Computer Standards & Interfaces 57 (2018) 76–82

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Elastic systems: Towards cyber-physical ecosystems of people, processes, and things

Daniel Moldovan, Georgiana Copil, Schahram Dustdar*

Distributed Systems Group, TU Wien, Austria

ARTICLE INFO

Keywords:
Elasticity
Cloud
IoT
Human-based computing

ABSTRACT

Pervasive mobility and an exponential increase in the number of connected devices are adding to IT complexity. Users are bypassing traditional IT to access cloud-based services. Boundaries between computing systems, people, and things are disappearing. New approaches are required to manage today's and tomorrow's increasingly connected and heterogeneous ecosystems of people, computing processes, and things. We envision *future elastic systems driven by business requirements, integrating computing, people, and things in open dynamic ecosystems in which all entities collaborate towards common goals*. We introduce elasticity as a means of integrating computing processes, people, and things. We identify the core computing fields enabling future elastic systems: (i) hardware and software reusability, (ii) smart things, (iv) adaptation, and (v) human-based computing. We look at the development of these fields, and identify fundamental properties for building future elastic systems. We further envision a new field of research: *Elastic Computing*. We identify and discuss challenges to be addressed by this field towards realizing future elastic systems: Are existing programming languages and models sufficient for designing and managing future elastic systems? How important are the interactions between people, computers, and things? Can people and things be monitored and controlled like computing resources?

1. Introduction

Boundaries between computing systems, people, and things are gradually disappearing. Pervasive mobility and an exponential increase in the number of connected devices are adding to IT complexity. Users are bypassing traditional IT organizations to access cloud-based services. Today's landscape of ever changing technologies and customer needs challenge the traditional IT models. IT business models focusing on strategic competitive advantages are being replaced with more agile methods. Innovation and continuous adaptation to changes are crucial for the business viability of today's IT organizations [1]. Today's digital technology has the potential to bring together businesses, software, and individuals, under the umbrella of an integrated digital ecosystem.

New approaches are required to manage today's and tomorrow's increasingly connected and heterogeneous ecosystems of people, computing processes, and things. Existing approaches deal with individual ecosystem components, such as autonomic computing focusing on computing processes, human-based computing on software-human relationships, or cyber-physical systems dealing with systems which span in the physical world. However, when people, computing processes, and things become interconnected in digital ecosystems, they evolve and adapt together, almost as a single entity. Changes in one ecosystem part trigger changes and adaptation in the other components. E.g., an increase in physical devices generating data might demand an increase in both the computing resources, and people processing and analyzing the data. Managing such ecosystems requires a unified end-to-end view and approach, considering the heterogeneity and interactions between system components.

To this end we envision *elastic systems spanning complete business processes, integrating computing, people, and things in an ecosystem in which all entities collaborate towards common goals*. Defined by Dustdar et al. [2], the *Principles of Elastic Processes* bring together three dimensions in managing elastic systems: *resource elasticity*, *cost elasticity*, and *quality elasticity*. These dimensions reflect not only computing-related aspects of elastic systems, but also the business aspects driving IT organizations. Resource elasticity focuses on mechanisms and capabilities for allocating/deallocating computing resources on demand, to align the IT infrastructure to changing load and business needs. Cost elasticity covers the business perspective of software systems, dealing with aspects influencing the cost and cost efficiency of systems. Quality elasticity focuses on capturing and adapting the quality of elastic systems according to business goals and available resources.

* Corresponding author.
E-mail addresses: d.moldovan@dsg.tuwien.ac.at (D. Moldovan), e.copil@dsg.tuwien.ac.at (G. Copil), dustdar@dsg.tuwien.ac.at (S. Dustdar).

https://doi.org/10.1016/j.csi.2017.04.002
Received 28 December 2016; Received in revised form 15 March 2017; Accepted 13 April 2017
0920-5489/

