Software Engineering

LECTURE 1: Software Engineering Trends and Challenges

Topics

- Introduction to Software Engineering
  - Trends in Software Engineering
  - Challenges in Software Engineering

Software Domain

- Software is more and more pervasive and it cannot be considered anymore as a minor element of a complex systems.

- In domains like cloud computing, big data, Internet of Things (IoT), Cyber-Physical Systems (CPS) it is the core element.

Software Domain

- Software does not require complex machinery to be developed, it can be created on personal computers that today are accessible to almost all people in the society.

- This gives the impression that it can be developed by anyone with good technical skills and willing to learn some simple-to-use programming language

Trends in Software Engineering

- Software Engineering methods and tools has evolved over past years from stand alone applications to a current trend of system of systems;
- These is mainly due to adoption to daily changes in user requirements and also technology advancement over years

Standalone Applications

- Applications that run on a local computer
- Normally they have all functionality and no need to be connected to network.
- Examples are; office applications, CAD programs, photo manipulation software and etc

Interactive transaction-based applications

- This applications are normally executed on a remote computer and can be accessed by users on their own PCS/terminals
- These include web apps, e-commerce, business systems, cloud based systems (mail, photos sharing)
- Apps normally generates a large amount of data that can be accessed and updated on each transaction

Embedded control systems

- These applications control and operates hardware devices
- Basically there are probably more embedded systems than other types of software
- Examples include software that control mobile phones, control of anti-lock braking system, and software that control home appliances and etc
- When more that one embedded system communicate, ie device to device communication lead to a field called IoT

Modelling and simulation systems

- These are applications that are developed for purpose of modelling physical process/situations
- They include separate interacting objects
- They are often computationally intensive thus require high performance parallel system for execution.

More on Trends in Software Engineering

- Agile Software development
- Big Data
- ANDROID COMPUTING

Trends in Software Engineering

- Agile Software development
  - Systems are becoming increasingly reliant on software due to needs for rapid fielding of interoperability, net-centricity, and rapid adaptation to change. The need for rapid adaptation and releases led to increased interest in agile methods of software development.

  - Project Management
    - Approaches like Scrum and XP accelerate project cycles require developers to interact with their managers more frequently but for shorter periods as daily contact is the norm in most agile processes.

Trends in Software Engineering

- Big Data
  - “Big Data” is famously known for software system utilize the Operational Data (OD) for software design and maintenance activities.
  - The structure and unstructured data in operational support tools have long been prevalent in software engineering field.
  - It will be necessary to develop basic principles and tools that allow having effective use of software engineering in OD system

Trends in Software Engineering

- ANDROID COMPUTING
  - Proliferations of Android device and application services have created demands that are applicable for software testing techniques
  - Previous research focuses on unit and GUI testing of Androids applications.
  - EvoDroid is an evolutionary approach for system testing in Android applications.

Trends on Software Engineering

- Requirement elicitation

Questionnaires/interviews ⇒ scenarios ⇒ use cases ⇒ ethnography ⇒ crowd sourcing

- Software development

Blockchain ⇒ Progressive Web app (PWAs) ⇒ IoT ⇒ Artificial intelligence ⇒ Mixed reality ⇒ cybersecurity

Challenges in Software Engineering

- The software design challenges regard the design and development of novel software engineering approaches, taking into account the transition from monolithic applications to applications developed based on the adoption of microservices-based paradigms and the need for novel approaches that facilitate service/applications composition.

Challenges in Software Engineering

- The software placement/implementation challenges concern the adoption of novel virtualization approaches and the development of software that can take advantage of them (e.g. IoT based software in the form of containers, distributed applications consisted of micro-services packaged as uni-kernels) and the associated security, distribution and delivery challenges.

Challenges in Software Engineering

- The orchestration middleware challenges regard the need for design and development of orchestration mechanisms able to facilitate optimal placement and execution of applications over programmable infrastructure, taking into account the programmability of the infrastructure as well as the reconfigurability of the applications' execution context

Challenges in Software Engineering

- The software quality challenges refer to the need for adoption of approaches that will facilitate development of qualitative software, such as collaboration driven software development and testing processes ensuring interoperability and user acceptance, e.g., introduction of novel software engineering tools, adoption of open-source solutions, quality assessment, revolutionary methods for collecting feedback prior - during and after the deployment, evolution of a validation and verification culture for any software related product or service, utilisation of available (customer) data for improving assumptions during design/development, cloud services interoperability frameworks.

