SOFTWARE ENGINEERING
FOR SERVICES AND APPLICATIONS:
CURRENT AND FUTURE CHALLENGES

SE4SA CLUSTER CONTRIBUTION TO
THE 2020-2027 RESEARCH AND INNOVATION ICT WORK PROGRAMME
Date: 5.11.2017
Executive summary

ICT and, in particular, software is more and more pervasive. It is affecting our lives, but also business, manufacturing, agriculture, health, and other fields in a way that could have never been even imagined a century ago. At the same time, keeping software engineering processes under control and ensuring that software is actually fully trustable still requires a significant amount of work. Retrospectively, we can report several cases where the lack of well-established approaches has caused significant problems.

The last prominent case has happened very recently, in February 2016, when the Japanese Hitomi spacecraft broke off for a problem that was caused by a number of software failures. Other recent issues concern British Airways that suffered an incident that stopped Gatwick and Heathrow airports. Even if British Airways claimed that it was due to a power failure, data center experts are doubting such argument as they did have available backup powerage.

Still, software availability greatly simplifies and enable our human activities. An interesting example is given by Tesla that, upon request of a customer in Florida, could be able to boost, through a free software update, the battery capacity of all those living in the same area to give them the possibility to escape from Hurricane Irma.

Among the others, outstanding application fields for software concern: i) any kind of smart space, including, in particular, industrial environments where an advanced digitalization could be dramatically beneficial; ii) autonomous driving, where machines would be required to take autonomous decisions in particularly unstructured environments controlled by a high number of variables (think of a street close to a residential area where a large number of kids walk from home to school and vice versa at some specific hours of the days) and we should be able to certify the software that is behind this decision making process; iii) the Internet of Things context, where the presence of massively replicated, autonomous and embedded software requires that we understand how to ensure proper behaviors, proper collaboration and self-adaptation between the various software parts; iv) societal computing, where aspects concerned with privacy, management of socio-technical ecosystems, sustainability of software, governance patterns, machine to society interaction are to be analyzed deeply and shaped based on the needs of today's society.

In these contexts, software engineering needs to consolidate and identify new approaches and tools to guarantee that software is trustworthy, exploits effectively today hardware and middleware technologies, can be certified, is developed by involving end users, is reliable, robust, performant, and many others (see Figure 1). This requires the enhancement and evolution of processes, tools, frameworks, and programming models. In particular, the following questions become particularly important and critical:

- Which programming models and design approaches are needed to build trustworthy software?

  New advances in hardware and infrastructure software are offering very powerful tools to start building complex, intelligent and autonomous systems able to cope with the growing needs of new application fields, from smart environments, where software is pervasively controlling complex physical contexts, to autonomous driving, which requires a significantly high level of

Software Engineering for Services and Applications
software intelligence and trustworthiness. These advancements, however, have still to be translated into consolidated and reliable programming, design and verification approaches able to guarantee the important properties of software that the application domain requires. In particular, new approaches to fully exploit the massive parallelization of hardware, the interplay between edge devices and cloud (fog computing), the possibility of relying on quantum computers for high complexity computations, the possibility of creating collective and autonomous systems (swarming intelligence) starting from simple components, should be one of the main objectives of future research.

- **Which verification and validation approaches are needed to certify the properties of trustworthy software?** While the problem of verifying and validating software is an old one, the increasing dependency of our lives and business from software requires the development of more solid approaches that finally lead to the certification of software in multinational and multi-regulation contexts. Analysis of big log and testing data as well as learning techniques will have to be adapted to this context and blended with in depth knowledge of the reference application domains for the considered software.

- **How to ensure that trustworthy software is sustainable (energy efficient)?** In a context where software is continuously running and offering important services to the society, its sustainability and energy efficiency becomes a critical aspect to be considered during the software design phase and fine-tuned during the execution phase.

- **What technologies are required to be able to realise smart applications and environments?** As we have already mentioned, technologies for computing have advanced quite significantly and now include things such as fog and quantum computing, in what we can call a computing continuum. In this context, infrastructural technologies that enable users to exploit such a continuum in a seamless way are needed. Execution containers as well as monitoring and self-adaptation mechanisms become important assets and require an in depth investigation.

- **How can humans participate in building trustworthy software?** Research has shown that user involvement is a major success factor for software projects. Involving citizens in the functional design of the systems of their interest has the potential to guarantee high service quality and user satisfaction. Thus, methods and processes for a democratic mass participation in software development need to be developed. Users of software systems, acting as non-professional software developers, should be supported in the creation, modification or extension of a software artefact.

- **How to define a safe interaction between humans and autonomous machines?** In a context where humans are surrounded by autonomous machines and robots, it is of paramount importance to understand how to let the two interact in a proper way. On the one side, we need to understand what kinds of tools we should offer to humans to program autonomous machines and how to program them, respecting their autonomous capabilities. On the other side, we need to ensure that machines behave “appropriately” in their interaction with humans. This is a multidisciplinary area where software engineering is required both to revise the way
design and programming paradigms are conceived to enable programmability of autonomous machines and to provide analysis and testing approaches to enable the verification and certification of autonomous machines.

In the following of this document we present in further details the issues listed above, motivating their importance and the potential effects of not taking them into the due consideration.

This work is the result of the ongoing discussion among the members of the Cluster on Software Engineering for Services and Applications and aims at constructively contributing to the development of a new European research agenda on software and software engineering issues.
Contributors

Elisabetta Di Nitto - Politecnico di Milano, DICE project
Orlando Avila-García - ATOS, MegaMart2 project
Cristina Chesta - Santer Reply, HyVar project
Davide Fucci - HITeC, OpenReq project
Yosu Gorroñogotia - ATOS, SUPERSEDE, STAMP projects
Patxi Gortázar - URJC, ElasTest project
Geir Horn, MELODIC project
Andreas Ocklenburg - cloudSME, COLA project
Leire Orue-Echevarría - TECNALIA, DECIDE project
Anna Perini - FBK, SUPERSEDE project
Clara Pezuela - ATOS, TANGO project
Pierluigi Plebani - Politecnico di Milano, DITAS project
Vlado Stankovski - University of Ljubljana, SWITCH and ENTICE projects
George Suciu - BEIA Consult, SWITCH-project.eu
Angelo Susi - FBK, SUPERSEDE project
Zhiming Zhao - University of Amsterdam, SWITCH project

Moderator

Odysseas Pyrovolakis - European Commission

Projects involved in the cluster

Aligned, AppHub, ARCADIO, Artist, CloudTeams, CloudWave, COLA, DECIDE, DICE, DITAS, ElasTest, ENTICE, Envisage, HyVar, MELODIC, MODAClouds, MONDO, OpenReq, Prowess, Q-Rapids, RISCOSS, SeaClouds, S-CASE, STAMP, SUPERSEDE, SWITCH, TANGO, U-QASAR

More information about the cluster is available here:

https://eucloudclusters.wordpress.com/software-engineering-for-services-and-applications/
TABLE OF CONTENTS

1. Introduction .................................................................................................................................... 8
2. Outstanding software application domains .................................................................................... 9
3. Challenges and trends for the next decade .................................................................................. 11
   Which programming models and design approaches are needed to build trustworthy software? 13
   Which verification and validation approaches are needed to certify the properties of trustworthy software? .......................................................................................................................................... 15
   How to ensure that trustworthy software is sustainable (energy efficient)? ................................. 17
   What technologies are required to be able to realise smart applications and environments? ......19
   How can humans participate in building trustworthy software? ..................................................... 21
   How to define a safe interaction between humans and autonomous machines? ........................... 23
4. Conclusion ..................................................................................................................................... 24
5. References .................................................................................................................................... 25
6. Appendix A: Contributions from the cluster projects to the definition of challenges .......... 26
   Support to the life-cycle (development and operation) of complex software ......................... 26
   Tools to facilitate the development of resilient, secure and self-adaptive software .................. 26
   Formal methods for taming the complexity of software systems .............................................. 28
   Cognitive Model-Driven Engineering ......................................................................................... 30
   AI Process Automation ................................................................................................................ 33
   Management principles for software fragments, executable images of Virtual Machines and containers ..................................................................................................................................... 36
   Security and privacy ....................................................................................................................... 37
   Development of Secure Software ................................................................................................. 37
   Privacy-aware software engineering ............................................................................................. 38
   Verification, testing and compliance issues ................................................................................... 40
   Functional testing for native-cloud Apps ..................................................................................... 40
   Context-aware testing in the field ................................................................................................. 41
   Integration of (safety critical) systems in a multi-national regulation ......................................... 42
   Dependability ................................................................................................................................ 44
   Software Reliability in Large Scale Distributed Cloud ................................................................. 44
   Collective Self-Adaptive Systems ................................................................................................. 45
   Design and programming approaches ......................................................................................... 49
1. **Introduction**

ICT and, in particular, software is more and more pervasive. It is affecting our lives, but also business, manufacturing, agriculture, health, and other fields in a way that could have never been even imagined a century ago. According to Shackelford & Jankowski [1], economies such as Taiwan, South Korea and USA spend a significant amount of their R&D investments in ICT, with Taiwan (more than 70%) and South Korea (more than 50%) mostly focusing on hardware, and USA (more than 40%) on both hardware and software. According to the same authors, the situation in Europe varies from country to country, but in all cases ICT is significantly less prominent with about 20% of R&D investments for France and Germany.

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 willingness to learn some simple-to-use programming language. At the same time, its intangibility makes it invisible and, thus, suggests that it is only a minor part of the devices it is controlling, while, in many cases, it is a core part of it. For these reasons, historically, there has been a tendency to direct investments and attention to the devices rather than to the software itself, and to assume that software development and operation approaches and tools should have been defined within specific devices, so to address the specific problems of specific application domains. Also, in many cases, the developers of such approaches and tools as well as of the software itself were application domain specialists rather than software engineers. This is especially true for scientific software, as discussed on Nature, where Merali [2] points out at a number of issues generated by this tendency.

Retrospectively, we can see that the lack of mature software engineering expertise or practices have caused software bugs that, in turn, have determined loss of significant quantities of money in the best cases. The last prominent case has happened very recently, in February 2016, when the Japanese Hitomi spacecraft broke off for a problem that was caused by a number of software failures [3]. Other recent issues concern British Airways that suffered an incident that stopped Gatwick and Heathrow airports¹. Even if British Airways claimed that it was due to a power failure², data center experts are doubting such argument³, as they did have available backup powerage. This situation, however, has unveiled a debate on outsourcing (common practice in large companies to cut costs) as well as on investment cuts in software development⁴. In 2014, the UK went under the same issue when a claimed software bug caused the UK airspace to be closed for several hours⁵. These incidents clearly show that research and investment in technologies to improve the way in which software is developed and operated is needed.

---

¹ [https://www.economist.com/blogs/gulliver/2017/05/going-nowhere](https://www.economist.com/blogs/gulliver/2017/05/going-nowhere)
³ [https://www.theguardian.com/business/2017/may/30/british-airways-it-failure-experts-doubt-power-surge-claim](https://www.theguardian.com/business/2017/may/30/british-airways-it-failure-experts-doubt-power-surge-claim)
As researchers in software engineering have been highlighting whenever possible, high quality software requires specific skills and the adoption of good and controlled development and operation practices and processes. Software engineering has the mission to offer the right tools and methods to guide users in all activities connected to the lifecycle of software and of the services created from software, through the usage of technologies and new paradigms, still ensuring productivity of processes and quality of software (fit for purpose, performance, availability, evolvability, reliability, security, or compliance, to name a few). In the last years, advancements have led to an increasing automation of aspects such as mining users’ reviews to elicit requirements [4], perform testing, deployment, management of new software releases [12], and, at the same time, have allowed researchers and practitioners to identify new approaches for creating and operating software and services (think of DevOps [5] as an example). However, there are still many challenges to tackle. Some of them are old ones that we still need to address. Others derive by the emerging of new problems, new application domains and new technological advancements. In [6] we have highlighted a first set of short-term challenges. In this document we try to identify others with a longer time horizon in mind. The aim is to constructively contribute toward the development of a new European research agenda focusing on software and software engineering issues.

The aim of this document is to present the challenges that the members of the Cluster on Software Engineering for Services and Applications [7] consider to be crucial for the years to come. The presented analysis is based on the contributions of the cluster members and on the inputs they have collected from other researchers in the field.

The document starts with the identification of the novel application domains that appear to challenge software engineering the most (Section 2), then presents the main challenges and trends we have identified as the result of the synthesis of various contributions (Section 3) and, finally, draws the conclusions (Section 4). The initial collection of contributions from which we have started our analysis is presented in Appendix A.

2. OUTSTANDING SOFTWARE APPLICATION DOMAINS

While software is not evident nor tangible, it is at the center of the modern society and production plants. Software defects can create today significant problems and loss of money and even human lives. At the same time, software availability greatly simplifies and enable our human activities. An interesting example is given by Tesla that, upon request of a customer in Florida, could be able to boost, through a free software update, the battery capacity of all those living in the same area to give them the possibility to escape from Hurricane Irma6.

Among the others, outstanding application fields for software are the following:

---

- **Smart specialization platforms** and **Industry 4.0**: These are areas where the European Union is making significant investments with the aim of increasing the effectiveness of our factories, the organization of smart cities and wellbeing of citizens. In these contexts, important software issues concern data fusion, management of big data, machine to machine interaction, and machine to human interaction. On this last point, we highlight that, till a few years ago, the focus was on improving the way humans could interact with machines that were mostly passive and reactive to human actions. Now the interaction tends to become more and more bidirectional as autonomous machines and robots are being created. This poses the problem of how to ensure that machines behave “correctly” in the interaction with humans and how to allow humans to program the behavior of machines without negatively impacting on their autonomous abilities.

