# Chapter 5 AGILE DEVELOPMENT

# Chapter 5 AGILE DEVELOPMENT

**1. The modern business environment that spawns computer-based systems and software products is fast-paced and ever-changing. Agile software engineering represents a reasonable alternative to conventional software engineering for certain classes of software and certain types of software projects. It has been demonstrated to deliver successful systems quickly.**

**2. The agile software engineering philosophy encourages customer satisfaction and early incremental delivery of software; small, highly motivated project teams; informal methods; minimal software engineering work products; and overall development simplicity.**

**3. Software engineers and other project stakeholders (managers, customers,end users) work together on an agile team—a team that is self-organizing and in control of its own destiny. An agile team fosters communication and collaboration among all who serve on it.**

# Chapter 5 AGILE DEVELOPMENT

**The basic framework activities of agile development—communication,planning, modeling, construction, and deployment(generic process, or standard process)—remain. But they morph into a minimal task set that pushes the project team toward design,construction and delivery (some would argue that this is done at the expense of problem analysis and solution design).**

**Both the customer and the software engineer have the same view—the only really important work product is an operational "software increment" that is delivered to the customer on the appropriate commitment date.**

# Chapter 5 AGILE DEVELOPMENT

**Agile development can provide important benefits, but it is not applicable to all projects, all products, all people, and all situations. It is also not antithetical to solid software engineering practice and can be applied as an overriding philosophy for all software work.**

## 5.1 WHAT IS AGILITY ?

**Agility has become today's buzzword when describing a modern software process. Everyone is agile. An agile team is a nimble team able to appropriately respond to changes. Change is what software development is very much about. Changes in the software being built, changes to the team members, changes because of new technology, changes of all kinds that may have an impact on the product they build or the project that creates the product.**

# Chapter 5 AGILE DEVELOPMENT

## 5.2 AGILITY AND THE COST OF CHANGE

**The conventional wisdom in software development (supported by decades of experience) is that the cost of change increases nonlinearly as a project progresses ( Figure 5.1 , solid black curve). It is relatively easy to accommodate a change when a software team is gathering requirements (early in a project). A usage scenario might have to be modified, a list of functions may be extended, or a written specification can be edited. The costs of doing this work are minimal, and the time required will not adversely affect the outcome of the project. But what if we fast-forward a number of months? The team is in the middle of validation testing (something that occurs relatively late in the project), and an important stakeholder is requesting a major functional change. The change requires a modification to the architectural design of the software, the design and construction**

