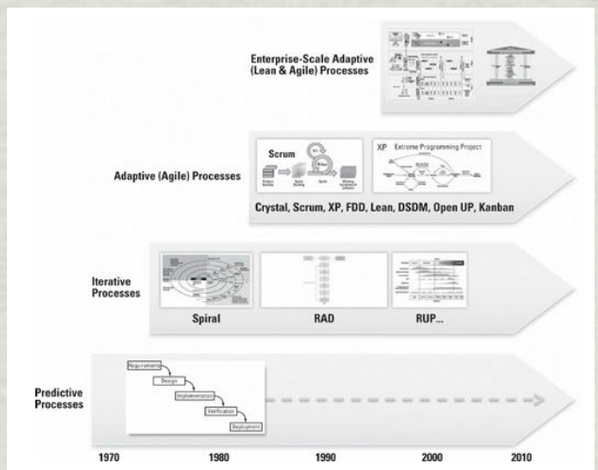


AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Rapid Application Development

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.



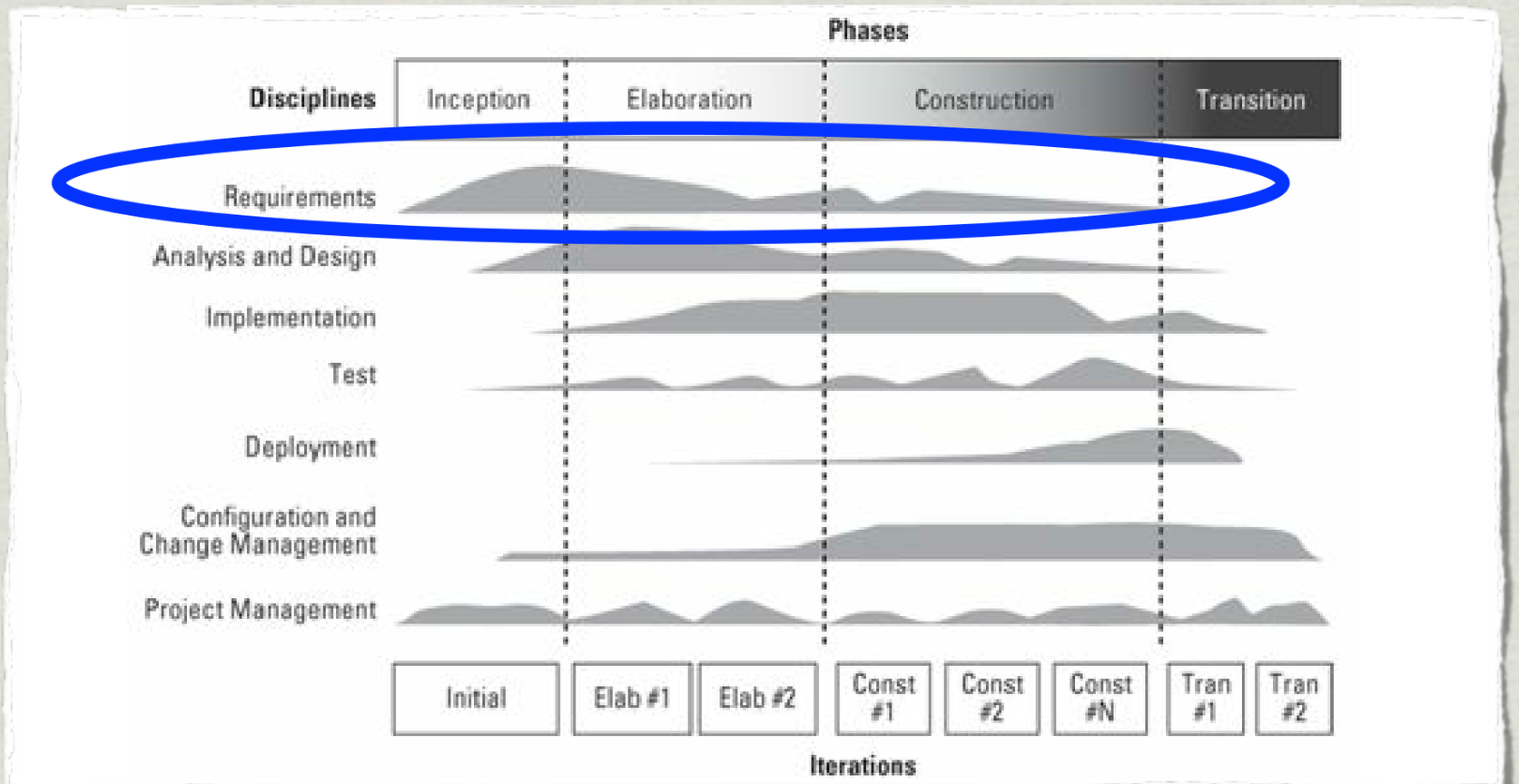


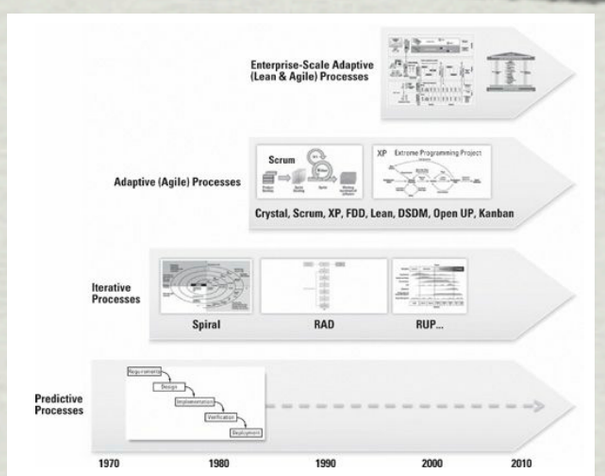
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Rapid Application Development

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- Recognise overlapping activities.



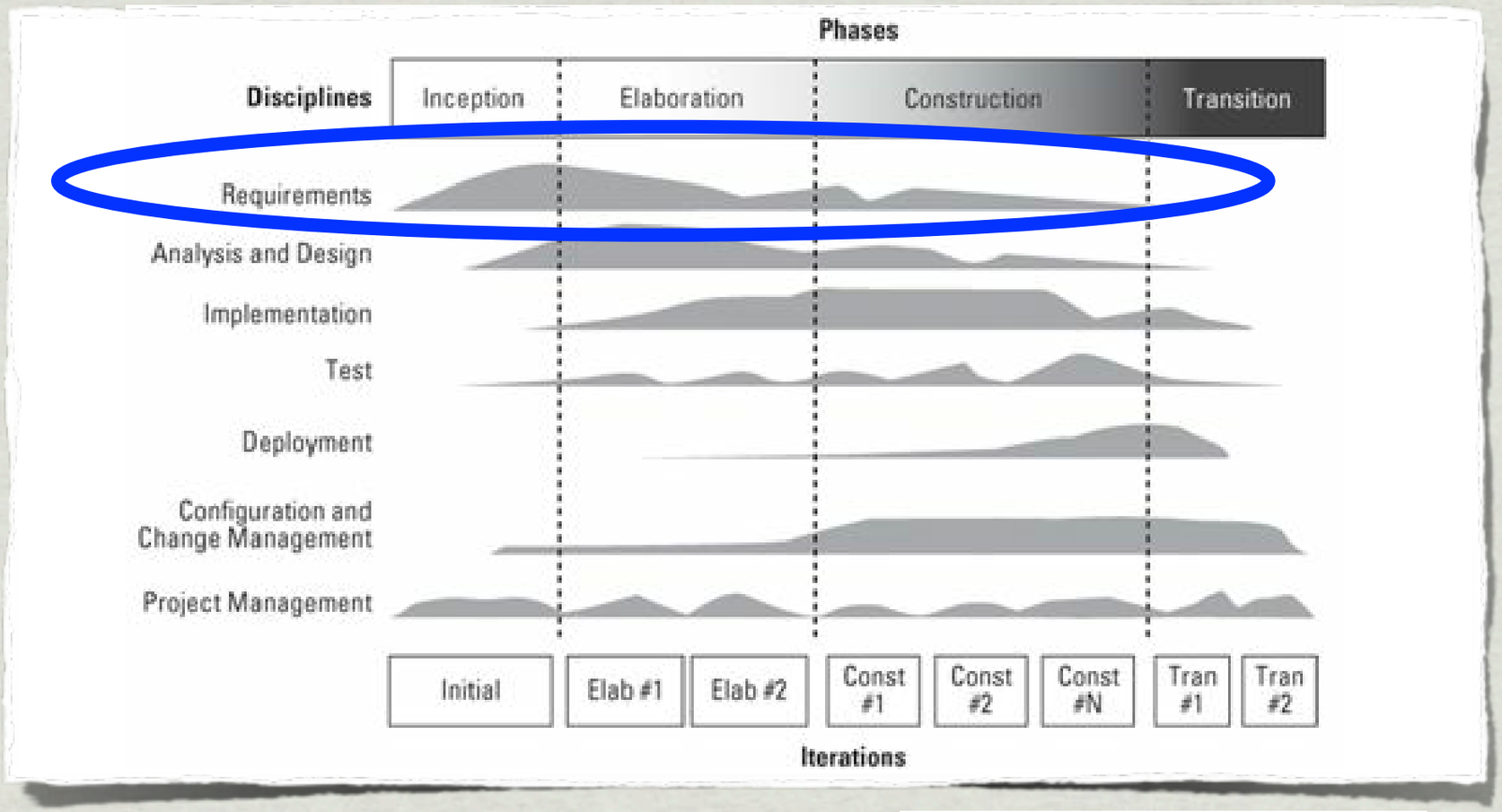


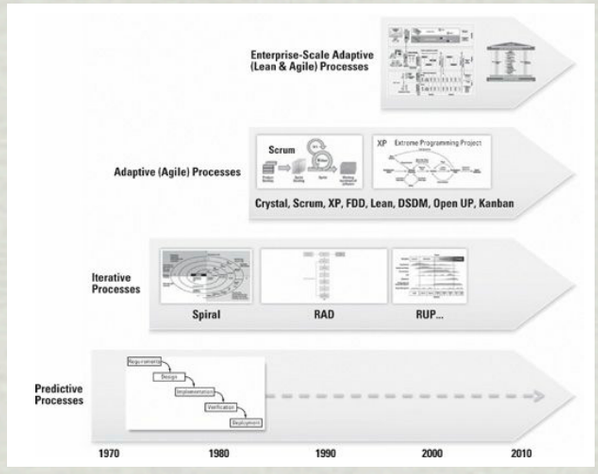
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Rapid Application Development

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- Recognise overlapping activities.
- Move away from big up-front requirements and design.



