Massivizing Computer Systems: a Vision to Understand, Design, and Engineer Computer Ecosystems through and beyond Modern Distributed Systems

Alexandru Iosup
Department of Computer Science,
Faculty of Sciences, VU Amsterdam,
The Netherlands
A.Iosup@vu.nl

Alexandru Uta
Department of Computer Science,
Faculty of Sciences, VU Amsterdam,
The Netherlands
A.Uta@vu.nl

The AtLarge Team*
VU Amsterdam and
Delft University of Technology,
The Netherlands,
https://atlarge-research.com

Abstract—**Our society is digital: industry, science, governance, and individuals depend, often transparently, on the inter-operation of large numbers of distributed computer systems. Although the society takes them almost for granted, these computer ecosystems are not available for all, may not be affordable for long, and raise numerous other research challenges. Inspired by these challenges and by our experience with distributed computer systems, we envision Massivizing Computer Systems, a domain of computer science focusing on understanding, controlling, and evolving successfully such ecosystems. Beyond establishing and growing a body of knowledge about computer ecosystems and their constituent systems, the community in this domain should also aim to educate many about design and engineering for this domain, and all people about its principles. This is a call to the entire community: there is much to discover and achieve.**

1. Introduction

The modern lifestyle depends on computer[1] *ecosystems*. We engage increasingly with each other, with governance, and with the Digital Economy [2] through diverse computer ecosystems comprised of globally distributed systems, developed and operated by diverse organizations, interoperated across diverse legal and administrative boundaries. These computer ecosystems create economies of scale, and underpin participation and innovation in the knowledge-based society: for example, in the European Union, information and communication technology (ICT)[2], for which all services are migrating to computer ecosystems[3], accounts for nearly 5% of the economy and accounts for nearly 50% of productivity growth[4]. However positive, computer ecosystems are not merely larger, deeper structures (e.g., hierarchies) of distributed computer systems. Although we have conquered many of the scientific and engineering challenges of distributed computer systems[5], computer ecosystems add numerous challenges stemming from the complexity of structure, organization, and evolving and emerging use. We envision in this work how computer systems can further develop as a positive technology for our society.

> **Vision**: We envision a world where individuals and human-centered organizations are augmented by an automated, sustainable layer of technology. At the core of this technology is ICT, and at the core of ICT are computer ecosystems, interoperating and performing as utilities and services, under human guidance and control. In our vision, ICT is a fundamental human right, including the right to learn how to use this technology.

We see a fundamental crisis, the *ecosystems crisis*, already at work and hampering our vision. The natural evolution from early Computer Systems to modern Distributed Systems has been until now halted by relatively few crises, among which standing out is the software crisis of the 1960s, due to unbounded increase in complexity [9], [10]. We see

- *The AtLarge team members co-authoring this article are: Georgios Andreadis, Vincent van Beek, Erwin van Eyk, Tim Hegeman, Sacheendra Talluri, Lucian Toader, and Laurens Versluis.*

1. The analysis by E.W. Dijkstra [1] explains the main differences between "computer" and "computing" science: origins in the US vs. in the (current) EU, respectively, with American CS seen in the past as "more machine-oriented, less mathematical, more closely linked to application areas, more quantitative, and more willing to absorb industrial products in its curriculum". The differences have now softened, and participants beyond US and EU have since joined our community.

2. ICT loosely encompasses all technology and processes used to process information (the "I") and for communications (the "C"). Historically, the distinction between the "I" and the "C" in ICT can be traced to the early days of computing, where information was stored and processed as *digital* data, and most communication was based on *analog* devices. This distinction has lost importance starting with the advent of all-digital networks, completed in the 1990s Internet.

3. Computer ecosystems build a world of cloud computing [3], artificial intelligence [4], and big data [5], underpinned by diverse software systems and networks interconnecting datacenters [6], and edge [7]/smart devices.

4. Correspondingly, ICT receives about 25% of all business R&D funding and is at the core of EU's H2020 programme, see https://ec.europa.eu/programmes/horizon2020/en/area/ict-research-innovation.

