Continuous, Trustless, and Fair: Changing Priorities in Services Computing

Stefan Tai

TU Berlin, Germany

Abstract. Services computing research and practice traditionally has focused on the objectives of business alignment, software systems interoperability and on leveraging the Web as a compute platform. Corresponding technology solution stacks and architectural styles have been promoted. Today, and probably for the next decade to come, different objectives are replacing these original ones and, correspondingly, different solution stacks and architectural styles are emerging. Most notably, challenges such as frequent delivery of service systems, decentralization and business disintermediation, and "socially aligned" service systems lead us to continuous computing, trustless computing, and fair computing – three major trends that we expect to become the driving force behind next-generation service systems. In this paper, we discuss these trends and identify major research directions to deliver on these changing priorities.

Keywords: continuous computing, trustless computing, fair computing

1 Introduction

After 15+ years of research and practice the services computing community is undergoing a fundamental change: newer technology is replacing older technology, research efforts of the past without any significant impact to-date are discontinued, and new research challenges are appearing. In this invited paper, we argue that the primary objectives of services computing are changing – from business alignment, software interoperability and web computing initially to continuous computing, trustless computing, and fair computing today and tomorrow – and that the community consequently must part from older themes and instead focus on addressing current and future priorities.

2 Services computing – A brief historical sketch

In the early 2000s, a time where XML was popular and interoperability of heterogeneous software components and systems was a priority objective, WS-* was born. SOAP and WSDL, along with the manifold WS-* specifications addressing all kinds of enterprise concerns, were promoted as industry standards pushed by large corporations. Correspondingly, the services computing research

community explored service systems from a variety of angles related to the primary objectives of software interoperability and web computing, driven by the idea of establishing a rich computing model based on the services abstraction to well-align IT services and business services. In the mid- and late 2000s, themes including service discovery, (business process-driven) service composition, and service semantics were on the research agenda, with WS-* being the natural choice for proof-of-concept and implementation.

At around the same time, REST emerged as the more lightweight computing model and alternative to WS-*. REST is an architectural style using ubiquitous, foundational web technology like HTTP. REST thereby defines architectural constraints but, unlike WS-*, does not introduce new technology standards. Less motivated by interoperability concerns especially of large IT corporations, REST focuses on leveraging the Web and its manifold resources for purposes of services computing.

Today, especially from a research perspective, WS-* is mostly history and best remembered as a set of XML specifications, which encode proven principles of enterprise computing, but which also tend to (invite to) introduce computational overhead. While SOAP and WSDL are still in use in many enterprise systems, the majority of the WS-*-specifications did not have any significant practical impact and – due to their focus on standardizing interoperability concerns – need no longer be subject of current or future research.

REST, on the other hand, today is by far the more popular services computing model. A large body of best practices is available, making REST a commonly applied and principally well-understood computing model. Few if any critical REST-specific research challenges are left open.

A third, more recent trend in services computing are microservices. Driven by the need to ease change management, microservices must be seen in the context of DevOps-based organizations: they directly link the software service artifact to the development and operations team that builds and manages the artifact – an aspect that both WS-* and REST have ignored – and emphasize communication between different teams by means of APIs. Like REST, microservices are not about standards, but about architecture. Unlike REST, microservices promote an architectural decomposition into individual business functions, where each business function may be a full vertical cut across multiple system layers including the data and resource management layer. Microservices thereby loosely couple the business functions, but tightly couple business logic and data management.

Yet another difference lies in the deployment and runtime environments that the three services computing models propose: WS-* advocates a traditional enterprise middleware environment, REST relies on the web itself, and microservices lend themselves naturally to cloud systems, especially deployments using container technology. Microservices can be seen as a native cloud-based services computing model, whereas WS-* and REST describe models that were originally developed and proposed prior to the cloud-era.

3 Some lessons learned

WS-*, REST, and microservices describe three different services computing models. WS-* is mostly history, REST is current engineering practice, and microservices, along with related concepts of lambda-services and serverless architectures, are gaining momentum. Before we discuss future research directions, we can conclude:

1. Services computing fundamentally is about architecture – "making non-trivial decisions that are documented and are based on a clear rationale" [4]. Such decision-making is influenced by the services computing model chosen and the corresponding objectives associated with the model. The architectural principles are what drives and distinguishes service systems.
2. Architecture does not need complex and rich technology standards; the most basic and simple standards suffice. The engineering, the proposition and the use of rich standards, correspondingly, is not critical to service architectures and need not be on the services research agenda.
3. Architectural constraints change as the service engineering culture and technology evolves. With WS-*, interoperability was a priority objective. REST put web principles to the front. Microservices emphasize ease of change management. With different priorities in mind, different architectural solutions have been and will continue to be born.
4. Future services computing models will natively reflect advancements in computing infrastructure.

4 Research Ahead: Changing Priorities

We observe three major trends in services and cloud computing, which each replace former thinking and former priorities with newer thinking and newer priorities: *continuous computing*, *trustless computing*, and *fair computing*. These three trends, individually and in combination, reflect changing needs: frequent, i.e. almost 'continuous' delivery of systems, disintermediation of businesses and decentralized applications, and "social alignment" beyond "business alignment".

