- ΠΡΟΧΩΡΗΜΕΝΑ ΘΕΜΑΤΑ ΜΗΧΑΝΙΚΗΣ ΛΟΓΙΣΜΙΚΟΥ
- ΠΡΟΣΧΕΔΙΑΣΜΕΝΟΣ & ΕΥΕΛΙΚΤΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

- ADVANCED SOFTWARE ENGINEERING
- PLAN-DRIVEN & AGILE PROGRAMMING

Prerequisite & Desirable Courses

- *Procedural Programming (C),*
- *Principles of Software Engineering*
- *OOD*
- *Data Bases,*
- *HCI*

AGILE SOFTWARE ENGINEERING

Artificial Intelligence and Systems Engineering Lab
Department of Informatics Engineering and Electical Engineering, TEI of Crete

NiLE Laboratory

Dr. Vidakis Nikolas

What is Agile?

- The ability to create and respond to change in order to succeed in an uncertain and turbulent environment.
- Agile --readiness for motion, nimbleness, activity, dexterity in motion

- Agility

  The ability to both create and respond to change in order to profit in a turbulent business environment
  - Companies need to determine the amount of agility they need to be competitive
- Chaordic
  - Exhibiting properties of both ***cha***os and ***ord***er
    - The blend of chaos and order inherent in the external environment and in people themselves, argues against the prevailing wisdom about predictability and planning
    - Things get done because people adapt, not because they slavishly follow processes

What is Agile Software Development?

- Agile Software Development is an umbrella term for a set of methods and practices based on the values and principles expressed in the Agile Manifesto.
- Solutions evolve through collaboration between self-organizing, cross-functional teams utilizing the appropriate practices for their context.
- **Agile software development** is a conceptual framework for software engineering that promotes development iterations throughout the life-cycle of the project.
- Software developed during one unit of time is referred to as an iteration, which may last from one to four weeks.
- Agile methods also emphasize working software as the primary measure of progress

Characteristics of Agile Software Development

- Light Weighted methodology
- Small to medium sized teams
- vague and/or changing requirements
- vague and/or changing techniques
- Simple design
- Minimal system into production

A Short History of Agile

- In the late 1990's, several methodologies began to gain increasing public attention, each having a different combination of old and new ideas.
- These methodologies emphasized close collaboration between the development team and business stakeholders; frequent delivery of business value, tight, self-organizing teams; and smart ways to craft, confirm, and deliver code.
- The term "Agile" was applied to this collection of methodologies in early 2001 when 17 software development practitioners gathered in Snowbird, Utah to discuss their shared ideas and various approaches to software development.
- This joint collection of values and principles was expressed in the Manifesto for Agile Software Development and the corresponding twelve principles.
- Agile Alliance was formed shortly after this gathering to encourage practitioners to further explore and share ideas and experiences.
- Agile Alliance continues to curate resources to help you adopt Agile practices and improve your ability to develop software with agility.

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.

The 12 Agile Principles

The following principles are those that differentiate agile processes from others.

12 Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Alliance AgileAlliance.org

Principle 1: Our Highest Priority is to Satisfy the Customer through Early and Continuous Delivery of Valuable Software

- Research has indicated that a number of practices have significant impact upon quality of final system:
- A strong correlation between quality and early delivery of a partially functioning system.
  - The less functional the initial delivery, the higher the quality of the final delivery.
- Another strong correlation exists between final quality and frequently deliveries of increasing functionality.
  - The more frequent the deliveries, the higher the final quality.
- Agile processes deliver early and often. Rudimentary system first followed by systems of increasing functionality every few weeks.
  - Customers my use these systems in production, or
  - May choose to review existing functionality and report on changes to be made.

Principle 2: Welcome changing Requirements, even late in Development. Agile Processes harness change for the Customer's Competitive Advantage.

- This is a statement of attitude. The participants in an agile process are not afraid of change.
  - View changes to the requirements as good things, because they mean that the team has learned more about what it will take to satisfy the market.
- An agile team works very hard to keep the structure of their software flexible, so that when requirements change, the impact to the system is minimal.
- Moreso, the principles of object oriented design help us to maintain this kind of flexibility.

Principle 3: Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale.

- We deliver working software.
  - Deliver early and often.
  - Be not content with delivering bundles of documents, or plans.
  - Don't count those as true deliverables.

- The goal of delivering software that satisfies the customer's needs.