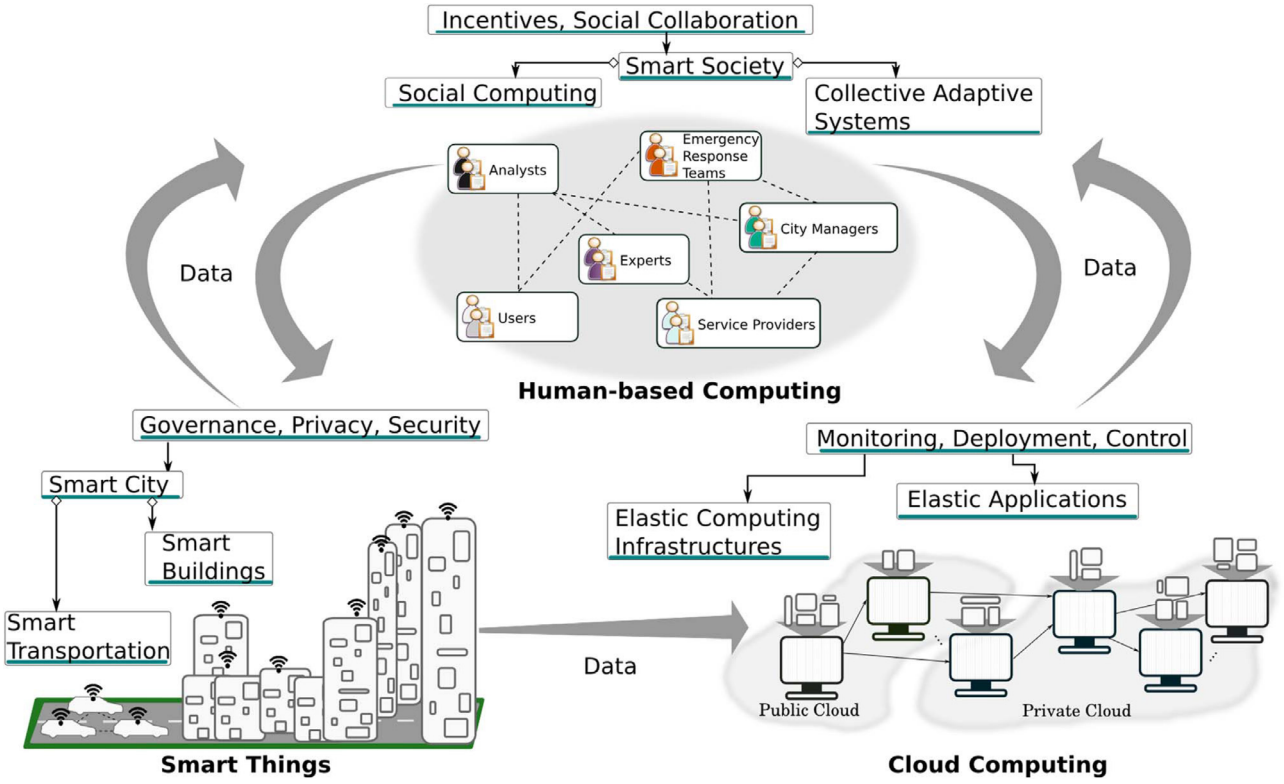

Fig. 1. Towards cyber-physical ecosystems of people, processes, and things.

Following the elasticity principles, elastic systems would capture, manage, and adapt to the needs of all involved actors, irrespective of their nature, from people, to computing processes, and physical devices. We identify and depict in Fig. 1 the three main components of future elastic systems: cloud computing, smart things, and human-based computing. To achieve such systems, we can start from existing approaches in managing individual ecosystem components. Approaches from *Smart Things* allow things to connect, collaborate, adapt, and provide complex computation capabilities. Developments in *Cloud Computing* enable a wide range of computing resources and software to be offered as services accessed remotely over a network. Approaches from *Human-based Computing* provide the foundation for combining people and software in building future elastic systems, leveraging their individual properties and capabilities. Combining smart things, cloud computing, and human-based computing, we can build systems capable of managing computing infrastructures, physical things from the real world, and the people interacting with them.

In the next sections we first look at the future of elastic systems, highlighting their unique properties and capabilities. We then discuss the development through time of the properties and concepts critical for building elastic systems. We look at how concepts such as time-sharing, socio-technical systems, or wearable computers have paved the way towards future elastic systems. We focus on four core research areas: (i) hardware and software reusability, (ii) smart things, (ii) human-based and hybrid computing, and (iv) adaptive, autonomic, and intelligent systems. We outline our vision over a new computing field, *Elastic Computing*, dealing with the study of elastic systems. We discuss the challenges to be addressed by the field towards building elastic systems. We start with interactions between computing systems, people, and things. We discuss if existing programming languages and models are sufficient for capturing and managing the complexity of elastic systems. Finally, we outline and discuss challenges in monitoring and controlling cyber-physical ecosystems of people, processes, and things Fig. 2.

The rest of this paper is structured as follows. In Section 2 we discuss our vision over elastic systems integrating people, processes and things. Section 3 provides an overview over the historical development and state of the art in the four core research areas providing properties and concepts critical for building elastic systems. Section 4 covers the field of human-based computing and discusses its implications over the development of elastic systems. Section 5 outlines our vision over a new research field in computer science *Elastic Computing*. Section 6 concludes the paper Fig 3.

2. Emerging elastic systems

In 2011 Dustdar et al. [2] introduced the *Principle of Elastic Processes*, defining cost, quality, and resources as the basic elasticity dimensions, forming the foundations of elastic systems. Focusing on uniform management of people and computing resources as functional units of the same system, in 2012 Tai et al. [3] introduced the *Design by Units* principle. The principle defines the *Unit* as an abstraction over both people and computing resources. Noticing that people have become entangled in bigger heterogeneous systems, Anderson et al. [4] have defined the concept of *Collective Adaptive Systems*. Collective Adaptive Systems focus on the societal aspects of systems in which people, processes, and things, evolve, collaborate, and function as a part of an artificial society. Based on these recent developments we have a better understanding on how to manage with the help of elastic systems today's and tomorrow's increasingly connected and heterogeneous ecosystems of people, computing processes, and things. In the following we discuss in detail the properties and capabilities future elastic systems must have to achieve this vision Fig. 4.

