Agile Software Development

Lecturer: **Raman Ramsin**

Lecture 1

Agile Development: Basics

Software Development Methodology (SDM)

- A framework for applying software engineering practices with the specific aim of providing the necessary means for developing software-intensive systems

- Consisting of two main parts:
  - A set of modeling conventions comprising a *Modeling Language* (syntax and semantics)
  - A *Process*, which
    - provides guidance as to the order of the activities,
    - specifies what artifacts should be developed using the *Modeling Language*,
    - directs the tasks of individual developers and the team as a whole, and
    - offers criteria for monitoring and measuring a project's products and activities.

Object-Oriented Software Development Methodology (OOSDM)

- Specifically aimed at viewing, modeling and implementing the system as a collection of interacting objects
- First appeared in late 1980s
- Categorized as
  - *Seminal (First and Second Generations)*
  - *Integrated (Third Generation)*
  - *Agile*
- UML was the result of the 'war' among seminal methodologies
- Process has now replaced modeling language as the main contentious issue

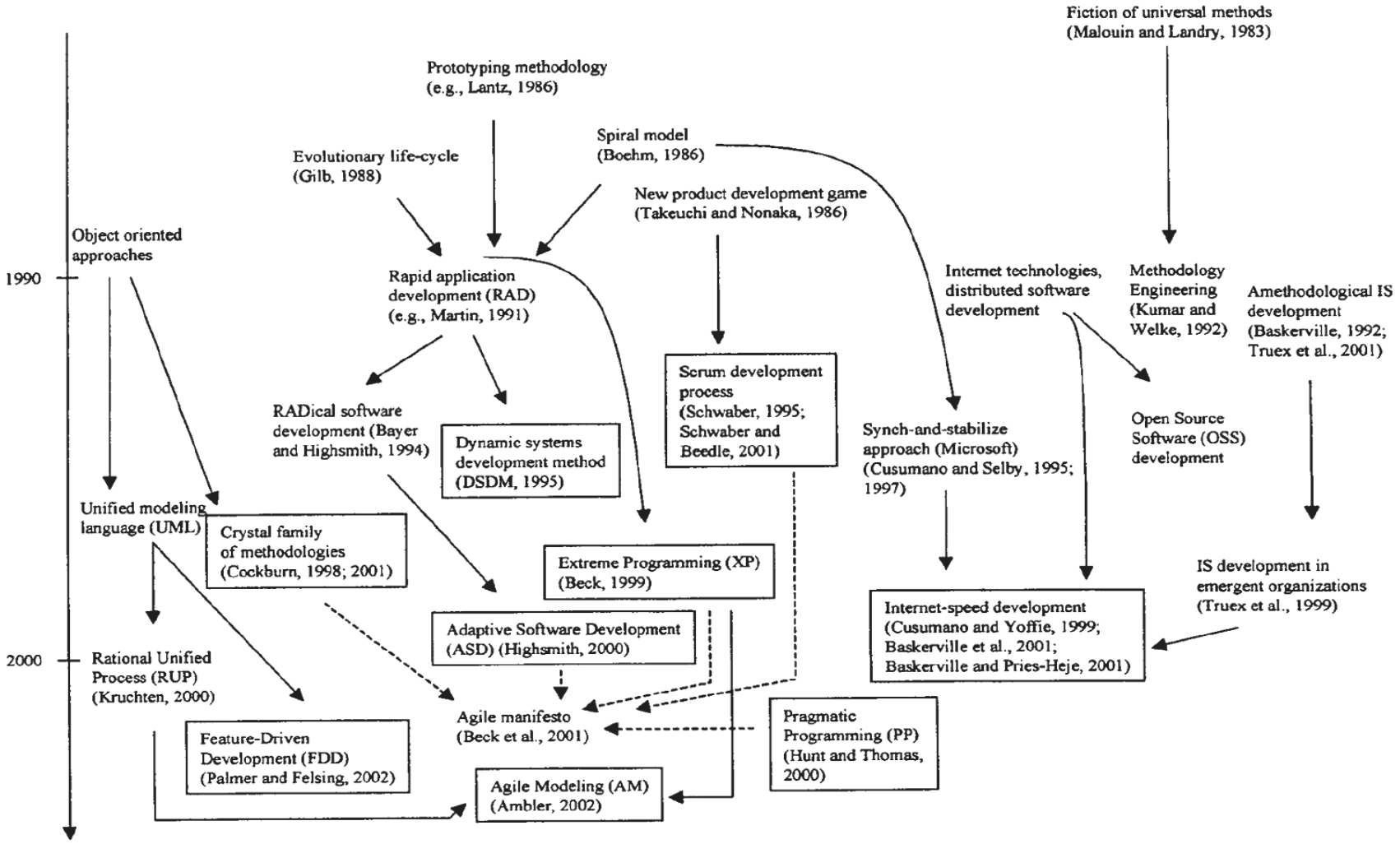
Agile Development: Brief History

- First appeared in 1995.
- The once-common perception that agile methodologies are nothing but controlled code-&-fix approaches, with little or no sign of a clear-cut process, is only true of a small – albeit influential – minority.
- Essentially based on practices of program design, coding and testing that are believed to enhance software development flexibility and productivity.
- Most agile methodologies incorporate explicit processes, although striving to keep them as lightweight as possible.