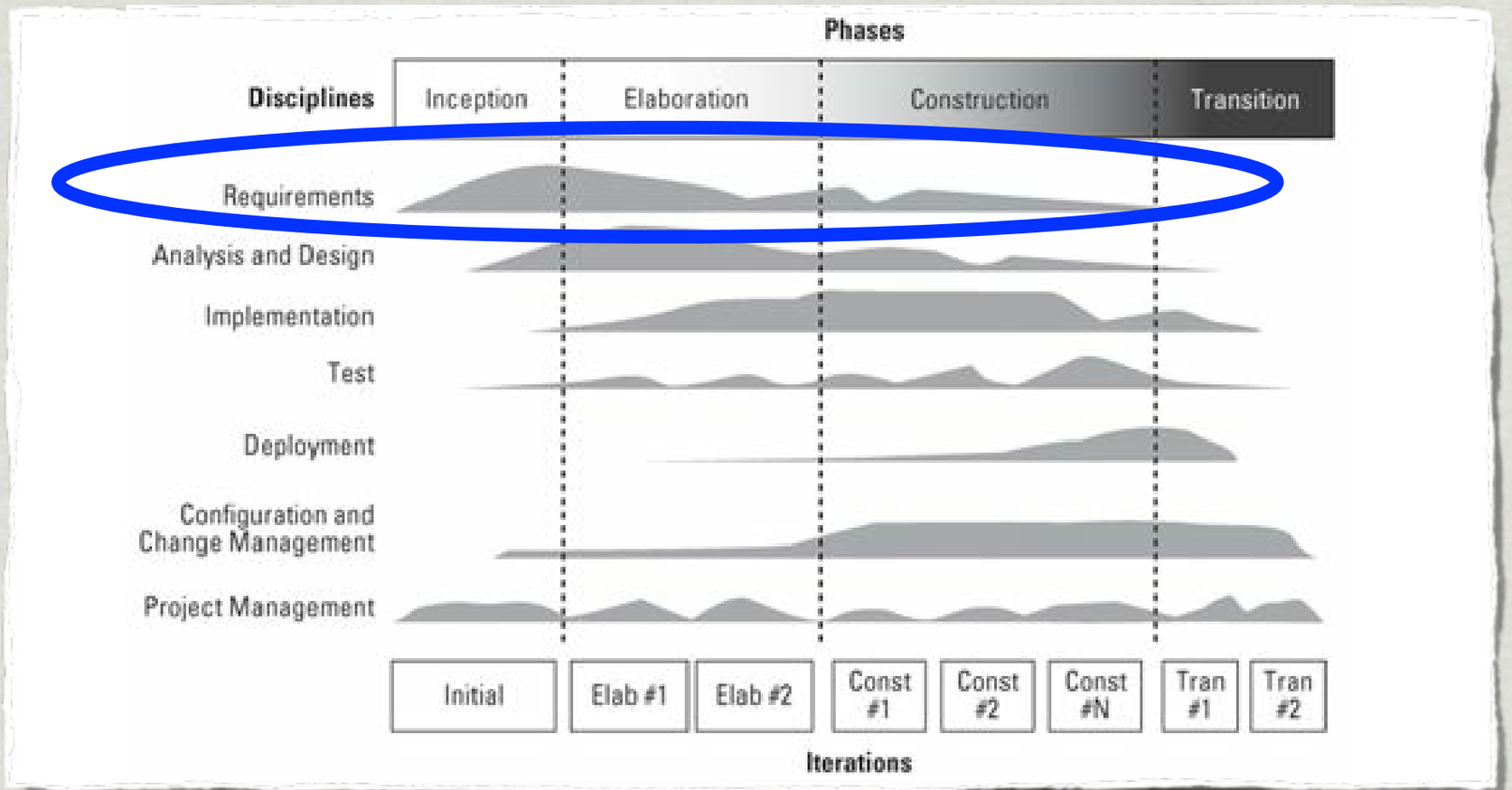


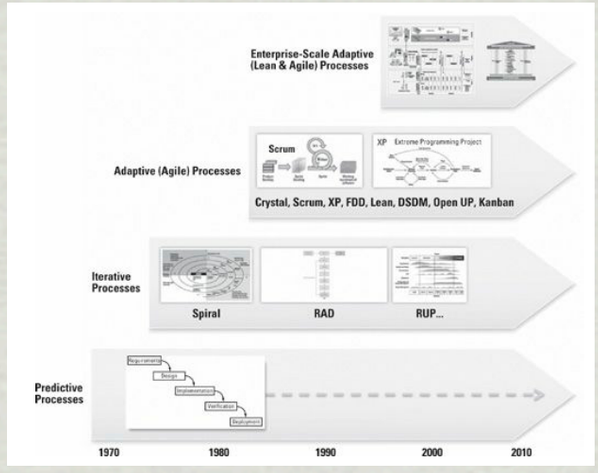
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Rapid Application Development

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- Recognise overlapping activities.
- Move away from big up-front requirements and design.
- Based on Spiral-for large-scale projects



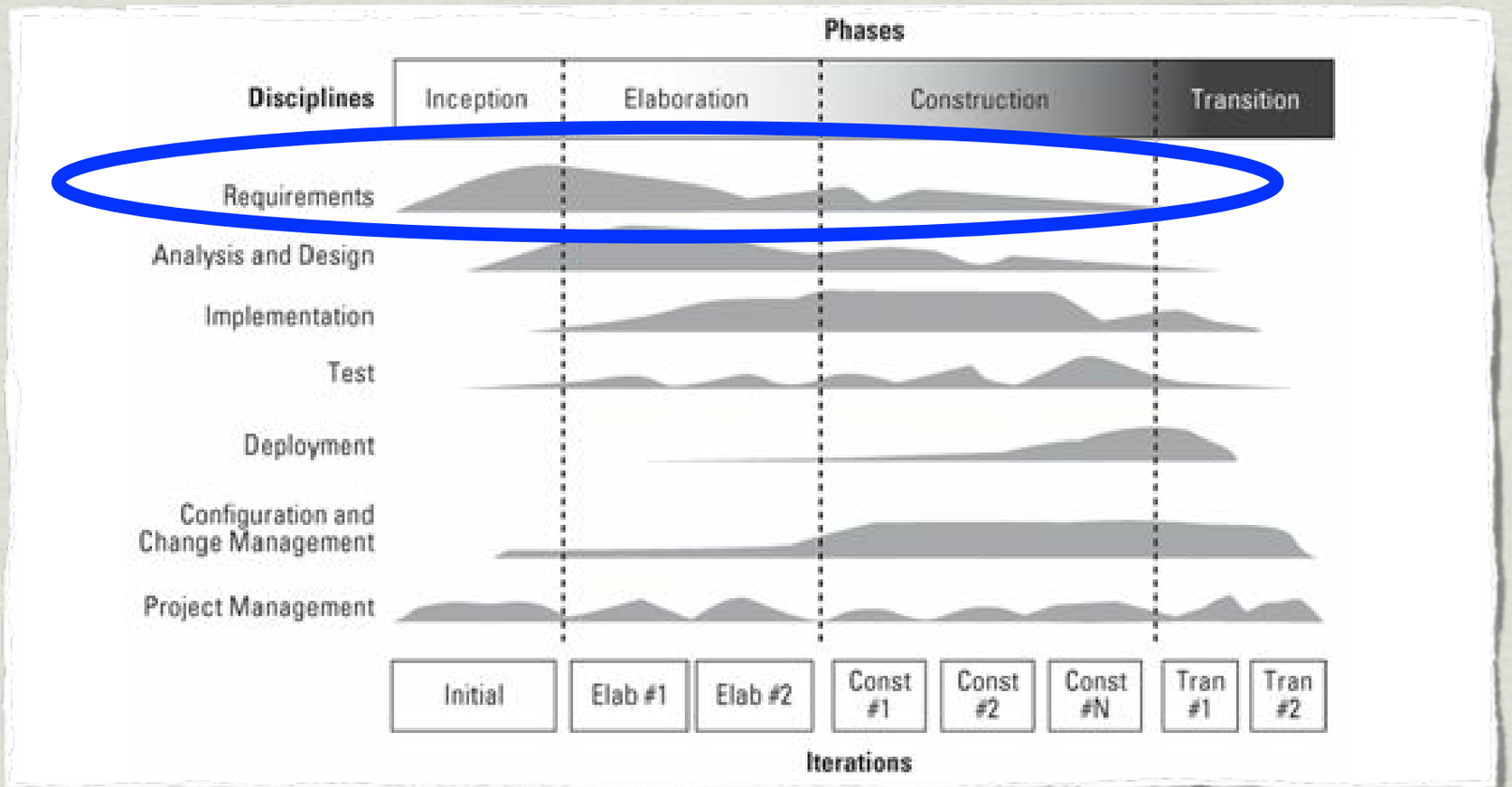


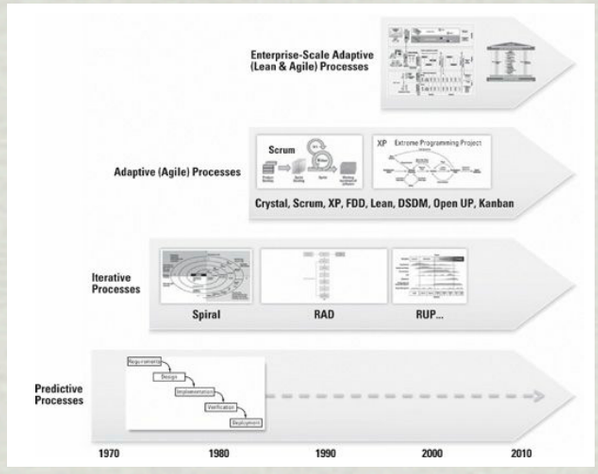
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Rapid Application Development

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- Recognise overlapping activities.
- Move away from big up-front requirements and design.
- Based on Spiral-for large-scale projects
- Agile RUP and openUP are recent lightweight versions



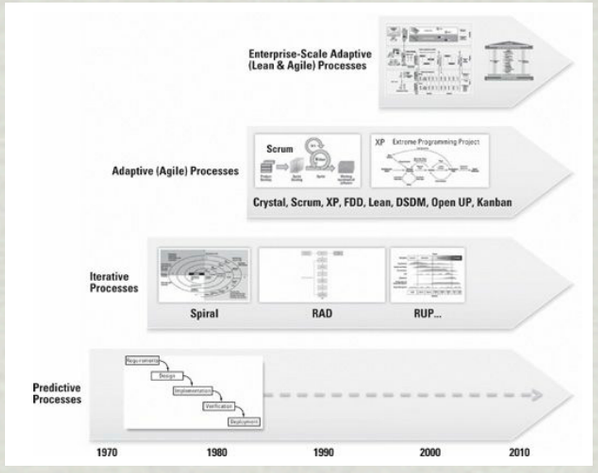


AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Agile

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

Individuals and interactions	over	[redacted]
Customer collaboration	over	[redacted]
Working software	over	[redacted]
Responding to change	over	[redacted]

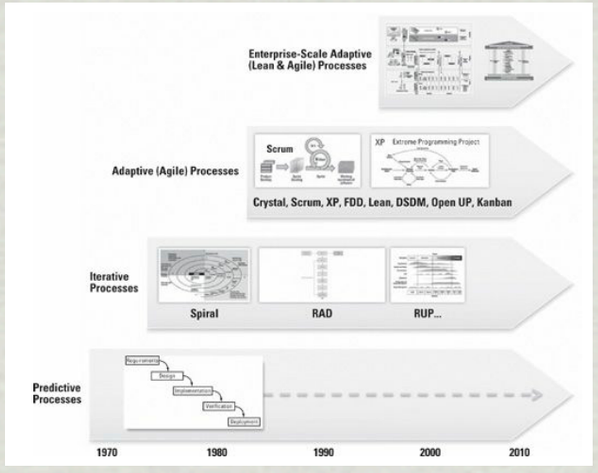


AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Agile

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

Individuals and interactions	over	processes and tools
Customer collaboration	over	
Working software	over	
Responding to change	over	

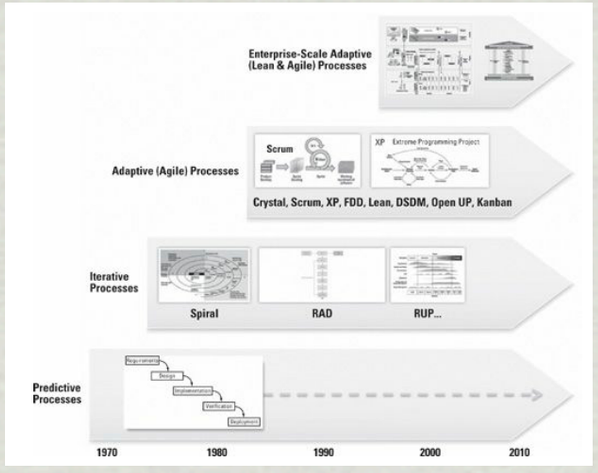


AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Agile

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

Individuals and interactions	over	processes and tools
Customer collaboration	over	contract negotiation
Working software	over	[redacted]
Responding to change	over	[redacted]

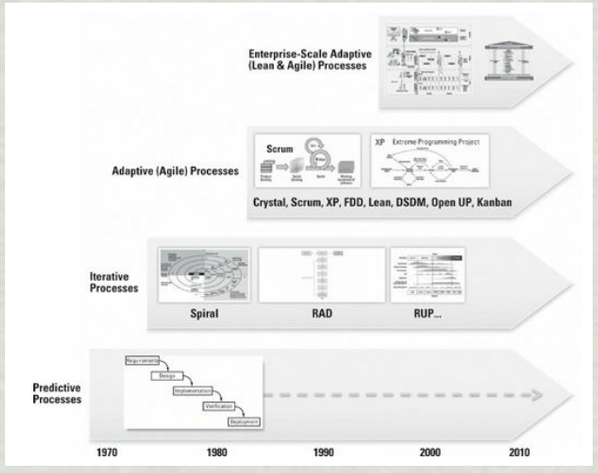


AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Agile

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

Individuals and interactions	over	processes and tools
Customer collaboration	over	contract negotiation
Working software	over	comprehensive documentation
Responding to change	over	[redacted]

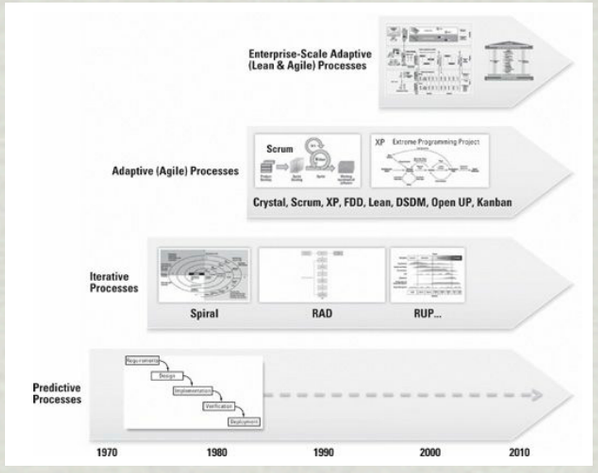


AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Agile

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

Individuals and interactions	over	processes and tools
Customer collaboration	over	contract negotiation
Working software	over	comprehensive documentation
Responding to change	over	following a plan