2.1. Connect multiple computer science fields

We believe our vision comes as a natural consequence of all the historical developments done in computer science so far. Approaches from hardware and software reusability such as service-oriented architectures

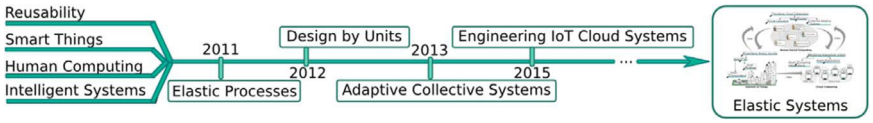

Fig. 2. Emergence of elastic systems.

or cloud computing bring the fundamental concepts and mechanisms for adapting computing processes at run-time to fulfill diverse goals. Approaches from the area of smart things and cyber-physical systems are crucial for combining physical things and software solutions in integrated systems. People can be managed using approaches originating from human computing. Finally, methods and techniques from adaptive, autonomic, or intelligent systems are useful in automating run-time system adaptation. The autonomic vision deals with making systems smarter to the point they do not require human intervention. We start from that vision and also analyze and describe how humans operate and link multiple computing systems. Thus, we can design and analyze larger computing ecosystems which combine humans and computing systems Fig. 5.

Considering this synergy of approaches and technologies from different research areas, to design and build elastic systems, one must be able to apply tools, mechanisms, concepts, and techniques from multiple computer science fields. An elastic system should *leverage and combine developments from multiple fields of computer science*, to achieve its goals. This multi-disciplinary approach provides to elastic systems the *necessary capabilities to adapt and change* with respect to the concerns and requirements of people, processes, and things over both physical and cyber worlds Fig. 6.

2.2. Heterogeneous

Elastic systems include people, processes, and things, heterogeneous entities with individual properties and capabilities. They can further span both computing and business domains, including a variety of use cases. There is a need to manage such heterogeneous units in an uniform way, allowing them to act as functional system units. The *Design by Units* principle [3] allows one to consider and manage both people and computing resources as system *units*. Each unit, irrespective whether it is human or not, exposes in a common way its properties and functionality. Developments in smart things are also important here, towards seamless integration and management of physical things with computing systems. Advancements in ubiquitous computing and cyber-physical systems provide the necessary tools, concepts, and mechanisms for managing things and processes. Advancements in human computing, such as crowdsourcing or adaptive collective systems, provide the means for capturing the properties and capabilities of humans, integrating them as functional units in computing systems.

Thus, *an elastic system is composed of heterogeneous units, i.e., people, processes, and things, working together*.

2.3. Replaceable, self-contained units of functionality

Elastic systems must adapt to changes originating from various sources. They should be able to add, remove, or reconfigure functional units on-demand, depending on requirements. To achieve such dynamic behavior, system units should be loosely coupled. The system's functionality should be distributed between people, processes, and things, in a manner creating high-cohesion units of functionality. Such units should be self-contained for easy replacement, and expose their functional capabilities through well defined interfaces. This provides run-time discovery and replacement of components, depending on requirements. Developments in the area of hardware and software reusability such as component-based systems, service-oriented architectures, or cloud computing can provide fundamental concepts and mechanisms for achieving replaceable software and hardware components. Replaceable human or hybrid human-compute units can be achieved relying on fundamental developments in the area of human-computing. Human-computing implies viewing humans as functional units of computing systems, modeling their input, output, and describing mechanisms for quantifying their output quality. This enables the optimization, assignment, and replacement of humans in larger computing ecosystems just like any other software component.

Elastic systems should be built from replaceable self-contained units of functionality, each unit exposing its functionality through a well defined interface.

2.4. Dynamic perspective

Elasticity is change-driven, and the elasticity capabilities of elastic systems should be considered first class citizens. Elasticity capabilities define how the system may change. The expected impact of each capability on the system's units has to be well understood before enforcing it. Enforcing capabilities can become challenging for complex systems, in which a capability enforcement can produce both desired and adverse effects, depending on the time of enforcement, and particular system implementation. Considering these issues, elastic systems should be designed and built to support adding, removing, or reconfiguring units at run-time. Computing programs should be described and implemented to run interchangeably on computing resources, people, and things, adapting to their functionality, limitations, and particularities. This implies writing programs using abstractions, capturing and describing the inputs, outputs, and behavior for both humans and software components. We believe until now humans where captured more implicit in the design of software systems, and argue an explicit abstraction and description of the human in software terms would enable such hybrid human-software systems to cover end-to-end real-life systems.

Replaceable computing units can be achieved through developments in the area of hardware and software reusability such as component-based systems, service-oriented architectures, or cloud computing. Developments in the area of intelligent systems such as artificial intelligence and autonomic computing have brought techniques for automating the run-time change of elastic systems. Developments from the area of human computing such as crowdsourcing have provided the foundation for composing humans and computing processes in elastic systems.

Elastic systems must have a strong focus on change, from design time, when elasticity capabilities are defined, to run-time and operation, when desired changes occur by enforcing the capabilities of different units.

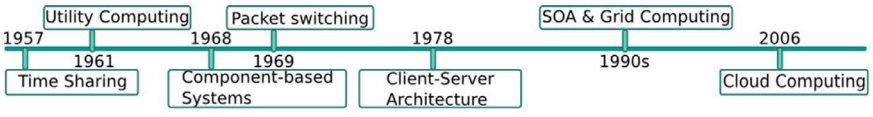

Fig. 3. Hardware and software reusability development.

