OOAD & Software Engineering (UE18CS353)
Unit 5

Aronya Baksy

April 2021

1 Ethics in SE

- Professional ethics: guiding principles for ideal behaviour and actions in a professional environment
- SE Ethics becomes important when design/implementation/maintenance decisions taken during the development affect real people.
- All decisions taken during software engineering must be guided by technical as well as ethical considerations. Software design must incorporate ethics.
- Code of Ethics is a generic term used for a document that describes the interaction between ethics and technology. CoE ensures that this interaction is structured, and may/may not carry legal weight.

1.1 Code of Ethics

- Guiding principles in understanding boundaries and guides the behaviour of free agents in taking decisions
- Integrity, objectivity, competence, confidentiality, behaviour wrt. resource usage, respect for Intellectual Property

1.2 Code of Conduct

- Day-to-day behaviour of employees at the workplace.
- Equality, fairness, empathy, respect, compliance with laws/standards/guidelines, portraying realistic competence level.

1.3 Eight Principles

- Act with **pulbic** interest in mind
- Act in a manner that is in the best interest of **client and employer** while keeping the above in mind
- Build **products** that meet the highest professional standards of quality.
- Maintain integrity and independence in professional **judgement**
- Promote an ethical approach to **management** of Software development and maintenance
- Advance integrity and reputation of the **profession**.
- Be fair to and support your **colleagues**
- **Self**: engage in lifelong learning, promote ethical approach to the professsion of software engineering.

1.4 Hacking

- Hackers solve problems in non-standard methods (e.g. exploiting weaknesses of systems). Motivated more by novelty/challenges as against traditional rewards like $$$ and power.
- Hacking involves poking around with multiple solutions to see which one works best. May lead to traceability issues due to the lack of documentation associated with hacking.
- Engineers, by contrast, take solutions to existing problems and seek to improve their non-functional attributes (aesthetics, reliability, performance benchmarks) through technical solutions, within a budget and time constraint.
- Engineering involves crafting a solution understanding why and the considering the best practices.
- Computer hackers are classified as
  - **White-Hat**: improve organization security by finding vulnerabilites, design solutions to patch them up
  - **Black-Hat**: look for exploits in organization security that can be used for gathering data for purposes such as corporate espionage, nation-state hacking etc.

2 Global Software Engineering

- Traditional approach: co-located teams working on inter-related goals.
- Global Development Team: Use virtual teams to develop software, linked by communication technologies and working remotely.
- Global distance = geographical distance + linguistic distance + temporal (time-zone) distance + cultural distance
- Software teams can be organized as **co-located**, **multi-site** or **global** (multi-site teams organized across > 1 country)
- Advantages of global development:
  - Increase productivity hours, leads to faster delivery
  - Larger pool of global developers, keep teams closer to clients
  - Diverse stakeholders with diverse knowledge and experience.
- Communication, Teaming, Collaboration and Project management practices and processes if put in plan can mitigate the challenges of Global development.

2.1 Challenges and Solutions in Global SE

- **Communication**: involve less informal communication, build trust through team-building activites
- **Coordination**: ensure shared sense of urgency, and task awareness
- **Control**: use accurate tools for tracking issues and progress, maintain uniform process across locations
- **Culture**: Build awareness about social backgrounds, attitudes, cultural distances

3 ITSM and ITIL

3.1 IT Service Management (ITSM)

- ITSM processes manage deployment of software products in real-world environments.
- ITSM is about how an IT organization manages IT services for customers and provides a stable IT environment that supports the business.
- THe objectives of ITSM are:
  - Improved availability, security, reliability of IT infrastructure
  - Increased flexibility, productivity, scaling
  - Predictable support and reduction in costs

3.2 ITSM Processes

3.2.1 Availability Management

- Manage expectations of agreed-upon level of functioning of services in a cost-effective and efficient manner
- The 7 Rs (Reliability, Redundancy, Repairability, Recoverability, Responsiveness, Robustness, Reputation)

3.2.2 Performance/Tuning

- Performance optimization for increased throughput, minimized response time even in the face of dynamic workloads
- For networks, storage devices, servers, databases

3.2.3 Acceptance

- Methodology for consistently deploying software to a production environment regardless of environment etc.
- Maintain integrity of application post deployment

3.2.4 Change Management

- Changes made to IT environment for improvement of performance/reliability etc., or for fixing existing issues in the environment
- Involves change control (request, prioritize, approval) and co-ordination (collaboration, schedule, communicate and implement)

3.2.5 Problem Management

- Log, track, analyze and resolve problems raised in the IT environment.
- When client initiates a call, the problem is analyzed and added for tracking and logging

3.2.6 Storage and Network Management

- Increase performance, reliability, utilization of storage and network devices.