Major Agile Methodologies

- DSDM – Dynamic Systems Development Method (1994..2014)
- Scrum (1995..2020)
- XP – Extreme Programming (1996, 1999, 2004, 2013)
- ASD – Adaptive Software Development (1997)
- Crystal Family: Orange, Orange Web, Clear (1998, 2001, 2004)
- FDD – Feature-Driven Development (1999, 2002)
- AUP – Agile Unified Process (2005)
- DAD – Disciplined Agile Delivery (2012, 2020)

Agile Methodologies: Evolution Map



[Abrahamsson et al. 2003]

Agile Methodologies: Agile Manifesto

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

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

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

Agile Methodologies: Principles

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.

Agile Methodologies: Principles (Contd.)

- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

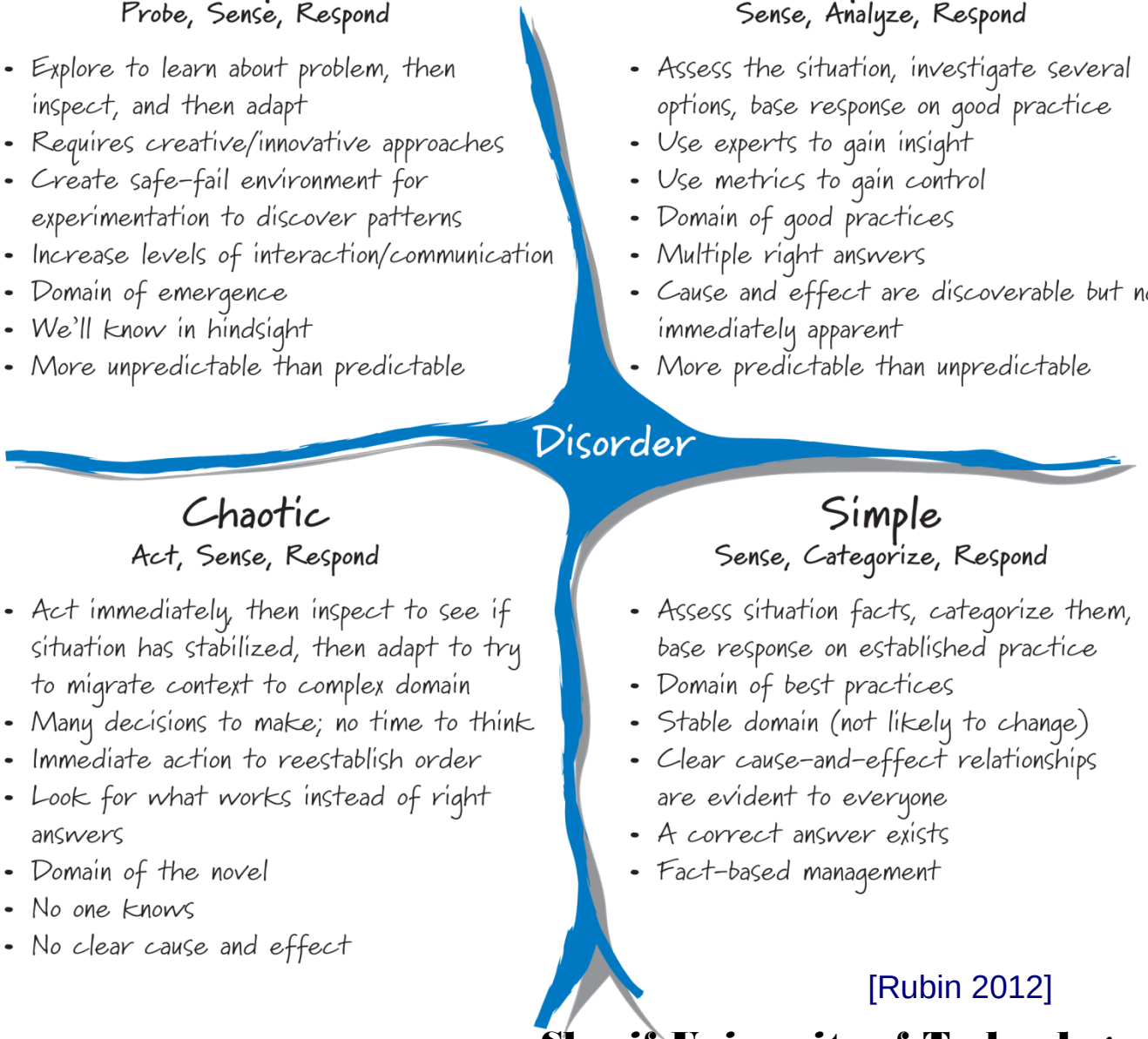
Agile Methodologies: Principles (Contd.)