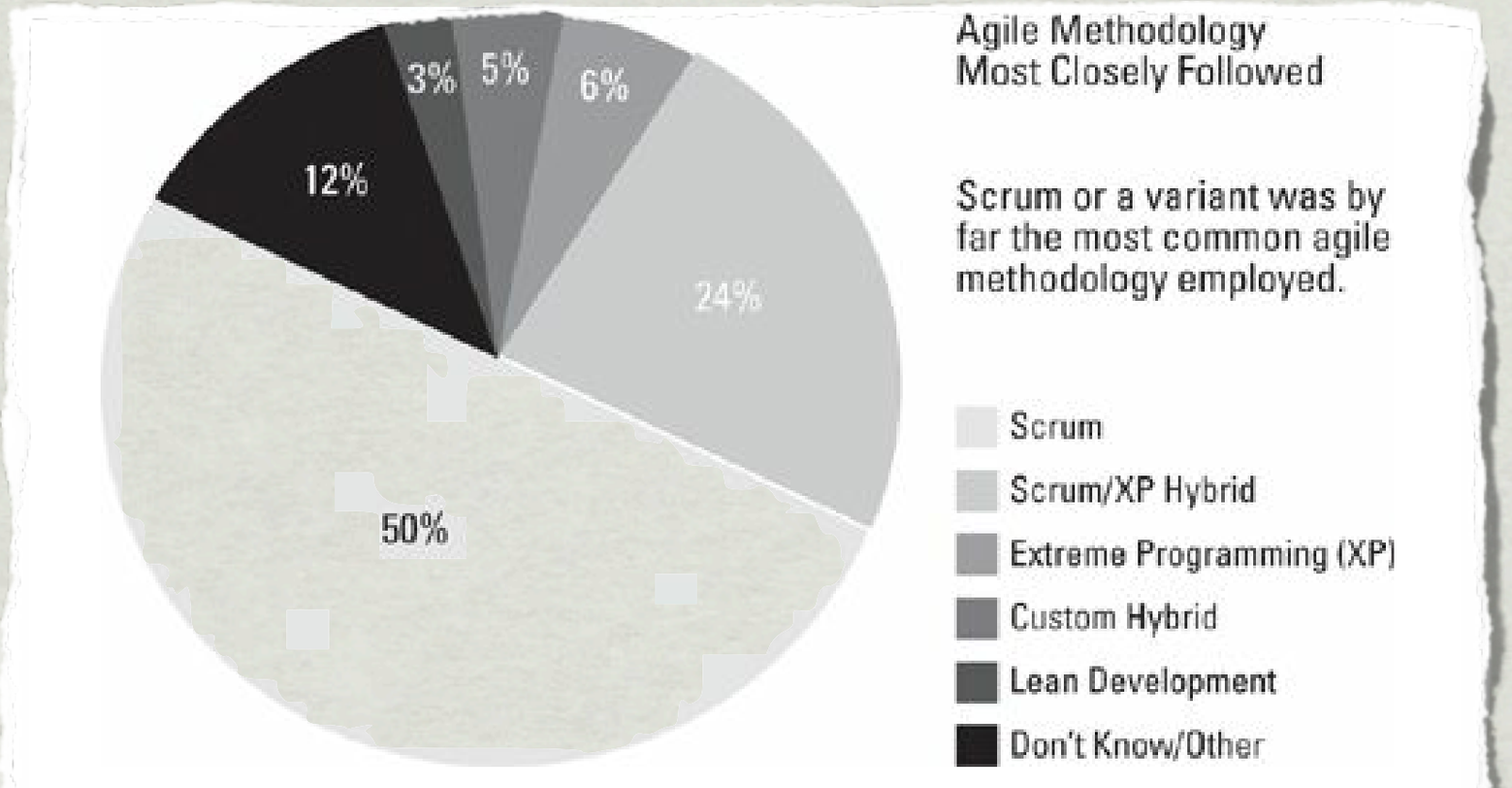


AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

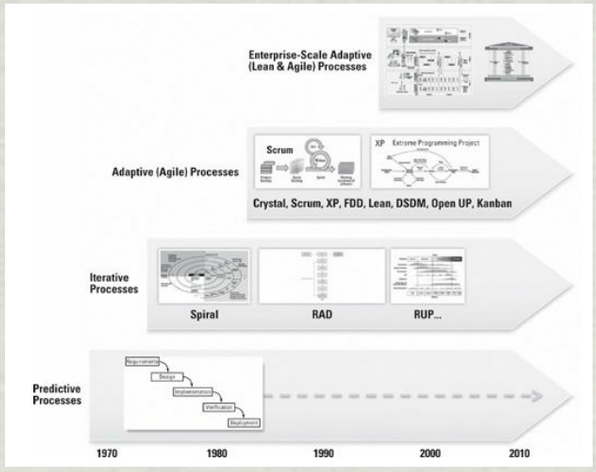
Agile

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

Individuals and interactions	over	processes and tools
Customer collaboration	over	contract negotiation
Working software	over	comprehensive documentation
Responding to change	over	following a plan



Courtesy of VersionOne, Inc.

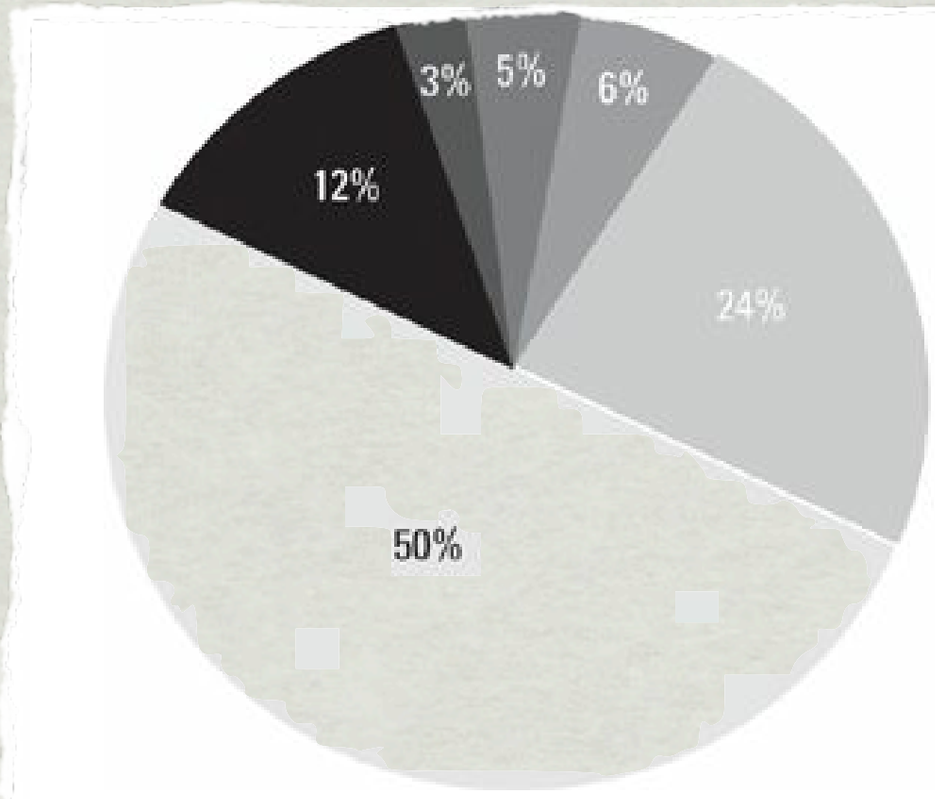


AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Agile

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

Individuals and interactions	over	processes and tools
Customer collaboration	over	contract negotiatio
Working software	over	comprehensive do
Responding to change	over	following a plan



Agile Me
Most Clo

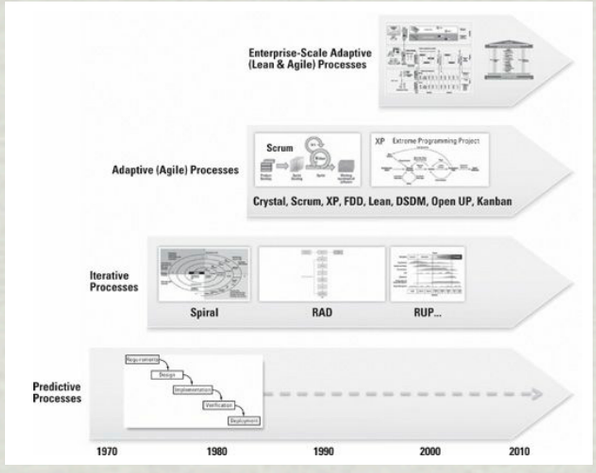
Scrum or a
far the most common agile
methodology employed.

- Scrum
- Scrum/XP Hybrid
- Extreme Programming (XP)
- Custom Hybrid
- Lean Development
- Don't Know/Other

Courtesy of VersionOne, Inc.

Source: Version One's 2009 Agile Methodology Survey

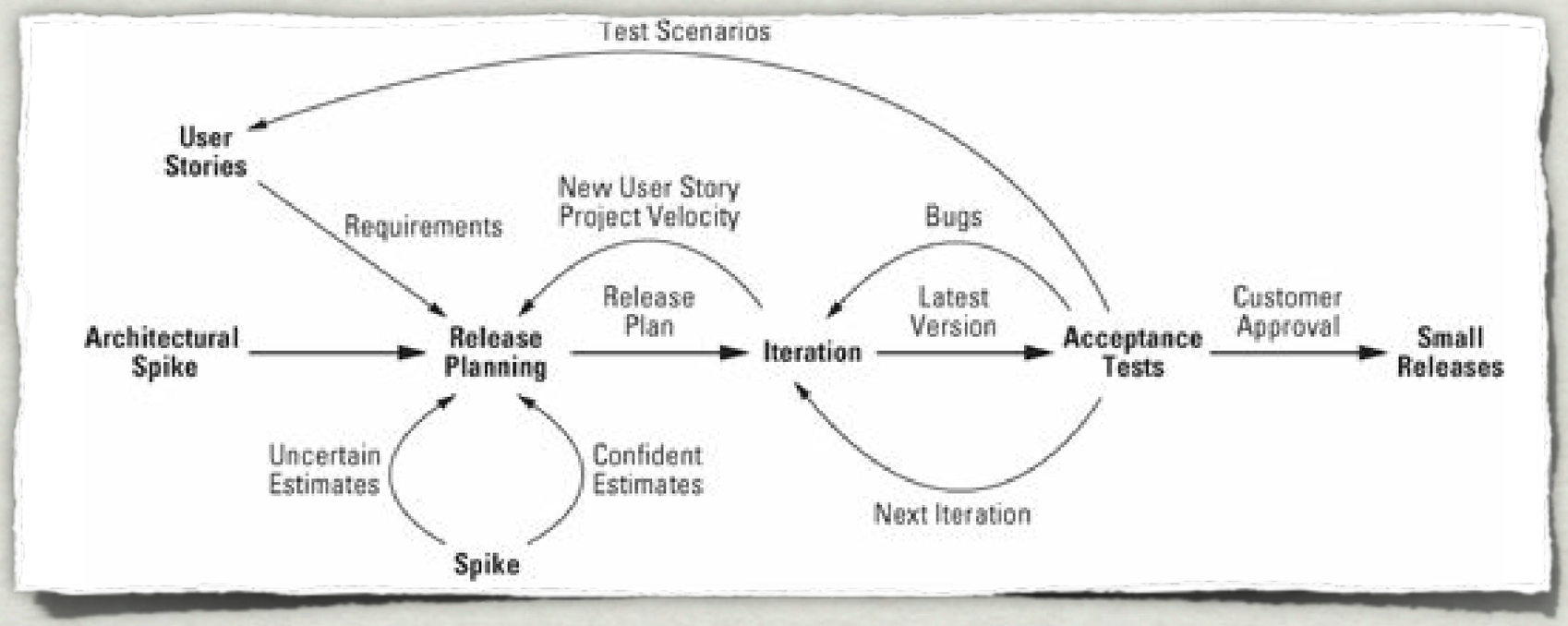
Sample size?
Types of organisations in sample?
Sample selection method?
VersionOne sells Agile tools?

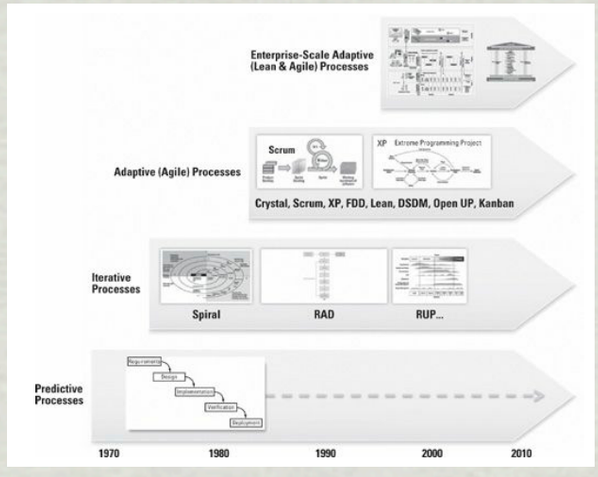


AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Extreme Programming (XP)

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.