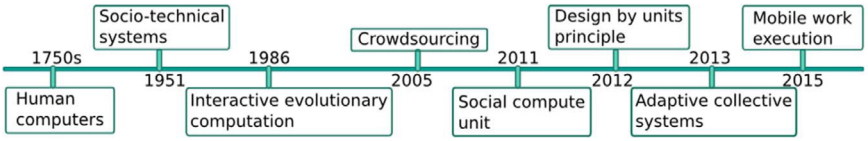

Fig. 4. Human computing development.

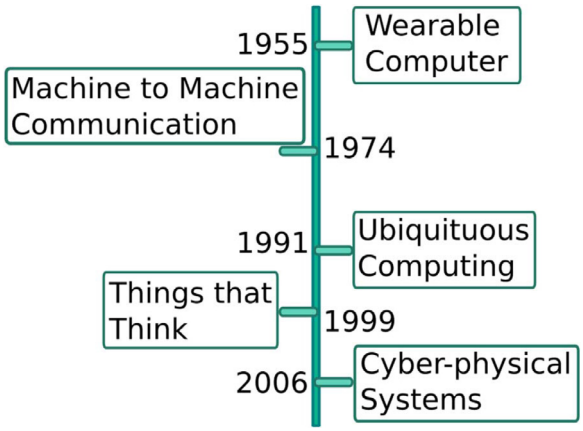

Fig. 5. Smart things development.

2.5. Business perspective

Elastic systems are guided in their evolution by *business requirements* defining what is desired by system stakeholders in terms of behavior and business goals. To manage people, processes and things, stakeholders' requirements must guide the system. The behavior of elastic systems has to be monitored and analyzed with respect to business requirements, over all heterogeneous units that compose it. To fulfill the goals of multiple types of stakeholders, elastic units rely on enforcing elasticity capabilities considering their impact on business requirements. The degree with which a system unit fulfills business goals should be analyzed over all units of the system, and over the system as a whole. Business orientation makes elastic systems suitable for supporting and executing business processes.

Elastic systems must also consider stakeholders' business requirements for achieving desired business goals.

3. Foundations of elastic systems

To realize elastic systems integrating people, processes, and things, we can start from concepts and techniques developed throughout the history of computing. *Hardware and Software Reusability* principles are crucial in supporting dynamic hardware and software run-time reconfiguration. Developments in *Smart things* must be considered in order to properly manage physical things. Developments in *Human Computing* are necessary for managing people as functional units of elastic systems. Finally, properties found in *Adaptive, Autonomic, or Intelligent Systems* are crucial in reducing the complexity of managing cyber-physical ecosystems composed of people, processes, and things. In the following we take a look at the development through time of the properties and concepts critical for building elastic systems.

3.1. Software and hardware reusability

Today we find a large array of computing functionality exposed as a service, accessible over a network, under different pricing schemes. This provides the necessary capabilities for reusing and replacing computing services, crucial in building elastic systems which can change at runtime.

Hardware and software *reusability* has gradually developed through the history of computer science. In 1957 Bob Bemer introduced the concept of *Timesharing* [5]. The timesharing principle enables many users to run simultaneous tasks on the same machine. As the timesharing principle was gradually implemented in mainframe computers, the vision of *Utility Computing* has emerged in the beginning of 1960s, promoting the idea of organizing computers as a public utility, similar to telephone companies [6,7]. The utility computing principle has heavily influenced developments in computing systems in the next years. In 1965 the Multics operating system was developed by Corbató et al. [8] to meet all requirements of a computer utility, and was designed with elasticity in mind. It could grow and shrink its computing power by varying the number of processor units or the configuration of drum and disk equipment.

In the meantime, work to connect geographically distributed computing systems lead to the emergence of ARPANET in 1969, the first computer network to implement TCP/IP, later becoming the foundation for today's Internet [9]. The possibility to interconnect previously isolated systems has enabled the distribution of computing functionality between separate machines. As computing systems became more and more complex, so did their programming. Software was written unstructured, and hard to test, extend, or reuse. This has lead to the first NATO Software Engineering Conference in 1968, with the aim of promoting *software engineering*. In this conference Douglas McIlroy introduced the idea of *component-based software systems* [10]. His vision focused on mass produced reusable software components which can later be combined into larger systems. Software started to be designed with reusability and distribution in mind, leading to the emergence of the two-tier *Client-Server* architecture in 1978 [11]. Client-Server architectures distribute functionality in two components: a client and a server. Unlike component-based systems which had as result integrated systems, a Client-Server architecture has two distinct software components, which can run on separate machines. The Client-Server architecture has marked an important point in the history of reusable software, marking the beginning of systems composed of standalone units of functionality which communicate over a network. The 1990s brought the emergence of the *Service Oriented Architecture (SOA)* [12], promoting dynamic discovery and composition of functionality exposed as a service. In SOA systems are able to discover and use services through their APIs, without requiring knowledge about their implementation.