5. M. van Steen and A. Tanenbaum provide an introduction [8].

the ongoing ecosystems crisis as due to similar reasons, and leading the Distributed Systems field to a fundamental deficit of knowledge and of technology[6], with abundant forewarnings. In Section 2, we define and give practical examples of five fundamental problems of computer ecosystems that we believe apply even to the most successful of the tech companies, such as Amazon, Alibaba, Google, and Facebook, but even more so to the small and medium enterprises that should develop the next generation of technology: (i) lacking the core laws and theories of computer ecosystems; (ii) lacking the technology to maintain today's computer ecosystems; (iii) lacking the instruments to design, tune, and operate computer ecosystems against foreseeable needs; (iv) lacking peopleware[7] knowledge and processes; and (v) going beyond mere technology.

Our vision, of *Massivizing Computer Systems*, focuses on rethinking the body of knowledge, and the peopleware and methodological processes, associated with computer ecosystems. We aim to reuse what is valuable and available in Distributed Systems, and in the complementary fields of Software Engineering and Performance Engineering, and to further develop only what is needed. Grid computing and cloud computing, which both leverage the advent of the Networked World[8], of modern processes for the design and development of software systems, and of modern techniques for performance engineering, are sources of technology for utility computing[9]. However, grid computing has succumbed to the enormous complexity of the ecosystems crisis, for example, it did not reach needed automation for heterogeneous resources and non-functional requirements such as elasticity, and did not develop appropriate cost models. Armed with knowledge and practical tools similar to grid computing, the pragmatic and economically viable Distributed Systems domain of cloud computing started with the limited goal of building a digital ecosystem where the core is largely homogeneous, and is still primarily operated from single-organization datacenters. Attempts to expand to more diverse ecosystems have led to problems, some of which we cover in Section 2. Edge-centric computing [7] borrows from peer-to-peer computing and proposes to shift control to nodes at the edge, closer to the user and thus human-centric in its security and trust models, but still relies on current cloud technology instead of explicitly managing the full-stack complexity of ecosystems.

6. Like Arthur [11, Ch.2], we refute the dictionary definition of technology, which superficially places technology in a role secondary to (applied) science. Instead, we use the first-principle definition provided by Arthur: technology is (i) use-driven, (ii) a group of practices and components, typically becoming useful through the execution of a sequence of operations, (iii) the set of groups from (iv) across all engineering available to a human culture, forming thus Kevin Kelly's "technium".

7. "If the organization is a development shop, it will optimize for the short term, exploit people, cheat on the workplace, and do nothing to conserve its very lifeblood, the peopleware that is its only real asset. If we ran our agricultural economy on the same basis, we'd eat our seed corn immediately and starve next year." [12, Kindle Loc. 1482-1484].

8. Of which the Internet is a prominent example, but which further includes networking in supercomputing, telco, and IoT-focused industries.

9. We trace the use of "utility computing" in scientific publications to Andrzejak, Arlitt, and Rolia [13], and to Buyya [14].

We propose to complement and extend the existing body of knowledge with a focus on Massivizing Computer Systems, with the goal of defining and supporting the core body of knowledge and the skills relevant to this vision. (This path is successfully followed by other sciences with significant impact in the modern society, such as physics and its impact on high-precision industry, biology and its impact on healthcare, ecology and its impact on wellbeing, etc.) Toward this goal, we make a five-fold contribution:

1) We propose the premises of a new field[10] of science, design, and engineering focusing on MCS (in Section 3). To mark this relationship with the vision, we also call the field MCS. We define MCS as a part of the Distributed Systems domain, but also as synthesizing methods from Software Engineering and Performance Engineering.
2) We propose ten core principles (Section 4). MCS has not only a technology focus, but also considers peopleware and co-involvement of other sciences. One of the principles has as corollary the periodic revision of principles, and MCS will apply it—a community challenge.
3) We express the current systems, peopleware, and methodological challenges raised by the field of MCS (in Section 5). We cover diverse topics of research that evolve naturally from ongoing community research in Distributed Systems, Software Engineering, and Performance Engineering. We also raise challenges in the process of designing ecosystems and their constituent systems.
4) We predict the benefits MCS can provide to a set of pragmatic yet high-reward application domains (in Section 6). Overall, we envision that computer ecosystems built on sound principles will lead to significant benefits, such as economies of scale, better non-functional properties of systems, lowering the barrier of expertise needed for use, etc. We consider as immediate application areas big and democratized (e-)science, the future of online gaming and virtual reality, the future of banking, datacenter-based operations including for hosting business-critical workloads, and serverless app development and operation.
5) We compare MCS with other paradigms (Section 7). We explicitly compare MCS with the paradigms emerging from Distributed Systems, including grid, cloud, and edge-centric computing. We further compare MCS with paradigms across other sciences and technical sciences.