- **Autonomous driving and Artificial Intelligence**: Today we already have undergrounds functioning without drivers that are in operation in many urban areas. In the future, the aim is to extend this idea to other transportation means, in particular, cars and trucks that have to perform their task in a significantly less controllable context. This opens up a large quantity of challenges that include image recognition issues, ability to sense a very large quantity of stimuli from the external environment, ability to process stimuli and take decisions within hard time constraints, interaction with sensors and actuators, and so on. Also in this context, the issues concerned with machine to human interaction become of paramount importance, and so are the approaches for certifying software behavior and also the lifecycle of software. For instance, in a context where overnight a new software component is installed on a car, we need to make sure that this is done considering all possible harmful circumstances.

- **Internet of Things**: the Internet of Things (IoT) is an enabling technology for the creation of smart environments in many different application domains. Also, it is posing by itself some important challenges to software engineering. One of the most notable ones is that the presence of massively replicated, autonomous and embedded software within IoT requires that we understand how to ensure proper behaviors and proper collaboration and self-adaptation between the various software parts.

- **Social and Societal Computing**: According to Carnegie Mellon University, societal Computing is the branch of computer science that designs computational technology to shape tomorrow's digital world and uses computational methods to understand the societal challenges a digital world poses. It seeks to discover the scientific principles and laws governing the relationship between the design of computing technology and the nature of digital society. In this context, aspects concerned with privacy, management of socio-technical ecosystems, sustainability of software, governance patterns, machine to society interaction are to be analyzed deeply and shaped based on the needs of today’s society.

---

7 [http://s3platform.jrc.ec.europa.eu/home](http://s3platform.jrc.ec.europa.eu/home)
3. CHALLENGES AND TRENDS FOR THE NEXT DECADE

The purpose of current and future software engineering is to build software that we can trust and that we can exploit to support the application fields we have presented in the previous section and to enable new applications in these fields. Figure 1 shows a map that highlights the relationship between trustworthy software and a number of aspects that are relevant in the identification of challenges and trends for future software engineering.

More in detail, the development of trustworthy software requires new research aiming at building effective processes and tools. Among the others aspects, the areas that we should certainly investigate concern: i) the technological infrastructures we make available for software development and operations (cloud, fog, parallelization frameworks like Spark or Flink, combination and extension of these, ...), ii) the design and programming models these offer and their sustainability in the context of multi-disciplinary teams, and, more and more important, iii) the new style of human participation in the software development and evolution process. While so far users and developers had specific and differentiated roles, now the role of users is becoming closer and closer to the one of developers. End-users development has became a discipline per se and requires the presence of robust and well-designed tools to enable end-users to program specific behavior of software without creating potentially dangerous cases.

The properties that a trustworthy software should have and that processes and tools should check and guarantee are many. Among the others, correctness, compliance, reliability, availability, performance, safety, security, maintainability and evolvability are the classical ones, which are now side by side with others such as privacy of data, energy efficiency, sustainability of software, and certified interaction with humans. The importance of this last property is increasing together with the level of intelligence and autonomicity of software: the more software is able to actuate actions in response to cognitive decisions, the more we need to discipline the way decisions are made and actions are taken. An outstanding case today is autonomous driving, where the decision on whether to give priority to the life of passengers as opposite to the life of the others, for instance, those walking on the street, is raising a very complex debate.

Of course, the development, verification and validation of trustworthy software showing some of the listed properties and exploiting new design and programming models, new technologies and offering new styles of human participation, requires effective processes, of which DevOps appears at the moment an interesting instantiation, and effective tools and techniques supporting all aspects of the software lifecycle.
Figure 1. Important aspects associated to software.

Within the context of the map in Figure 1, we see many open questions and challenges to be addressed (see Appendix A for an incomplete list of challenges that have been contributed by the cluster participants). In the following of this section we elaborate on a number of questions that summarize the main challenges we have identified so far. These are:

- Which programming models and design approaches are needed to build trustworthy software?
- Which verification and validation approaches are needed to certify the properties of trustworthy software?
- How to ensure that trustworthy software is sustainable (energy efficient)?
- What technologies are required to be able to realise smart applications and environments?
- How can humans participate in building trustworthy software?
- How to define a safe interaction between humans and autonomous machines?

For each question, we provide a short description of it and then we discuss on the consequences in case the challenge is not addressed as well as on the impact the question has on technology, society and European leadership.
WHICH PROGRAMMING MODELS AND DESIGN APPROACHES ARE NEEDED TO BUILD TRUSTWORTHY SOFTWARE?

Description

New advances in hardware and infrastructure software are offering very powerful tools to start building complex, intelligent and autonomous systems able to cope with the growing needs of new application fields from smart environments, where software is pervasively controlling complex physical contexts, to autonomous driving, which requires a significantly high level of software intelligence and trustworthiness. These advancements, however, have still to be translated into consolidated and reliable programming, design and verification approaches able to guarantee the important properties of software that the application domain requires. Examples are the following:

Massively parallel computing: today most computing systems are multi-core, but such feature is more used to instantiate several independent virtual machines rather than to enable smart parallel computations where the cores cooperate with each other to achieve a common goal. In most cases, in fact, the need for cooperation between software components is addressed at a higher level of abstraction allowing components to interact and coordinate with each other through the communication network rather than through the shared memory features offered by multi-core machines. The reason for this is that building a complex system in a multi-core context is a challenging and error prone activity that is carried out by specialists only for specific kinds of algorithms. One of the future challenges for software engineering is therefore to identify a proper programming model and corresponding supporting tools to simplify the development and verification of software with massive parallelism thus allowing average programmers to exploit the characteristics of modern computers.

Fog computing: Fog Computing is emerging as a paradigm aiming to create a continuum between the Cloud and the Edge of the network. While the Cloud provides a theoretical unlimited amount of resources and the Edge a low-latency computing power, Fog aims to seamless integrate these two environments, so to take advantage of the strengths from both.

Although researchers have started studying the implication of this new paradigm in terms of infrastructure and data management, the need of a specific programming model has not yet been addressed. If on the one side, microservices (or functional programming) could fit in this new heterogeneous, scalable, dynamic environment as the Fog is, the implication of these approaches need to be further studied in terms of performance, security, and privacy.

Quantum computing: Quantum computers exploit quantum-mechanical phenomena to perform operations on data. This approach has been experimented so far on a small scale but is proven to be successful in solving in polynomial time some NP problems. Its most prominent application is in cryptography, but, more generally, it is able to tackle optimization problems where traditional space exploration requires a significant amount of computational power by traditional Turing machine-based computers and can easily lead to local optima. The quantum approach allows us to quickly
Software Engineering for Services and Applications

find the optimal solution as well as a number of good alternatives to it. Application domains could range from optimization to machine learning, to Montecarlo simulations. Luckily, even if the field is still in its infancy, large companies like Google and IBM are aiming at developing quantum computers with a significant number of qubits that should demonstrate the advantages of this technology [8]. Some programming models exist but they appear to be suitable for researchers rather than for average programmers. The challenge is therefore to create development approaches, verification tools, languages and environments that enable the usage of quantum computing as part of complex software development activities.

**Collective Adaptive Systems:** The control of global behavior emerging from complex systems is a growing concern. Technologies such as Internet of Things (IoT) and Cyber Physical Systems (CPS) are expected to be the basis for the so-called collective adaptive systems (CAS). By nature, those are typically complex systems that tend to display emergent (bottom-up, rather than top-down/prescribed by design) behavior. Regardless of whether this emergence is intended or unintended, we must be able to keep control on it. However, complex systems with emergent behavior are, by definition, difficult to control as small changes in one element can lead to huge changes at the system level. Therefore, it would be important to devise principles, software development methods and reference architectures for realizing emerging and effective self-adaptation.

**Consequences in case the challenge is not addressed**

Not all aforementioned paradigms and technologies show the same level of maturity, but all of them represent different facets of a complex computing machinery that will enable dramatic advancements in the execution of complex reasoning. Losing the possibility to develop research and innovative solutions in these areas will result in the impossibility to increase the cognitive abilities of software thus excluding the possibility to develop advanced solutions in challenging application areas such as smart environments, autonomous driving and the like.

**Impact on technology**

Development of novel approaches and tools in the areas we have identified will determine a dramatic technological advancement and will enable large scale development and adoption of applications that at the moment are still at a prototypical stage.

Moreover, a technological challenge that will have to be considered consists in the integration among different computation approaches and models.

**Impact on society**

Society will certainly benefit from the application scenarios that will be enabled by the advancements in this area. Areas such as the organization, control and evolution of complex smart environments (industries, healthcare institutions, homes etc.), data science, autonomous driving, future Internet, will show significant progresses thanks to the improvement of design and programming models targeting the novel hardware architectures and middleware we have mentioned.
Impact on European leadership

Investments in these areas are high risk but could give back important results. Support by the European Union could help enterprises in taking the risks, in strengthening the collaboration with European research laboratories that are already active in these fields and in nurturing a new generation of researchers and engineers focusing on addressing high priority problems for our society. As for quantum computing, the European Union is going to grant significant funding to the development of quantum computers through the Qflagship program. From the information that are available, however, the emphasis of the current funding program is on the development of the quantum computer hardware rather than into the development of environments supporting the development of quantum computing software. Supporting the work in this area will be certainly complementary to the planned programme and will allow quantum computing to become more usable and effective.

WHICH VERIFICATION AND VALIDATION APPROACHES ARE NEEDED TO CERTIFY THE PROPERTIES OF TRUSTWORTHY SOFTWARE?

Description

More and more often software systems are influencing everyday life. These software systems become more complex over time. In addition, software is scattered across different devices, and applications are no longer a monolithic asset, but instead they are a composition of smaller components or services. This poses a big challenge when it comes to certifying the properties of trustworthy software.

An example of software systems exposing this kind of properties are the safety critical applications installed on railways, planes or health care devices. One of the major problems in this case is the capacity of checking the design and the requirements of these systems to early discover serious flaws and, at the same time, to validate their compliance with national and multinational norms and regulations. Very often these norms differ from country to country leading to serious issues when an integration of such systems has to be performed at an international level, as, for example, happened during the integration of high speed train control systems in the EU multinational area. This kind of large scale integration requires an intense human and computer decision-making and validation effort. This task should be performed following well-structured software engineering processes that allow to systematically check the specification of the systems against the different kinds of rules in the different countries and evolve these specifications according to the changes in these rules also exploiting conceptual modelling methods and AI based reasoning techniques such as Machine Learning or logic based reasoning.
In the case of **dynamic systems** composed by the interaction of different services and components, it becomes infeasible to reproduce in testing environment the many possible configurations, parameters and interactions happening on line [9], therefore for validating the adequacy of a system to given requirements, both functional and non-functional, testing must be moved to production and carried out in continuous way along with monitoring [10]. Hence, certifying properties of **trustworthy software requires new testing approaches**, including monitoring, introspection, log ingestion and analysis (big data), but also **new ways of using formal methods** and **new Artificial Intelligence techniques** to detect, from the big amount of information generated when running the tests, if there is any substantial difference between tests run, to detect early any regression symptom. Ensuring applications work correctly will require machine learning techniques mixing natural language recognition (for logs) and pattern recognition (for metrics) to provide a high level overview of the system as a whole, possible points of failures, bad-behaving components, among other features.

**Consequences in case the challenge is not addressed**

Verification remains the most costly and complex activity in software development. Current practices require about 35% of the total IT cost (with 40% predicted for 2018 and 2019) [11]. These costs will increase with the need to verify dynamic complex software systems. The challenge of doing adequate verification before a system is deployed cannot be met due to the dynamism and ultra large-scale of modern ICT systems. On the other hand, ICT systems are pervasive and if verification is not adapted to new paradigms as cloud, continuous integration, or DevOps, society will be subject to consequences of failures. The consequence to not having a precise Software engineering methodology to pave the way towards certification of software could determine delays and flaws in the integration phase, leading to delays in the use of the integrated services for the citizens, so compromising the capacity of EU to early expose the products of the research and technology transfer in EU area and worldwide.

**Impact on technology**

(Semi)-automated methods for regulation representation and reasoning techniques to perform check and validation will have to be identified. New solutions allowing the generation, launch, and control of tests for deployed systems will have to be developed. These will have to be tailored to the specific objectives and profile of usage. These facilities should be easy to use and automated, allowing clients and end-users to administer verification activities according to their purposes. Solutions for mitigating side-effects on actual usage should be conceived.

**Impact on society**

The solution of this challenge for different kinds of systems, and in particular safety critical systems, is relevant in several fields such as transportation, healthcare and other crucial services for EU citizens. The same solution should be generalised to the case of a set of systems/services that adhere to national/local rules/specifications and that have to be integrated together in order to satisfy new regulations/specifications minimising the effort and the costs related to this transformation and to minimise the time to market of the revised solutions. This should enable the
citizens in having a faster access to integrated services. Furthermore, improved security and reliability are achieved, and overall trust of society in ICT systems is increased.

**Impact on European leadership**

Europe is the perfect area for studying and applying the outcomes of this research when it comes to national/local rules due to its current multinational structure and to the existence of governmental bodies that release regulations and specifications affecting the specific regulations and specifications of the different countries and at the same time requiring integration among them. This panorama should allow to design methods and tools that should constitute a model to be proposed to other entities and institutions around the world so increasing the European capacity of taking leadership in this critical area.