4.1 Continuous computing

Continuous computing emphasizes the need to continuously, i.e., frequently, deliver a system. Continuity requires new engineering processes for delivery and a high degree of automation with appropriate tooling, along with organizational models that support these processes. In practice, "reducing the time between committing a change to a system and to place the change into normal production, while ensuring high quality" [1] – multiple times a day – is a critical requirement. Architectural abstractions in support of an effective change management consequently are a top priority. We can identify at least the following main research challenges:

- *Engineering microservices-based architectures.* Microservices, born out of DevOps-based organizations with continuous delivery pipelines, and related concepts of lambda services, describe different building blocks for service-oriented architectures than the traditional WS-* or REST services. Their tight integration with organizational and delivery aspects makes them a natural candidate to support continuous computing. An integrated approach to the design and use of microservices, their management in cloud-based deployment and runtime environments, and their role in the organizational context and continuous delivery processes is required.
- *Cloud service benchmarking.* Frequent changes and continuous delivery require evidence-based quality control and management. With cloud service benchmarking, we refer to recurring quality-oriented experimentation and analysis of services deployed in cloud environments, for the purpose of discovering quality insights otherwise unknown [2]. Cloud service benchmarking, in addition to functional testing and monitoring of production systems, is critical to understanding service systems and to both justify and guide system changes in continuous computing.

4.2 Trustless computing

In the past years, much attention was paid to trust in computing and trust models for services and cloud systems. This was largely driven by fear or risk aversion when outsourcing computing and data to external service and cloud providers. Trustful computing then suggested architectures that require complex security protocols, intermediation and, typically, some central authority to manage and/or warrant 'trust'.

Trustless computing deliberately breaks with such thinking and promotes decentralized solutions for the correct execution of 'transactions'. Unlike transactions as known from database systems, in trustless computing, no transaction manager and no concurrency and coordination control exist, but symmetric shared responsibilities, including transaction validation, by any node participating in the network. Peer-to-peer systems employing decentralized consensus protocols remove centralized control and allow for new forms of business disintermediation. Note that the term 'trustless' does not imply a lack of trust, but similar to the terms 'stateless' or 'serverless' in services computing, a change in perspective in how trust (or state, or servers) is managed.

To this end, one major research challenge stands out: *Blockchain-based application architectures.* Blockchains are decentralized, immutable ledgers for verifying and recording 'transactions'. Originally proposed along with the bitcoin cryptocurrency [5], blockchains today are the prime candidate solution for trustless computing in any application domain where trade occurs, and whenever trust is to be ensured through peers, but not by some central authority. Blockchains currently experience intensive debate and hype. We agree that there is a huge potential associated with blockchains to disruptively change entire application domains, but argue that a careful selection of application domains and much

more experimental research is needed. Blockchain-based applications are inherently distributed systems, and a distributed systems perspective is fundamental to building applications using blockchains. Solutions to deal with the typical fallacies of distributed computing are needed.

4.3 Fair computing

Third, we observe that services computing no longer is driven by business thinking alone, but increasingly also by aspects of social awareness and social responsibility. For example, complex challenges such as *privacy* go well beyond business concerns but must focus on the human individual or group as the main stakeholder. We refer to this trend as *fair computing*, deliberately calling out for a modern computer science notion of *fairness* that may draw from fairness as studied in other scientific communities, especially law and economics. Research in fair computing demands at first two strands:

- *Fair Information Practices.* Different fair information practice principles have been around for decades, including those published by the US Federal Trade Commission [3]. These may serve as a first step and as general guidelines for fair computing in today's and tomorrow's service systems – covering aspects of transparency, choice and consent, and information review, correction and protection. Nevertheless, we expect refinements to be necessary as digitization continues to transform every aspect of life with unprecedented speed and impact. Privacy is more a "social alignment" challenge, complementing the general "business alignment" objective that services computing traditionally has focused on.
- *Trade-off management.* Dealing with complex challenges such as privacy inherently induces dealing with conflicting objectives within such challenges. Typical trade-offs relate to 'anonymity versus accuracy' or, from a distributed systems perspective, '(desired) security (levels) versus (acceptable) performance (impact)'. In addition, 'fairness' itself is often regarded to be in a trade-off relationship with 'efficiency', typically, in the context of resource allocation problems. Balancing and overcoming trade-offs at different levels of abstraction is hardly possible in a generic way, but typically requires system/application-specific exploration that is evidence-based using quantifiable objectives and corresponding benchmarking methods.

5 Next steps

The three trends of continuous computing, trustless computing, and fair computing described above share significant commonalities. First, core principles of peer-to-peer computing are prominent in all three trends. Second, all three trends are potentially highly disruptive in nature, replacing older technology stacks and former architectural thinking with different technology stacks and newer thinking. Third, they all build on a notion of a 'distributed service', where each service

is tightly associated with critical, non-technical responsibilities – organizational aspects in continuous computing, independent validation in trustless computing, and compliance to fair practice principles in fair computing.

We expect architectural styles that define trend-specific sets of constraints to continue to emerge, and so will innovations and technology in support of all three trends. We do not expect a need to devise complex standards and standardization activities for such architectures, neither protocols or infrastructure, as long as fundamental architectural constraints and governing principles are agreed upon.

The services computing research community must re-focus by putting traditional and 'solved' (or 'failed') research topics aside, and instead focus on current and future priorities that are at the core of next generation service systems. Continuity of service delivery, decentralization, and fairness should move into the center of our attention.

References

1. Bass, L.J., Weber, I.M., Zhu, L.: DevOps - A Software Architect's Perspective. Addison-Wesley (2015)
2. Bermbach, D., Wittern, J.E., Tai, S.: Cloud service benchmarking. Springer (2017, forthcoming)
3. Federal Trade Commission: Privacy online: Fair information practices in the electronic marketplace, https://www.ftc.gov/reports/privacy-online-fair-information-practices-electronic-marketplace-federal-trade-commission (2000)
4. Hohpe, G.: 37 Things One Architect Knows About IT Transformation. Leanpub (2016)
5. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf (2008)