Ch.31 Project Management Concepts (Cont.)

March 29, 2015

- **Team Coordination & Communication**
  - *Formal, impersonal approaches* include software engineering documents and work products (including source code), technical memos, project milestones, schedules, and project control tools (Ch. 23), change requests and related documentation, error tracking reports, and repository data (see Ch.26).
  - *Formal, interpersonal procedures* focus on quality assurance activities (Ch.25) applied to software engineering work products. These include status review meetings and design and code inspections.
  - *Informal, interpersonal procedures* include group meetings for information dissemination and problem solving and "collocation of requirements and development staff."
  - *Electronic communication* encompasses electronic mail, electronic bulletin boards, and by extension, video-based conferencing systems.
  - *Interpersonal networking* includes informal discussions with team members and those outside the project who may have experience or insight that can assist team members.

- **The Product Scope**
  - Scope
    - *Context*. How does the software to be built fit into a larger system, product, or business context and what constraints are imposed as a result of the context?
    - *Information objectives*. What customer-visible data objects (Ch.8) are produced as output from the software? What data objects are required for input?
    - *Function and performance*. What function does the software perform to transform input data into output? Are any special performance characteristics to be addressed?
  - Software project scope must be unambiguous and understandable at the management and technical levels.

- **Problem Decomposition**
  - Sometimes called *partitioning* or *problem elaboration*
  - Once scope is defined …
    - It is decomposed into constituent functions
    - It is decomposed into user-visible data objects

    or
    - It is decomposed into a set of problem classes
  - Decomposition process continues until all functions or problem classes have been defined

- **The Process**
  - Once a process framework has been established
    - Consider project characteristics
    - Determine the degree of rigor required
    - Define a task set for each software engineering activity
      - Task set =
        - Software engineering tasks
        - Work products
        - Quality assurance points
        - Milestones

- **Melding the Problem and the Process**

COMMON PROCESS FRAMEWORK ACTIVITIES	communication	planning	modeling	construction	deployment
Software Engineering Tasks					
Product Functions					
Text input					
Editing and formatting					
Automatic copy edit					
Page layout capability					
Automatic indexing and TOC					
File management					
Document production					

- **The Project**
  - Projects get into trouble when …
    - Software people don't understand their customer's needs.
    - The product scope is poorly defined.
    - Changes are managed poorly.
    - The chosen technology changes.
    - Business needs change [or are ill-defined].
    - Deadlines are unrealistic.
    - Users are resistant.
    - Sponsorship is lost [or was never properly obtained].
    - The project team lacks people with appropriate skills.
    - Managers [and practitioners] avoid best practices and lessons learned.

- Common-Sense Approach to Projects
  - *Start on the right foot.* This is accomplished by working hard (very hard) to understand the problem that is to be solved and then setting realistic objectives and expectations.
  - *Maintain momentum.* The project manager must provide incentives to keep turnover of personnel to an absolute minimum, the team should emphasize quality in every task it performs, and senior management should do everything possible to stay out of the team's way.
  - *Track progress.* For a software project, progress is tracked as work products (e.g., models, source code, sets of test cases) are produced and approved (using formal technical reviews) as part of a quality assurance activity.
  - *Make smart decisions.* In essence, the decisions of the project manager and the software team should be to "keep it simple."
  - *Conduct a postmortem analysis.* Establish a consistent mechanism for extracting lessons learned for each project.

- To Get to the Essence of a Project(W^5HH)
  - *W*hy is the system being developed?
  - *W*hat will be done?
  - *W*hen will it be accomplished?
  - *W*ho is responsible?
  - *W*here are they organizationally located?
  - *H*ow will the job be done technically and managerially?
  - *H*ow much of each resource (e.g., people, software, tools, database) will be needed?

Barry Boehm [Boe96]

- Critical Practices
  - Formal risk management
  - Empirical cost and schedule estimation
  - Metrics-based project management
  - Earned value tracking
  - Defect tracking against quality targets
  - People aware project management

Ch.5 Agile Development

- **The Manifesto for Agile Software Development （2001）**

宣言

  - "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."