Europe has also a long tradition in testing, and it has consolidated research groups in testing in different countries. Some big companies in Europe (like Airbus) do need good testing practices, and Europe is leading the Industry 4.0 revolution, where new testing approaches will be needed in a world of interconnected devices. There’s a chance for leading the new approaches for certifying these modern highly distributed applications.

**HOW TO ENSURE THAT TRUSTWORTHY SOFTWARE IS SUSTAINABLE (ENERGY EFFICIENT)?**

**Description**

Sustainability of software services can be achieved by embedding energy efficiency and other non-functional requirement (NFR) optimization principles in software. The resources we have available are not unlimited. This is evident particularly with the rise of the Big Data problem. Thus, our software cannot and should not be made to use just all necessary resources, but, it should always strike balance between various requirements, such as performance versus energy efficiency. Moreover, different users have different requirements. For some, Quality of Service (QoS) and performance are important, for others, low-cost and energy-efficient operation. It is therefore necessary to embed automated mechanisms and adaptation strategies in software so that various optima can be reached.

Achieving such balance is even more important with the rise of various battery run devices, such as cars, and community systems of Renewable Energy Sources (RES) that can be used to refill such devices, in order to achieve sustainable operation. Thus, reaching a balance between NFRs for every particular case is important, and automated mechanisms for energy efficiency and other NFR management are still necessary.
Consequences in case the challenge is not addressed

Currently a lot of software is running in idle mode, and a lot of hardware too. Great improvements of the utilization of the computing resources are still possible. This would assure that generated energy is not spent in vain. It is our moral right and duty to embed such sustainability principles in software. Using new software technologies for energy efficiency of software services would improve the competitiveness of European companies against the rest of the World.

It is a common prediction that the computing requirements for smart IoT environments will grow in nearby future. This would increase the carbon footprint of European software, unless new methods, approaches and technologies are invented for energy-efficient software.

Impact on technology

Great improvements on the carbon footprint of software, through various smart new software engineering methods and technologies are still possible. Smart environments may be represented by various cars, robots, sensors, smartphones and similar.

New energy efficiency approaches for software could make an impact on the hardware industry. For example, algorithms for various low-precision, low-power calculations can be further implemented in hardware.

Impact on society

Current software engineering methodologies and tools seldom focus on energy efficiency. Society at large can benefit indirectly, via the reduction of the carbon footprint. Energy efficient devices will enlarge battery life, which is crucial for smartphones on extreme situations or for increasing the autonomy of hybrid/electrical cars that impacts on the carbon footprint.

Impact on European leadership

Energy efficiency is an area in which various countries such as the USA and Japan compete with Europe. Tackling these aspects of applications from software view point, rather than from hardware view point, would be novelty and would reinforce the European leaderships in the software services domain.
WHAT TECHNOLOGIES ARE REQUIRED TO BE ABLE TO REALISE SMART APPLICATIONS AND ENVIRONMENTS?

Description

We need to aim at developing technologies for the computing continuum. This continuum ranges from Internet of Things (IoT) devices to large computational machines.

With the rise of the IoT there is an increasing demand for the development of various smart environments. IoT generate a huge Big Data problem, since they generate a lot of data, change geographical locations, and require response to various events in real time. Thus, building applications that meet safety-critical and other dependability requirements is currently a challenge for the software industry. It may be expected that the operation of smart environments would necessitate the solution of complex problems, such as various optimizations, predictions, simulations and other hard problems. This translates to various requirements for resources, such as computing power, memory, storage, geographic positioning of services and similar.

Disruptively new software technologies are therefore needed to be able to rely on the computing continuum, that is, any adequate computing power and connectivity (wireless or wired) that can be used for applications, starting from the supercomputing / data centres, communities microservers, routers and other devices available at the edge of the network, as well as on fog (i.e. IoT). For example, software engineering environments should support the management of Non-Functional Requirement (NFR) (such as performance, energy efficiency, security, dependability) throughout the software life-cycle. Since the required computing infrastructures may be available within different administrative domains, technologies to acquire and use various computing resources should further be developed. Various improvements, such as optimized use of resources due to fine grained planning, scheduling and orchestration of services, could be also invented. Another typical technological aspect seen so far, is the constant shift towards lighter software packaging approaches, for example, from the use of Virtual Machines to containers. One could imagine significant progress in this domain as well, so that that only those essential parts of the code are stored, moved, executed and otherwise managed, which would contribute to improvements of the software life cycle. More monitoring and learning technologies are needed to be able to monitor Quality of Service (QoS), based on which software can be modelled and managed on a very fine grained level (e.g. parallelization of code for heterogeneous multi-processor systems.

Consequences in case the challenge is not addressed

New technologies for software engineering are being constantly invented. As we have seen in the course of the current period, there are still significant possibilities for improvement in the technologies which are used to deliver software of a very high quality.

For software companies across Europe it is a must to be able to capitalize on top of the IoT. Without improved software tools and methodologies to address the computing continuum, it would be
difficult to compete on the international market, and deliver products and services faster than the competition World-wide.

**Impact on technology**

Software is closely associated to hardware. Software can be implemented as hardware, and vice versa. There is, however, a growing industrial interest to build new hardware technologies with specific purpose, e.g. various precision calculation systems for low-energy computing. This requires new software engineering approaches, so that developers can benefit from the new developments. Similarly, new software technologies, may greatly influence the development of hardware, e.g. for seamless spinning on and off specially designed software services, which are highly optimized for performance, energy efficiency, security and other aspects, cashing various supporting functionalities of hardware systems, and other.

**Impact on society**

The impact on new technologies for software engineering on society could be large. While is increasingly hidden from humans, embedded in TV sets, refrigerators, cars, robots and similar, various optimization problems for humans remain. For example, one wishes to balance operational cost and quality of experience when playing a computer game. Software can be made responsive and adaptive to human decisions, such as a choice for low cost operation of a service, which may influence lower, but acceptable quality. In practically all smart environments human need to stay in the loop and make decisions. These decisions should affect the operation of the software, so that humans can actually steer software for their needs.

**Impact on European leadership**

Europe is already a leader in software services. That is why, various possibilities for improvements of the software engineering technology have been identified. Since there exist great possibilities for further improvement of common software engineering practices (e.g. DevOps), it is possible to imagine that other countries will invent new technologies, in case Europe does not take the lead. It is therefore necessary and required to invest in new software engineering technologies for the computing continuum.
HOW CAN HUMANS PARTICIPATE IN BUILDING TRUSTWORTHY SOFTWARE?

Description

Research has shown that user involvement is a major success factor for software projects. Involving citizens in the functional design of the systems of their interest has the potential to guarantee high service quality and user satisfaction. Thus, methods and processes for a democratic mass participation in software development need to be developed.

Users of software systems, acting as non-professional software developers, should be supported in the creation, modification or extension of a software artefact. The advent of massively interconnected objects, devices, and sensors — commonly referred to as the Internet of Things — raises equally massive challenges regarding the interfaces that will allow end-users to manage the complexity of such systems, to exploit the opportunities such technologies open up, and to tailor context-dependent interactive applications for their needs.

Consequences in case the challenge is not addressed

If Europe will not support the end-user development approach most of the personalization features available for non professional developers to address and manage emerging technological settings will be in the hands of the main ICT players, which are based outside Europe (e.g. Google, Apple, Facebook).

Impact on technology

Appropriate authoring environments and architectural infrastructures are needed to support End-User Development (EUD) in the context of the Internet of Things. New generation of EUD approaches, languages and tools are needed, allowing fast personalization of applications, sensors, smarts objects and intelligent environments. This personalization must be enabled through usable interaction metaphors and multimodal interfaces to identify and edit context-dependent events and conditions and how the application and the connected objects and services should react through different devices and interaction modalities possibly in an effective and persuasive manner. The multimodal authoring environments should support the use of graphical representations, subsets of natural language, gestural interfaces, and vibro-tactile feedback to ease end user development of personalization rules. Moreover, the environment will have to be able to identify potential conflicts, and simulate and debug such rules.

An example of EUD tooling that supports massive user participation is Requirements Bazaar (http://requirements-bazaar.org)—a prototype tool for social requirements engineering (SRE) which aims at supporting user participation by bringing together communities and service providers. Tools to systematically gather the voice of users (e.g., through app stores) are already important source of information for software projects. However, such tools do not address issues such as users having contradicting positions and opinions. Also, they reach their limits when the amount of users grows.
What is needed is a tool support that offers solutions for a scalable, structured, and democratic approach to gather feedback, contributions, and opinions and distills them to requirements. Both processes and tools are needed to mitigate the challenges of opening requirements engineering to the public—or in other words democratizing requirements engineering—particularly when it comes to the development of E-Government platforms. The challenges of SRE that will impact technology are: scalability, contribution quality, and conflicting contributions.

**Impact on society**

Regarding development of software systems, professional developers cannot foresee all the possible and even unpredictable situations that applications would encounter during use and whether the produced results will be actually meaningful to end users. End users assume an increasing number of responsibilities traditionally held by professional developers. As users continue to grow in number and diversity, EUDs are likely to play an increasingly central role in shaping software to meet the broad, varied, rapidly changing needs of the world. Some use cases include enabling users to take advantage of massive amounts of data accessible through the web; empowering end users to create a variety of software on their own, to define and tailor the functions of their systems in order to satisfy their personal, local, and often task-specific needs; or enabling a participatory, collaborative process in which end users, domain experts, and developers contribute with their different expertise at various stages of the process in co-creating the software artefacts to obtain meaningful results in real contexts and address concrete end user needs.

This is working also for citizens’ acceptance of E-government systems. This acceptance will improve as citizens will become active players in the field, driving the design of the systems they use. This, in turn, will improve trust towards the government due to the increased transparency and democratization of the process.

**Impact on European leadership**

EUD methodologies allow the software community to benefit from the creative power of end users and domain experts that will generate a large set of personalized versions of existing applications and services that will flexibly meet disparate and dynamic needs, and lead to completely new business models, which would positively influence European economy. In the specific topic of e-government, due to higher transparency and closer proximity to governmental processes, the European Union will increase the trust between central governments, the local municipal institutions, its citizens, and among European citizens. Consequently, the European Union will reinforce its position as a democratic establishment and bring citizens’ representation to software technologies and their development. The Union will set a new example for e-democracy through an endeavor complementary to the existing ones, such as e-voting, already in place. E-government platforms, designed through citizens participations, will set the condition for a wider political self-determination pivotal for a democratic society.
HOW TO DEFINE A SAFE INTERACTION BETWEEN HUMANS AND AUTONOMOUS MACHINES?

Description

In a context where humans are surrounded by autonomous machines and robots, it is of paramount importance to understand how to let the two interact in a proper way. On the one side, we need to understand what kinds of tools we should offer to humans to program autonomous machines and how to program them, respecting their autonomous capabilities. On the other side, we need to ensure that machines behave “appropriately” in their interaction with humans. This is a multidisciplinary area where software engineering is required both to revise the way design and programming paradigms are conceived to enable programmability of autonomous machines and to provide analysis and testing approaches to enable the verification and certification of autonomous machines.

Consequences in case the challenge is not addressed

A well-known Tesla accident occurred in 2016 and caused the death of the car owner. The accident occurred because the car was being driven by the autopilot and its sensors could not distinguish a white tractor-trailer against a bright sky. This is a clear example of an autonomous machine improperly assessing the situation and behaving in an incorrect way based on this. Even worse situations could be envisaged in the case the machine selects behavior that purposely harm humans using or around it. While the entire society has to reason on the identification of the right behavior (this is especially true when an autonomous machine has to choose between actions that can be harmful for different sets of people), software engineering can offer the technical means to ensure the execution of the right behavior when needed.

Impact on technology

The field of robotics, which includes devices supporting factory automation, autonomous driving, smart homes and the like, is now gaining momentum and achieving important results. For this reason, the issues of programmability of autonomous machines and of good machines behavior is becoming of paramount importance. Software Engineering is required in both areas. Autonomous machines programming should inherit and extend the experience in the programming languages area. Also, it should exploit formal verification techniques to check “good behavior of machines”. Of course, in both cases it is not just a matter of applying existing techniques to a new problem, but it requires a profound adaptation and evolution of existing concepts to make them suitable in the new context.

---

Impact on society

This research area is very critical for ensuring a proper and controlled adoption of autonomous machines in many different application domains. The risks of not considering machine-humans interaction as a critical issue in the first case can be very high for human life and for the wellbeing of the entire society.

Impact on European leadership

This area is at the edge of technology and has the potential to determine very important achievements in the context of autonomous machines. Considering the primary role of Europe in the areas where autonomous machines are having and could have an important role (like Industry 4.0 or autonomous vehicles), a strong participation of Europe in this research area could imply an important strategic advantage.

4. Conclusion

Software is influencing today all pillars of our society and, as such, it should be trustworthy. In this context, the main challenges that we need to face concern the need to increase our capability of building faster, more effective and more autonomous software, as well as our ability to verify and certify the characteristics of a trustworthy software not only with respect to functional aspects but also with respect to the many other properties that software should fulfill (availability, performance, security, privacy, sustainability, among others).

While these challenges seem to call for methods and tools devoted to highly skilled and specialized software professionals, at the same time, there is also the need to give power to end users both in participating to the design of complex systems and in programming part of them without breaking their main characteristics.

Wherever software performs tasks that require it to be more and more autonomous another important question is how to keep the relationship and the interaction between software and human beings under control so to ensure safety.