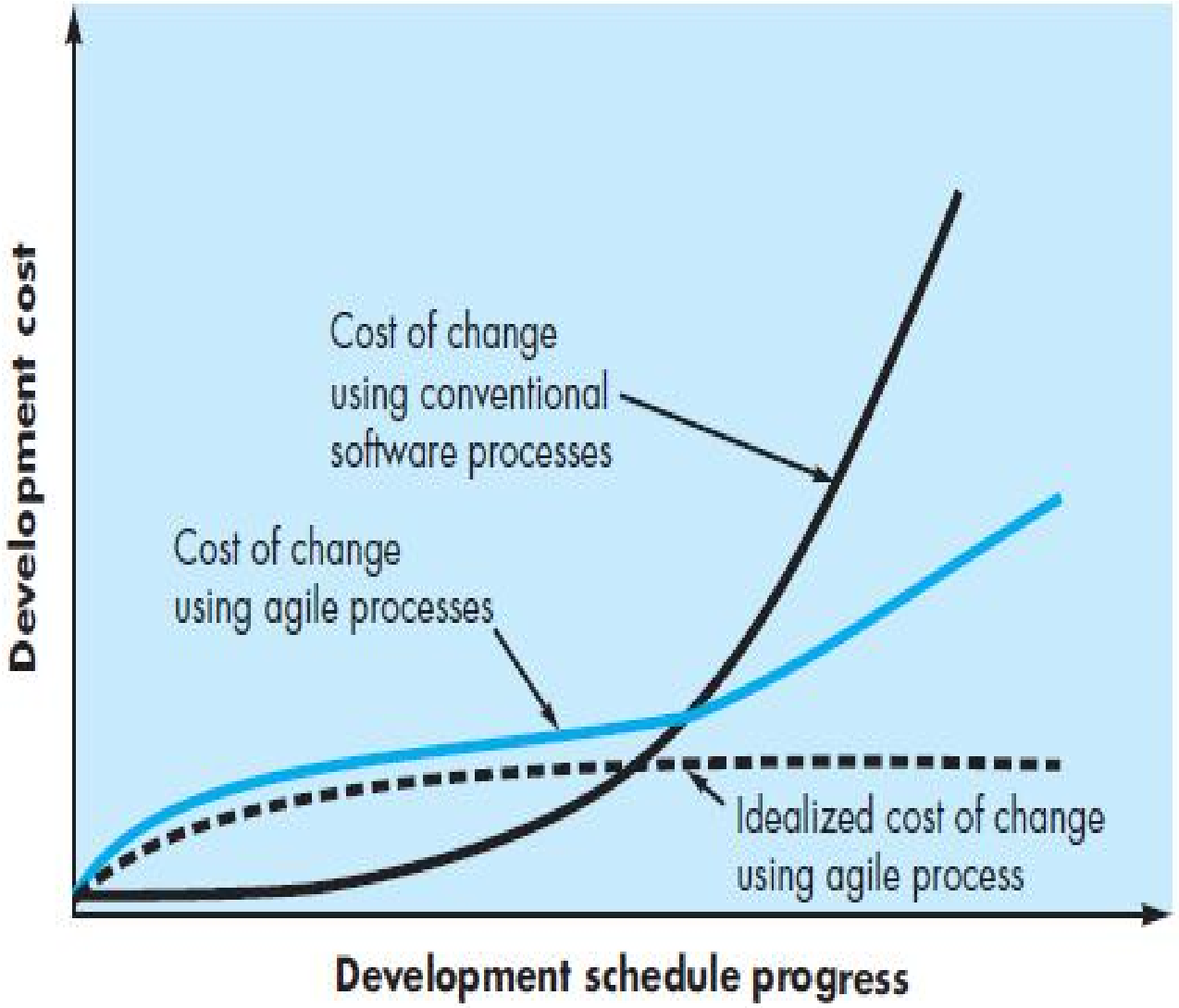
# Chapter 5 AGILE DEVELOPMENT

of three new components, modifications to another five components, the design of new tests, and so on. Costs escalate quickly, and the time and cost required to ensure that the change is made without unintended side effects is nontrivial.

Proponents of agility argue that a well-designed agile process "flattens" the cost of change curve ( Figure 5.1 , shaded, solid curve), allowing a software team to accommodate changes late in a software project without dramatic cost and time impact. You've already learned that the agile process encompasses incremental delivery. When incremental delivery is coupled with other agile practices such as continuous unit testing and pair programming (discussed later in this chapter), the cost of making a change is attenuated. Although debate about the degree to which the cost curve flattens is ongoing, there is evidence to suggest that a significant reduction in the cost of change can be achieved.

**FIGURE 5.1**

Change costs as a function of time in development



# Chapter 5 AGILE DEVELOPMENT

## 5.3 WHAT IS AN AGILE PROCESS ?(read by yourselves as exercise)

**Any agile software process is characterized in a manner that addresses a number of key assumptions about the majority of software projects:**

① **It is difficult to predict in advance which software requirements will persist and which will change. It is equally difficult to predict how customer priorities will change as the project proceeds.**

② **For many types of software, design and construction are interleaved. That is, both activities should be performed in tandem so that design models are proven as they are created. It is difficult to predict how much design is necessary before construction is used to prove the design.**

# Chapter 5 AGILE DEVELOPMENT

③ **Analysis, design, construction, and testing are not as predictable (from a planning point of view) as we might like.**