Challenges in Software Engineering

- The services/applications lifecycle management challenges refer mainly to the need for interconnection of software engineering and DevOps functionalities/approaches and the support of continuous deployment approaches (e.g., tools for automated or semi-automated deployment of applications taking into account the existence of several components along with their requirements/constraints).

Future Challenges

- Process, Methodologies and Productivity

Future Challenges

- Process, Methodologies and Productivity
  - In the context of process, methodologies and productivity, existing concepts need to be redefined for meeting the current needs of the industry. Software process is a well-investigated research area, but nowadays there are several new advancements in technology and practice that introduce significant changes in this aspect. A new notion of productivity should be defined, where “lines of code” is not anymore the right measure of productivity, but software is measured in terms of its other qualities, usability, reliability, scalability. New possibilities to easily gather user feedback and monitoring information have the potential to enable an informed evolution of software while shorter development cycles call for novel software production methodologies to actually enable controlled management of such short development cycles. With DevOps there is also a need to shift deployment decisions and resource management from the deployment phase to the design phase of software engineering4, making efficient use of resources and supporting architecture level analysis, optimization of deployment decisions, as well as automated placement and orchestration of applications/services (e.g. specification of novel software development paradigms based on micro-services) and adopting infrastructure-as-a-code approaches for eliminating the needs for configuration placement and management over programmable infrastructures, with an emphasis on their utilization from small companies.

Future Challenges

- Application Contexts
  - Software is also the driving force behind the CPS and IoT paradigms, and their further evolution is heavily linked to the ability of software to be both dependable and adaptable to real time changes, thus enable different Application Contexts. The main challenges raised by IoT-enabled CPS include the development of models, methods and design tools for IoT/CPS-enabled applications going beyond formal methods research to create abstractions and formalisms for constructing and reasoning about systems with diverse and more difficult-to-characterize components. Moreover, the needs brought by CPS for novel methods of software Adaptability, Scalability and Maintainability, which are not envisaged at design time as such systems, need to be continually modified and maintained to meet changing requirements, run time adaptation of software Quality Assurance in large scale open CPS environments able to deal with uncertainty and variability at the same time, software-awareness of hardware to ensure Web-scale performance, flexibility and agility and meet the “software-defined anything” paradigm.

Future Challenges

- Design Patterns Development for a Systems of Systems Approach
  - As suggest above, and having in mind the different application context, there is also a need to further research into design patterns development for a systems-of-systems approach. New patterns at the architectural level describing the obligations/constraints to be fulfilled by the system in which the software is running, and to validate and standardize them are needed and methods on how to apply them into a dynamic, ever-changing context environments6. As such, issues such as frame of references, unifying lexicons, visualisations, design architecture and interoperability, modelling languages, tools integration and simulation and analysis should be tackled.

Future Challenges

- Quality Guarantees
  - Design patterns, as discussed previously, will allow software to reach a better level of quality. However, they alone are not enough. The rapid growth in the last years of agile delivery methods in the context of DevOps, as well as the need to reduce the development time as much as possible, call for research approaches that can increase the anti-fragility of systems, reduce the meantime-to-restore-service (MTRS), and develop accelerated methodologies to test quality through staging and canary testbeds. In parallel, although Big Data offers the ability to capture large amounts of monitoring data on the behaviour of an application, limited progress has been achieved in developing feedback analysis tools, thus further research is envisaged in the architectural level, in the ability to pinpoint specific root causes of performance degradation in the application code, and in the application of machine learning methods to quality engineering7. Last but not least, there is a shortage of standardized reference quality benchmarks for code and extra-functional properties in many classes of applications and domains.