3.2.7 Configuration Management

- Document relationships between different versions of software and hardware components in the IT infrastructure

3.2.8 Capacity Management

- Predict and provision resources as needed (when, what type, how much) needed based on predicted workload of the system

3.2.9 Strategic Security Management

- Safeguard the security, integrity and confidentiality of the IT environment against unauthorized access, modification and deletion
- Achieved through security testing, security reviews and security incidents.

3.2.10 Business Continuity Process

- Managing normal continuous operation of the environment in the event of disasters that affect the environment.
- Business continuity involves risk identification (in terms of disasters), risk mitigation plans (to minimize the risk impact) and risk recovery plans to get the environment back to operation soon after a disaster occurs.

3.3 IT Infrastructure Library (ITIL)

- A framework or a set of ITSM best practices that focus on aligning business goals with IT development goals
- These best practices are not organization or technology specific, but are generic, used for establishing integration with the business strategy, generating value and maintaining basic competency.
- ITIL is a public-domain framework. Started in the UK in the 1980s, with around 40 volumes, current ITIL v4 (2019) is around 60 volumes long.

3.3.1 ITIL v3 life cycle

- **Service Strategy**: Understand business objectives and customer needs, provide strategic guidance for investments in services. Includes service value definition, business-case development, service assets, market analysis, and service provider types
- **Service Design**: Turn the strategy into a detailed plan that outlines the delivery of the service and business objectives
- **Service Transition**: Develop capabilities for introducing new services in an existing environment, relates to the "delivery" of services.
- **Service Operation**: Manages services in supported environments, provide best practice for achieving the delivery of agreed service level both to end-users and the customers. Includes Ops management, Service management, Service desks etc.
- **Service Improvement**: Incremental and large-scale improvements in delivered services.

3.3.2 ITIL v4

- Consists of 34 management practices subdivided into:
- **General management practices** including architecture management, measurement and reporting, risk management and project management
- **Technical management practices** which include Infrastructure and platform management and software development and management
- **Service management practices** like Availability management, Capacity management and performance management, Incident management etc.

4 DevOps

- DevOps is the combination of philosophies, practices and tools that increase the ability of an organization to deliver (i.e. deploy and support) effective software applications at high velocity.
- It is the result of Software Development and IT Operations working in synchronization
- The software devleopment team's activites are controlled by the SDLC, while the IT Operations team takes care of utilization of IT infrasturcture owned by the developing organization.
- DevOps aims to remove repetitive manual processes; these are automated as much as possible.
- DevOps follows the Agile principle of prioritizing individuals and interactions, over processes and tools.
- DevOps leads to faster delivery and deployment (maybe several times a day using CI/CD pipelines)
- The four common themes or pillars that are required for implementation of DevOps in an organization are:
  - Collaboration
  - Affinity
  - Tools
  - Scaling

4.1 Pillars of DevOps

4.1.1 Collaboration

- Working towards a common objective with the interaction and support of multiple teams and individuals
- Collaboration is based on:
  - Communication
  - Equal participation
  - Theory of Mind, which is the ability to recognize one's perspective and understanding that others have distinct perspectives based on their own context
- Collaboration necessitates relationships based on trust and empathy (empathy for different socio-cultural, economic and professional backgrounds, as well as different cognitive styles and professional goals/needs)
- Less established hierarchies, more supporting opportunities (mentorships, sponsorships)

4.1.2 Affinity

- The measure of strength of a relationship between individuals, teams, business units or even companies.
- Relationships are strengthened by navigating differing goals or metrics while keeping in mind shared goals, as well as creating empathy and learning between different groups.
- Affinity is **measured** using:
  - Shared time
  - Reciprocity of stories and support
  - Intensity of relationship
- Affinity is **built** using:
  - Shared values
  - Team cohesion
  - Strong and consistent team culture

4.1.3 Tools

- Tools drive change based on current culture and direction. They are the common language using which different teams communicate
- Usability of tools drives team culture, a tool must be usable by all members of a team in order to build cohesion and trust in the team.
- Examples of tools used in DevOps are unit testing tools, build tools, monitoring tools, tracking tools for issues etc.