**Given these three assumptions, an important question arises: How do we create a process that can manage unpredictability?The answer, as we have already noted, lies in process adaptability (to rapidly changing project and technical conditions). An agile process, therefore, must be adaptable.**

**Hence, an incremental development strategy should be instituted. Software increments (executable prototypes or portions of an operational system) must be delivered in short time periods so that adaptation keeps pace with change (unpredictability). This iterative approach enables the customer to evaluate the software increment regularly, provide necessary feedback to the software team, and influence the process adaptations that are made to accommodate the feedback.**

# Chapter 5 AGILE DEVELOPMENT

## 5.3.1 Agility Principles (read by yourselves，as exercise)

**The Agile Alliance defines 12 agility principles for those who want to achieve agility:**

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.**

# Chapter 5 AGILE DEVELOPMENT

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.**

# Chapter 5 AGILE DEVELOPMENT

⑪ **The best requirements analysis, 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.**

**Not every agile process model applies these 12 principles with equal weight, and some models choose to ignore (or at least downplay) the importance of one or more of the principles. However, the principles define an agile spirit.**

## 5.3.2 The Politics of Agile Development(read by yourselves)

## 5.4 EXTREME PROGRAMMING

**In order to illustrate an agile process in a bit more detail, we'll provide you with an overview of Extreme Programming (XP), the most widely used approach to agile software development.**

# Chapter 5 AGILE DEVELOPMENT

**A variant of XP, called Industrial XP(IXP), refines XP and targets the agile process specifically for use within large organizations.**

## 5.4.1 The XP Process

**Extreme Programming uses an object-oriented approach as its preferred development paradigm and encompasses a set of rules and practices that occur within the context of four framework activities: planning, design, coding,and testing. Figure 5.2 illustrates the XP process and notes some of the key ideas and tasks that are associated with each framework activity.**

**Planning. The planning activity (also called the planning game ) begins with listening —a requirements gathering activity that enables the technical members of the XP team to understand the business context for the software and to get a broad feel for required output and major features and functionality. Listening**

# Chapter 5 AGILE DEVELOPMENT

leads to the creation of a set of " stories " (also called user stories ) that describe required output, features, and functionality for software to be built. Each story (similar to use cases described in Chapter 8) is written by the customer and is placed on an index card. The customer assigns a value (i.e., a priority) to the story based on the overall business value of the feature or function. Members of the XP team then assess each story and assign a cost —measured in development weeks—to it. If the story is estimated to require more than three development weeks, the customer is asked to split the story into smaller stories and the assignment of value and cost occurs again. It is important to note that new stories can be written at any time.

Customers and developers work together to decide how to group stories into the next release (the next software increment) to be