In this report we touch on the above aspects trying to spell out issues, areas to be investigated, the consequences of not investigating them and the impact that the development of these areas could have on technology, society and European Leadership.
5. REFERENCES


6. **Appendix A: Contributions from the Cluster Projects to the Definition of Challenges**

**Support to the Life-Cycle (Development and Operation) of Complex Software**

**Tools to Facilitate the Development of Resilient, Secure and Self-Adaptive Software**

**Description**

More and more, software applications need to be resilient and self-adaptive. Applications nowadays are deployed on heterogeneous resources, with different computational needs, in different technologies (and/or languages), are more complex, needing to aggregate, curate and process data coming from different sources and in the most varied of formats. Furthermore, software has become inherent to the functioning of the modern world. Everything contains software: planes, trains, critical infrastructures. The world is no longer understood without software. Therefore, when there is a software failure, it has the potential to make the world stop. The development of such resilient, self-adaptive, self-healing, complex and critical (not only for the business continuity but also for the daily life) software, in most cases with distributed teams in different locations, observes the need to create or extend methods / techniques / tools that facilitate the development, automated testing and automated deployment, update and self-healing of such complex and secured applications.

Current trends in that respect involve DevOps. However, these DevOps approaches focus predominantly on continuous integration, continuous quality and continuous delivery, avoiding early stages of the software development lifecycle and not paying enough attention in most cases to the health of the application (e.g. compliance of non-functional requirements).

Furthermore, in DevOps, security is traditionally under the cycle of Operations, but in fact, it also affects development and must be realized since the early stages of the design of the application. Thus, there is a need for novel solutions enabling organizations to benefit from the values of DevOps in developing complex applications (e.g. CPS, multi-cloud, fog computing) and at the same time be able to address today’s growing security concerns both during development and operations cycles. Therefore, in order to apply DevOps in such complex environments security needs to be considered both for the delivery pipeline as well as for the feedback loop. Hereby two options can be considered: 1) secdevops in which security is addressed from the requirements phase, architecture and design or 2) devsecops in which development comes first, then security is added and finally the application is deployed.
Impact on technology

With such tools and methods, aiding developers and operators in the development and maintenance of complex applications, applications will be more stable, trustworthy, safe and secure.

Impact on society

Software is a key enabler for business.

Organizations need to be sure that their applications deliver added-value and customer experiences of high quality. However, application complexity and current manual processes stop faster application delivery. Design complexity is often considered as one of the top technical challenges in the development and operation of applications.

Lack of automated tools is considered also as a significant impediment to deliver applications in a fast-paced manner. This is especially relevant in the case of deployments, where manual ones are often prone to human error, which must be discovered, evaluated, and resolved.

The provisioning of such methods and tools for complex, secure and critical software will impact in the software industry as a whole and in the domains for which the software is developed, e.g. Industry 4.0, CPS, safety critical, eHealth, and so on.

Impact on European leadership

Europe is leader in several sectors such as the automotive industry[1], aeronautics[2], where secure, self-healing applications are needed, while still achieving the regulations and certifications demanded. Furthermore, the digital transformation of the industry is a challenge that remains to be solved[3] and presents a huge growth potential in Europe. The provisioning of such tools and methods to help in the creation of new software, as well as the modernization of existing one towards the path for digital transformation of European industry can have huge impact.

References


Consequences in case the challenge is not addressed

Recently, British Airways suffered an incident that stopped Gatwick and Heathrow airports⁹. Even if British Airways claimed that it was due to a power failure¹⁰, data center experts are doubting such argument¹¹, as they do have available backup powerage. This situation, however, has unveiled a

______________________________

⁹ https://www.economist.com/blogs/gulliver/2017/05/going-nowhere
¹¹ https://www.theguardian.com/business/2017/may/30/british-airways-it-failure-experts-doubt-power-surge-claim
debate on outsourcing (common practice in large companies to cut costs) as well as on investment cuts in software development\textsuperscript{12}. In 2014, the UK went under the same issue when a claimed software bug caused the UK airspace to be closed for several hours\textsuperscript{13}. These incidents clearly show that research and investment in technologies to improve the way in which software is developed and operated is needed.

\section*{FORMAL METHODS FOR TAMING THE COMPLEXITY OF SOFTWARE SYSTEMS}

\subsection*{Description}

Software applications, services and systems are increasing more and more their complexity in the last decades, being almost impossible to be managed and understood by individuals (e.g. architects, developers, maintainers, etc) or small teams. Their size can range, in term of SLOC, in case of small-medium application/systems, from thousands to hundred of thousands of SLOC, and in case of large applications, from hundred of thousands to millions. Their complexity can also be measured in term of the number of their modules, components, etc, and the inter-dependencies among them. In terms of the size of their development (and maintenance) teams, the number of developers involved range from dozens to thousands, eventually structured hierarchical into small teams who cope with the development of specific parts of the entire software system.

Coping with such his complexity is not possible without powerful techniques and tools that assist developers to understand these software systems as a whole, proving the means to obtain overall representations of them at different levels of abstraction, and for different concerns. Therefore techniques to automate the program comprehension of large software systems, providing and managing the abstraction of their representations, supporting techniques for model slicing, concern/facet separation and representation, reverse modeling, abstraction raising, etc., are required to tame this increasing complexity. Despite a significant progress obtained by the research on this direction, their industrial adoption is still scarce, partially due to the extra complexity added by proposed methods and techniques, particularly for raising or lowering the abstraction level. Formal methods, based on AI techniques can automate these abstraction-switching techniques required to manage complex software systems at different comprehension scales.

\subsection*{Impact on technology}

Understanding software is vital to stop maintenance costs soaring: "Software maintenance gobbles up 90\% of the cost of software over its lifetime - and the reason typically goes back to its original development...Studies have shown that most maintenance costs arise from gaining an

\begin{thebibliography}{9}
\bibitem{13} http://www.telegraph.co.uk/news/aviation/11290489/UK-flights-grounded-as-London-airspace-closed-live.html
\end{thebibliography}
understanding of the software...These costs arise because most software is not maintained by those who developed it”

Taming the increasing software complexity will dramatically reduce the development and maintenance costs, increasing the quality of the development source, its interoperability, maintainability, adaptability, suitability, compatibility, efficiency, usability, reliability, security or portability.

**Impact on society**

There is a profound economic impact on the society, due to the highly increasing costs of coping with the software complexity and its maintenance. There are also strong impact on the costs associated to software failures, software service outages, etc., in most cases, caused by the barriers imposed by the software complexity to identify potential security breaches, software bugs, that are hidden by the complexity.

**Impact on European leadership**

The European industry of software plays a leading role in a highly competitive global market, threatened by expanding Asian software industry and the leading US industry. In this global market, releasing high-quality software systems with restrained manufacturing and maintenance costs is vital to become competitive. Lowering the cost required to acquire and maintain the know-how of complex software systems results on reducing overall software costs.

**Consequences in case the challenge is not addressed**

Increasing software complexity, particularly on mission-critical systems, is identified as a potential source of risk, which, if not tamed, can lead to operational failures that comprise the European industry and economy. Some examples of software systems which complexity was not tamed, leading to catastrophic consequences follow:

*Less than a week into 2016, HSBC became the first bank to suffer a major IT outage. Millions of the bank’s customers were unable to access online accounts. Services only returned to normal after a two-day outage. The bank’s chief operating officer Jack Hackett blamed a “complex technical issue” with its internal systems.*

*In March 2015 the UK government was forced to delay launch of its £154 million rural payments system, a new online service for farmers to use to apply for Common Agricultural Policy payments from the EU. The service was originally supposed to be up and running by May 2015 but integration problems between the portal and the rules engine software proved ‘too difficult to overcome’ by then.*

---

Cognitive Model-Driven Engineering

Description

Model-Driven Engineering (MDE) tools facilitate the automation of software engineering, capturing and automating complex decision processes. The gains in productivity, quality and time has been extensively shown. However, current MDE methods and tools are characterized by a total lack of convenience. MDE tools and service providers must address a digital transformation in order to become people first, showing more consideration for how professionals think and organize\(^\text{15}\) and providing engaging experiences to them. The final goal should be to provide software and systems architects, designers and developers with tools that are simple, intuitive, and that return immediate value. One of the main challenges here is the complexity of the MDE tools life-cycle and the practice known as “domain engineering,” where common and variable design decisions are be analyzed. Actually, this is the reason why engineering teams providing the tools are normally distinct from those that use them. The provider-consumer relationship typical from MDE tool plays against the high levels of agility required to software professionals, who usually see how new (decision) variability points arise and the scope of the domain shifts. When this happen, software engineers enter a time-consuming and laborious process where they work, on the one hand, to either bypass or modify the tool to change the decision process or, on the other hand, to either abandon the tool or wait for the tool provider to listen to their change request.

Cognitive MDE means to build an intelligence layer on top of current MDE methods and tools. This layer would take the form of intelligent virtual assistants which communicate with software professionals in natural form, make decisions, take actions with few or no input from the user and learn from experience both professional’s preferences and the best decisions in different contexts. Many of those decisions and actions would refer to complex MDE tasks, such as model construction, analysis and validation, where modeling assistants would apply AI techniques such as machine learning to automate tasks. These software robots or modelbots would effectively augment the intelligence of software professionals (following the so-called Cognitive Computing paradigm) and empower them to carry out MDE in a more rapid, easier and more intuitive manner. These assistants eliminate the provider-consumer dichotomy by empowering the software architects, designers and developers to modify themselves the automated decision processes by configuring (or “training”) the modelbot using demonstrative examples or any other form of natural communication. This would effectively realize the idea of self-service MDE.

Impact on technology

The intelligence augmentation of software professionals realizes as an intelligence layer on the top of current MDE technologies. This layer further operationalizes through intelligent assistants or

**modelbots** that become the central element in modeling activities and workbenches. These assistants would be equipped with the following capabilities:

- Communicate in **natural** (e.g., conversational) ways.
- Let (assist, recommend) users to make more, faster and better (design) decisions when creating models of complex systems of systems. This transforms the software automation problem in a more natural **decision delegation problem**.
- Help to manage and keep consistency among architecture views and, in general, navigate, verify and validate **complex systems of systems architectures**.
- Let software professionals specify the requirements and constraints for the **automation of new tasks / decision processes** (e.g., model analysis, model validation, or model construction), including the steps in the process where the modelbot must receive further instructions (variability points).
- Learn from experience, based on examples, to automate complex MDE tasks by means of machine learning, including paradigms such as **deep learning** (for example, LSTM networks, which have shown great success in language processing), **reinforcement learning** and/or innovative forms of **active learning** (where the learning system gets involved in the learning process rather than being passive).

Modelbots increases the productivity and agility of software architects, designers and developers when developing complex systems of systems. They provide intelligence augmentation to software professionals by giving assistance in the form of advice, recommendations and robotic process automation (RPA) when managing a high volume and variety of architecture views. In this kind of systems, change speed (velocity) and veracity (e.g., consistency) are also important challenges; robotic assistance helps to tackle those difficulties. Nevertheless, that means the MDE intelligence layer must integrate not only with specific MDE technologies but also with whole architecture frameworks, architecture management tools and application intelligence suites.

Technologies for building modelbots may take ideas from the so-called ChatOps approach to DevOps, where developers communicate with software robots through the same channels as they communicate with the rest of the team. These channels are usually messaging platforms such as Slack. Software robots become thus an integral part of the team, understanding the project context, passing messages and notifying events (e.g., alarms) to them, and answering their questions about the deployment, provision and monitoring of the systems. They also can follow more or less direct commands that automate rote systems tasks. These capabilities are realized by means of conversational agents or chatbots technologies, such as automatic speech recognition (ASR), natural language processing (NLP) and semantic search.

**Impact on society**

Creating model-driven engineering tools is a complicated, time-consuming and error-prone task that requires higher levels of software engineering skills. The dream of a future where the user (software
professional) modifies its own tools has not come true. The truth is that development and evolution of the software automation tools are usually beyond the knowledge and experience of most software architects, designers and developers. This has for decades limited the impact of software automation, in general, and MDE, in particular.

In the case of Cognitive MDE, software professionals are able to communicate in natural form with modelbots to tell them their needs and desires, perhaps by giving demonstrative steps, rather than program their behavior through code-based instructions. This in an intuitive manner to configure (“teach”) a tool to do something once and again, that is, repeat a decision process. Therefore, Cognitive MDE will have a tremendous impact across industries and ultimately in economy and society, which will benefit from more and better software in less time.

**Impact on European leadership**

Software professionals are no longer in a position to compromise their agility by using MDE tools, which usually do not satisfy their changing demands. The software life-cycle of those tools is too long and complicated so as to keep the tools synched with the problem and solution at hand. It is important to understand that MDE methods and tools provide agility to engineering during the application engineering phase but not during the so-called domain engineering, that is, the development phase of the tools themselves. This might has been the cause of the rising popularity of scripting languages being used to automate systems engineering tasks, or the spectacular popularity gained by the agile methodologies (that reject the use of any kind of model) in the last decades at the expense of MDE. MDE is at risk of becoming obsolete and with it the software automation principles it brings with it.

Cognitive MDE would eliminate the chasm between agile and MDE principles and practices, which will become an authentic revolution in the software industry. It will provide new orders of magnitude of operational effectiveness, quality and cost efficiency to software professionals. With a strong MDE tradition, European industries and academic institutions would become the forefront of research and innovation in this new form of software production. Furthermore, cyber-physical systems and Industry 4.0, which are highly dependant on software, will greatly benefit from cheaper, faster and better software.