Principle 4: Business People and Developers Must Work Together Daily throughout the Project.

- In order for a project to be agile, there must be significant and frequent interaction between the customers, developers, and stakeholders.
- An agile project is not like a fire-and-forget weapon.
- An agile project must be continuously guided.

Principle 5: Build Projects around Motivated Individuals. Give them the environment and support they need, and trust them to get the job done.

- An agile project is one in which people are considered the most important factor of success.
  - All other factors, process, environment, management, etc., are considered to be second order effects, and are subject to change if they are having an adverse effect upon the people.
- For example, if the office environment is an obstacle to the team, the office environment changes.
- If certain process steps are an obstacle to the team, the process steps change.

Principle 6: The Most Efficient and Effective Method of Conveying Information to and within a Development Team is face-to-face Communications.

- In an agile project, please *talk* to each other.
  - The primary mode of communication is conversation.
  - Documents may be created, but there is no attempt to capture all project information in writing.
- An agile project team **does not demand** written specs, written plans, or written designs.
  - They may create them if they perceive an immediate and significant need, but they are not the default.
  - The default is conversation.

Principle 7: Working Software is the Primary Measure of Progress

- Agile projects measure their progress by measuring the amount of software that is working.
- They don't measure their progress in terms of the phase that they are in, or by the volume of documentation that has been produced, or by the amount of infrastructure code they have created.
- They are 30% done when 30% of the necessary functionality is working.

Principle 8: Agile Processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

- An agile project is not run like a 50 yard dash; it is run like a marathon.
  - The team does not take off at full speed and try to maintain that speed for the duration.
  - Rather they run at a fast, but **sustainable**, pace.
- Running too fast leads to burnout, shortcuts, and debacle.
- Agile teams pace themselves.
  - They don't allow themselves to get too tired.
  - They don't borrow tomorrow's energy to get a bit more done today.
  - They work at a rate that allows them to maintain the highest quality standards for the duration of the project.

Principle 9: Continuous attention to technical excellence and good design enhances agility.

- High quality is the key to high speed.
  - The way to go fast is to keep the software as clean and robust as possible.
  - Thus, all agile team-members are committed to producing only the highest quality code they can.
  - They do not make messes and then tell themselves they'll clean it up when they have more time.
    - If they make a mess, they clean it up before they finish for the day.

Principle 10: Simplicity – the art of maximizing the amount of work not done – is essential.

- Agile teams do not try to build the grand system in the sky.
  - Rather they always take the simplest path that is consistent with their goals.
  - They don't anticipate tomorrow's problems and try to defend against them today.
  - Rather they do the simplest and highest quality work today, confident that it will be easy to change if and when tomorrows problems arise.

