



Ch.6 Human Aspects of Software Engineering





6.1 Characteristics Of A Software Engineer

- Traits of Successful Software Engineers



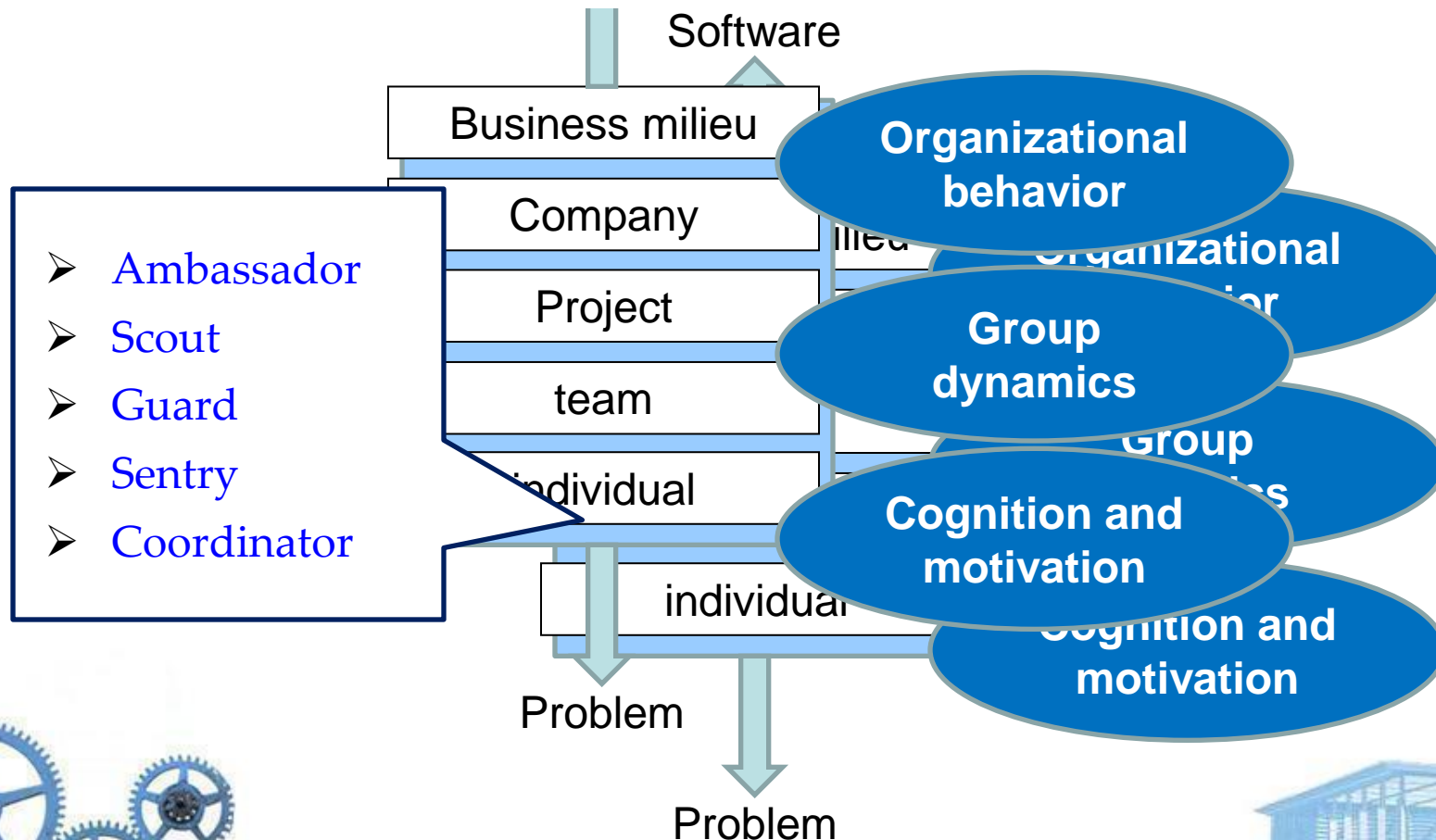
- Sense of individual responsibility
- Acutely aware of the needs of team members and stakeholders
- Brutally honest about design flaws and offers constructive criticism
- Resilient under pressure
- Heightened sense of fairness
- Attention to detail
- Pragmatic