**Consequences in case the challenge is not addressed**

The current european leadership in model-driven engineering (MDE) and MDE technologies is at risk of obsolescence due to the long software life cycles of MDE tools. Current MDE solutions devised by european research institutions and industries are at risk of becoming too slow, difficult and costly to use and evolve for the digital world. Like any other product and service in the digital era, MDE tools must become convenient and people/user-first; this requires tools providers to undergo an authentic digital transformation to make their offering attractive to software professionals.

Unless we introduce a layer of intelligence on the top of MDE methods and tools, to provide convenience and self-service capabilities to software professionals (e.g., to modify MDE tools in order to rapidly and easily adding and/or updating automated decision processes), european...
software industry is at risk of losing its competitiveness advantage in domains where traditionally has excel (e.g., safety-critical, embedded) in favour of other regions producing more and better software in less time.

**AI Process Automation**

**Description**

The development, operation and evolution of AI systems, specially when based on machine learning techniques such as deep learning and reinforcement learning, are substantially different from those of “traditional” software products and applications. AI systems engineering requires elaborated machine learning (ML) pipelines integrating data management (indexing, storage, quality assurance), data collection and processing (cleaning, formatting), model building (training, tuning), model validation, model deployment and model monitoring activities. Moreover, ML pipelines can become complex workflows which incorporates diverse stakeholders: from data analytics and data scientists to data engineers and production (systems infrastructure) engineers.

AI systems can be developed and operated independently as (micro-)services in a service-oriented architectures. However, they must work in collaboration (orchestration) with other components to provide higher level functions and services; these other components of the architecture will normally be developed using traditional software engineering techniques. In the case of autonomous cyber-physical systems, such as driverless vehicles, these architectures require complicated systems engineering\(^{16}\), for example, integrating deep learning in automotive software\(^{17}\).

The challenges are big:

1. Current AI systems engineering lacks proper tooling, specially for end-to-end management and streamlining of data processing and analytics pipelines in real-time.
2. There is a growing interest in the so-called automated machine learning (AutoML) paradigm, which intents to automate the ML pipelines. Toolsets which depart from “traditional” systems engineering technologies are being developed.
3. AI systems engineering lacks proper modeling and abstraction tools allowing for effective reuse and portability of solutions. Companies are working on their own reuse formalisms and frameworks. This is resulting in a lack of standards and a growing uncertainty about the best way to address the design of those systems.
4. AI systems are not planned to replace (at least in the foreseen future) whole software applications and services but to seamlessly integrate with them as (micro-)services in large- -


\(^{17}\) [https://www.computer.org/software-magazine/2017/06/13/deep-learning-in-automotive-software](https://www.computer.org/software-magazine/2017/06/13/deep-learning-in-automotive-software)
scale distributed architectures. These architectures and many of their building blocks are developed following more “traditional” engineering techniques.

5. AI systems provide unique adaptation and runtime evolution capabilities; however, to be effective in larger distributed architectures, they will have to be consistent with autonomic computing, the discipline that has traditionally tackle this problems in software engineering.

6. AI systems engineering process need to seamlessly integrate and be compatible with current software and systems engineering infrastructure and tools in order to optimize resources (including human ones).

AI process automation proposes the use of concepts, methods and tools (even technologies) coming from more traditional software engineering in order to end-to-end streamline AI systems processes. For example, a software and systems architecture approach would bring abstraction, separation of concerns and composition techniques that would result in higher levels of standardization, reuse and automation in AI systems processes.

Impact on technology

AI systems will be realized as an intelligence layer on top of (or an intelligence fabric interwoven with) current software systems. Therefore, software and systems engineering practices and technologies are expected to be impacted by this new kind of software component. This is especially true in the case of software process automation as we plan to use current technologies to streamline AI processes (e.g., software and systems architecture), which will result in new requirements and constraints for that technology. Furthermore, this alignment and integration of toolsets will affect the kind of technologies data engineers use for developing and operating data processing and data analytics (e.g., machine learning) pipelines.

AI process automation proposes an unified approach to systems engineering; this means “tradicional” application platforms will have to give support to AI-based components. In other words, current industry-specific and/or application-specific platforms are expected to be extended to give support to the development and (integrated) operation of AI-based components. In the immediate future, application and technology platforms for sales, marketing and cyber-physical systems (e.g., autonomous cars, autonomous robots, unmanned aerial vehicles, etc) are going to be among the most impacted ones.

Impact on society

Security and safety of systems (and systems of systems) are becoming a growing source of concern in safety-critical engineering. AI-based systems and some of their properties are worsening the challenge: algorithmic bias, lack of transparency, stochastic processes, etc. It is important that AI systems build on top of well-sounded engineering practices and techniques to ensure dependability and security.

AI process automation promises to incorporate state-of-the-art methods, tools and technologies in AI systems engineering. This approach will help industries incorporate new levels of convenience and
adaptation in its products and services without compromising dependability, reaping the AI benefits while avoiding potential pitfalls.

**Impact on European leadership**

Digital platforms need to give support to the development of large-scale distributed systems where AI components play an important role. These systems are expected to be a blend of AI and more “traditional” software engineering components. With AI process automation, industries that rely heavily on software will be able to handle the infusion of AI in their products and services in a cost-effective manner. They will also benefit from the high levels of dependability and predictability coming from software process automation and standardization. Industry- and application-specific platforms such as automotive, industrial robotics or unmanned aerial vehicles are expected to be greatly benefited from the industrialization of the production of AI systems.

With a strong MDE (Model-Driven Engineering) tradition, in particular, and safety-critical and embedded systems engineering, in general, European industries and academic institutions working on AI process automation will be in the forefront of a new software revolution. This revolution will bring together engineering process automation methods and tools for AI and non-AI systems, resulting in higher levels of quality, productivity and time-to-market in AI-based systems development. AI process automation would also mean that systems with increasing levels of intelligence remain dependable and cost-effective. This is particularly important to ensure European leadership in the Industry 4.0.

**Consequences in case the challenge is not addressed**

There exists a risk of disintegration of practices and tools in AI and non-AI systems engineering. The development of new engineering tools and techniques for dealing with AI systems would mean that we fail to build on top of existent software and systems engineering. This might slow down the rise of AI systems or, even worse, make them much more unreliable, unsecure and unsafe than (or even compromise those properties in) current systems.

The development of alternative toolsets for developing AI systems would also widen the distance between data engineering and more “traditional” software engineering practices. The lack of an integrated approach would increase the costs of systems engineering incorporating increasing levels of AI.
MANAGEMENT PRINCIPLES FOR SOFTWARE FRAGMENTS, EXECUTABLE IMAGES OF VIRTUAL MACHINES AND CONTAINERS

Description

The ENTICE project uses analysis and synthesis mechanisms for VM images, so that they can be automatically composed and decomposed (treated no longer as monolithic entities) in logical functional components, such as software packages, libraries, executables, services, components, OS kernels, files and so on. For the ENTICE distributed repositories of VM and container images, this kind of fragmentation brings a number of benefits, such as improved VM or container creation times by assembling VM images from fragments, improved performance by minimising the number of fragments in a VM and consequently its size, and optimised storage by sharing common fragments, such as popular libraries or OS kernels.

The proposal is therefore to create a new challenge that focuses on the management of software fragments for improved software operations.

Impact on technology

This would provide much finer grained mechanisms for the management of software components, reducing cost of storage for example, improving reusability of software and similar.

Impact on society

Faster software production and lower cost of storing and manipulating of storage, as suggested by the ENTICE project.

Impact on European leadership

New software technologies for the management of software as executable images and parts of such images.

Consequences in case the challenge is not addressed

Component based engineering is currently on the rise. Europe will be less competitive in the software domain.
SECURITY AND PRIVACY

DEVELOPMENT OF SECURE SOFTWARE

Description

Secure Software Development, intended as the practices to ensure that the code and processes that go into developing applications are as secure as possible, is assuming high importance, with the vast amount of threats that constantly pressure companies and governments.

As technology advances, application environments become more complex and application development security becomes more challenging. Applications, systems, and networks are constantly under various security attacks such as malicious code or denial of service.

The emergence of Microservices and cloud-native applications (which make heavy use of cloud resources as a service) has posed a challenge from the point of view of securing the applications. With an application composed by several services, the attack surface is bigger, and compromising a service may compromise other services.

Impact on technology

Security should be planned and managed throughout every stage of software development life cycle. There are a number of basic principles and practices already in place, however a lot of work has still to be done both at the level of standardization and at the level of adapting to evolving technologies (for example IoT, devices for smart environments or AI).

Impact on society

Trust in technology is the basis for its adoption. Therefore correctly address security aspects would have an impact on several application domains, such as health and assisted living applications, remote payment, chatbots and virtual assistant technologies, autonomous driving, that deeply involve personal data and impact the individual.

Impact on European leadership

Security is an important driver in many fields and serves to secure Europe's competitiveness as we all depend on safe and reliable information technology – be it in the fields of energy supply, healthcare, logistics and transport or in the field of finance.

Consequences in case the challenge is not addressed

Some recent examples of not addressing this challenge are reported below:

March 2016: Microsoft is "deeply sorry for the unintended offensive and hurtful tweets" generated by its renegade artificial intelligence chatbot, Tay, which was abruptly taken offline Thursday after one day of life. (Source: USA Today)
Dec 2016: According to US intelligence officials, Russian hackers made repeated attempts before this year’s election to get into major US institutions, including the White House and the state department. The tactics were simple: send out volleys of phishing emails and hope that someone clicked. (Source: The Guardian)

May 2017: “Two days before France’s recent presidential election, hackers leaked nine gigabytes of emails from candidate Emmanuel Macron’s campaign onto the web” (Source: Wired)

August 2017: Equifax reported a data breach that has affected 143 million people

Industry 4.0 is one of the application fields that poses specific requirements on security. The risk of not addressing them could inhibit the growth of the Industry 4.0 concept due to lack of assurance from customers and vendors.

PRIVACY-AWARE SOFTWARE ENGINEERING

Description

The increasingly huge amount of data that is continuously produced by today’s digital environment (sensors, mobile devices, social networks, etc) has led over the last years to the development of many technologies enabling the exploitation of what is generally referred to as Big Data, which is now leading to increasing demand for data-intensive computing. Although today we have the necessary technology for storing and processing huge amounts of data, there are still many open issues connected to Big Data. Some of the most critical are related to data privacy, as in many cases Big Data are personal and sensitive data. The problem of data protection becomes more and more complex as the collection and processing of personal data becomes more pervasive, such as in sensor-based and fog computing systems.

In this context, the design and development of data-intensive systems needs new methods and tools to deal with data security and privacy concerns, in such a way to deliver software that is privacy-aware by design.

Data security usually deals with guaranteeing the confidentiality (i.e. protection from unauthorized access), integrity (i.e. protection from unauthorized modification) and availability of data. Privacy, although strictly connected with the concept of confidentiality, additionally impose to take into account individual privacy preferences as well as privacy regulations. All of this issues must carefully be taken into account when designing and developing modern data-intensive systems. Nevertheless, there has been so far only little effort along providing software engineering methods and tool for achieving this goal. In this context, the challenges are manyfold. Some of them are for instance:

enforcing access control policies in big data stores and applications, techniques to control which
data are collected and how they are used, approaches for letting data subjects to easily and directly
control their privacy, bringing cryptographic and data anonymization concepts and techniques to big
data stores and applications and many others. This problems, along with the new, complex and data-
centric technological environment in which they occur, define overall a new research direction that
we may call privacy-aware software engineering, i.e. engineering software to provide strong data
privacy guarantees by design.

**Impact on technology**

The following technologies and approaches are influenced by privacy issues and will have to be
extended and integrated within the software lifecycle to enable the development of privacy-aware
software:

- anonymization of big data sets and data streams,
- access control techniques for big data stores,
- new approaches for having users to directly take part into the process of securing their own
data,
- technique to monitor and control how data intensive applications use sensitive data,
- methods and tools for managing and enforcing user privacy preferences in data-intensive
applications,
- programming models and techniques for identify portions of code that deal with sensitive
data,
- methods for letting data-intensive application to seamless work on anonymized data,
- formal methods to analyse and identify data security threats,
- tool for estimating the privacy risks,
- methods for understanding and lowering privacy implications on data quality,
- methods for handling data acquisition in a privacy preserving manner.

**Impact on society**

As we live in a society in which our personal data are more and more gathered and used by third
parties in order to provide ICT services, which, in turn, are more and more pervasive in our life, data
security and privacy concerns simply cannot be ignored and must become primary requirements for
modern software. Developing methods and tools for privacy-aware software design and
development of modern data-intensive systems will help in bridging such unacceptable gap.

**Impact on European leadership**

It is commonly recognized that in the European culture there is a stronger expectation of privacy
with respect to United States. Indeed, privacy in the EU is protected as a "fundamental right" under
the European Union's Charter of Fundamental Rights. This is confirmed also by the all-encompassing
data protection framework currently in vigor in Europe, compared to the federal or sector-specific
US privacy laws. Nevertheless, the risk of compromising customer privacy is taken much more
seriously by US companies, which eavily invest in privacy preserving solutions since breaches of data
protection laws can reflect in serious penalties, than in European ones. This research would then
foster in Europe the adoption of privacy preserving techniques and the development of privacy-aware software, thus bridging the gap between the European sensibility for privacy and the current lack of an appropriate enforcement effort.