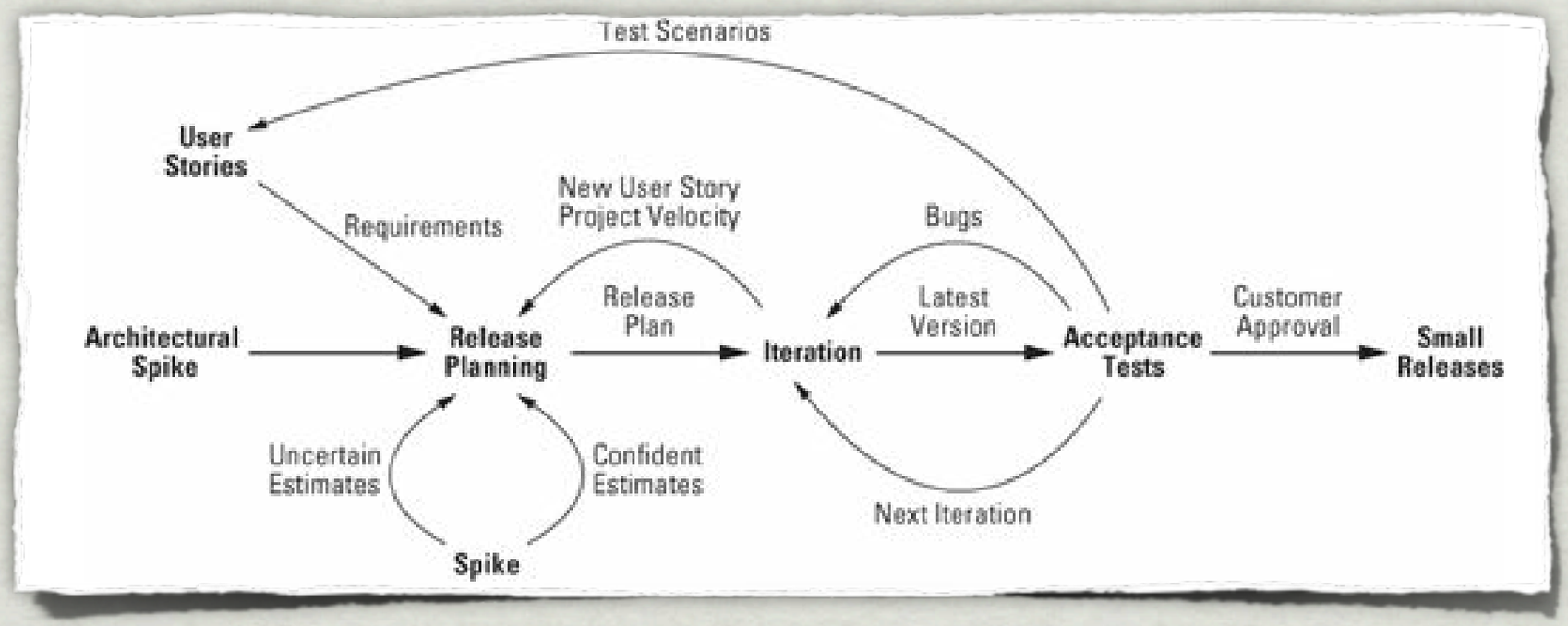


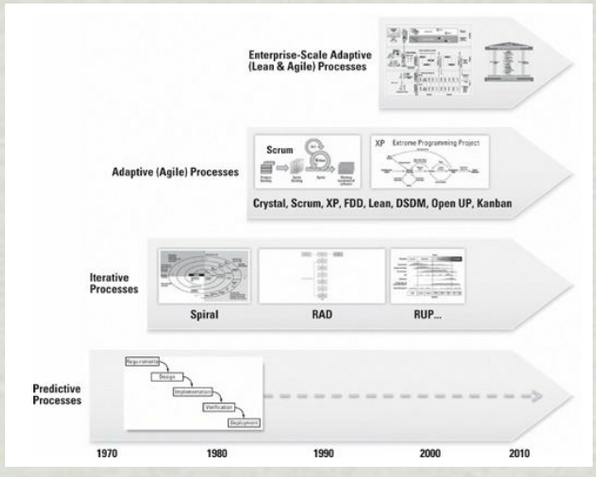
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Extreme Programming (XP)

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- A team of five to ten programmers work at one location with customer representation on-site.



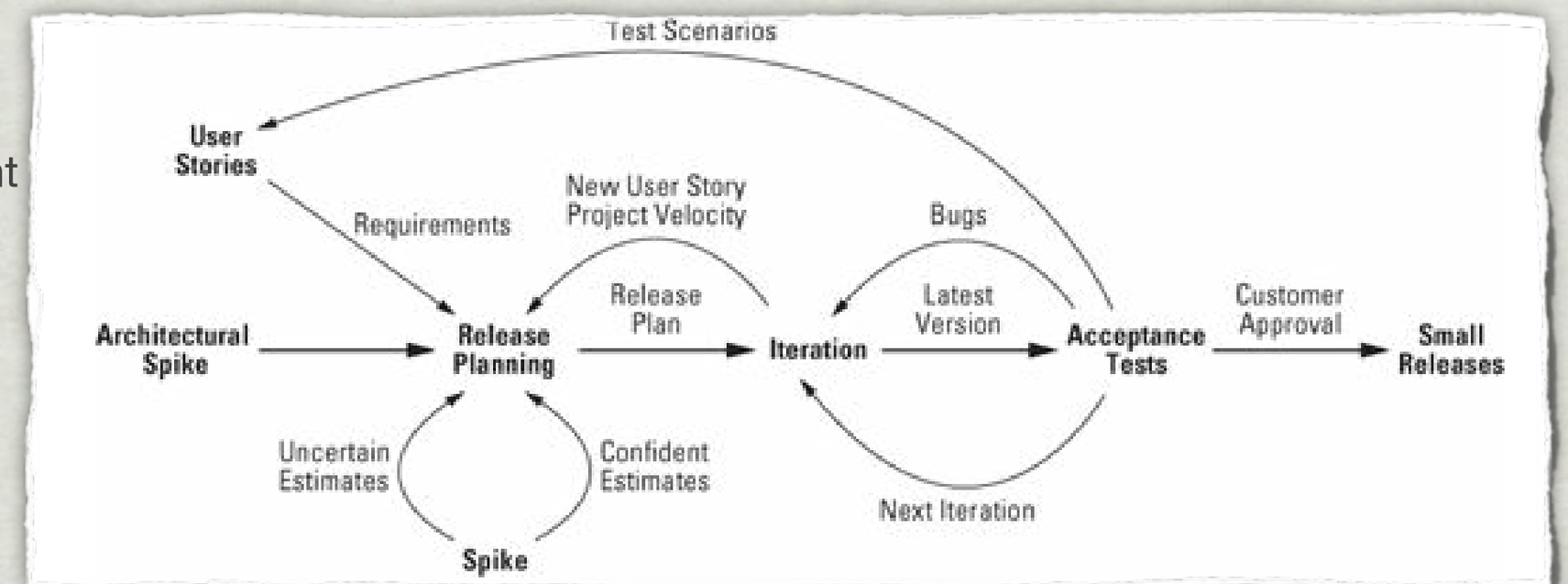


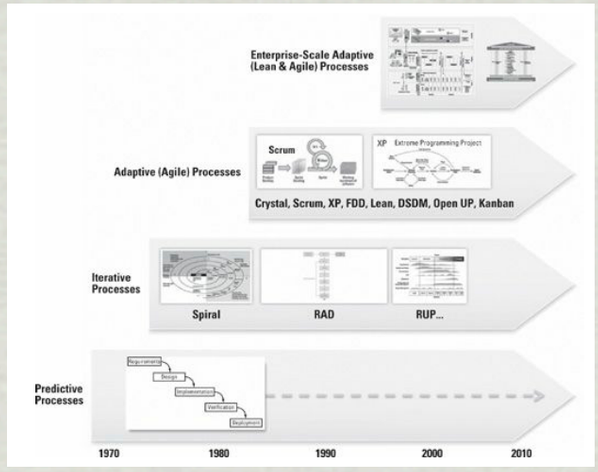
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Extreme Programming (XP)

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- A team of five to ten programmers work at one location with customer representation on-site.
- Development occurs in frequent builds or iterations, which may or may not be releasable, and delivers incremental functionality.



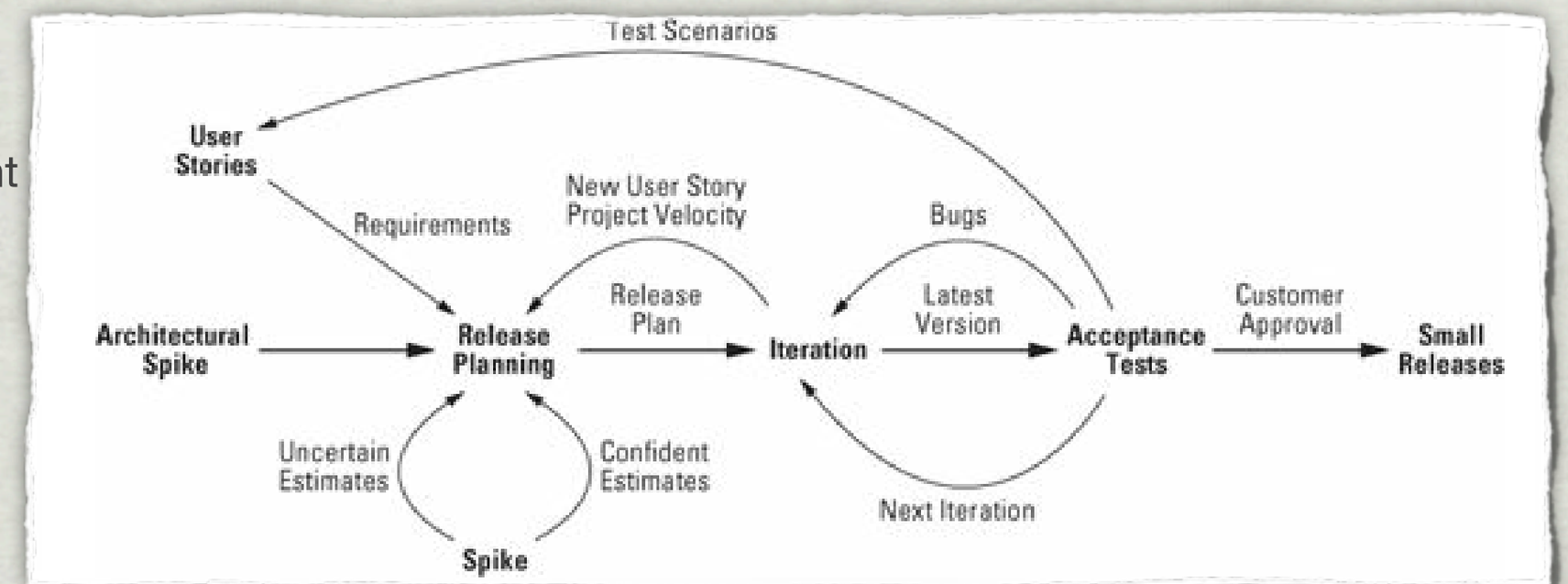


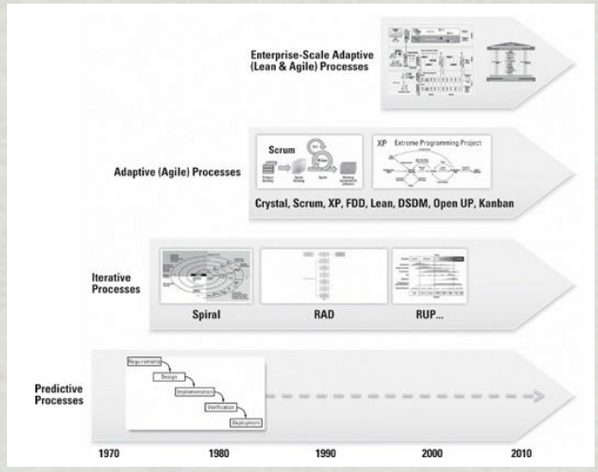
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Extreme Programming (XP)

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- A team of five to ten programmers work at one location with customer representation on-site.
- Development occurs in frequent builds or iterations, which may or may not be releasable, and delivers incremental functionality.
- Requirements are specified as user stories, each a chunk of new functionality the user requires.