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

- Avoid Team “Toxicity”

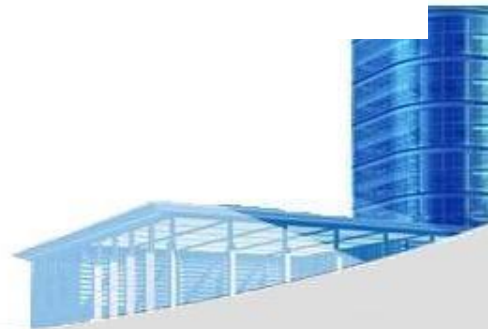
- **A frenzied work atmosphere** in which team members waste energy and lose focus on the objectives of the work to be performed.
- **High frustration** caused by personal, business, or technological factors that cause friction among team members.
- **“Fragmented or poorly coordinated procedures”** or a poorly defined or improperly chosen process model affecting accomplishment.
- **Unclear definition** of roles resulting in a lack of accountability and resultant finger-pointing.
- **“Continuous and repeated exposure to failure”** that leads to a loss of confidence and a lowering of morale.





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

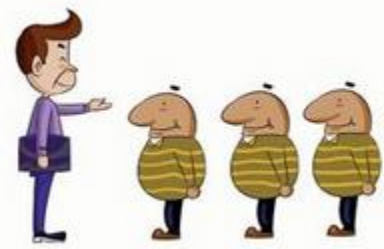




6.4 Team Structures

- Organizational Paradigms

- **Closed paradigm** —structures a team along a traditional hierarchy of authority
- **Random paradigm** —structures a team loosely and depends on individual initiative of the team members
- **Open paradigm** —attempts to structure a team in a manner that achieves some of the controls associated with the closed paradigm but also much of the innovation that occurs when using the random paradigm
- **Synchronous paradigm** —relies on the natural compartmentalization of a problem and organizes team members to work on pieces of the problem with little active communication among themselves





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, 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

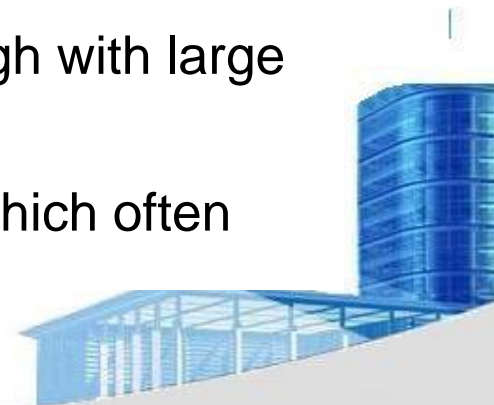


- **Benefits**

- 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





6.8 Collaboration Tools

- Services of collaborative development environments(CDEs)
 - **Namespace** that allows secure, private storage or work products
 - **Calendar** for coordinating project events
 - **Templates** that allow team members to create artifacts that have common look and feel
 - **Metrics support** to allow quantitative assessment of each team member's contributions
 - **Communication analysis** to track messages and isolates patterns that may imply issues to resolve
 - **Artifact clustering** showing work product dependencies





6.9 Global Teams

- Team Decisions Making Complications
 - Problem complexity
 - Uncertainty and risk associated with the decision
 - Work associated with decision has unintended effect on another project object (law of unintended consequences)
 - Different views of the problem lead to different conclusions about the way forward
 - Global software teams face additional challenges associated with collaboration, coordination, and coordination difficulties





6.9 Global Teams

- Factors Affecting Global Software Development Team