**Consequences in case the challenge is not addressed**

The risk of privacy breach is well known and has always existed since the birth of the internet and the world wide web. It is indeed quite easy to come to know about famous privacy breaches that have happened in the past. It is clear that the more the information technologies become pervasive and the more our society becomes data-centric, the more the risk and the number of privacy violations augment. On the long run this can have a strong impact on the trust people give to information systems.

**VERIFICATION, TESTING AND COMPLIANCE ISSUES**

**FUNCTIONAL TESTING FOR NATIVE-CLOUD APPS**

**Description**

Applications exposing a microservice architecture, and native-cloud applications in general, impose a different approach for functional testing. The application is scattered among different services, and ensuring functional tests cover what we really want to cover requires new testing approaches, including: partial deployments to test part of the services (integration tests), monitoring, introspection, log ingestion and analysis (with relations to big data), but also new IA techniques to detect, from the big amount of information generated when running the tests, if there's any substantial difference between test runs, to detect early regressions. Ensuring such applications work correctly will require machine learning techniques mixing natural language recognition (for logs) and pattern recognition (for metrics) to provide a high level overview of the system as a whole, possible points of failures, bad-behaving components, among other features.

Given that current testing practices suppose about 35% of the total IT cost (with 40% predicted for 2018 and 2019)\(^\text{20}\), these costs can even be increased with the need to test cloud-native applications.

**Impact on technology**

The challenge would require to develop novel test techniques, including:

- Microservice language specification: a language to describe the microservices and their relationships
- Partial deployments of those microservices to test specific integrations of services

\(^{20}\) [https://www.capgemini.com/resources/world-quality-report-2016-17](https://www.capgemini.com/resources/world-quality-report-2016-17)
● Failure injection to ensure the proper mechanisms to handle load balancing, circuit breaker, etc, are well implemented
● Load injection
● New tools for log analysis are required
● Testing in production

Impact on society

With more and more companies pushing their assets (software) to the cloud, ensuring it works as expected is critical. From a social point of view, users will see less defects, more resilient and robust applications, and in general, a sense of security when using cloud applications.

Impact on European leadership

Europe has a long tradition in testing, with events like Test Bash or the European Testing Conference, and it has consolidated research groups in testing in different countries. Some big companies in Europe (like Airbus) do need good testing practices, and Europe is leading the Industry 4.0 revolution, where new testing approaches will be needed in a world of interconnected devices. There’s a chance for leading the new approaches for testing these modern cloud-native highly distributed applications.

Consequences in case the challenge is not addressed

Other stakeholders may take part on this and lock-in users. AWS, for instance, is paying much attention to development tools (AWS Code Commit, AWS Code Deploy etc.). It would be a natural consequence that they embraced app testing as well. They already have a “service” description language (cloudformation), evolving it to support high level descriptions for services that can be used when testing the application could be a matter of time.

CONTEXT-AWARE TESTING IN THE FIELD

Description

For dynamic systems composed by the interaction of different services and components, it becomes infeasible to reproduce in testing environment the many possible configurations, parameters and interactions happening on line\textsuperscript{21}, therefore for validating the adequacy of a system to given requirements, both functional and non-functional, testing must be moved to production and carried out in continuous way along with monitoring\textsuperscript{22}.

\textsuperscript{21} https://thenewstack.io/distributed-systems-hard/
\textsuperscript{22} https://opensource.com/article/17/8/testing-production
Impact on technology

New solutions for allowing the generation, launch, control of tests for deployed systems, that are tailored for the specific objectives and profile of usage should be found. These facilities should be easy and automated allowing for the clients and end-user to administer the testing according to their purposes. Solutions for mitigating side-effects on actual usage should be conceived.

Impact on society

Improved security and reliability are achieved, and overall trust of society in ICT systems is increased.

Impact on European leadership

Europe has a long tradition in testing, with events like Test Bash or the European Testing Conference, and it has consolidated research groups in testing in different countries. Some big companies in Europe (like Airbus) do need good testing practices, and Europe is leading the Industry 4.0 revolution, where new testing approaches will be needed in a world of interconnected devices. There’s a chance for leading the new approaches for testing these modern cloud-native highly distributed applications.

Consequences in case the challenge is not addressed

Testing remains the most costly and complex activity in software development. The challenge of doing adequate testing before a system is deployed cannot be met due to the dynamism and ultra large-scale of modern ICT systems.

On the other hand, ICT systems are pervasive and if testing is not adapted to new paradigms as cloud, continuous integration, DevOps, society will be subject to consequences of failures.

INTEGRATION OF (SAFETY CRITICAL) SYSTEMS IN A MULTI-NATIONAL REGULATION

Description

More and more often software systems are influencing everyday life. An important category of software systems are the safety critical applications installed on railways, planes or health care devices. One of the major problems in this case is the capacity of checking the design and the requirements of these systems to early discover serious flaws and, at the same time, to validate their compliance with (multi-)national norms and regulations. Very often these norms differ from country to country leading to serious issues when an integration of such systems has to be performed at an international level, as, for example, happened during the integration of high speed train control systems in the EU multi-national area. This kind of large scale integration requires an intense human and computer decision-making and validation effort. This task should be performed following well structured software engineering processes that allow to systematically check the specification of the systems against the different kinds of rules in the different countries and evolve these specifications.
according to the changes in these rules also exploiting AI based reasoning techniques such as Machine Learning or logic based reasoning.

**Impact on technology**

The possibility to study (semi)-automated methods for regulation representation and reasoning techniques to perform check and validation.

**Impact on society**

The solution of this challenge for different kind of systems, and in particular safety critical systems, is relevant in several fields such as transportation, healthcare and other crucial services for EU citizens. The same solution should be generalised to the case of a set of systems/services that adhere to national/local rules/specifications and that have to be integrated together in order to satisfy new regulations/specifications minimising the effort and the costs related to this transformation and to minimise the time to market of the revised solutions. This should enable the citizens in having a faster access to integrated services.

**Impact on European leadership**

Europe is the perfect area for studying and applying the outcomes of this research due to its current multinational structure and to the existence of governmental bodies that release regulations and specifications affecting the specific regulations and specifications of the different countries and at the same time requiring integration among them. This panorama should allow to design methods and tools that should constitute a model to be proposed to other cases around the world so increasing the European capacity of taking leadership in this critical area.

**Consequences in case the challenge is not addressed**

The consequence to not having a precise Software engineering methodology in this case could determine delays and flaws in the integration phase, leading to delays in the use of the integrated services for the citizens, so compromising the capacity of EU to early expose the products of the research and technology transfer in EU area and worldwide.
The next evolution in cloud computing will be towards a more distributed nature. This is already foreseen by some in the industry [horowitz]. We already see this with the emergence of edge and fog computing and their applications to not only regular IT applications but also to the 5GPPP domains. Many of the use cases that are mentioned in the 5GPPP vision paper assume that:

1. the supporting telecommunication infrastructure can deliver the latency (computation in a timely manner for the use) required for critical applications or services that are used for safety critical systems (e.g. car, factory [Industrie 4.0], emergency services) and
2. by inference, the software using the underlying infrastructure a) takes avail of all features and facilities of that (available services [owner’s and 3rd-party/outsourced], placement of data/computation sensitive to latency of end-user) and b) that the software that executes upon this infrastructure does so reliably [definition23].

All future applications are also subject to the ever increasing pace of application delivery. As the need for delivery acceleration is needed, this motivates ever more the need for reliability as this acceleration is brought forth by automation and the automation of unreliability is to be avoided at all costs.

Impact on technology - what needs to be updated?

There are already formal methods (Coq, TLA+) that can provide significant improvements in software reliability, however to the regular software developer they are out of reach and appear "too academic". current software development practises and processes do not allow for the use of these systems. Such systems need to be integrated into the software lifecycle. Even with today's software engineering practises, cloud native approaches are in the minority, with developer's having to wrestle with legacy maintenance. Much research into how to make reliability

Impact on society

Provide meaningful scenarios where addressing the challenges could be helpful

Reliability in scenarios described/referenced above can be confidently enabled. This further maintains and increases the European consumer/prosumer’s trust in software and future similar software that will be delivered upon new infrastructure, many of these changes transparent. If

there’s no reliability, this transparency will be shattered reducing the trust in new technology investments.

Impact on European leadership

Addressing this challenge will set Europe as the lead or at least the fore-runner in reliable applications requiring 5G. Having reliable methods applied to software engineering will allow European consumers and prosumers to further extend and use software services and software running upon this new infrastructure and ever more pushing the service-based economy forward.

Consequences in case the challenge is not addressed

Investments in existing cloud research will fail to impact over time.

COLLECTIVE SELF-ADAPTIVE SYSTEMS

Description

Self-adaptive systems are systems that can adapt themselves dynamically (at runtime) to a change in their internal or external environment. MAPE-K\(^{24}\) is a high-level reference model for self-adaptive systems, representing the essential functions for self-adaptation. Traditionally, a centralized approach to realise those functions has been taken. More recent research proposes decentralized schemes where MAPE loops are coordinated in different ways\(^{25}\). However, important challenges still lay ahead:

- Guiding principles for decentralised self-adaptation.
- Coordination mechanisms and interaction protocols between managing systems (autonomic managers) for different types of adaptation goals.
- Emergence of adaptation functions in complex managing systems.
- Self-organization to achieve self-adaptation by means of massive interconnection between (managing and managed) systems, decentralized decision-making and distributed problem-solving.

The control of global behavior emerging from complex systems is a growing concern. Technologies such as Internet of Things (IoT) and Cyber Physical Systems (CPS) are expected to be the based for


the so-called **collective adaptive systems (CAS)**\(^{26}\). By nature, those are typically complex systems that tend to display emergent (bottom-up, rather top-down/prescribed by design) behavior. Regardless of whether this emergence is intended or unintended, we must be able to keep control on it. However, complex systems with emergent behavior are by definition difficult to control as small changes in one element can lead to huge changes at the system level. This has become an important issue for achieving the goal of self-adaptation.

A related but distinct problem is the realization of adaptation functions as emergent behavior coming from complex interplay between distributed managing systems, rather than prescribed in a top-down fashion (by design) and implemented in a centralized way.

A self-adaptive system must continuously monitor changes in its context and react accordingly; an important issue is deciding (and continuously change or adapt) what aspects of the environment to monitor and where to act to get the desired behavioral change. Self-organization can help to solve this problem in a more cost-effective and resilient way. Therefore, it would be important to devise principles and reference architectures for realizing self-adaptation of self-organized managed systems and/or by self-organized managing systems. Self-organization is a classical mechanism to realize and control emergent behavior.

Adaptive Systems research has a long history with successes in various fields. Its symbiosis with classical cybernetics, control theory and modern control engineering has been specially fruitful. It provides well-established mathematical models, tools and techniques to analyze dynamical (complex) systems and control (regulate) their behavior through feedback loops. Adaptive systems have extensively use cybernetic models to study emergent behavior in multi-agent systems, where each agent is equipped with feedback loop mechanisms to realize reactive (local) control instead of elaborate planning or decision-making. The massive interaction (communication) between simple agents results in elaborated emergent behavior at the (multi-agent) system level. Collective Adaptive Systems (CAS) is a new field that emphasises the "collective" nature of the adaptive system, incorporates related fields such a self-organization and collective (swarm) intelligence, and brings together research and engineering communities working on these issues. In order to solve the problem of emergent behavior and/or self-organization in managing and managed systems, **the field of self-adaptation must evolve to integrate CAS concepts, principles, methods, tools and techniques.**

**Impact on technology**

Internet of Things (IoT) and Cyber Physical Systems (CPS) are expected to have a wide impact in all sectors and industries, especially those related to Industry 4.0. This complex system architectures

are starting to be understood as large-scale ensembles of collective adaptive computing systems (CAS)\(^\text{27}\):

\begin{quote}
**tight entanglements of real world physical objects (things, appliances) and processes (services), with their digital data representation and computations in communication networks (the "cyber").** Embedded, wirelessly connected tiny compute platforms equipped with a multitude of miniaturized sensors collect data about phenomena, analyze and interpret that data in real time, reason about the recognized context, make decisions, and influence or control their environment via a multitude of actuators. Sensing, reasoning and control, thus, are tightly interconnecting the physical and digital domains of the world, with feedback loops coupling one domain to the other. They implement notions of **autonomous adaptive behavior**.
\end{quote}

CAS exhibit properties typically observed in **complex systems**:

- spontaneous, dynamic network configuration, with
- individual nodes acting in parallel,
- constantly acting and reacting to what the other agents are doing, and
- where the control tends to be highly dispersed and decentralized. If there is to be any coherent behavior in the system, it
- has to arise from competition and cooperation among the individual nodes, so that the overall behavior of the system is the result of a huge number of decisions made every moment by many individual entities.

In the upcoming years, much research and engineering work will be carried out to ensure CAS remain under control. Self-adaptive systems is a discipline that has traditionally dealt with this problem in “simple” systems. The evolution of it towards the integration with concepts, principles, methods, tools and techniques from CAS will have a profound impact on the technology for self-adaptation as well as the whole technology development around Internet of Things (IoT) and Cyber Physical Systems (CPS).

**Impact on society**

In a digital world, the increasing introduction of Internet of Things (IoT) and Cyber Physical Systems (CPS) will result in increasingly complex systems that interwove ICT with human behaviour. Economic, social and political systems are becoming highly reliant on the dependability, security and resilience of those systems and their emerging behavior.