In parallel with developments in software reusability, work was done on increasing hardware reusability. In the 1990s, *Grid Computing* emerged as a means of making computing as easily accessible as an

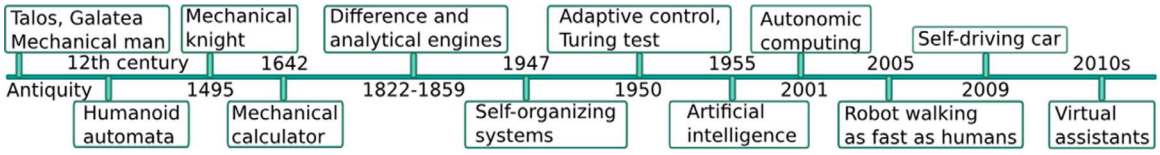

Fig. 6. Adaptive, autonomic, intelligent systems development.

electric power grid [13]. Grid users where able to access large pools of computing resources as long as needed, and release them when no longer required. In 2006, Amazon introduced its Elastic Compute Cloud[1], offering computing resources under a pay-per-use model. Amazon popularized the way to do computing by allocating/deallocating on-demand resources hosted by a third party. This will end up to be called *Cloud Computing* [14]. More business-oriented than grid computing, cloud computing focuses on providing easy access to computing resources under pay-per-use pricing models.

This brings us to today's state of the art in hardware reusability. Today's computing resources are exposed as services. We find a wide range of cloud vendors providing almost anything under the form of a service in a pay-per-use manner, from hardware Infrastructure as a Service (IaaS), to Platform (PaaS), and Software as a Service (SaaS). Cloud users can discover and use these services, and change them at run-time to fulfill certain requirements. Advancements in hardware and software reusability provide the necessary capabilities to change and replace system components at run-time, a crucial property in building elastic systems.

3.2. Smart things

The concept of *smart things* has steadily developed in time, increasingly adding computing capabilities to physical objects. Today we find computing capabilities embedded in everyday physical things, from sensors to household appliances. The computing capabilities of physical things are fundamental for realizing elastic systems able of performing computation over people, processes, and things.

The first wearable computer can be traced to 1955 a small device for predicting roulette results [15]. As devices evolved, so did the need for them to be smarter. We can trace the beginning of smart things to the *machine to machine* concept. The concept was coined during 1974 by Theodore G. Paraskevakos in the context of caller identification for telephone communications [16]. In order for the telephone to be able to read the caller's telephone number, it must possess intelligence. With developments in computing devices and miniaturization, *Ubiquitous Computing* emerged around 1991 as "a vision for activating the world" by providing hundreds of wireless computing devices of all scales, for each person and office [17]. This vision has become almost reality in today's world, in which people are surrounded and aided by various devices performing computing. Building on this trend of connecting everything, in 1999 Gershenfeld et al. [18] presented the concept of integrated smart daily life objects. Their vision focused on the importance of things being *smart*. Through smart things, a world is envisioned in which everything from shoes to books can communicate and exchange data. Up to this point the visions where focusing on connecting and making things smart. Around 2006, researchers predominantly from real-time and control systems coined the term *Cyber-physical Systems* to describe the important area at the interface of the cyber and physical worlds [19]. In such systems, embedded computers and networks monitor and control the physical things, and vice versa. Advancements in the development of smart things such as remote communication and distributed ubiquitous computation are crucial in building future elastic systems connecting physical things and computing infrastructures.

4. Human-based computing

Human computing systems have emerged from the need for well-defined interactions between humans and computing systems. Capturing the interactions between people and computing systems have been one of the focuses in computer science research throughout time. Even before the invention of electronic computers, complex mathematic computations were necessary in other fields, such as astronomy. The first recording of a very complex computation, and its division of labor, was undertaken by Clairaut, Lalande and Lepaute [20], for computing the trajectory and expected date of return of the Halley comet. Persons undertaking such laborious computations were called computers. In World War II, so-called *Human Computers* played an essential role through the computations undertaken in laboratories for determining artillery trajectories or end-points for aerial bombings (e.g., Pearson's Biometrics Laboratory in UK, or Aberdeen Proving Ground in US). During that time, *Socio-Technical* systems were introduced, capturing interactions between society's complex infrastructures and people. However, for the first general-purpose electronic computers (e.g., ENIAC), people were just in charge of programming and interpreting computing results [21].

To allow humans to better interact with computers and control them, there has been sustained effort for improving computer-human interactions. Currently, people can be contacted by computers through e-mail or file synchronization. Such advances have given raise to online staffing platforms and crowdsourcing Internet marketplaces which use human intelligence to perform tasks of various complexity, such as Elance[2], oDesk[3], or MTurk[4].

Using people to solve small tasks that are unfit for computers is not a new concept. Fields such as *Interactive Evolutionary Computation (IEC)* employ people to evaluate fitness functions that cannot be easily expressed in a computer-understandable manner [22], such as personal preference on visual appeal or attractiveness. The term *Crowdsourcing* was coined in 2005, referring to the process of obtaining ideas, services, or content through contributions from a large group of people. The *Social Compute Unit*, defined by Dustdar et al. [23], goes further and includes human and software-based computing in a unified framework allowing instantiation and control of both human and software-based services. One step further towards the tighter integration between people and machines was made through the development of mobile work execution solutions. One such solution is Jennifer[5], a mobile application that guides workers in warehouses, telling them what to do in each point in time, for increasing their productivity.