4.1.4 Scaling

- Application of DevOps principles and pillars as organizations change in size and structure
- Involves technical and cultural considerations of operating at different scales
- Scaling could be for Organization, infrastructure, teams (hiring, retention, outsourcing), Complexity, Workload

4.2 DevOps Pipeline

4.2.1 Version Control

- Record changes to files stored within a repository that is shared by multiple developers
- Changes can be saved by developers using commit operations. Changes may be made by an individual or group of developers.
- Richer collaboration is provided by the ability to compare changes, merge changes and restore past versions of the repository.

4.2.2 Continuous Integration (CI)

- Merging branches of the repository owned by individual developers with the master branch as frequently as possible (multiple times a day)
- Multiple developers can checkout their own branch from master, make changes on their own branch, and merge their branches with the master concurrently.
- Merging of branches happens multiple times a day. Merge conflicts, if any, can be handled and the merge takes place. Metadata about each revision is stored by the system.
- CI is in contrast to big-bang integration which has a higher chance of leading to integration failures or merge conflicts, and hence causing build failure later on.
- Benefits of CI:
    1. Early error detection, reduced debugging effort and time
    2. Small and incremental integrations are easier to manage
    3. Increased visibility and communication

4.2.3 Continuous Build

- Building an executable application from the input code files.
- Involves static analysis (linting, data-flow or control-flow analysis), followed by actual build and sanity testing
- **Sanity testing** checks that the build was successful, includes all dependencies and is in a runnable state.

4.2.4 Continuous Delivery

- Frequent deployment of code to a production or test environment.
- Supports quick releases in a sustainable manner.
- CD can be triggered automatically by a trigger at the end of the build process.
- Supports faster time to market, and deployment on demand.
- Tools that support CD: Jenkins, CircleCI, GitLab, AWS CodeDeploy

4.2.5 Continuous Testing

- Executing automated tests frequently as part of the DevOps pipeline allows the code to be in a deployable state most of the time, while detecting bugs as early as possible.
- Tests are designed to execute with minimal wait time, and provide instant feedback and bug discovery/prevention.
- Tests involve static code analysis, validation of both functional and non-functional requirements
- Tests can be executed several times a day, using a trigger on the version control system (i.e. everytime someone pushes to the master branch, run the test suite)

4.2.6 Continuous Deployment

- A software release process that uses delivery mechanisms for deploying the validated product, immediately and autonomously to a production environment.
- Validated and integrated components are batched together and are then deployed into customer environment
- Removes the need for moving code between 2 environments, checking if it works as expected (typically error prone and resource-heavy process).
- Tools automate the entire deployment process, allows more focus on business golas than infrastructure overheads.

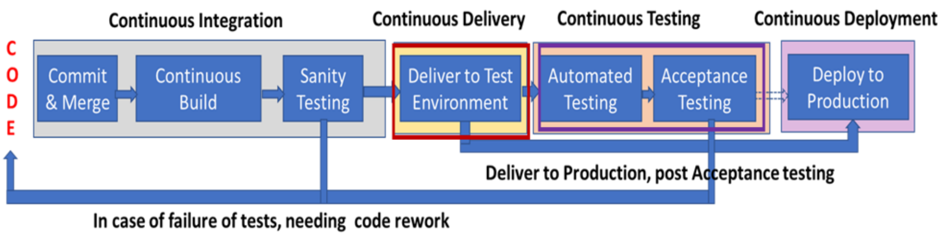

Figure 1: DevOps Pipeline

DevOps	**Agile**
A culture and approach which looks to remove the silos of Development activities of building a product and the Operations activities of deployment, support and upkeep.	A process that supports changes, ensures more collaborations between developers and other stakeholders, reduces the planning overhead and delivers products or part of products periodically
Focus on constant test and delivery	Focus on constant change
Target is end-to-end business solutions and fast delivery	Target is efficient software development
Operational and business readiness	Functional and non-functional readiness