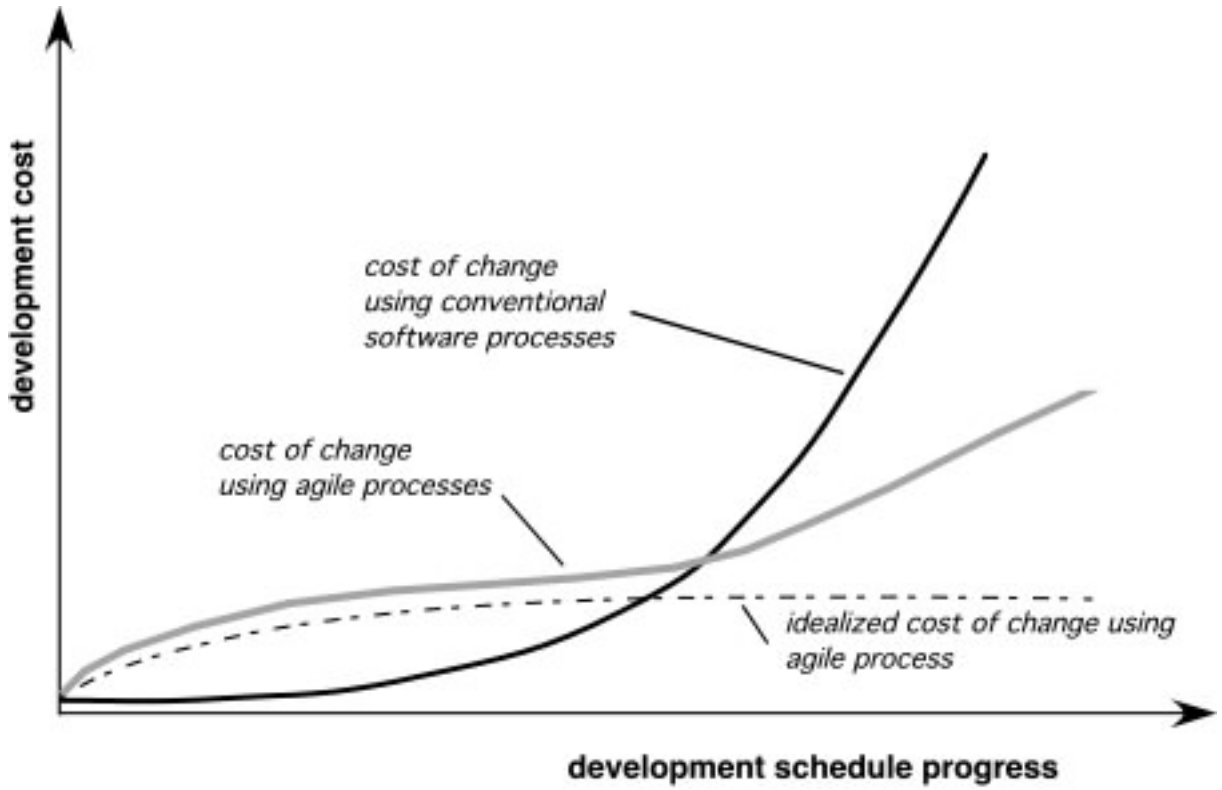
    – Kent Beck et al

- **What is "Agility"?**
  - 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**



- **An Agile Process**
  - Is driven by customer descriptions of what is required (scenarios)
  - Recognizes that plans are short-lived
  - Develops software iteratively with a heavy emphasis on construction activities
  - Delivers multiple ‘software increments’
  - Adapts as changes occur

- **Agility Principles - I** 利用

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.

- **Agility Principles - II**

可持续的

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.

- **Human Factors**
  - the process molds to the needs of the people and team, not the other way around
  - key traits must exist among the people on an agile team and the team itself:
    - Competence.
    - Common focus.
    - Collaboration.
    - Decision-making ability.
    - Fuzzy problem-solving ability.
    - Mutual trust and respect.
    - Self-organization.