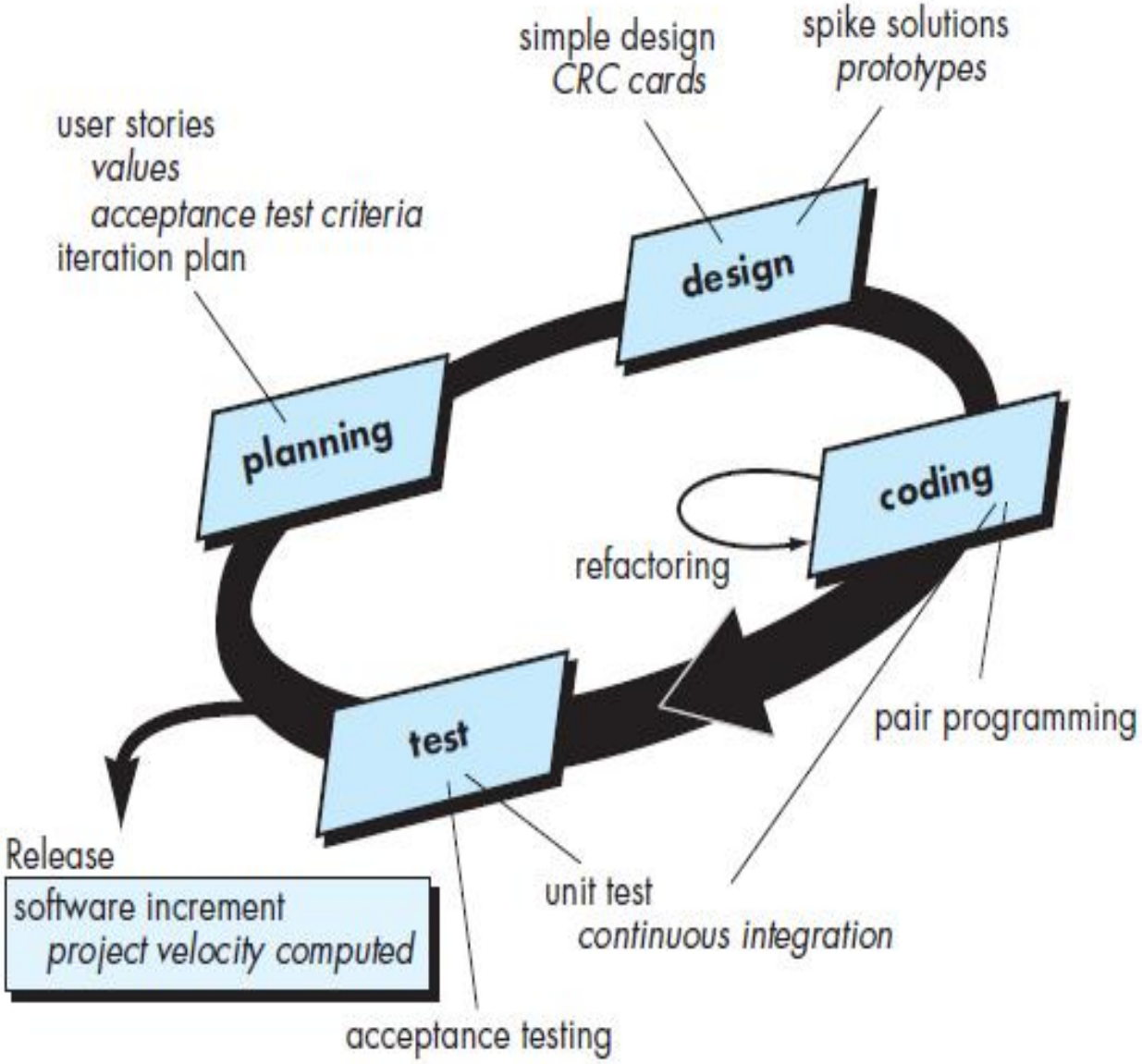
# Chapter 5 AGILE DEVELOPMENT

**developed by the XP team. Once a basic commitment (agreement on stories to be included, delivery date, and other project matters) is made for a release, the XP team orders the stories that will be developed in one of three ways: (1) all stories will be implemented immediately (within a few weeks), (2) the stories with highest value will be moved up in the schedule and implemented first, or (3) the riskiest stories will be moved up in the schedule and implemented first.**

**After the first project release (also called a software increment) has been delivered, the XP team computes project velocity. Stated simply, project velocity is the number of customer stories implemented during the first release. Project velocity can then be used to (1) help estimate delivery dates and schedule for subsequent releases and (2) determine whether an**

**FIGURE 5.2**

The Extreme Programming process



# Chapter 5 AGILE DEVELOPMENT

**overcommitment has been made for all stories across the entire development project. If an overcommitment occurs, the content of releases is modified or end delivery dates are changed.**

**As development work proceeds, the customer can add stories, change the value of an existing story, split stories, or eliminate them. The XP team then reconsiders all remaining releases and modifies its plans accordingly.**