From the human computer era to these days, progress has been made towards using better the particular capabilities of both people and computers. Today one can build systems composed out of both humans and computers, in which all components, regardless of type, work for a common goal. Such advancements are crucial in integrating and combining people with software processes, towards building future elastic systems connecting people, processes, and things.

4.1. Intelligent systems

Creating systems that through automation or intelligence improve people's lives has been a long standing dream of humanity. Today we find various degrees of intelligence and adaptation embedded in every physical device and software. Approaches from adaptive, autonomic, and intelligent systems are fundamental in enabling elastic systems to adapt to changing needs and requirements.

In Antiquity, the Greek mythology created Talos[6], a giant android made of bronze, which was supposed to protect Europa. In the same period, in China Yan Shi is said to have designed for King Mu of Zhou a mechanical man [24], a construction of leather and wood which walked and sang. In the 12th century, Ismail al Jazari [25] designed human-like automatons, with a focus on increasing people's comfort through

[1] https://aws.amazon.com/about-aws/whats-new/2006/08/24/announcing-amazon-elastic-compute-cloud-amazon-ec2-beta.

[2] https://www.elance.com.

[3] http://odesk.com.

[4] https://www.mturk.com.

[5] http://www.lucasware.com/jennifer-mobile/.

[6] http://www.greekmythology.com/Myths/Creatures/Talos/talos.html.

mechanical aides (e.g., robotic waitress, musical robot band). Hundreds of years later, in 1495 Da Vinci sketched a mechanical knight[7], and described mechanisms for its movement. In 1642, Blaise Pascal introduced the first *mechanical calculator* [26]. The first steps towards programmable computers where done by Lovelace and Babbage [27], who envisioned writing a method for calculating a sequence of Bernoulli numbers.

Noticing that computing systems are getting more complex, in 1947 by Ross Ashby defined *Self-organizing Systems* [28]. A self-organizing system, no matter if natural or artificial, is defined as a dynamic system, capable of automatically changing from a bad organization to a good one. Three years later, Turing discusses what do intelligence and thinking mean for a computing system [29]. Turing highlights that an intelligent machine should be able to adapt and take decisions such that it would fool an outsider into thinking the decisions are taken by a human. The same year, 1950, NASA produced the first proposal of *Adaptive Control* in the context of autopilots for the aerospace industry [30]. In 1955 "The Dartmouth Conference" [31] introduced the term *Artificial Intelligence*, laying the foundations for a new research area dealing with intelligent systems. The conference introduced multiple areas related to intelligent systems, areas still relevant today, such as natural language processing, neuron nets (i.e., now neural networks), and self-improvement.

During years of development, the level of complexity in computing systems increased, systems becoming more and more difficult to manage by people. As a means of tackling the ever increasing complexity of managing computing systems, IBM defined in 2001 the *Autonomic System* [32]. An autonomic system should exhibit self-awareness, self-configuring, self-optimizing, self-healing, self-protecting, and should be context-aware, open and participatory. The principle of autonomic systems was applied to many fields, from computing to robotics, where it enables robots to act independently. Example of autonomic robots are the ASIMO robot[8], that in 2005 was able to walk as fast as humans, or MIT cheetah [33] that is able to run and jump over hurdles. More recent examples are autonomous cars (e.g., Google's self-driving car[9]), or virtual assistants (e.g., Siri[10], Cortana[11]).

Principles developed in adaptive, autonomic, and intelligent systems are fundamental towards realizing elastic systems capable of adapting to changing needs and requirements of highly heterogeneous entities such as people, processes, and things.

5. Towards elastic computing

The individual properties of elastic systems described in Section 2 can be achieved through approaches developed in particular computing fields, as highlighted in Section 3. However, to realize elastic systems it is not sufficient just to combine these approaches. Instead, new concerns must be addressed, originating from the heterogeneity and dynamic behavior of elastic systems. To this end we identify the need for a new *Elastic Computing* field, dealing with the study of elastic systems and addressing their particularities. This new field should analyze and define the interactions between people, computers, and things, crucial in realizing systems interconnecting them. Defining computing processes executing on top of people, computers, and things should be investigated, towards leveraging the heterogeneity of elastic systems. Ethical aspects should be considered and addressed in the context of monitoring and controlling people and things. In the following we layout our initial ideas and challenges to be answered in addressing these concerns.

5.1. How important are the interactions between people, computers, and things?

When computation becomes distributed among independent resources (e.g., distributed computing, parallel computing), the communication and interactions between resources become crucial in achieving the computation's goals. Similar, when computation is distributed among people, computers, and things, the interactions between such heterogeneous units must be well-defined and understood. Several well defined and developed communication principles, such as message passing, can be applied to any such unit. However, the communication mechanism typically differs with each individual unit. People might use e-mail for message passing. Computers would use message queues. In turn, things could use both, depending on the message target. The types of interactions between people, processes, and things are still not fully defined. Each of these units has particular properties and capabilities, which determine the type and duration of each interaction. The type and duration of each interaction must be well-defined, for interacting parties to consider them in internal processes. The interactions should cover use cases present in both the physical and cyber-worlds, dealing with their specific concerns. Elastic computing should analyze and address the challenges in managing interactions between people, computers, and things.