- **Extreme Programming (XP)**

- The most widely used agile process, originally proposed by Kent Beck in 1996

- XP Planning
  - Begins with the creation of "***user stories***"
  - Agile team assesses each story and assigns a ***cost***
  - Stories are grouped to for a ***deliverable increment***
  - A ***commitment*** is made on delivery date
  - After the first increment "***project velocity***" is used to help define subsequent delivery dates for other increments

• Extreme Programming (XP)

- XP Design
  - Follows the ***KIS (Keep It Simple)principle***
  - Encourage the use of ***CRC cards*** (see Chapter 8)
  - For difficult design problems, suggests the creation of "***spike solutions***"—a design prototype
  - Encourages "***refactoring***"—an iterative refinement of the internal program design

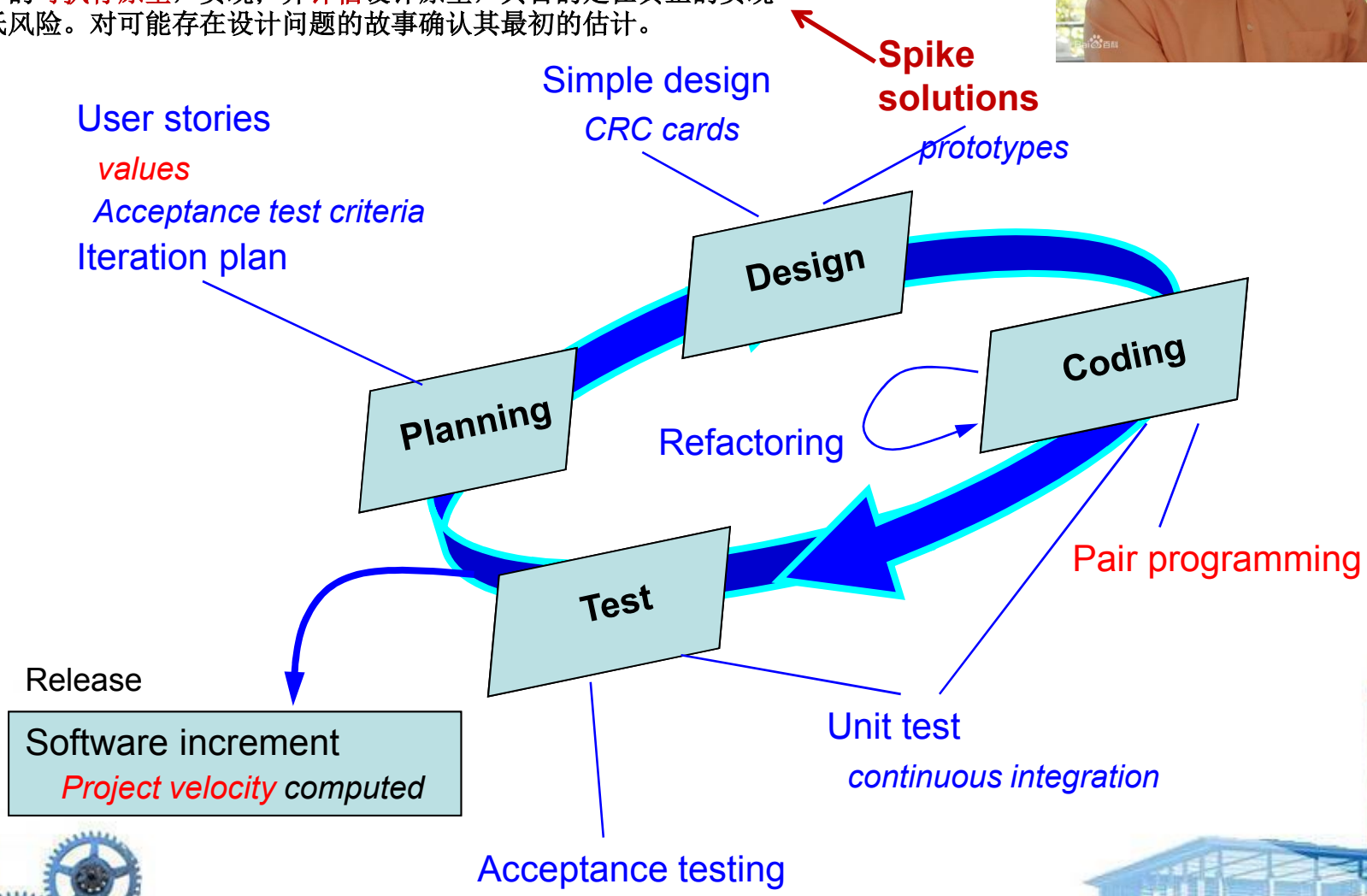
Class-**R**esponsibility-**C**ollaborator