- Continuous attention to technical excellence and good design enhances agility.
- Simplicity—the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Applicability of Agile Development

- Agile development is not the proper solution in all problem situations.
- We will discuss agile development's applicability based on the categories of problem situations proposed by the **Cynefin Framework**.
  - Cynefin, pronounced 'ku-nev-in', is a Welsh word for **habitat**:
    - It signifies the factors in our environment/experience that influence us in incomprehensible ways.
  - The Cynefin Framework is a sense-making framework that helps us understand the situation in which we have to operate, and decide on a situation-appropriate approach.
    - Defines and compares the characteristics of five different domains: **Simple (Obvious)**, **Complicated**, **Chaotic**, **Complex**, and **Disorder**.

Cynefin Framework



[Rubin 2012]

Cynefin: Complex Domain

- In complex problems, things are more unpredictable than they are predictable.
- If there is a right answer, we will know it only with hindsight; this is the domain of emergence.
- We need to **probe** to learn about the problem, then **sense** and **respond** based on our learning.
- Working in complex domains requires creative and innovative approaches. Routine solutions simply don't apply.
- We need to create a safe-fail environment for experimentation so that we can discover important information.
- In this environment high levels of interaction and communication are essential.
- Innovative new-product development falls into this category, as does enhancing existing products with innovative new features.
- Agile methodologies are particularly well suited for operating in a complex domain.

Cynefin: Complicated Domain

- In complicated problems, things are more predictable than unpredictable.
- Complicated problems are the domain of good practices dominated by experts.
- There might be multiple right answers, but expert diagnosis is required to figure them out.
- The approach is to **sense** what's coming in, **analyze** using expert knowledge, and **respond** by deciding on and applying good practices.
- Suitable methodologies:
  - Lighter agile methodologies such as Scrum can work in a complicated domain (e.g., software maintenance), but they might not be the best solutions.
  - Heavier agile methodologies such as DSDM and DAD are better suited because they are analysis and design processes based on expert knowledge and good practices.
  - As complexity increases, model-based heavyweight methodologies prevail.

Cynefin: Simple (Obvious) Domain

- In simple (obvious) problems, the domain is stable and everyone can see cause and effect.
- Often the right answer is obvious and undisputed.
- This is the domain of legitimate best practices; there are known solutions.
- Once we **assess** the facts of our situation and **categorize** them, we can **respond** by determining and applying the proper predefined solution.
- Agile methodologies can be used in a simple (obvious) domain, but they may not be the most efficient tools.
  - Using a process with a well-defined, repeatable set of steps that are known to solve the problem would be a better fit.
  - For example, if we want to reproduce the same product over and over again, a well-defined assembly-line process would be a better fit than Scrum.

Cynefin: Chaotic Domain

- In chaotic problems, there is a crisis and we need to act immediately to prevent further harm and reestablish at least some order.
- There is no clear cause and effect.
- There are many decisions to make, and no time to think.
- No one knows the answers.
- We have to look for what works instead of the right answers.
- We need to **act** immediately, then **inspect** to see if the situation has stabilized, and then **adapt** to migrate the context to the complex domain.
- Agile methodologies are not the best solutions in a chaotic domain.
  - In such domains, we are not interested in prioritizing a backlog of work and determining what work to perform in the next iteration. We need to *act* fast.

Cynefin: Disorder

- You are in the disorder domain when you don't know which of the other domains you are in.
  - This is a dangerous place to be because you don't know how to make sense of your situation.
  - In such cases, people tend to interpret and act according to their personal preferences for action; this is often a recipe for disaster.
- When in the disorder domain, the way out is to break down the situation into constituent parts and assign each to one of the other four domains.
- You are not trying to apply methodologies in the disorder domain; you are trying to get out of this domain.

Interrupt-Driven Contexts

- Plan-based agile methodologies (such as Scrum) are not well suited to **highly interrupt-driven** (or request-driven) work.
  - Interrupts may disrupt your plans and constantly change your priorities, prohibiting reliable planning of iterations of a week or more.
  - For example, during product support, the plan (or backlog) is populated continuously as you receive support requests.
- In interrupt-driven environments you would be better off considering an alternative agile approach called **Kanban**.
  - Kanban is not a stand-alone process solution, but instead an approach that is overlaid on an existing process.

Cynefin and Software Development

- The many facets of software development and support will not fit nicely into just one Cynefin domain.
- Most software development work falls in the domains of complicated or complex.
- However, software development is a rich endeavor, with aspects that overlap and activities that fall into all of the different domains.
  - It should be noted that the spectrum of work can range from innovative new-product development to maintenance/support.

References

- Ramsin, R., Paige, R.F., "Process-centered review of object-oriented software development methodologies." *ACM Computing Surveys*, Vol. 40, No. 1 (February), Article 3, pp. 1–89, 2008.
- Abrahamsson, P., Warsta, J., Siponen, M.T., Ronkainen, J., "New directions on agile methods: A comparative analysis." In *Proceedings of the International Conference on Software Engineering (ACM/ICSE 2003)*, pp 244–254, 2003.
- Beck, K., et al., "Manifesto for Agile Software Development." 2001, Available online at: http://agilemanifesto.org (Last visited: 14 September 2024).
- Snowden, D.J., Boone, M.E., "A Leader's Framework for Decision Making." *Harvard Business Review*, November 2007.