**Design. XP design rigorously follows the KIS (keep it simple) principle. XP encourages the use of CRC cards (Chapter 10) as an effective mechanism for thinking about the software in an object-oriented context. CRC(class-responsibility-collaborator) cards identify and organize the object-oriented classes that are relevant to the current software increment. The CRC cards are the only design work product produced as part of the XP process.**

# Chapter 5 AGILE DEVELOPMENT

**If a difficult design problem is encountered as part of the design of a story, XP recommends the immediate creation of an operational prototype of that portion of the design. Called a spike solution , the design prototype is implemented and evaluated. The intent is to lower risk when true implementation starts and to validate the original estimates for the story containing the design problem.**

**Coding. After preliminary design work is done, the team does not move to code, but rather develops a series of unit tests that will exercise each of the stories that is to be included in the current release (software increment). Once the unit test has been created, the developer is better able to focus on what must be implemented to pass the test. Once the code is complete, it can be unit-tested immediately, thereby providing instantaneous feedback to the developers.**

# Chapter 5 AGILE DEVELOPMENT

**A key concept during the coding activity (and one of the most talked-about aspects of XP) is pair programming. XP recommends that two people work together at one computer workstation to create code for a story. This provides a mechanism for real-time problem solving (two heads are often better than one) and real time quality assurance (the code is reviewed as it is created). It also keeps the developers focused on the problem at hand.**

**As pair programmers complete their work, the code they develop is integrated with the work of others. In some cases this is performed on a daily basis by an integration team. In other cases, the pair programmers have integration responsibility. This "continuous integration" strategy helps to avoid compatibility and interfacing problems and provides a "smoke testing" environment that helps to uncover errors early.**

# Chapter 5 AGILE DEVELOPMENT

**Testing. The unit tests that are created should be implemented using a framework that enables them to be automated (hence, they can be executed easily and repeatedly). This encourages a regression testing strategy whenever code is modified.**

**As the individual unit tests are organized into a "universal testing suite", integration and validation testing of the system can occur on a daily basis.**

**XP acceptance tests, also called customer tests, are specified by the customer and focus on overall system features and functionality that are visible and reviewable by the customer. Acceptance tests are derived from user stories that have been implemented as part of a software release.**

# Chapter 5 AGILE DEVELOPMENT

## 5.4.2 Industrial XP(read by yourselves)

## 5.5 OTHER AGILE PROCESS MODELS

**As we noted in the last section, the most widely used of all agile process models is Extreme Programming (XP). But many other agile process models have been proposed and are in use across the industry. In this section, we present a brief overview of a few common agile methods.**