- XP Coding
  - Recommends the ***construction of a unit test*** for a store before coding commences
  - Encourages "***pair programming***"

- XP Testing
  - All ***unit tests are executed daily***
  - "***Acceptance tests***" are defined by the customer and executed to assess customer visible functionality

Extreme Programming (XP)
--- by Kent Beck in 1996

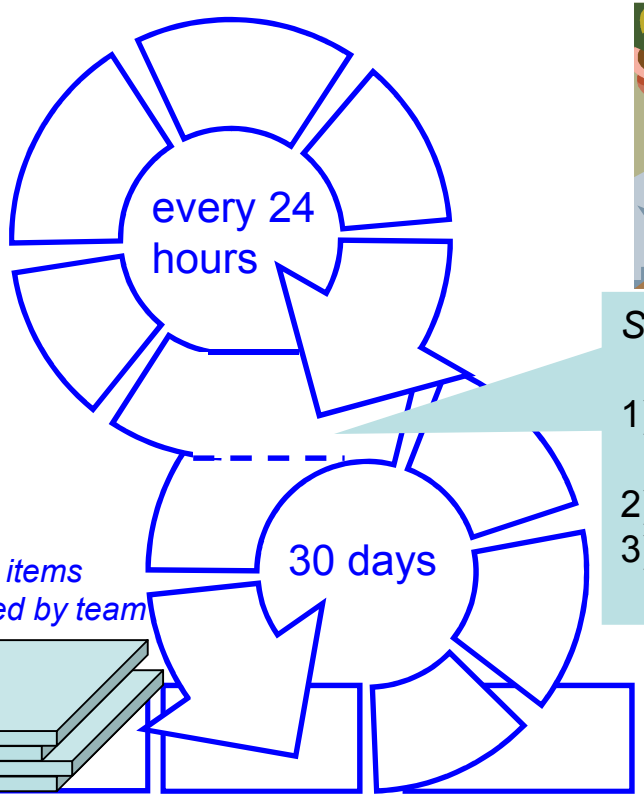
严格遵循原则；鼓励使用CRC卡；在某个故事设计中遇到困难时，立即建立这部分设计的可执行原型，实现；并评估设计原型，其目的是在真正的实现开始时降低风险。对可能存在设计问题的故事确认其最初的估计。



- **Industrial XP (IXP)**
  - IXP has greater inclusion of management, expanded customer roles, and upgraded technical practices
  - IXP incorporates six new practices:
    - Readiness assessment (on board?)
    - Project community
    - Project chartering
    - Test driven management
    - Retrospectives
    - Continuous learning

- **Scrum ---**by Schwaber and Beedle
  - Scrum—distinguishing features
    - Development work is partitioned into "***packets***"
    - ***Testing and documentation are on-going*** as the product is constructed
    - Work occurs in "***sprints***" and is derived from a "***backlog***" of existing requirements
    - ***Meetings are very short*** and sometimes conducted without chairs
    - "***demos***" are delivered to the customer with the time-box allocated