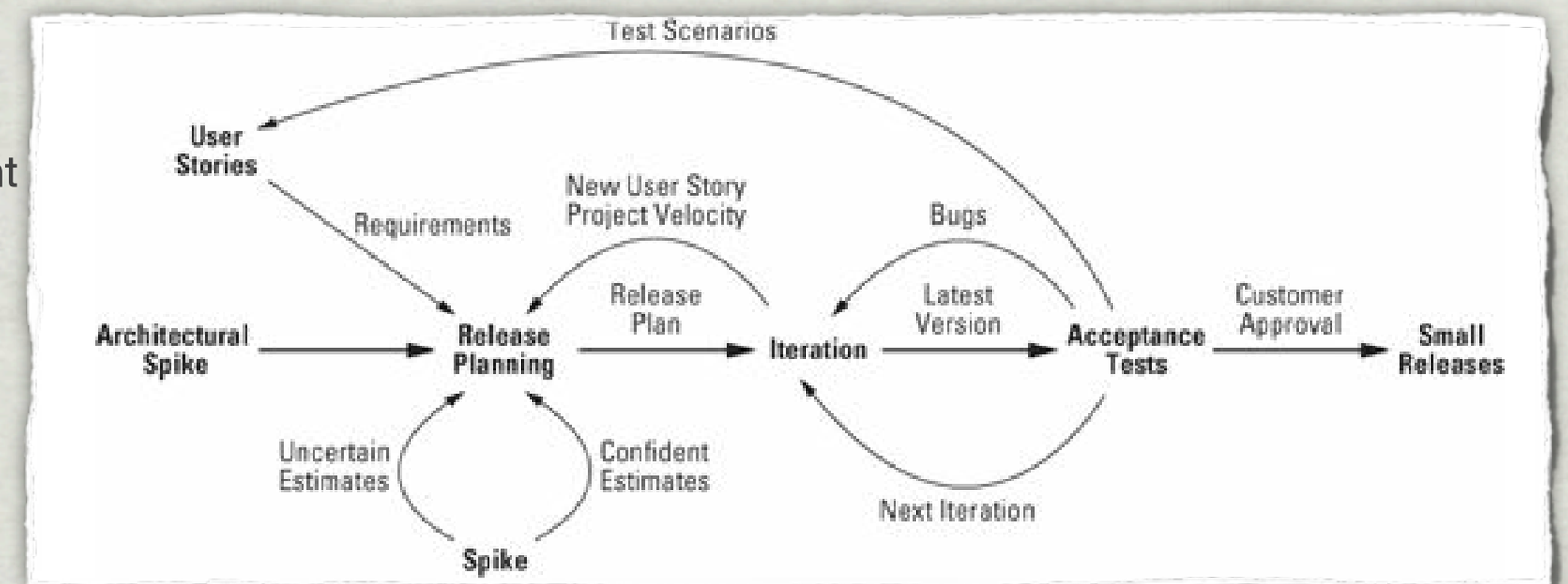


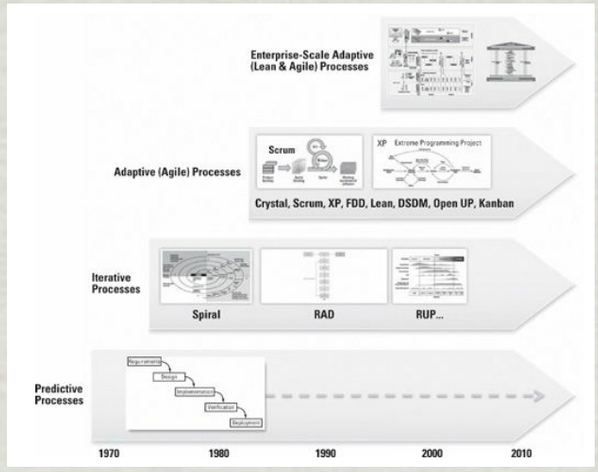
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Extreme Programming (XP)

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- A team of five to ten programmers work at one location with customer representation on-site.
- Development occurs in frequent builds or iterations, which may or may not be releasable, and delivers incremental functionality.
- Requirements are specified as user stories, each a chunk of new functionality the user requires.
- Programmers work in pairs, follow strict coding standards, and do their own unit testing. Customers participate in acceptance testing.



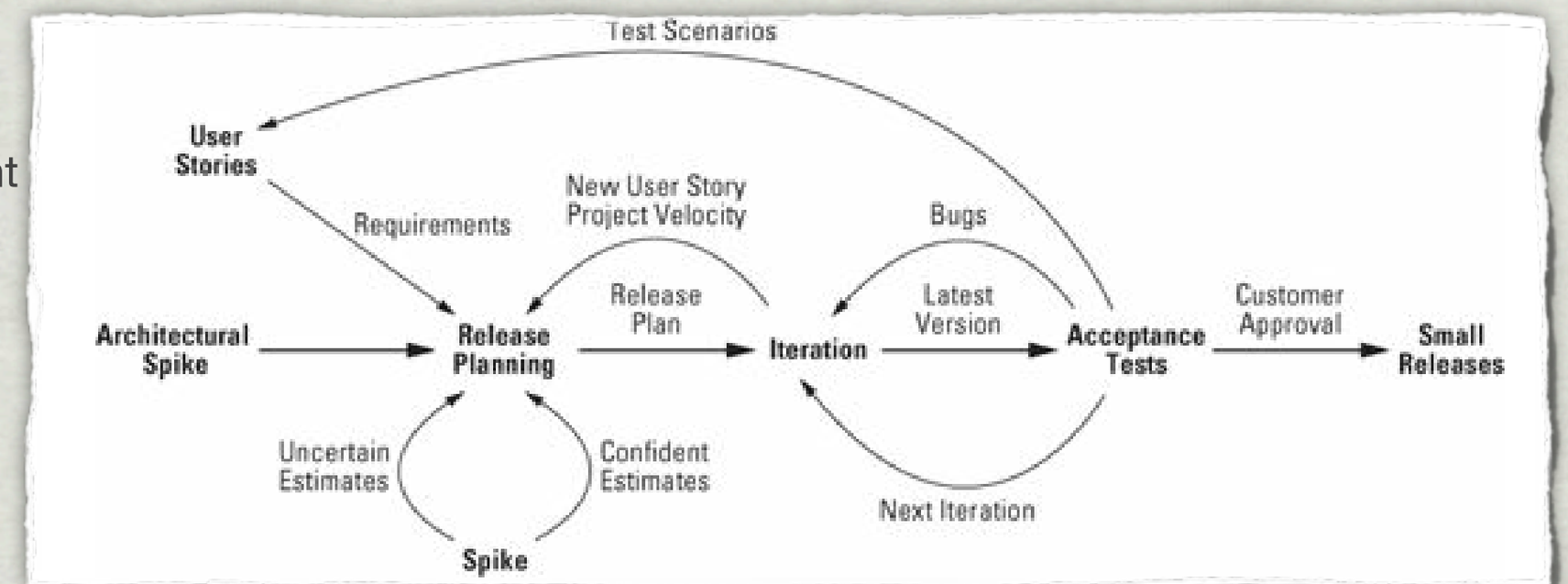


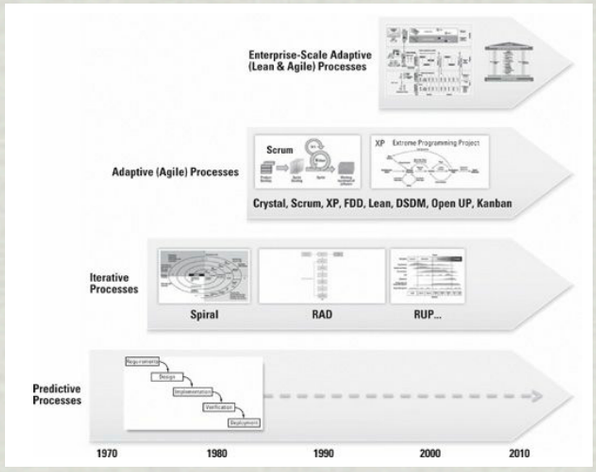
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Extreme Programming (XP)

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- A team of five to ten programmers work at one location with customer representation on-site.
- Development occurs in frequent builds or iterations, which may or may not be releasable, and delivers incremental functionality.
- Requirements are specified as user stories, each a chunk of new functionality the user requires.
- Programmers work in pairs, follow strict coding standards, and do their own unit testing. Customers participate in acceptance testing.
- Requirements, architecture, and design emerge over the course of the project.



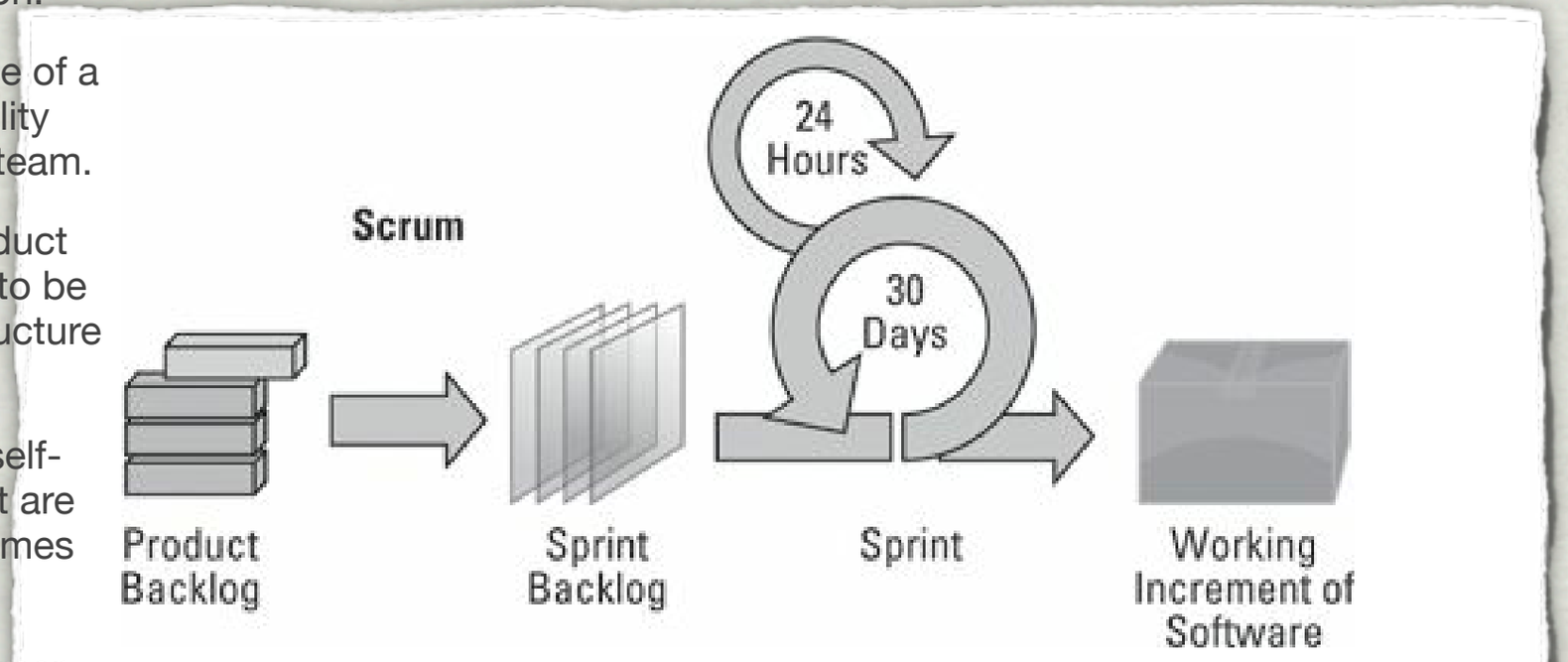


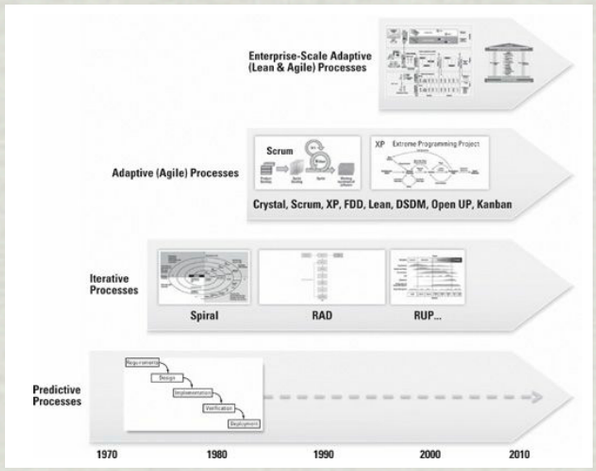
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Scrum

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

- Work is done in “sprints,” which are timeboxed iterations of a fixed 30 days or fewer duration.
- Work within a sprint is fixed. Once the scope of a sprint is committed, no additional functionality can be added, except by the development team.
- All work to be done is characterized as product backlog, which includes new requirements to be delivered, the defect workload, and infrastructure and design activities.
- A Scrum Master mentors the empowered, self-organizing, and self-accountable teams that are responsible for delivery of successful outcomes at each sprint.
- A product owner plays the role of the customer proxy.
- A daily stand-up meeting is a primary communication method.
- A heavy focus is placed on timeboxing. Sprints, stand-up meetings, release review meetings, and the like are all completed in prescribed times.
- Typical Scrum guidance calls for fixed 30-day sprints, with approximately 3 sprints per release, thus supporting incremental market releases on a 90-day time frame.





AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

- (with agile) instead of investing months in building detailed software requirements specifications...teams focus on delivering early, value-added stories into an integrated baseline.

Early delivery serves to test the requirements and architectural assumptions, and it drives risk out by proving or disproving assumptions about integration of features and components.

- No longer do management and the user community wait breathlessly for months, hoping the team is building the right thing. At worst, the next checkpoint is only a week or so away, and...users may be able to deploy even the earliest iterations in their own working environment.

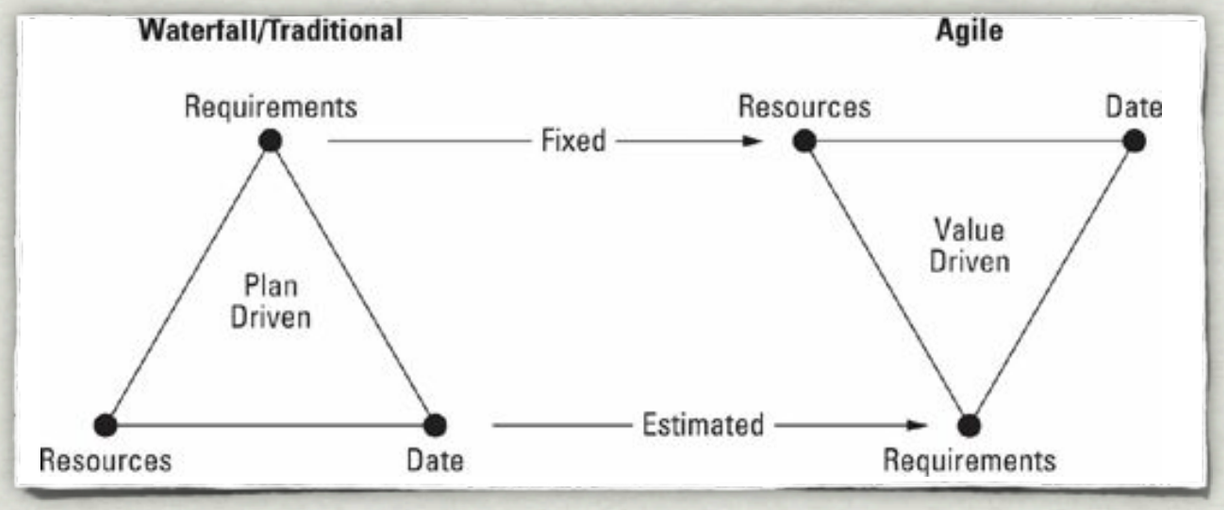
--Leffingwell, D, (2007) *Scaling Software Agility*

Agile Requirements is Fundamentally Different

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

Agile Requirements is Fundamentally Different

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.



Agile Requirements is Fundamentally Different

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

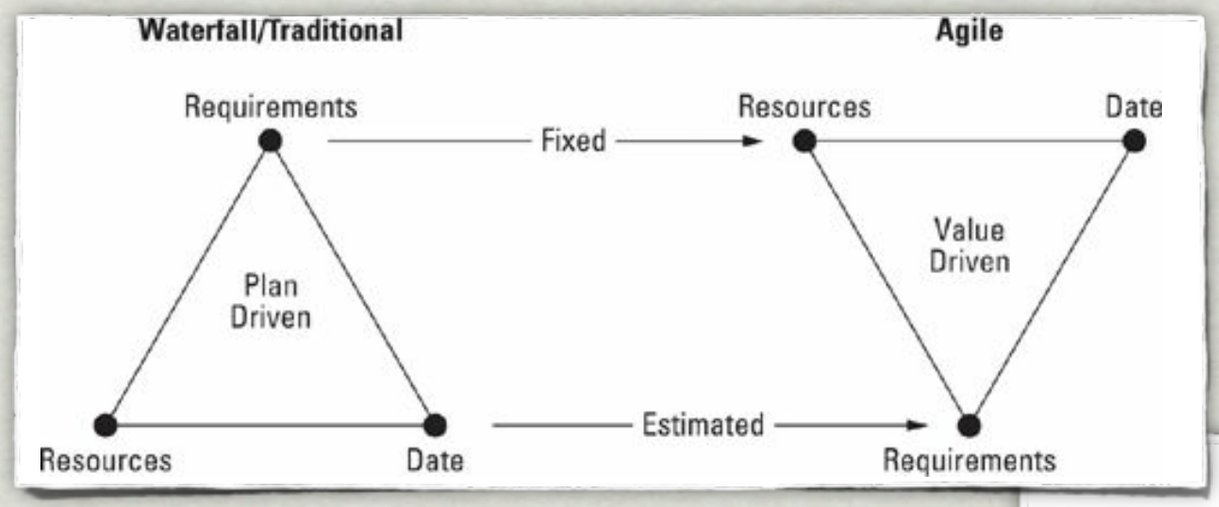


Figure 1-9. Value delivery and ROI in waterfall versus agile

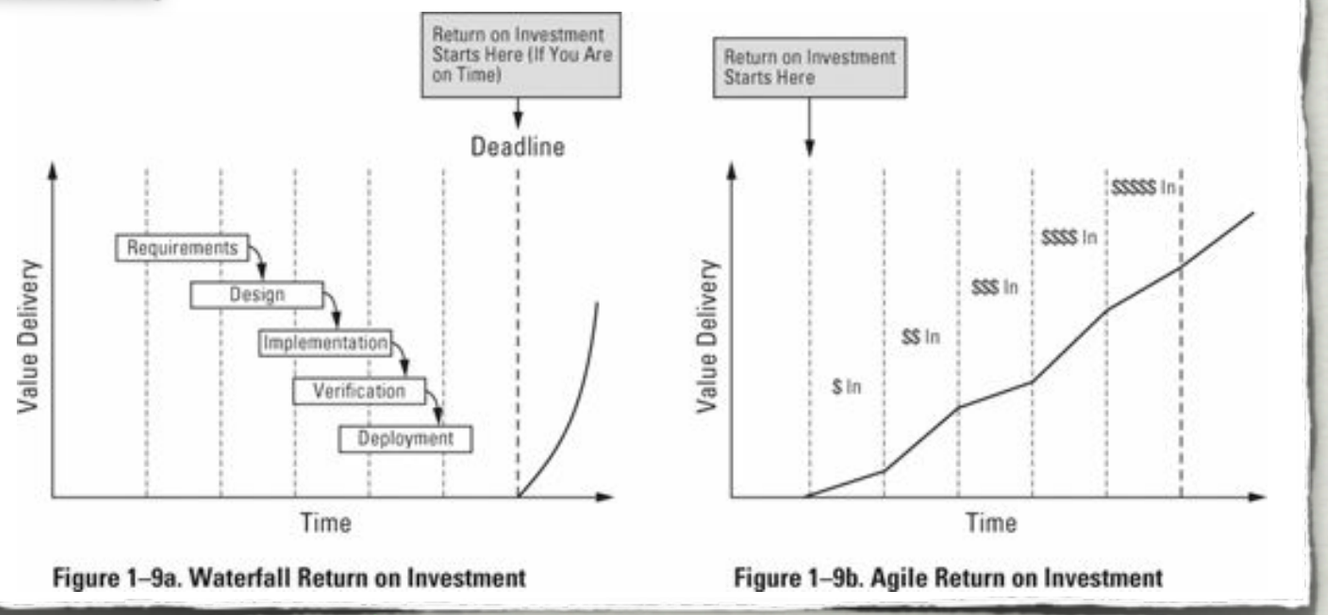


Figure 1–9a. Waterfall Return on Investment

Figure 1–9b. Agile Return on Investment

Agile Requirements is Fundamentally Different

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

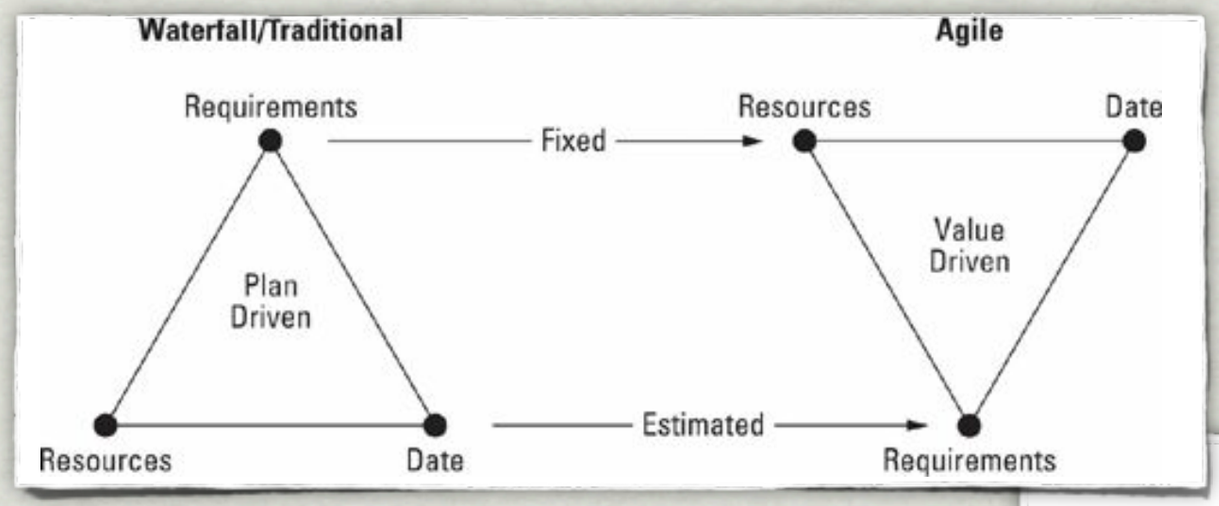


Figure 1-9. Value delivery and ROI in waterfall versus agile

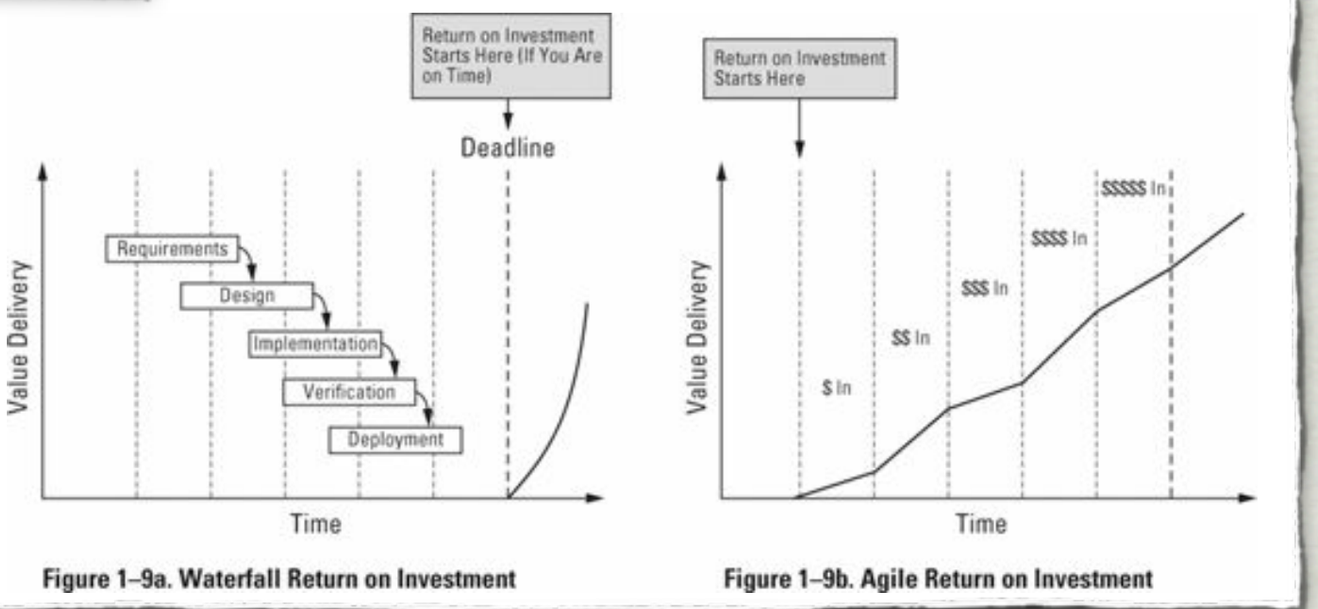
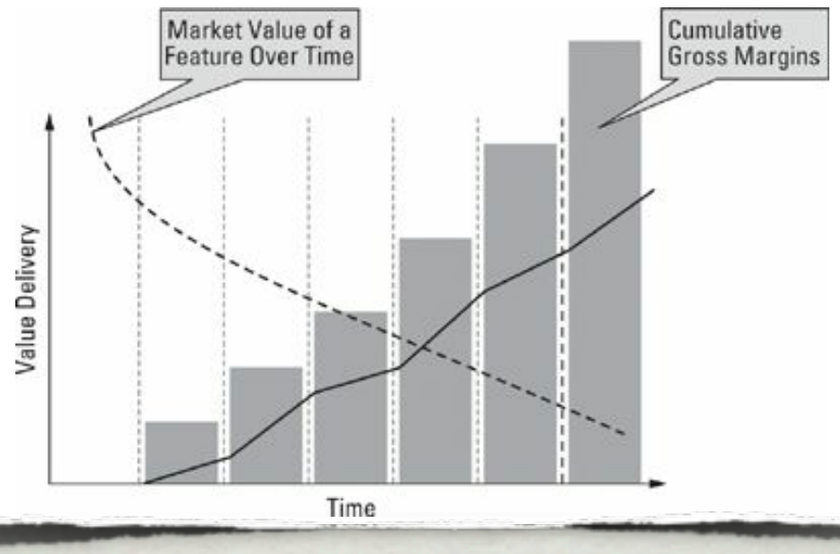