### 5.5.1 Scrum and DevOps(9/e, 3.5.3节)

**Scrum：**

**Refer to appemdix Scrum agile developoment.pdf**

**DevOps:**

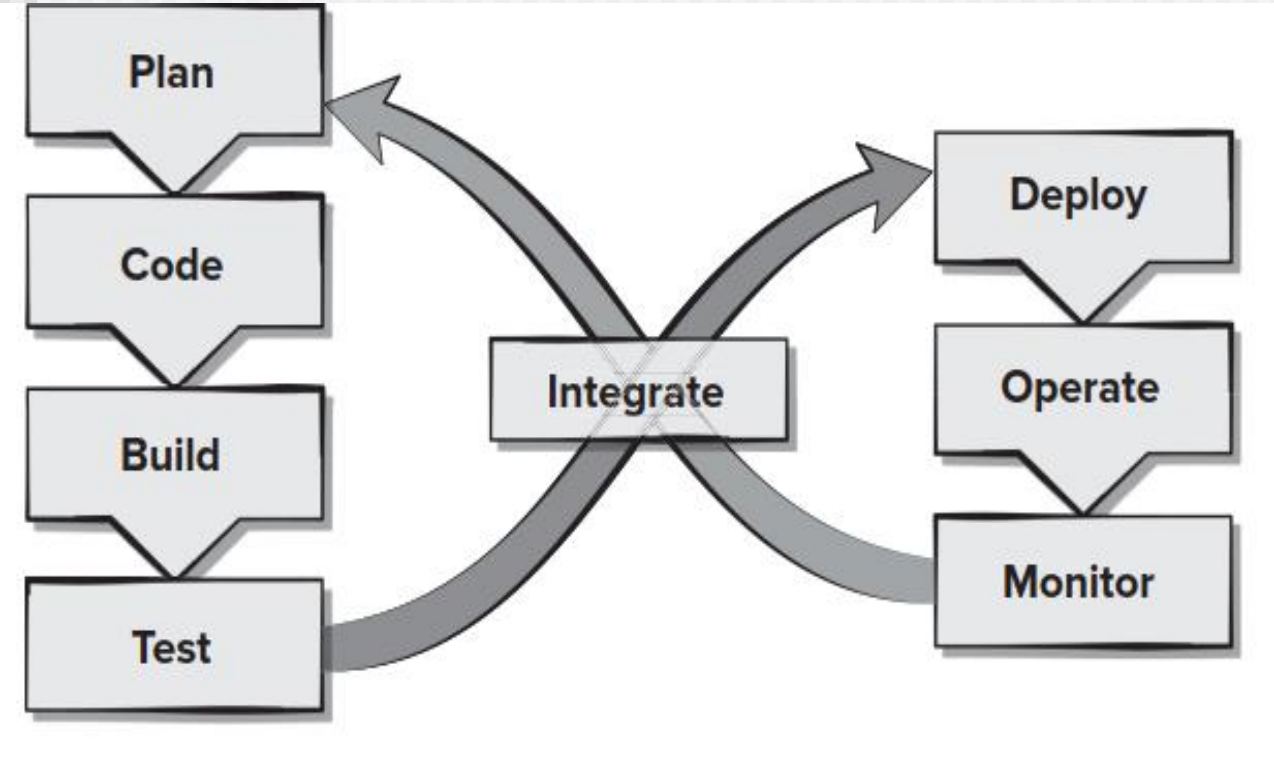
# Chapter 5 AGILE DEVELOPMENT

**DevOps was created by Patrick DeBois to combine Development and Operations. DevOps attempts to apply agile and lean development principles across the entire software supply chain. Figure 5.3 presents an overview of the DevOps workflow. The DevOps approach involves several stages that loop continuously until the desired product exists:**

- **Continuous development. Software deliverables are broken down and developed in multiple sprints with the increments delivered to the quality assurance members of the development team for testing.**
- **Continuous testing. Automated testing tools are used to help team members test multiple code increments at the same time to ensure they are free of defects prior to integration.**
- **Continuous integration. The code pieces with new functionality are added to the existing code and to the run-time environment and then checked to ensure there are no errors after deployment.**

# Chapter 5 AGILE DEVELOPMENT

**Figure 5.3**



# Chapter 5 AGILE DEVELOPMENT

- **Continuous deployment. At this stage the integrated code is deployed (installed) to the production environment, which might include multiple sites globally that need to be prepared to receive the new functionality.**
- **Continuous monitoring. Operations(运维) staff who are members of the development team help to improve software quality by monitoring its performance in the production environment and proactively looking for possible problems before users find them.**

**DevOps enhances customers' experiences by reacting quickly to changes in their needs or desires. This can increase brand loyalty and increase market share. Lean approaches like DevOps can provide organizations with increased capacity to innovate by reducing rework and allowing shifts to higher business value activities. Products do not make money until consumers have access to them, and DevOps can provide faster deployment time to production platforms**

# Chapter 5 AGILE DEVELOPMENT

## 5.6 A TOOL SET FOR THE AGILE PROCESS

**Some proponents of the agile philosophy argue that automated software tools(e.g., design tools) should be viewed as a minor supplement to the team's activities, and not at all pivotal to the success of the team. However, Alistair Cockburn suggests that tools can have a benefit and that "agile teams stress using tools that permit the rapid flow of understanding.**

**Exercise:find a project that developend by XP or Scrum, and discuss it.**