5.2. Are existing programming approaches sufficient for enabling elastic systems?

Elastic systems connect people, processes, and things, spanning business domains. To create programs running on elastic systems, the characteristics and properties of each system unit should be described, and taken into consideration. However, existing programming languages usually consider people as system users, and not as functional system units. Thus, further research is required to understand how much information a programmer should understand and capture using programming languages about the type of units executing particular phases of elastic programs. Future programming languages should achieve a level of elasticity in which the same program phases could be executed by people, computers, or things. The interactions between various system units should be described, defining how a program designed for elastic systems should behave when a system unit fails to execute a task. Care must be taken to consider the heterogeneity of system units. One must be able to capture interaction scenarios between people, processes, and things, such as how we would expect a computing process to behave when it is waiting for a human who failed to execute or report on its task (or the inverse scenario). Another issue is describing business requirements, and ensuring system compliance. Existing approaches such as SLAs can provide a starting point. However, there is a need for tighter integration between high-level business objectives and the properties of elastic systems. Most importantly, SLAs covering human computing units should be considered and integrated with software level SLas, enabling hybrid human-software systems to adapt and change with changing business goals and requirements. Novel programming languages and models need to be developed in the field of elastic computing to address the needs of such new systems.

5.3. Can people and things be monitored and controlled like computing resources?

Even only considering computing resources, their monitoring and control is challenging in the context of elastic systems. In elastic systems, units can appear, disappear, or be reconfigured at run-time depending on requirements. Adding people and things further increases the complexity [34]. People, things, and computing resources act, report, and react in different ways. They have different capabilities, properties, their actions being measurable with different metrics. For

[7] http://history-computer.com/Dreamers/LeonardoAutomata.html.
[8] http://asimo.honda.com.
[9] https://www.google.com/selfdrivingcar.
[10] http://www.apple.com/ios/siri/.
[11] http://windows.microsoft.com/en-us/windows-10/getstarted-what-is-cortana.

computing resources, traditional resource usage metrics such as CPU usage might be appropriate. For things, battery life might be a crucial metric. Humans might be evaluated on entirely different metrics, such as trust, reliability, or accuracy. Such heterogeneous system units have individual run-time change capabilities, which impact their systems in different ways. New monitoring mechanisms and systems must be developed, capable of dealing with this heterogeneity. Monitoring should depart from just collecting metrics, and also analyze the behavior of complete elastic systems, and their individual units. Monitoring should be able to monitor any system unit, using appropriate mechanisms for people, computing resources, and things. Control of elastic systems must consider their particularities, such as heterogeneity, replaceable units, and business requirements orientation. Elastic systems should capture and understand how different types of units react to control actions, and plan accordingly. Ethical aspects should not be neglected. The type and amount of monitoring with respect to privacy and security aspects should be considered. E.g., do we monitor all tasks the people perform, or only the quality of the final result? Novel mechanisms should be investigated in elastic computing for monitoring and controlling people, things and computing resources, considering their heterogeneity and ethical aspects.

6. Conclusions

Elasticity is the means of managing today's and tomorrow's increasingly connected and heterogeneous systems consisting of people, computing processes, and things.

However, elasticity requires a change of perspective. Designing and managing elastic systems implies considering and embracing heterogeneity and change at every stage in their development. Future elastic systems should be designed with architectures that provide necessary capabilities for adding, removing, and replacing functional units at runtime. Future control mechanisms should be able to understand the particularities of people, processes, and things, and exploit them to their maximum potential.

Elasticity is also required from the perspective of the techniques, methods, and processes used to design and manage future IT systems. Previous developments from multiple areas of computer science provide the building blocks for future cyber-physical ecosystems of people, processes, and things. However, it is not enough to only rely on approaches from individual computing areas. Techniques, methods, and processes developed in different areas of computer science should be combined and applied. Elastic systems should be designed with hardware and software reusability in mind. They should consider and capture the particularities and capabilities of humans interacting with them. They should include smart things, capable of executing partially or completely computing processes.

Elastic systems also bring new research challenges to be addressed in a new emerging computer science field dealing with the study of elastic system: *Elastic Computing*. Novel models and techniques are needed for capturing and understanding the relationships between people, processes, and things. Novel programming approaches are required for specifying computing processes spanning people, processes, and things. Novel mechanisms for monitoring and controlling people and things are required, considering particular ethical aspects. These and other challenges remain to be addressed in the future of elastic computing.

References

[1] R. McGrath, A. Gourlay, The End of Competitive Advantage: How to Keep Your Strategy Moving as Fast as Your Business, Harvard Business Review Press, 2013.

[2] S. Dustdar, Y. Guo, B. Satzger, H.-L. Truong, Principles of elastic processes, Internet Comput. 15 (5) (2011) 66–71, doi:10.1109/MIC.2011.121.

[3] S. Tai, P. Leitner, S. Dustdar, Design by units: abstractions for human and compute resources for elastic systems, Internet Comput. 16 (4) (2012) 84–88 ⟨http://dx.doi.org/http://doi.ieeecomputersociety.org/10.1109/MIC.2012.81⟩.

