Chapter 6

- **Human Aspects of Software Engineering**

Slide Set to accompany

Software Engineering: A Practitioner's Approach, 8/e

by Roger S. Pressman and Bruce R. Maxim

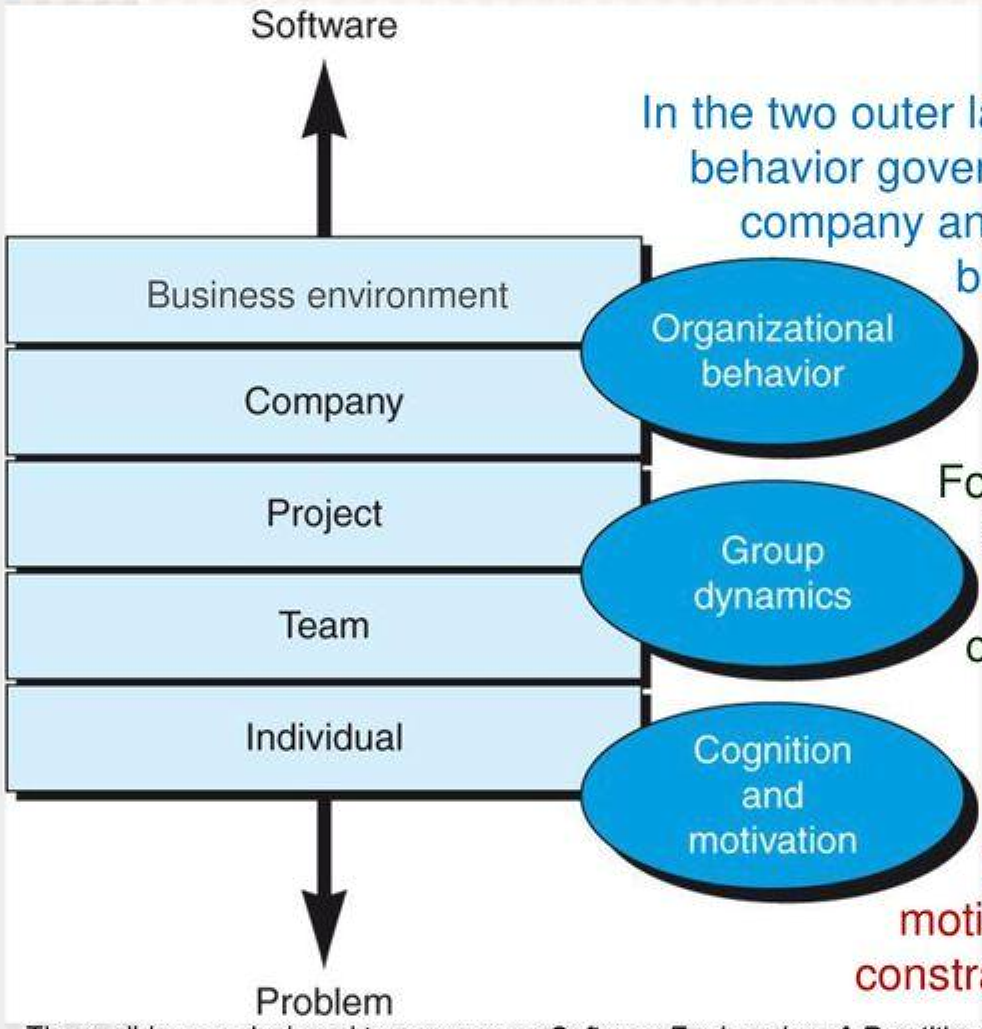
It is essential that we understand …

- that the human aspects of software engineering often have as much to do with the success of a project as the latest and greatest technology
- that without skilled and motivated people, success of a software project is unlikely
- that software engineers on a team must play well with each other and with other project stakeholders
- the characteristics of successful software engineers
- the structure & dynamics of successful project teams

Characteristics of Successful Software Engineers

- Sense of individual responsibility – driven to deliver as promised
- Acutely aware of the needs of team & stakeholders
- Brutally honest – realistic and truthful about design flaws and schedule slippage, and offers constructive criticism
- Resilient under pressure – is able to manage the pressure so that his performance does not suffer
- Heightened sense of fairness – shares credit, avoids conflict of interest, does not sabotage other's work
- Attention to detail – carefully considers the technical decisions he makes on a daily basis against broader criteria (e.g., performance, cost, quality)
- Pragmatic – they adapt based on the circumstances

Layered Behavioral Model for Software Development



In the two outer layers, organizational behavior governs the actions of the company and its response to the business environment

Focus is on team structure and social factors that govern success. Group communication, collaboration, and coordination are as important as the skills of team members

Focus is on identifying the problem, the skills required to solve it, and the motivation to finish the solution within the constraints of the outer layers in the model

Boundary Spanning Team Roles

- Teams often establish artificial boundaries that reduce communication and, as a consequence, reduce the team effectiveness
- To overcome this problem, a set of *boundaries spanning roles* should be defined to allow members of a software team to effectively move across team boundaries
- These roles may be assigned explicitly or can evolve naturally

Boundary Spanning Team Roles

Ambassador – represents team to outside constituencies

Scout – crosses team boundaries to collect information

Guard – protects access to team work products

Sentry – controls information sent by stakeholders

Coordinator – communicates across the team and organization

Characteristics of Effective Software Teams

Members of effective teams are more productive and motivated than average. They possess:

- a sense of purpose – have a common definition of success
- a sense of involvement – allowing every member to feel that their skills and contributions are valued
- a sense of trust – in the knowledge and skills of the other team members
- a sense of improvement – a focus on learning, reflecting, and improving continuously
- diverse skill sets