Self-adaptive systems research has traditionally addressed the problem of creating systems that are able to adapt themselves dynamically (at runtime) to a change in their internal or external environment. This has proved to be a promising strategy to ensure dependability, security and resiliency of systems. However, the progressive emergence of CAS represents an important

challenge to the field: they nurture a new kind of system that exhibits properties typically observed in complex systems and, therefore, by definition, difficult to control as small changes in one element can lead to huge changes at the system level. Understanding, analysis, development and operation of collective self-adaptation is a promising approach to tackle complexity and keep the performance of complex systems under control.

Impact on European leadership

Europe aspires to lead the Industry 4.0 revolution. Internet of Things (IoT) and Cyber Physical Systems (CPS) are central in that revolution. In order to remain in leadership in industrial automation, Europe needs extensive research and engineering effort regarding not only Collective Adaptive computing Systems (CAS) but also mechanisms to keep their performance under control. Self-adaptive systems research has traditionally addressed the issues of dependability and resilience in computing systems. Its evolution towards Collective Self-Adaptive Systems (CSAS), integrating concepts, principles, methods, tools and techniques coming from CAS, will ensure that European industry is provided with tools to keep complex CPS under control and provide highly dependable, secure and resilient solutions.

Consequences in case the challenge is not addressed

The deployment of solutions based on the Internet of Things (IoT) and Cyber Physical Systems (CPS) might result in complex uncontrolled (and uncontrollable) systems to be deployed in safety-critical situations in the near future. Intendedly or unintendedly, CPS can easily become Collective Adaptive computing Systems (CAS), which exhibit properties typically observed in complex systems. We need to start working on the dependability, security and resiliency of that kind of systems, which are different in nature to the kind of “simple” systems we are used to design, develop and operate. This requires a radically new approaches to reliability, security and safety, in general, and self-adaptation, in particular. We call Collective Self-Adaptive Systems (CSAS) to this radical new way of thinking, design, build and operate self-adaptive systems. If we fail to develop this new approach we risk to be completely unable to ensure dependability, security and resiliency of safety-critical systems in the near future, at least with the current state of self-adaptive systems.
DESIGN AND PROGRAMMING APPROACHES

DESIGN APPROACHES FOR MASSIVELY PARALLEL COMPUTING

Description

Research on computer architectures has lead to the development of multi-core systems that, however, are rarely used as cooperating cores for performing smart parallel computations. In fact, highly data intensive computations, required in many application domains ranging from autonomous driving to intelligent industrial plants to medical diagnosis and the like, are being performed by distributed processes that cooperate through the network via message exchange. On the long term this approach is not feasible as it is significantly impacting on network performance and is wasting the large amount of resources that are available within each single multi-core processor. For this reason, we need to develop design paradigms and tools (ranging from simple to use design and coding languages to analysis and testing approaches as well as debugging approaches) that take advantage of the parallelization possibilities given by multi-core processors and we need to blend them with those offered by distributed computation.

Impact on technology

Development of novel approaches and tools in this area will determine a dramatic technological advancement and will enable large scale development and adoption of applications that at the moment are still at a prototypical stage.

Impact on society

Society will certainly benefit from the application scenarios that will be enabled by the advancements in this area. Areas such as the organization, control and evolution of complex smart environments (industries, healthcare institutions, homes, ...), data science, autonomous driving, future Internet, will show significant progresses thanks to the improvement of design and programming models for massively parallel computing.

Impact on European leadership

Investments in this area is today still high risk but could give back important results. Support by the European Union could help enterprises in taking the risks, in strengthening the collaboration with European research laboratories that are already active in these fields and in nurturing a new generation of researchers and engineers focusing on addressing high priority problems for our society.

Consequences in case the challenge is not addressed

The full development of massively parallel computing has the potential to enable dramatic advancements in the execution of complex reasoning. Losing the possibility to develop research and innovative solutions in this area will result in the impossibility to increase the cognitive abilities of
software thus excluding the possibility to develop advanced solutions in challenging application areas such as smart environments, autonomous driving and the like.

SOFTWARE FOR QUANTUM COMPUTING

Description
Quantum computing is being successfully used to solve in polynomial time some NP problems. Its most prominent application is in cryptography, but, more generally, it is able to tackle optimization problems where traditional space exploration requires a significant amount of computational power by Turing machine-based computers and can easily lead to local optima. The quantum approach allows us to quickly find the optimal solution as well as a number of good alternatives to it. As discussed in [D-Wave], application domains could range from optimization, e.g., in radiotherapy finding the optimal radiation plan, to machine learning, e.g., detecting the presence of specific objects in an image, to Montecarlo simulations. [Gay 2006] provides a survey of programming languages for quantum computing and others, for instance [Wecker and Svore 2014] and propose an architecture and programming approach. However, the field is still in its infancy both from the perspective of the hardware development -- even if D-Wave has productized a specific hardware implementation, several research labs are performing research on combining a few qubits -- and from the perspective of design, programming and verification approaches, with the available programming languages that, in most cases, appear to be suitable for researchers rather than for average programmers. The challenge is therefore to tackle these issues and to create development approaches, languages and environments that enable the usage of quantum computing as part of complex software development activities.

Impact on technology
The adoption of quantum computing could not be boost without the development of proper programming languages, paradigms and environments. Thus, expanding research on programming languages, verification approaches, and IDEs to quantum computing and to the integrated usage of quantum computers as well as traditional ones, has large potentials in terms of increasing the development and adoption of quantum computing.

Impact on society
Exploiting quantum computing could significantly speed up the execution of complex algorithms that address important issues related to fields like:

- Autonomous driving, for instance, the need for recognizing in real-time the presence of other cars or unforeseen obstacles;
- Health care, for instance, defining personalized plans for treatments that are optimal for the patient given many different variables;
● Support for decision making, for instance, assessing the probability that a certain event will happen under certain initial conditions.

Impact on European leadership

The company that is currently offering quantum computing services, D-Wave is based in Canada. Google and IBM are aiming at developing quantum computers with a significant number of qubits that should demonstrate the advantages of this technology [Courtland 2017], Microsoft has been developing research on the programming model for a long time. In Europe, knowledge in this area appears to be more limited as witnessed by a smaller number of papers written by Europeans in this area. At the same time, the European Union is going to grant significant funding to the development of quantum computers through the Qflagship program. From the information that are available, however, the emphasis of the current funding program is on the development of the quantum computer hardware rather than into the development of environments supporting the development of quantum computing software. Supporting the work in this area will be certainly complementary to the planned programme and will allow quantum computing to become more usable and effective.

Consequences in case the challenge is not addressed

With the availability in the near future of powerful quantum computing machine as claimed by Google, being able to exploit them will certainly be of paramount importance to be able to tackle the complex challenges related to improving the possibility of automating complex tasks. Currently available programming languages and models are not suitable for developing large scale solutions and do not offer tools to verify the effectiveness, correctness, safety of quantum computing software.

References


FOG COMPUTING PROGRAMMING

Description

Fog Computing is emerging as a paradigm aiming to create a continuum between the Cloud and the Edge of the network. While the Cloud provides a theoretical unlimited amount of resources and the Edge a low-latency computing power, Fog aims to seamlessly integrate these two environments able to take advantage of the strengths from both.

Although researchers have started studying the implication of this new paradigm in terms of infrastructure and data management, the need for a specific programming model has not yet been addressed. If on the one side, microservices (or functional programming) could fit in this new heterogeneous, scalable, dynamic environment as the Fog is, the implication of these approaches needs to be further studied in terms of performances, security, privacy.

Impact on Technology

As the Fog Computing implies to rely on a very heterogeneous environment where applications and services can be executed on both computational-limited devices as well as on a virtual environment, programmers can really and easily create polyglot applications relying on a very dynamic infrastructure where, on the one side, the Cloud ensures a set of reliable and powerful resources, while Edge a set of owned, ready-to-use, and quick resources. In addition, as most of the work done so far involved the data transfer from the Edge to the Cloud, Fog Computing can enable the development of applications which can take advantage of the computational power at the edge (e.g., actuators) as well as the data transfer from the Cloud to the Edge when decisions need to be made at the edge.

Impact on Society

Impacts on society can be manyfold. On the one side, with Fog Computing programmers can develop applications and services able to really integrate the mobile and the cloud environments where not only data can be transferred between these two worlds but also the computation. On the other side, this will enable for new solutions in the IoT scenario as more and more integration is required between the devices installed in the so-called smart buildings or smart factories and the cloud environment where the computation must occur due to privacy/security restrictions or latency requirements.

Impact on European Leadership

Fog Computing could increase the adoption of Cloud solutions. Indeed, as one of the reasons why Cloud is not fully accepted in some domains is the need for security and privacy that does not allow to move data to the cloud. With Fog Computing, assuming that the secure data process will always occur on the edge-side, the cloud-side can be exploited to run the other functions in a seamlessly.
Consequences in case the challenge is not addressed

If Fog Computing fails to be accepted, we will risk to have Cloud based solutions with some information coming from the Edge, as well as the Edge solutions where the computation cannot easily take advantage of the Cloud environments.

**BLOCKCHAIN IMPACT IN IOT / SW ENGINEERING**

**Description**

The blockchain technology has the power to make the world more transparent, safer and honest place. It is essentially a tool that records every single transaction and digital event that happen in the virtual world. Blockchain can be considered as the missing link to settle scalability, privacy and reliability concerns in IoT.

For example, blockchain can be used in tracking billions of connected devices, enable the processing of transactions and coordination between devices. It also allows for significant savings to IoT industry manufacturers. Moreover, the cryptographic algorithms used by blockchains would make consumer data more private [1].

Fig1: Visual illustration of blockchain

What do you find inside a block?

In simple terms, the data is contained inside the blocks as well as an arbitrary integer (called nonce) needed to produce the proof of work. In Bitcoin blockchain, the block contains a header and relevant transaction data. A Merkle transaction tree is created and the root hash is included in the header.
A merkle tree is a complete binary tree with a hash value. At the bottom of the tree, each transaction has a number that satisfies the hash value. Then, the tree is constructed so that the parent node has a hash value of the data contained in its concatenated children.

The merkle tree data structure allows validation by constructing a merkle tree from the bottom of the tree to the root node.

In blockchain, the block header has a field for the previous block hash. Therefore, it is possible to construct a string of blocks. Sometimes a fork on the chain of blocks may occur. This is due to two blocks calculated at a very short time interval. In the later process of mining, one fork superior to the other, the longer chain is accepted by the network and the court would not be used unless its length exceeds the long chain in the future.

Impact on technology

Blockchain will allow direct communication between two or more devices, so that they are able to transact without going through an intermediary. It permits users to synchronise devices against a single system of authority that is distributed and censorship resistant. Blockchain brings up special solutions like establishment as a secure means of protecting financial data but flexible enough to be applied to any high stakes. [2]

For instance, security technologies use secure server added to other methods to keep the data safe, but blockchain will significantly reduce the means used to secure the data, using only nodes and algorithms.

Examples of blockchains:

- **EOS** is the first blockchain operating system designed to support decentralized business applications. It designed for large-scale development for large companies. Currently, not taken into account by existing blockchain platforms. EOS introduces asynchronous communication and parallel processing to support millions of transactions per second, higher than Facebook and Google. Moreover, EOS is designed to support distributed applications, without for example requiring the user to pay for each transaction. This allows new blockchain users to engage without having to navigate the process of acquisition of a crypto-money and this should help to develop this sector. This also means that any kind of application can be built, run and managed by the EOS operating system, creating a new level of transparency. The EOS project aims to solve the problem of the scalability of blockchains. The objectives are:
  - Capacity of the blockchain to support millions of users simultaneously
  - Free use
  - Easy update and flexibility in the face of bugs
  - Low latency
  - Capacity of the blockchain to optimize the load in computation so as to sequence and to parallelize the tasks.
  - A blockchain built with EOS will use three resource classes to deploy applications
- Bandwidth and log storage
- The computing power (CPU)
- Storage of the state of the blockchain (RAM)
  - "In order to ensure inclusion, EOS are not sold at a fixed price but at a price determined by market demand. This replicates mining but does not offer unfair advantages to large buyers," said Brendan Blumer, PDG of block.one, the free software company and blockchains that develops EOS. Block.one will distribute one billion EOS ERC-20 compatible tokens, called "EOS tokens" over 341 days starting June 26, 2017 at 13:00 UTC
  - New paradigm of "proof of stake" instead of "proof of work"
  - Regarding the denial-of-service attacks (DOS), this type of attack is when a malicious attacker spams a network with the traffic in order to prevent passing legitimate traffic. In fact, other Blockchain networks such as Ethereum network has been vulnerable to such DOS attacks, whereas EOS should be invulnerable to such attacks. The hackers can only consume the proportion of the network that their EOS tokens give them.

- Ethereum is in one sentence a global computer, which anyone can program and use as he wishes. This computer is always on, it is very secure, and everything that is done using this computer is public. The market cap of the digital currency Ether is at 30 billion Euro, with already 700 million Euro circulating supply. Also, other side-chains are connected via smart contracts.
  - Smart contract is just a phrase used to describe computer code that can facilitate the exchange of money, content, property, shares, or anything of value. When running on the blockchain a smart contract becomes like a self-operating computer program that automatically executes when specific conditions are met. Because smart contracts run on the blockchain, they run exactly as programmed without any possibility of censorship, downtime, fraud or third party interference.
● **Blockstream** presents itself as a blockchain technology company that works to accelerate innovation in crypto currencies, open assets, and smart contracts. Its products include elements, an open-source platform for building and testing applications; and liquid, which is designed for bitcoin exchanges and high-speed transactions. Its core area of innovation is focused around an idea called “sidechains”, bitcoin-like ledgers that operate independently of, but are pegged to, bitcoin. It allows its users to build a separate platform for a specific use, but still have access to the bitcoin blockchain. It was co-founded by Adam Back, Gregory Maxwell and Austin Hill and is based in San Francisco, California.
  ○ The Blockstream Satellite network envisions to broadcasts the Bitcoin blockchain from space, giving almost everyone on the planet the opportunity to join the economic revolution. Also, this enables virtual transmission of solar energy from space to earth as crypto-currency.