[4] S. Anderson, N. Bredeche, A. Eiben, G. Kampis, M. van Steen, Adaptive Collective Systems – Herding Black Sheep. ⟨http://dx.doi.org/http://www.focas.eu/documents/adaptive-collective-systems.pdf⟩.

[5] B.R.W., Automatic Control Magazine, Ch. How to Consider a Computer, MIT Press, 1957, pp. 66–69.

[6] R. Manna, H. Waldburger, D. Whitson, The emergence of the computer utility, in: Proceedings of the Spring Joint Computer Conference, ACM, May 16–18, 1972, pp. 827–831.

[7] S. Garfinkel, H. Abelson, Architects of the Information Society: 35 Years of the for Computer Science at MIT, MIT Press, 1999.

[8] F.J. Corbató, V.A. Vyssotsky, Introduction and overview of the multics system, in: Proceedings of the Fall Joint Computer Conference, AFIPS '65 (Fall, part I), ACM, New York, NY, USA, 1965, pp. 185–196. ⟨http://doi.acm.org/10.1145/1463891.1463912⟩.

[9] S. Lukasik, Why the arpanet was built, Annals of the History of Computing, IEEE 33 (3) (2011) 4–21, doi:10.1109/MAHC.2010.11.

[10] D. Mcilroy, Mass-produced software components, in: J.M. Buxton, P. Naur, B. Randell, (Eds.), Proceedings of Software Engineering Concepts and Techniques, NATO Science Committee, 1969, pp. 138–155. URL ⟨http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF⟩.

[11] Operating Systems: Theory and Practice, North-Holland Pub. Co., IRIA and Carnegie-Mellon University, Rocquencourt, France, 1979.

[12] T. Erl, Service-Oriented Architecture: Concepts Technology, and Design, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[13] I. Foster, C. Kesselman, (Eds.), The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[14] P. Mell, T. Grance, The NIST Definition of Cloud Computing, Tech. rep., 2011.

[15] E.O. Thorp, The invention of the first wearable computer, in: International Symposium on Wearable Computers, ISWC '98, IEEE Computer Society, Washington, DC, USA, 1998, p. 4. ⟨http://dl.acm.org/citation.cfm?Id=857199.858031⟩.

[16] P.T, Apparatus for generating and transmitting digital information, US Patent 3,812,296.

[17] M. Weiser, The computer for the 21st century, SIGMOBILE, Mob. Comput. Commun. Rev. 3 (3) (1999) 3–11 (URL ⟨http://doi.acm.org/10.1145/329124.329126⟩), doi:10.1145/329124.329126.

[18] N. Gershenfeld, When Things Start to Think, Henry Holt and Co., Inc., New York, NY, USA, 1999.

[19] K.-D. Kim, P. Kumar, Cyber-physical systems: a perspective at the centennial, in: Proceedings of the IEEE 100 (Special Centennial Issue), 2012, pp. 1287–1308.

[20] D.A. Grier, Human computers: the first pioneers of the information age, Endeavour 25 (1) (2001) 28–32.

[21] S. McCartney, ENIAC: The Triumphs and Tragedies of the World's First Computer, Walker & Company, 1999.

[22] R. Dawkins, The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe without Design, WW Norton & Company, 1986.

[23] S. Dustdar, K. Bhattacharya, The social compute unit, IEEE Internet Comput. 3 (2011) 64–69.

[24] B. Mazlish, The Fourth Discontinuity: the Co-evolution of Humans and Machines, Yale University Press, 1995.

[25] J.S. Badeau, J.R. Hayes, The Genius of Arab Civilization: Source of Renaissance, The MIT Press, 1983.

[26] F. Kistermann, Blaise pascal's adding machine: new findings and conclusions, Ann. Hist. Comput. IEEE 20 (1) (1998) 69–76.

[27] J. Fuegi, J. Francis, Lovelace and babbage and the creation of the 1843' notes', Annals of the History of Computing, IEEE, 25(4), 2003, pp. 16–26.

[28] W.R. Ashby, Principles of the self-organizing dynamic system, J. Gen. Psychol. 37 (2) (1947) 125–128.

[29] A.M. Turing, Computing machinery and intelligence, Mind (1950) 433–460.

[30] Z.T. Dydek, A.M. Annaswamy, E. Lavretsky, Adaptive control and the nasa x-15-3 flight revisited, Control Syst. IEEE 30 (3) (2010) 32–48.

[31] J. McCarthy, M.L. Minsky, N. Rochester, C.E. Shannon, A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, august 31, 1955, AI Magazine, 27(4), 2006, p. 12.

[32] P. Horn, Autonomic Computing: Ibm's Perspective on the State of Information Technology.

[33] S. Seok, A. Wang, M.Y. Chuah, D. Otten, J. Lang, S. Kim, Design principles for highly efficient quadrupeds and implementation on the mit cheetah robot, in: International Conference on Robotics and Automation, IEEE, 2013, pp. 3307–3312. ⟨http://dx.doi.org/10.1109/ICRA.2013.6631038⟩.

[34] O. Scekic, H.-L. Truong, S. Dustdar, Incentives and rewarding in social computing, Commun. ACM 56 (6) (2013) 72–82, doi:10.1145/2461256.2461275.