- **Scrum -- by Schwaber and Beedle**

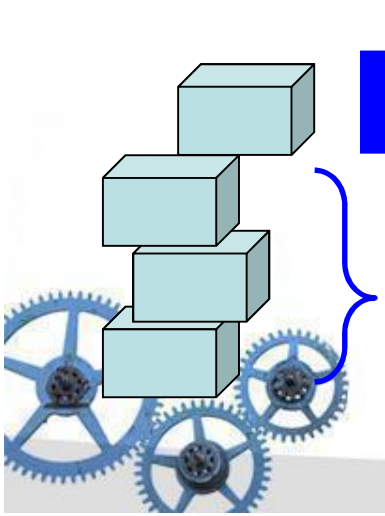


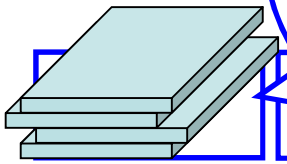
Scrum: 15-minute daily meeting.
Team members respond to basics:
1) What did you do since last Scrum meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?

冲刺　待定项

Sprint Backlog:
Feature(s) assigned to sprint

Backlog items expanded by team





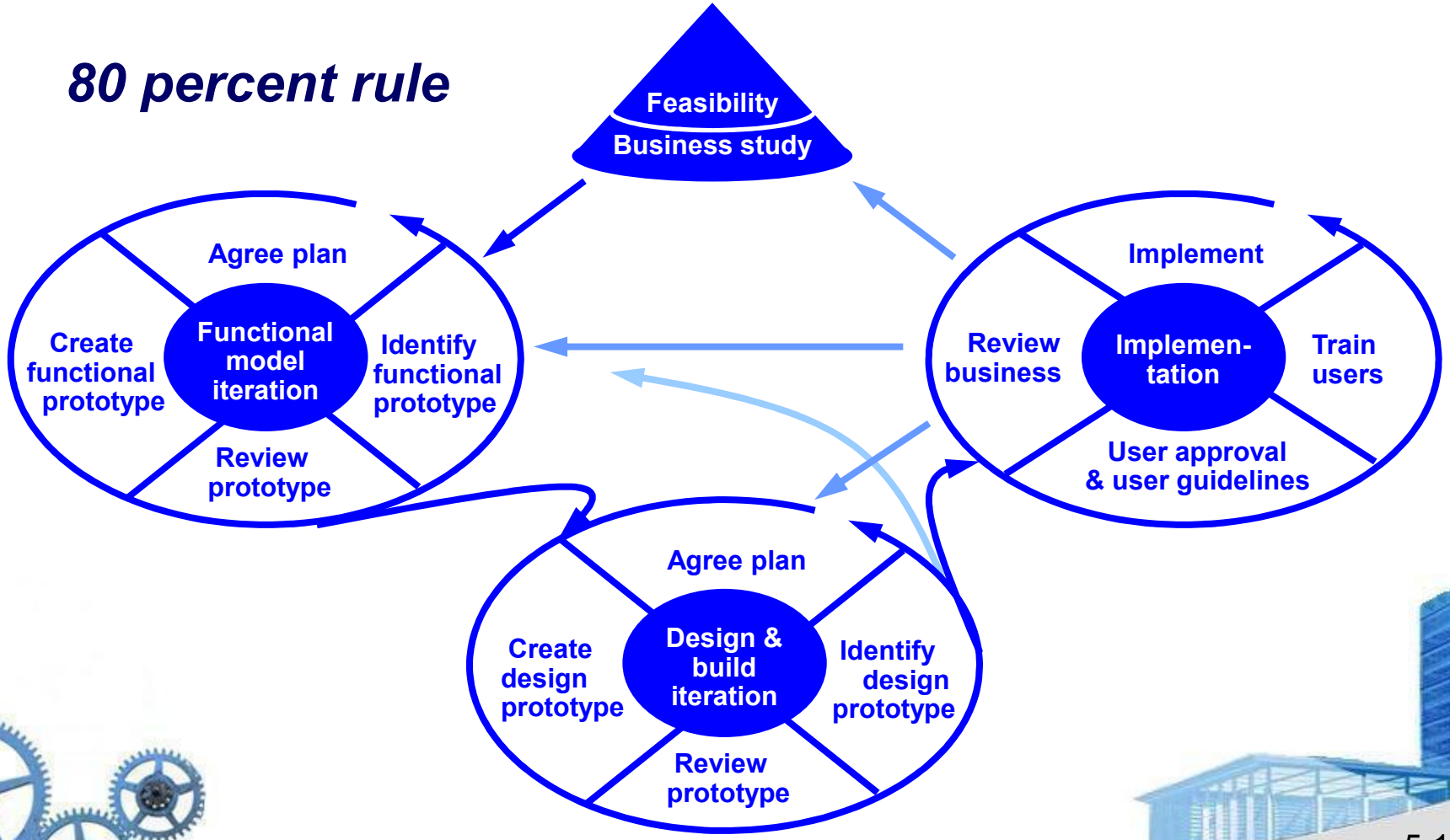
New functionality is demonstrated at end of sprint