Figure 1–9a. Waterfall Return on Investment

Figure 1–9b. Agile Return on Investment

Figure 1-10. Agile ROI, taking into account differential feature value over time

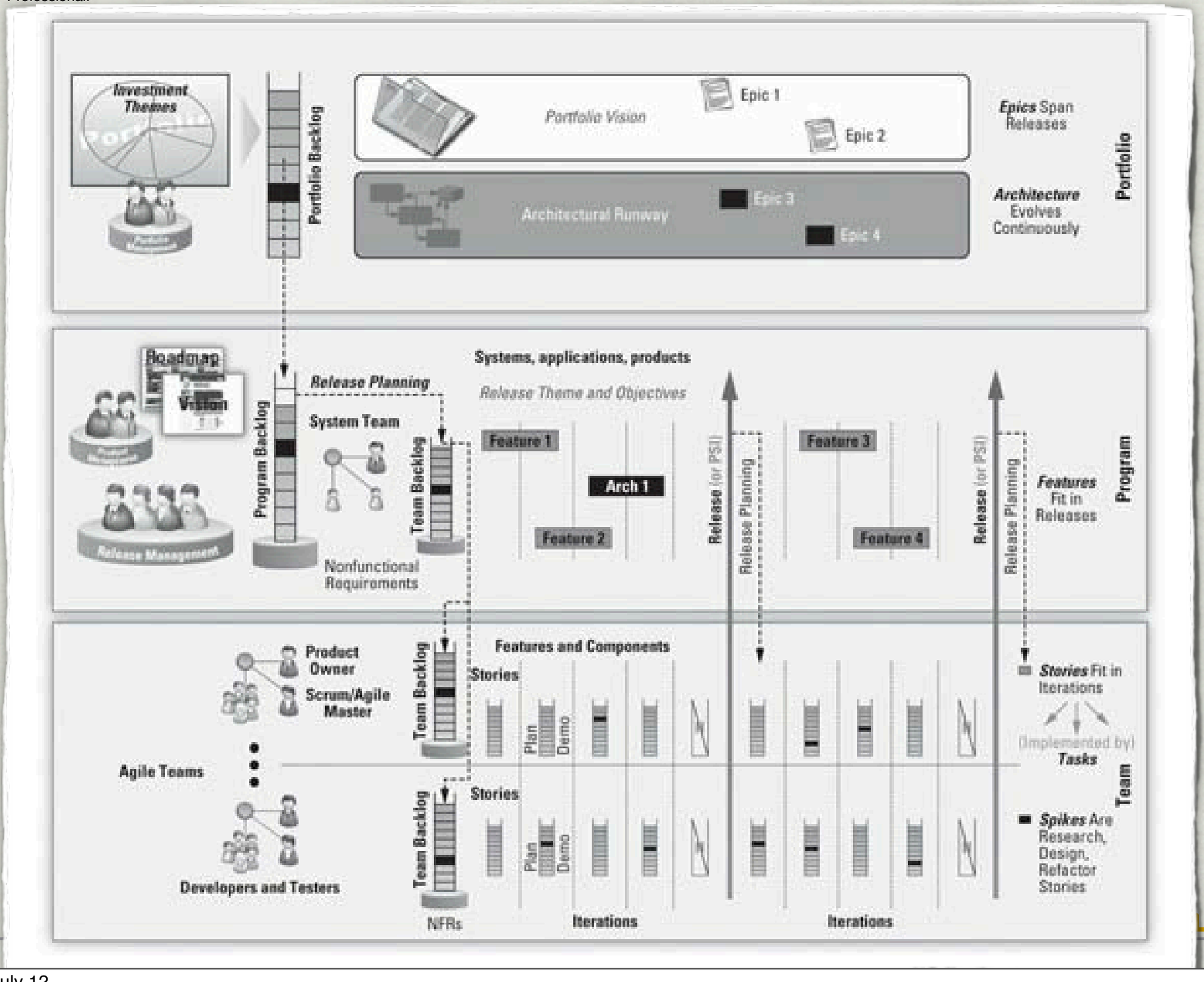


Agile for the Enterprise

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.

Agile for the Enterprise

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)* (1st ed. p. 560). Addison-Wesley Professional.



Your aim is maximise throughput of objects being passed.

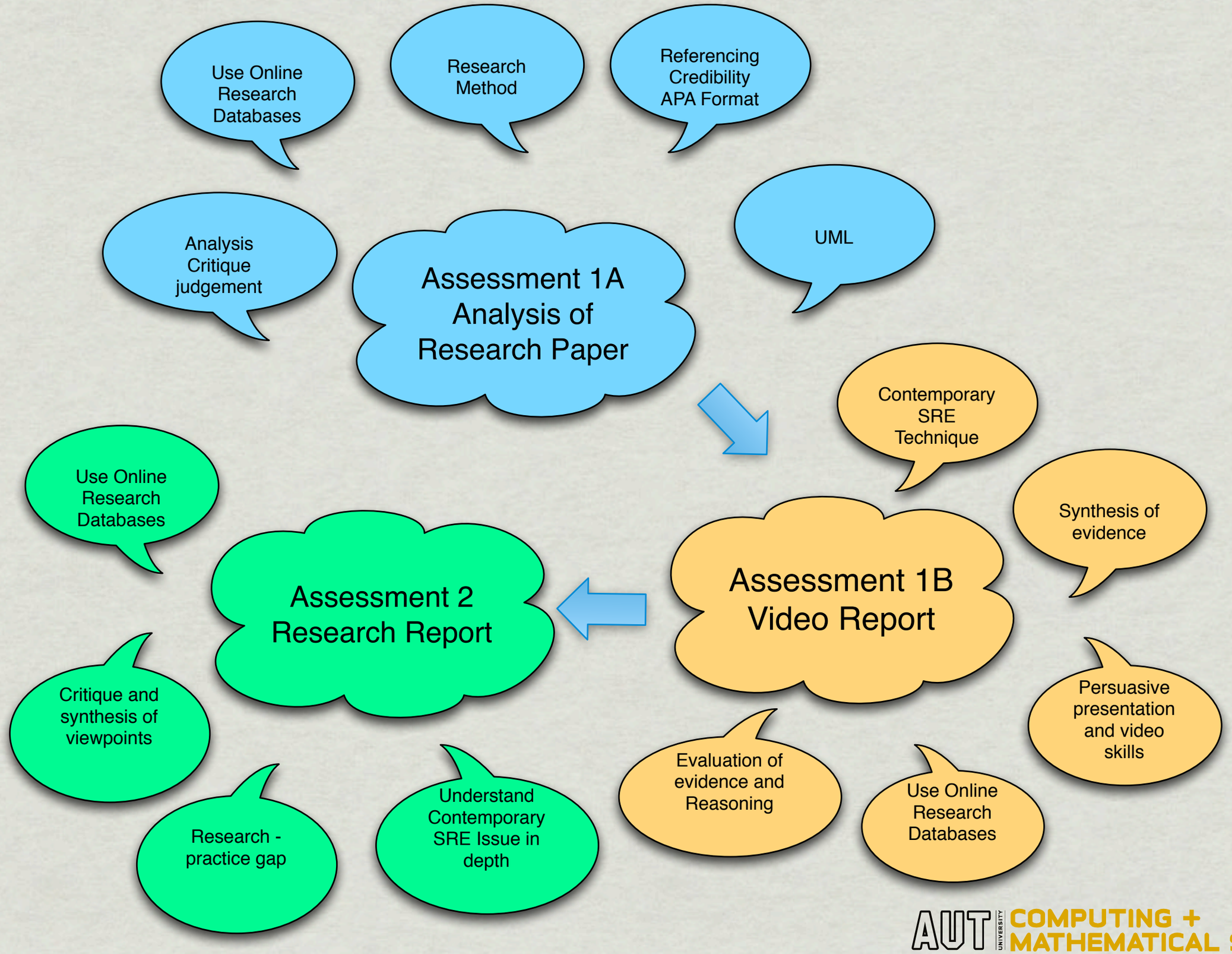
- **Form teams of 10-14**
- **Decide on a Team Name**
- **Plan process for estimation and recording on board**
- **Plan process for passing objects around legally**

Constraints

- All team members must be at least one arm's length from every other team member
- You **cannot** throw a ball to the person next to you
- You must say the name of the person you are throwing to
- The unit of throughput is the number of balls that have been through a complete cycle up (a part cycle is counted as zero throughput for a ball)
- You will have 1 minute to plan **and estimate** your throughput before each iteration. **During this time you must record your throughput for that iteration and the estimate for the next iteration on the board**.
- At the end of each iteration count the number of cycles each ball has done – the total is your throughput for that iteration.
- You will be allocated some resources randomly
- Priorities may change
- Balloons must be recycled before any other balls

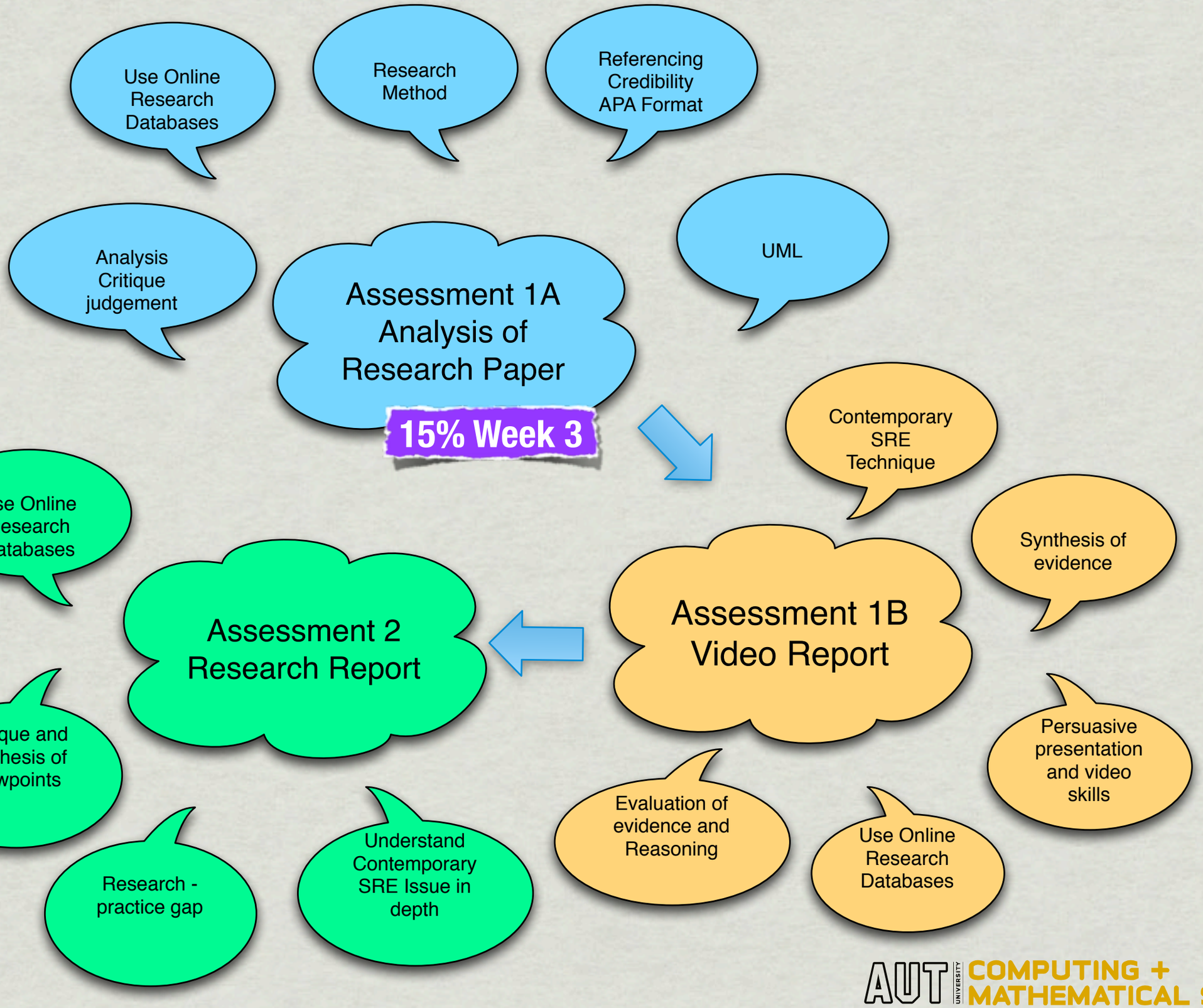
- How was the 1st estimate arrived at? What was the reasoning/model?
- What if you were forced to keep that estimate all the way through subsequent iterations – plans should change as you learn and gain experience!
- How accurate were estimates? How could you have done a more accurate estimate? Would it have been different if all tennis balls? (estimate model may include nbr people, nbr balls, how easy balls to handle, time given-maybe do some quick experiments with different types of balls)
- Did accuracy change? Why? –nothing beats experience-
- What issues did you solve in later iterations? Could you have foreseen these and planned for them?
- Did strategy change? Why? When?
- What if each iteration were 30 minutes, two iterations…what would have changed? Planning in small bites, estimating in small bites, reviewing in small bites, changes in small bites-if wrong or mismanaged consequences are in small bites
- How did teams organise? Make decisions? Leaders emerge? Handle Conflict? What did you do about people who couldn't throw or catch well? Or didn't want to participate? (may need some up-skilling ?)
- What was the effect of changes – balloons – like product owner changing what they want or re-prioritising?
- How would you describe the first iteration? (chaos, learning, unstructured, getting experience). Were later iterations better-could you have got to the better bits quicker with more planning?