Effective Teams Avoid the following Team **Toxins**

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

Team Structure Paradigms

Closed — structures a team along a traditional hierarchy of authority

Random — structures a team loosely and depends on individual initiative of the team members

Open —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 — 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

The choice of the right structure for a team depends upon …

- 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 & reliability of the system to be built
- the rigidity of the delivery date
- the degree of sociability (communication) required for the project

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 random, open, and synchronous structures
  - Significant autonomy
- Planning is kept to a minimum, constrained only by business requirements and organizational standards

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 current needs, not 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

Using Social Media in Software Development

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

Software Engineering using the Cloud

Benefits

- Provides access to all work products
- Removes device dependencies and available everywhere
- Provides avenues for distributing and testing software
- Allows 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, & reliability

Services Designed to Enhance Collaborative Software Development

- Namespace that allows secure, private storage of 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

Factors that Cause Complications in Team Decision Making

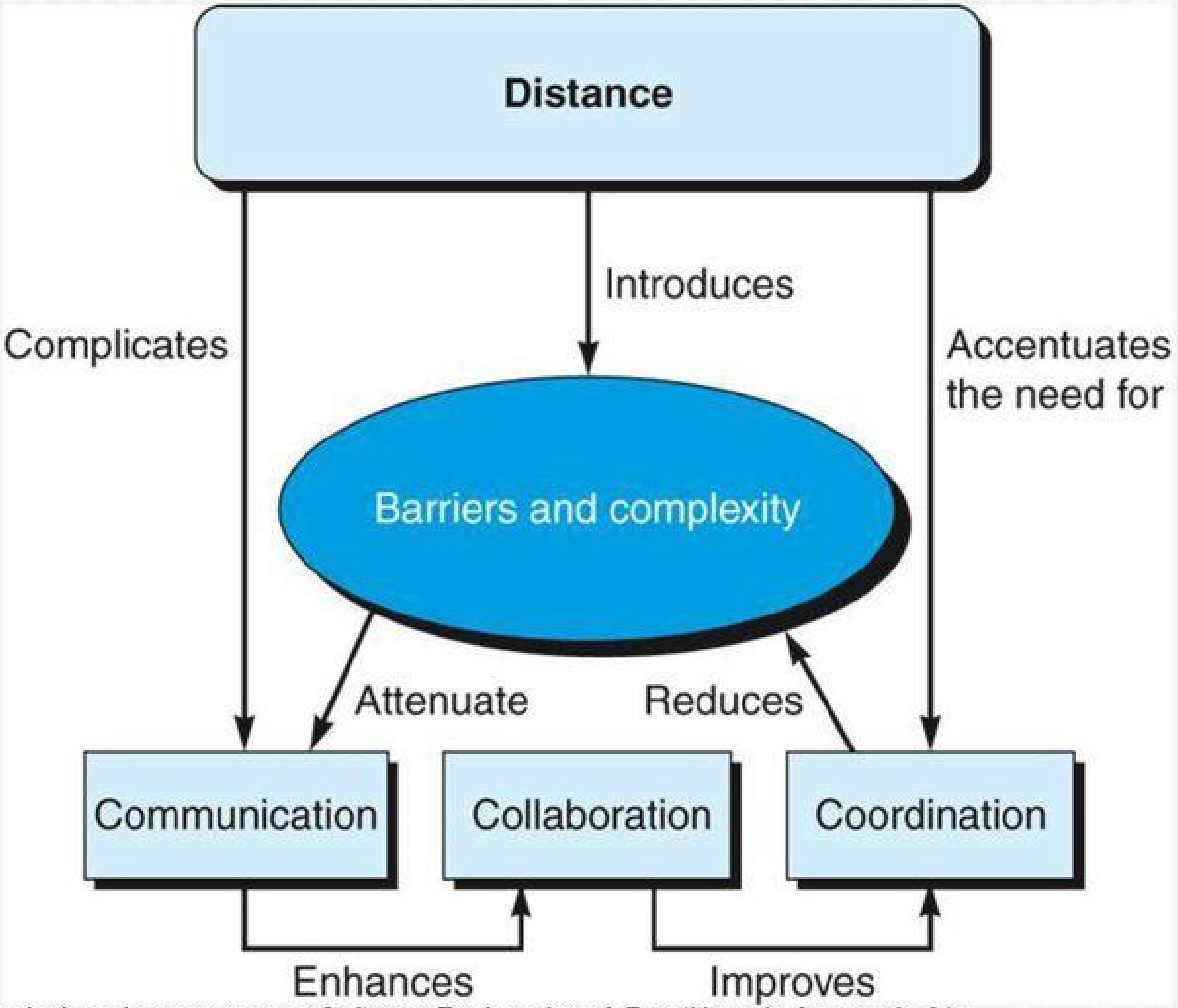
- Problem complexity
- Uncertainty and risk associated with the decision
- Work associated with decision has unintended effect on another project object (law of unintended consequences)
- Different views of the problem lead to different conclusions about the way forward
- Global software teams face additional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

Global Software Development Teams

- For the past few decades, an increasing number of major software products have been built by software teams that are often located on different continents

- These global teams have many of the characteristics of a conventional software team, but they also face other unique challenges that include coordination, collaboration, communication, and decision making

Factors Affecting Global Software Development Teams


Distance
Introduces
Complicates
Accentuates the need for
Barriers and complexity
Attenuate
Reduces
Communication
Collaboration
Coordination
Enhances
Improves