Product Backlog:
Prioritized product features desired by the customer

- **Dynamic Systems Development Method**
  - Promoted by the DSDM Consortium (www.dsdm.org)
  - DSDM—distinguishing features
    - Similar in most respects to XP
    - Nine guiding principles
      - Active user involvement is imperative.
      - DSDM teams must be empowered to make decisions.
      - The focus is on frequent delivery of products.
      - Fitness for business purpose is the essential criterion for acceptance of deliverables.
      - Iterative and incremental development is necessary to converge on an accurate business solution.
      - All changes during development are reversible.
      - Requirements are baselined at a high level
      - Testing is integrated throughout the life-cycle.

- **Dynamic Systems Development Method (DSDM)**
  - **Promoted by DSDM Consortium (www.dsdm.org)**

80 percent rule



- **Agile Modeling**
  - Originally proposed by Scott Ambler
  - Suggests a set of agile modeling principles
    - Model with a purpose
    - Use multiple models
    - Travel light
    - Content is more important than representation
    - Know the models and the tools you use to create them
    - Adapt locally

- **Agile Unified Process**
  - Each AUP iteration addresses these activities:
    - Modeling
    - Implementation
    - Testing
    - Deployment
    - Configuration and project management
    - Environment management

Ch.6 Human Aspects of Software Engineering

6.1 Characteristics Of A Software Engineer

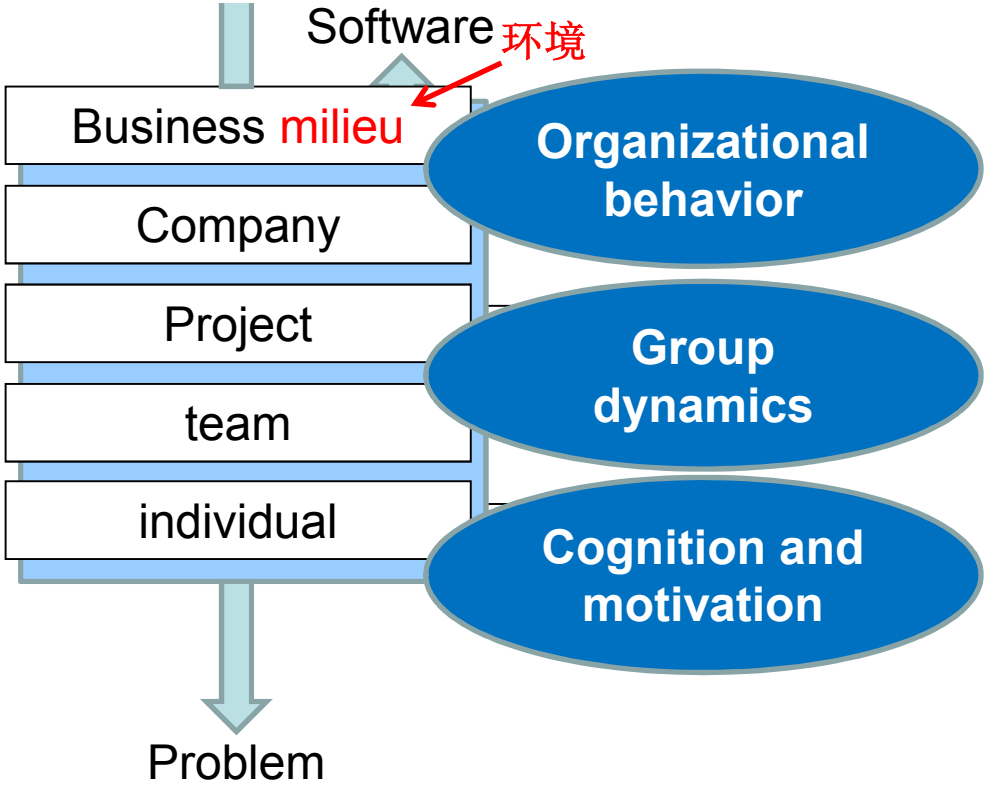
- Traits of Successful Software Engineers