Guiding learning with the assessments



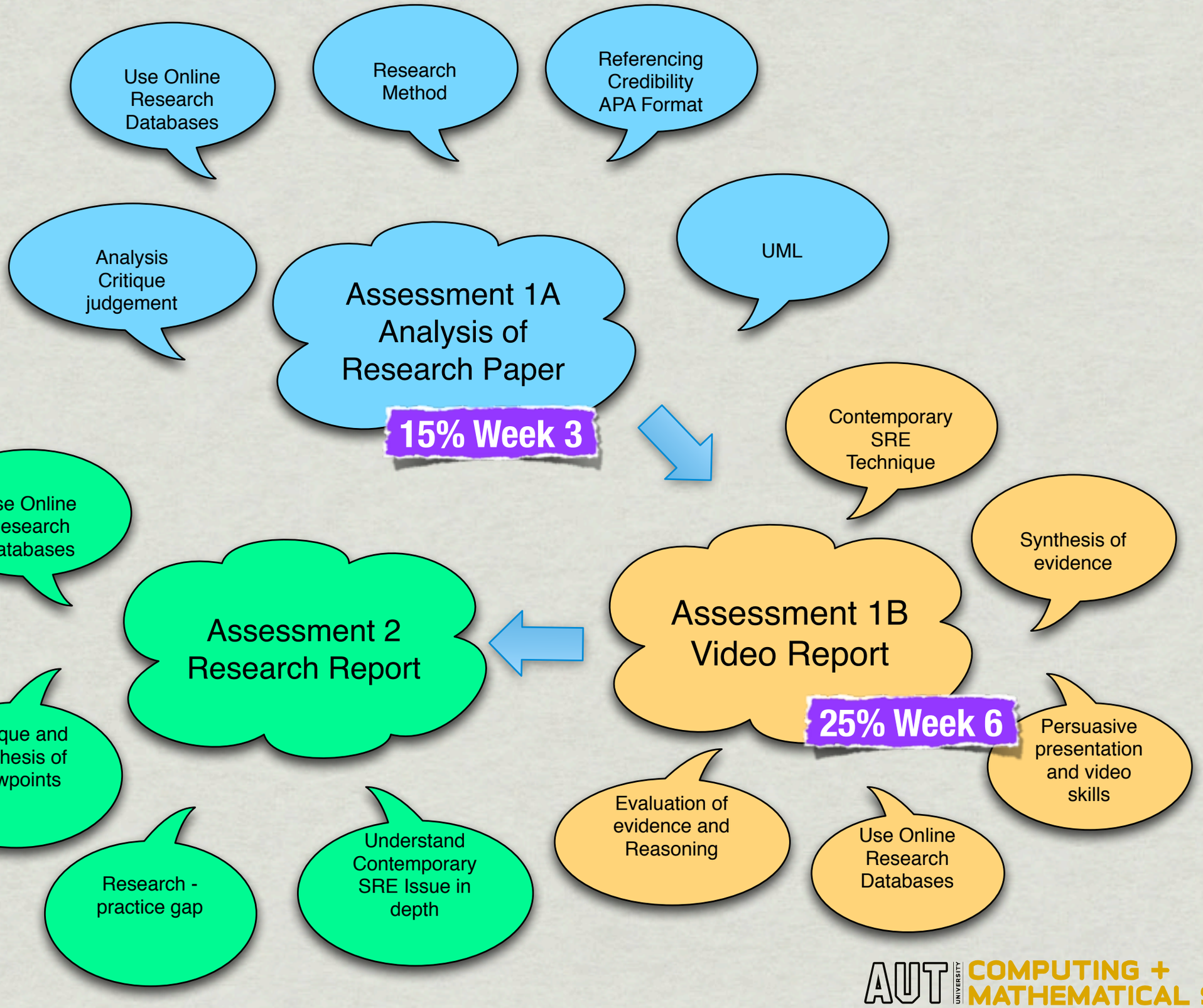
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Guiding learning with the assessments



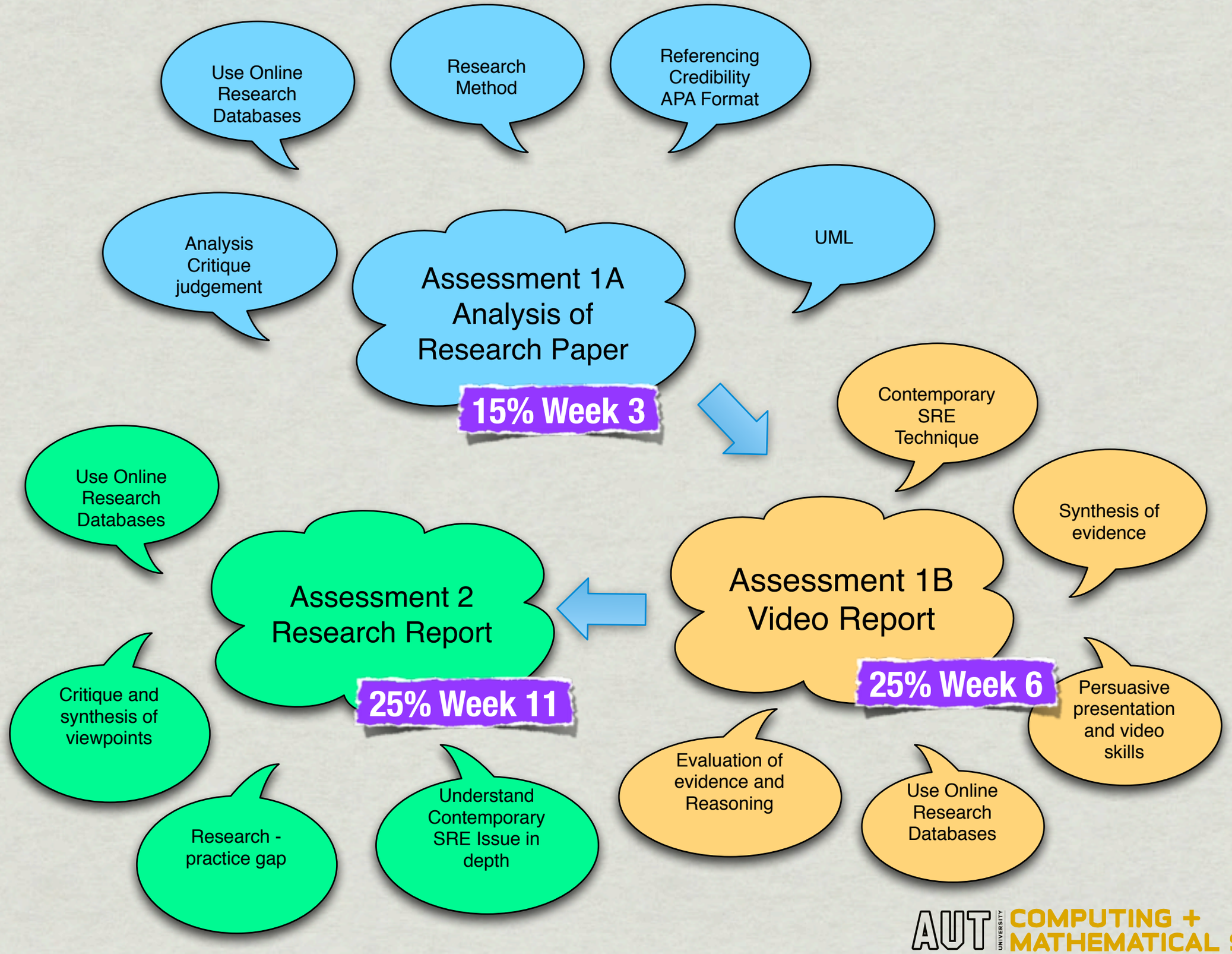
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Guiding learning with the assessments



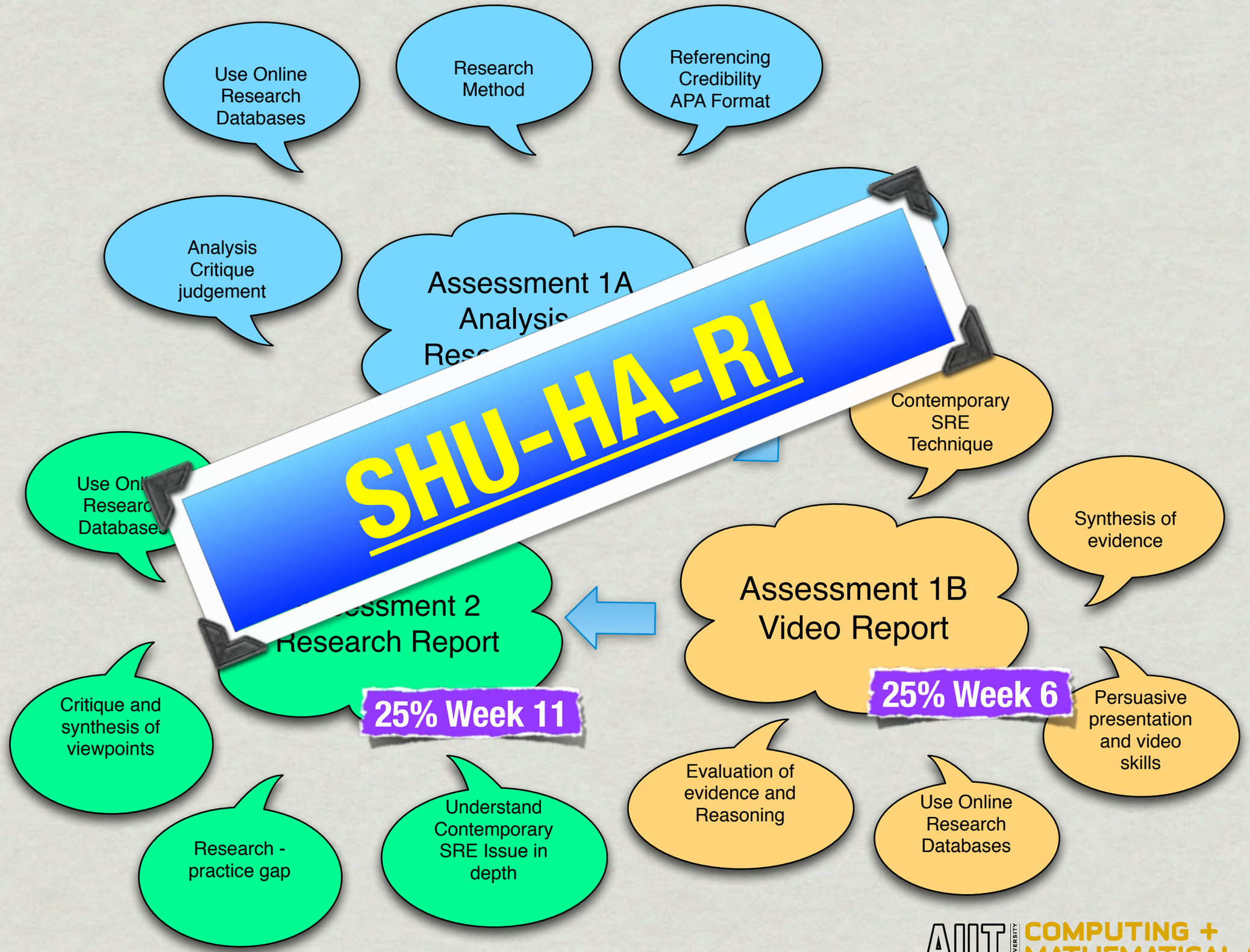
AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Guiding learning with the assessments



AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES

Guiding learning with the assessments



Course Overview

- Look at Handbook on AUTOnline

AUT UNIVERSITY COMPUTING + MATHEMATICAL SCIENCES