Future Challenges

- Requirement Engineering
  - One other core aspect of research, directly linked to any software engineering activity is that of requirements engineering. Going away from monolithic and stand-alone applications and adopting a Digital Single Market and Connected-world mentality increases complexity of knowledge capturing and representation. New devices, services and even individuals become part of a software-powered ecosystem and flexibility, constant evolution and interconnection contradicts current requirement engineering outputs, as existing approaches do not account for dynamicity of use and unknown requirements. There is the need for a radically divergent approach to capture emerging behaviour from systems and users. Emerging technologies and trends are shedding light on potential research topics such as multichannel big data analytics for requirements elicitation from large scale sites8 (like smart-city infrastructures which blend humans, machines and generally system characteristics and behaviour), novel methods for user engagement towards directly extracting requirements, privacy respective indirect requirements extraction paradigms exploiting context-awareness of individuals independently on the usage of a specific software, Human-Machine interface types taking into account CPS and new technologies that blend human and computer interactions and decisions, different kind of logics (both rational and behavioural), interconnection and interoperability with next-of-kin and other unrelated (at first sight) systems of a greater ecosystem, placement of requirements into production schedule dictates and relation with emerging business needs, unanimous description conventions and abstraction representation levels.

Future Challenges

- Privacy and Security by Design
  - Privacy and security at design time as well as runtime of software is another important aspect that should be tackled to comply with the evolution of software development methodologies (e.g. microservices based software engineering approaches) along with the placement of applications over virtualised environments in multiple formats (e.g. virtual machines, containers, unikernels). These issues have become highly critical since the the threat and vulnerabilities landscape is continuously expanding. Special care with regards to privacy and security has to be given in complex distributed systems that in many cases have to handle big data volumes in a distributed way. Challenges include the identification of contextual systems' patterns related to privacy leaking code snippets, secure computation of data structures, approaches for establishing optimality of encryption levels, continuous source code assessment at design time as well as vulnerability assessment of the developed applications9, secure packaging and placement mechanisms of the developed applications over programmable infrastructure, orchestration mechanisms supporting the secure and efficient policy-aware management of services and applications, real-time risk identification and assessment techniques along with the triggering of the appropriate mitigation actions. Special emphasis should be given in the topic of security and privacy by design software engineering approaches that contribute in the generation of software artefacts that can operated in multi-IaaS environment with increased security characteristics.

Future Challenges

- Big Data for Software Engineering
  - As datasets handled by software are constantly increasing, apart from supplying novel algorithms, new system architectures and software infrastructures able to cope with the 5Vs of Big Data, it's high time for software itself to benefit from the intelligence extracted from large sets of information such as software source code, commits and forks, bugs, warnings and notifications, issues from backtracking systems, logs of any kind, commits, demographics, coding patterns, requirements, user behaviours, user profiles, etc. Research challenges for software engineering in this direction include novel tools employing techniques of machine learning and data mining to reveal hidden knowledge aspects and extract information from sensor-based architectures, excavating knowledge which is impossible for humans to dig out, but is necessary to be brought into human attention and affection for improving software qualities, studying the evolution / discontinuation of application frameworks, open source components, analysis of user trends and preferences and behaviour with systems to better understanding users' needs, tools and methods for identifying feature and performance improvement opportunities, identifying root causes of failures and system halts based on log files (massively big (>>GBs) or lightning-fast updating) coming from various complex distributed systems and infrastructures10, insights collected at runtime on symptoms and context changes triggering adaptations, and perform predictive and prescriptive analytics for proactive planning and preparation of adaptation actions.

Future Challenges

- Finally
  - Finally, as a great proportion of the software developed today is of Open Source, there is a continuous need towards accelerating Open Source Software Innovation. Many projects lack proper community engagement and management structures, quality assurance, and a vision on how to contribute to the European open digital market. OSS governance includes a number of technical challenges related to software engineering and production processes, including methodologies and tool support for the detection and disposition of contradictions, ambiguities, and gaps in requirements specifications, decoupled architectures and production processes based on fault defensive and tolerant programming styles for distributed developers teams with different skill sets, interests and motivations, methodologies and tools for impact analyses of code additions and modifications11. Furthermore, OSS production processes also include organisational challenges that have to be met by an interdisciplinary approach aiming at the creation and management of communities of code contributors, reviewers, testers, first level users, etc. and a comprehensive development and communication approach combining existing tools under a set of common, formalized set of methodologies

The End