- Sense of individual responsibility
- Acutely aware of the needs of team members and stakeholders
- Brutally honest about design flaws and offers constructive criticism
- Resilient under pressure
- Heightened sense of fairness
- Attention to detail
- Pragmatic

务实的，实际的

6.2 The Psychology Of Software Engineering

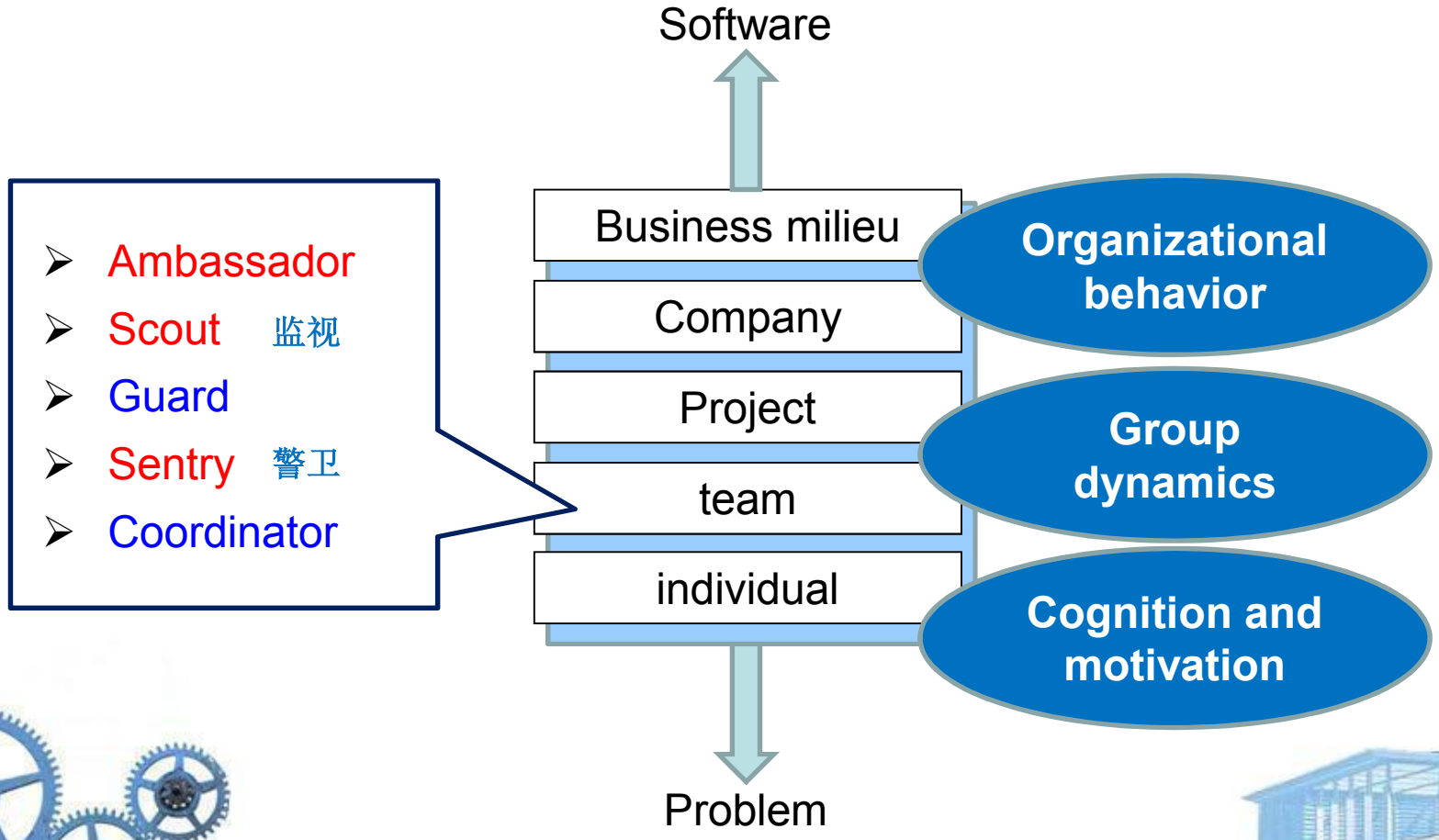
- Behavioral Model for Software Engineering
- Boundary Spanning Team Roles