● **IOTA** is a new cryptocurrency that focused on Machine-2-Machine (M2M) transactions. The main purpose of IOTA is to serve the machine economy by enabling feeless M2M (Machine-to-Machine) payments, offering open-source, scalable, decentralized cryptocurrency, engineered specifically for Internet of Things real-time micro transactions.
  ○ IOTA goes beyond blockchain, being the first cryptocurrency that provides the whole ecosystem based on Tangle, a distributed ledger which is scalable, lightweight and for the first time ever makes it possible to transfer value without any fees. Contrary to today’s Blockchains, consensus is no-longer decoupled but instead an intrinsic part of the system, leading to decentralized and self-regulating peer-to-peer network.

**Impact on society**

At its most basic, blockchain is a vast, global distributed ledger or database running on millions of devices and open to anyone, where not just information but anything of value – money, titles, deeds, music, art, scientific discoveries, intellectual property, and even votes – can be moved and stored securely and privately. Beyond financial services, blockchain has already disruptive impact in the following domains: authorship, media distribution, ownership, commodities, real estate, data management, energy, digital identity and authentication, gaming and gambling, e-government, supply chain management, job market, reputation management, sharing economy, social networks, IoT, network infrastructure,

On the blockchain, trust is established, not by powerful intermediaries like governments, banks and big technology companies, but through mass collaboration and smart code. Blockchains ensure integrity and trust between strangers and make it difficult to cheat, lowering transfer and interaction fees, while providing a high degree of openeness, transparency and dependability. Also, blockchain technology enable the integration of cyber and physical world by offering a way to represent nearly every asset as digital data.

Many argue that attacks are an inevitable result of how the blockchain is designed and that it is a "great attack surface".

More capacity on the platform means that there are more possibilities of problems, at least compared to other networks of chains of blocks, less ambitious.
Even if nodes no longer collapse completely, however, this results in a network that is more globally ready, making the Blockchain available to anyone who wants to turn a smart contract or send a transaction. Since the attacks, some users have reported difficulties accessing their funds.

One user even observed when switching pools that the profitability of mining has decreased for smaller pools, which is potentially of concern to an ecosystem that does not adapt to larger miners have more control.

The network is also bigger overall and all new domains do not work properly. However, some users seem intact, with many developers continuing to work on other projects. Two projects ethereum, FirstBlood and SingularDTV, had staffing to raise project funds in the middle of the attack.

**Impact on European leadership**

It is still impossible to fully measure the impact of blockchain technology on the financial sector - let's just say the best is yet to come. To go further, we know that the current financial / banking system is based on the notion of trust. Banks deal together whose role is to eliminate counterparty risks by ensuring the settlement and delivery of transactions. It is the systemic vision of banking exchanges.

From the point of view of the bank, as an organization, several processes of control, execution and reconciliation make it possible to ensure the smooth running of events. There is often talk of back-office work which is generally considered as a cost center for the bank.

The blockchain promises substantial savings for the financial industry from a perspective of reducing infrastructure costs for the management of international payments, securities trading and assurance of compliance with regulations.

This is reflected in a recent study by an European bank which estimates savings of between 15 and 20 billion per year by 2022. The Swiss bank UBS considers that blockchain applications Would reduce international payment delays from 3 to 4 days today to 15 seconds tomorrow. The bank explains in its white paper that the blockchain allows to override the various intermediaries intervening in the path of an international payment.

In insurance, the actors, traditionally vigilant against disruptive technologies, Also see in the blockchain opportunities to increase the fight control fraud by faster and more transparent access to data in a more automated way. Audit trail management is enhanced by the fact that the same transaction once performed is shared and distributed across the network, which prevents any modification or fraud.

**Consequences in case the challenge is not addressed**

Compare the bankruptcy of two different types of business: “money-acceptors” (like retailers), and “money-storers” (like banks). Money-acceptors can go bankrupt almost painlessly: Soon a new one will appear to fill the unmet market demand, and you can continue your shopping there (think of
Bitcoin’s “Silk Road(s)”). If you are extremely unlucky, and happen to be shopping mid-bankruptcy, you might permanently lose some value: that of the single transaction you were mid-conducting.

However, if a bank (money-storer) closes, an entire lifetime of savings can be wiped out permanently. If a new bank opened (from where does it get its initial deposits?), users would have to start saving all over again. Money-storage has higher stakes: owners/employees/hackers/lawyers/politicians can target the stored asset for theft, extracting a “ledger rent”. The damage is permanent, and the pain excruciating [7].

References

[5] Conferences TEDx about Blockchain impact on money and economy https://www.ted.com/talks/don_tapscott_how_the_blockchain_is_changing_money_and_business
https://www.ted.com/talks/bettina_warburg_how_the_blockchain_will_radically_transform_the_economy
[6] Benefits to use Blockchain and IoT together

INCLUSION OF HUMANS IN THE DESIGN AND INTERACTION WITH MACHINES

INTERACTION BETWEEN HUMANS AND AUTONOMOUS MACHINES (IN BOTH DIRECTIONS)

Description

In a context where humans are surrounded by autonomous machines and robots, it is of paramount importance to understand how to let the two interact in a proper way. We need to understand what kinds of tools we should offer to humans to program autonomous machines and what programming them, respecting their autonomous capabilities, means. On the other side, we need to ensure that
machines behave “appropriately” in their interaction with humans. This is a multidisciplinary area where software engineering is required, on the one side, to revise the way design and programming paradigms are conceived to enable programmability of autonomous machines. On the other side, it is required to provide analysis and testing approaches to enable the verification and certification of autonomous machines.

**Impact on technology**

The field of robotics, which includes devices supporting factory automation, autonomous driving, smart homes and the like, is now gaining momentum and achieving important results. For this reason, the issues of programmability of autonomous machines and of good machines behavior is becoming of paramount importance. Software Engineering is required in both areas. On the one side, autonomous machines programming should inherit the experience of almost one hundred years of achievements in the programming languages area. On the other side, ensuring “good behavior of machines” is a problem that can be addressed by exploiting formal verification techniques. Of course, in both cases it is not just a matter of applying existing techniques to a new problem, but it requires a profound adaptation and evolution of existing concepts to make them suitable in the new context.

**Impact on society**

This research area is very critical for ensuring a proper and controlled adoption of autonomous machines in many different application domains. The risks of not considering machine-humans interaction as a critical issue in the first case can be very high for human life and for the wellbeing of the entire society.

**Impact on European leadership**

This area is at the edge of technology and has the potential to determine very important achievements in the context of autonomous machines. Considering the primary role of Europe in the areas where autonomous machines are having and could have an important role, a strong participation of Europe in this research area could imply an important strategic advantage.

**Consequences in case the challenge is not addressed**

A well-known Tesla accident occurred in 2016 and caused the death of the car owner [https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk](https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk). The accident occurred because the car was being driven by the autopilot and its sensors could not distinguish a white tractor-trailer against a bright sky. This is a clear example of an autonomous machine improperly assessing the situation and behaving in an incorrect way based on this. Even worse situations could be envisaged in the case the machine selects behavior that purposely harm humans using or around it. While the entire society has to reason on the identification of the right behavior (this is especially true when an autonomous machine has to choose between actions that can be harmful for different sets of people), software engineering can offer the technical means to ensure the execution of the right behavior when needed.
INTEGRATION OF CITIZENS IN DESIGN AND DEVELOPMENT OF PUBLIC E-GOVERNMENT PLATFORMS

Description

The institutions of the European Union as well as the member states are working on digitalization concepts in many domains with many processes already digitalized. Political and social figures have are pushing digitalization in the field of communication and interaction between citizens and state institutions. At the same time, states and communities want to save resources while increasing the quality of the service they offer. Research has shown that user involvement is a major success factor for software projects. Although nonfunctional requirements such as usability, accessibility or security can be implemented mostly without additional contribution, the functional design should be developed together with the citizens to guarantee high service quality and user satisfaction. Thus, methods and processes for a democratic mass participation in software development need to be developed.

Impact on technology

Existing tools should support massive user participation. One example is Requirements Bazaar (http://requirements-bazaar.org)—a prototype tool for social requirements engineering (SRE) which aims at supporting user participation by bringing together communities and service providers.

Tools to systematically gather the voice of users (e.g., through app stores) are already important source of information for software projects. However, such tools do not address the following issues. First, users might have contradicting positions and opinions. Second, community features will soon reach their limits when the amount of users grows.

What is needed is a tool support that offers solutions for a scalable, structured, and democratic approach to gather feedback, contributions, and opinions and distills them to requirements. Both processes and tools are needed to mitigate the challenges of opening requirements engineering to the public—or in other words democratizing requirements engineering—particularly when it comes to the development of E-Government platforms.

The challenges of SRE that will impact technology are:

• Scalability. The quantity of user contributions in social systems imposes limits on how this information can be processed by other stakeholders (e.g., government). With increasing number of users and frequent contributions manual analysis techniques become infeasible. In general, informal data like natural language is hard to analyze automatically because of its high degree of freedom.

• Contribution Quality. If users contribute without professional support, the quality of resulting information is unpredictable, which can lead to misunderstandings. Users might express system properties using their own, possibly heterogeneous and ambiguous terminology.
• Conflicting Contributions. Different users will have different, possibly conflicting opinions, preferences, and expectations. To make decisions from this data, conflicts have to be identified and resolved. Continuous user contributions in social systems can lead to frequent conflicts, complicating manual identification and resolution techniques.

Impact on society

There is an deep socio-political impact in facilitating citizens democratic participation in the design of e-government platform. Governments will achieve social sustainability (on top of ecological and economical one) in the design of its infrastructures.

Citizens’ acceptance of E-government systems will improve as they are active players in the field who are driving the design of the systems they use. This, in turn, will improve trust towards the government due to the increased transparency and democratization of the process.

Increase of citizens and governments responsibilities, as not only the government will be responsible for the well-being of its citizens (e.g., through the services offered by e-government platform), but also the citizens themselves will acquire a sense of responsibility towards other citizens (i.e., other users of the platform) when participating to the platform design.

Impact on European leadership

Due to higher transparency and closer proximity to governmental processes, the European Union will increase the trust between them and the local municipal institutions, its citizens, and among European citizens.

Consequently, the European Union will reinforce its position as a democratic establishment and bring citizens’ representation to software technologies and their development. The Union will set a new example for e-democracy through an endeavour complementary to the existing ones, such as e-voting, already in place. E-government platforms, designed through citizens participations, will set the condition for a wider political self-determination pivotal for a democratic society.

Consequences in case the challenge is not addressed

A lack of citizens integration into the design of E-government platform will have three main consequences:

• Missed opportunity for the government to survey and address people’s needs in a democratic way through new, creative solutions.
• Decreased trust of the citizens which will result in less engagement with the platform and its services.
• The creation of e-government services risk to be influenced (e.g., in type of functionalities, user interfaces) by third-party contractors (e.g., software development agencies located outside Europe) rather than the citizens.
END USERS DEVELOPMENT APPLIED TO IOT

Description

Development of a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artefact.

The advent of massively interconnected objects, devices, and sensors — commonly referred to as the Internet of Things — raises equally massive challenges regarding the interfaces that will allow end-users to manage the complexity of such systems, to exploit the opportunities such technologies open up, and to tailor context-dependent interactive applications for their needs.

Impact on technology

Appropriate authoring environments and architectural infrastructures are needed to support End-User Development in the context of the Internet of Things. New generation of EUD approaches, languages and tools allowing fast personalization of applications, sensors, smart objects and intelligent environments through usable interaction metaphors and multimodal interfaces to identify and edit context-dependent events and conditions and how the application and the connected objects and services should react through different devices and interaction modalities possibly in an effective and persuasive manner. The multimodal authoring environments should support graphical representations, subset of natural language, gestural interfaces, and vibro-tactile feedback to ease end user development of personalization rules, and will be able to identify potential conflicts, and simulate and debug such rules.

Impact on society

Professional developers cannot foresee all the possible and even unpredictable situations that applications would encounter during use and whether the produced results will be actually meaningful to end users. End users assume an increasing number of responsibilities traditionally held by professional developers. As users continue to grow in number and diversity, EUD is likely to play an increasingly central role in shaping software to meet the broad, varied, rapidly changing needs of the world.

- Enable users to take advantage of massive amounts of data accessible through the web.
- Empower end users to create a variety of software on their own, to define and tailor the functions of their systems in order to satisfy their personal, local, and often task-specific needs.
- There is a need for a participatory, collaborative process in which end users, domain experts, and developers contribute with their different expertise at various stages of the process in co-creating the software artefacts to obtain meaningful results in real contexts and address concrete end user needs.
Impact on European leadership

EUD methodologies allow benefiting from the creative power of end users and domain experts that will generate a large set of personalized versions of existing applications and services that will flexibly meet disparate and dynamic needs, and lead to completely new business models, which would positively influence European economy.

Consequences in case the challenge is not addressed

If Europe will not support the EUD approach most of the personalization features available for non professional developers to address and manage emerging technological settings will be in the hands of the main ICT players, which are based outside Europe (e.g. Google, Apple, Facebook).