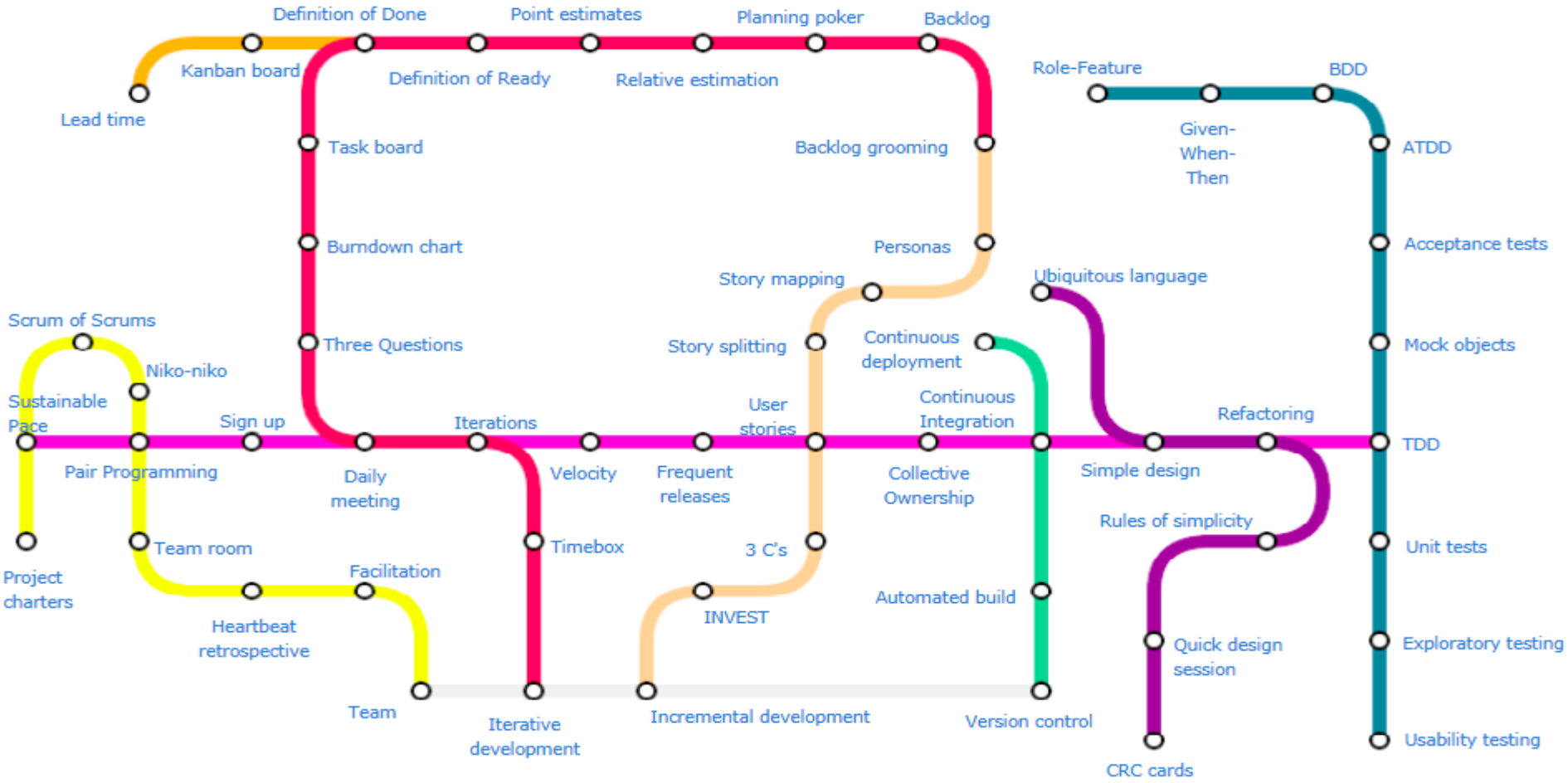
Principle 11: The Best Architectures, Requirements, and Designs emerge from Self-Organizing Teams

- An agile team is a self organizing team.
  - Responsibilities are not handed to individual team members from the outside. Responsibilities are communicated to the team as a whole, and the team determines the best way to fulfill them.
- Agile team members work together on all aspects of the project.
  - Each is allowed input into the whole.
  - No single team member is responsible for the architecture, or the requirements, or the tests, etc.
  - The team shares those responsibilities and each team member has influence over them.

Principle 12: At regular Intervals, the Team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

- An agile team continually adjusts its organization, rules, conventions, relationships, etc.
- An agile team knows that its environment is continuously changing, and knows that they must change with that environment to remain agile.

Subway Map to Agile Practices



Lines represent practices from the various Agile "tribes" or areas of concern:

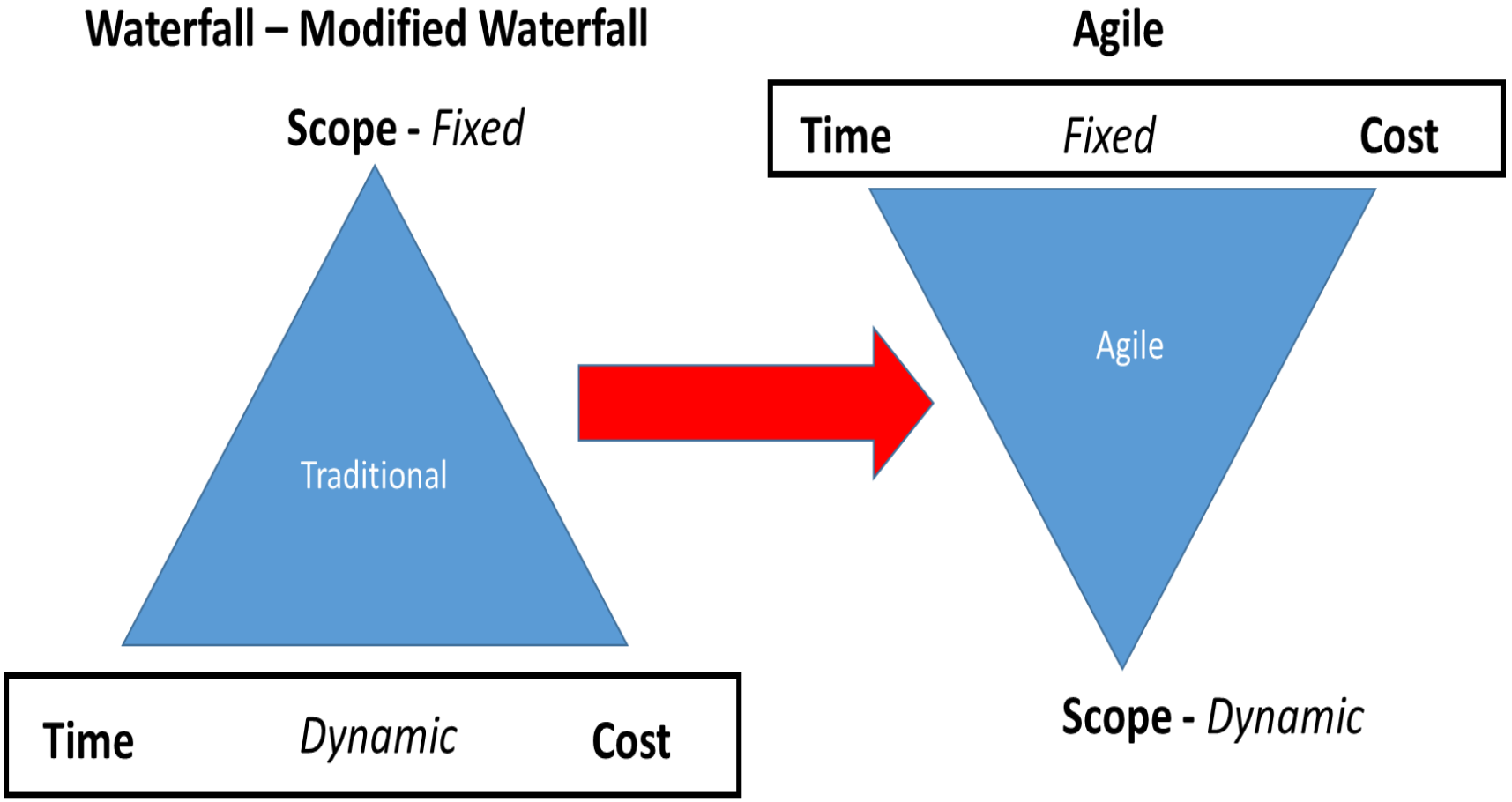
- Extreme Programming
- Teams
- Lean
- Scrum
- Product management
- Devops
- Design
- Testing
- Fundamentals

Agile Methods

Agile Methods

- Extreme Programming ("XP")
- Agile Unified Process
- Scrum
- Feature Driving Development (FDD)
- Lean Workflow
- Dynamic Systems Development Method (DSDM)
- Crystal Family of Methods

The Iron Triangle – Cost, Time and Scope Relationship



Extreme Programming ("XP")

- Most prominent Agile Software development method
- Prescribes a set of daily stakeholder practices
- "Extreme" levels of practicing leads to more responsive software.
- Changes are more realistic, natural, inescapable.

Agile Unified Process (AUP)

AUP is a simplified version of RUP

Phases of AUP

- Inception
- Elaboration
- Construction
- Transition

Disciplines of AUP

- Model
- Implementation
- Test
- Deployment
- Configuration Management
- Project Management
- Environment

SCRUM

- It is an Agile S/w development method for project management

Characteristics:

- Prioritized work is done.
- Completion of backlog items
- Progress is explained
- Agile Software Development

Useful Links

- https://www.agilealliance.org/
- http://www.agile.org/
- https://www.scrum.org/
- https://agilemanifesto.org/

REFERENCES

[1]. Abrahamsson P, Salo O and Ronkainen J. Agile software development methods (Review and analysis).

[2]. Scott W Ambler. Agile model driven development.

[3]. Cohen D, Lindvall M, Costa P. Agile software development.

[4]. http://en.wikipedia.org/wiki/Agile_Modeling.

[5]. http://en.wikipedia.org/wiki/Extreme_Programming.

[6]. http://en.wikipedia.org/wiki/Agile_Unified_process.

[7]. http://en.wikipedia.org/wiki/Scrum_28development29.

Useful Books