Problem

6.2 The Psychology Of Software Engineering

- Behavioral Model for Software Engineering
- Boundary Spanning Team Roles



6.3 The Software Team

- Effective Software Team Attributes
  - Sense of purpose
  - Sense of involvement
  - Sense of trust
  - Sense of improvement
  - Diversity of team member skill sets

6.4 Team Structures

- Factors Affecting Team Structure
  - the difficulty of the problem to be solved
  - the size of the resultant program(s) in lines of code or function points
  - the time that the team will stay together (team lifetime)
  - the degree to which the problem can be modularized
  - the required quality and reliability of the system to be built
  - the rigidity of the delivery date
  - the degree of sociability (communication) required for the project

6.5 Agile Teams

- Generic Agile Teams
  - Stress individual competency coupled with group collaboration as critical success factors
  - People trump process and politics can trump people
  - Agile teams as self-organizing and have many structures
    - An adaptive team structure
    - Uses elements of Constantine's random, open, and synchronous structures
    - Significant autonomy
  - Planning is kept to a minimum and constrained only by business requirements and organizational standards

6.5 Agile Teams

- XP Team Values
  - Communication – close informal verbal communication among team members and stakeholders and establishing meaning for metaphors as part of continuous feedback
  - Simplicity – design for immediate needs nor future needs
  - Feedback – derives from the implemented software, the customer, and other team members
  - Courage – the discipline to resist pressure to design for unspecified future requirements
  - Respect – among team members and stakeholders

6.6 Impact of Social Media

- Blogs – can be used share information with team members and customers
- Microblogs – allow posting of real-time messages to individuals following the poster (e.g. Twitter)
- Targeted on-line forums – allow participants to post questions or opinions and collect answers
- Social networking sites– allows connections among software developers for the purpose of sharing information (e.g. Facebook, 人人，LinkedIn)
- Social book marking– allow developers to keep track of and share web-based resources (e.g. Delicious, Stumble, CiteULike)

6.7 Software Engineering using the Cloud

- Benefits
  - Provides access to all software engineering work products
  - Removes device dependencies and available every where
  - Provides avenues for distributing and testing software
  - Allows software engineering information developed by one member to be available to all team members
- Concerns
  - Dispersing cloud services outside the control of the software team may present reliability and security risks
  - Potential for interoperability problems becomes high with large number of services distributed on the cloud
  - Cloud services stress usability and performance which often conflicts with security, privacy, and reliability

Task

- **Review** Chapter 31,5,6,
- **Finish** “Problems and points to ponder” in Ch. 31,5,6
- **Preview** Ch. 7,8,9
- **Prepare for Requirement Report!!**