10. As conjectured by Denning [15], there is a high threshold for becoming a field of science, paraphrasing: focus on the natural and artificial processes of a pervasive phenomenon, a body of knowledge and skills that can be codified and taught, experimental methods of discovery and validation, reproducibility of results and falsifiability of theoretical constructs, the presence of meaningful discovery itself. Even if MCS does not pass this threshold, the process of exploring it as a new field can lead to surprising discoveries, as in other sciences (see Section 7).

2. The Problem of Computer Ecosystems

In this section, we introduce systems, ecosystems, and five fundamental problems of computer ecosystems.

2.1. What Are Systems and Ecosystems?

We use Meadows' definition of systems [16, p.188]:

Definition: A system is "a set of elements or parts coherently organized and interconnected in a pattern or structure that produces a characteristic set of behaviors, often classified as its "function" or "purpose."

The system elements or parts can be systems themselves, producing more fine-grained functions. We see *computer ecosystems* as more than just complex computer systems, in that they interact with people and have *structure* that is more advanced, combinatorial and hierarchical as is the general nature of technology [11], etc.:

Definition: A computer ecosystem is a heterogeneous group of computer systems and, recursively, of computer ecosystems, collectively *constituents*. Constituents are autonomous, even in competition with each other. The ecosystem *structure* and *organization* ensure its collective responsibility: completing functions with humans in the loop, providing desirable non-functional properties that go beyond traditional performance, subject to agreements with clients. Ecosystems experience short- and long-term *dynamics*: operating well despite challenging, possibly changing conditions external to the control of the ecosystem.

Collective Responsibility: The ecosystem is designed to respond to functional and non-functional requirements. The ecosystem constituents must be able to act independently of each other, but when they act collectively they can perform *collective functions* that are required and that are not possible for any individual system, and/or they can add useful non-functional characteristics to how they perform functions that could still be possible otherwise. At least some of the collective functions involve the collaboration of a significant fraction of the ecosystem constituents.

Beyond Performance: When collaborating, the ecosystem constituents optimize or satisfice a decision problem focusing on the trade-off between subsets of both the functional and the non-functional requirements, e.g., correct functional result and high performance vs. cost and availability. The non-functional requirements are diverse, beyond traditional performance: e.g., high performance, high availability and/or reliability, high scalability and/or elasticity, trustworthy and/or secure operation.

Autonomy: ecosystem constituents can often operate autonomously if allowed, and may be self-aware as defined by Kounev et al. [17, Def.1.1]: they could continuously

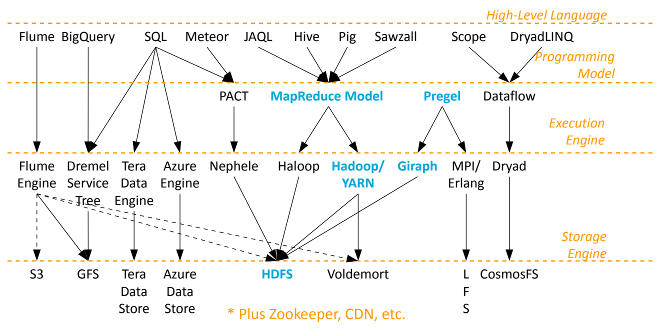

Figure 1. A view into the ecosystem of Big Data processing. (Reproduced and adapted from our previous work [22].) The four layers, *High-Level Language*, *Programming Model*, *Execution Engine*, and *Storage Engine*, are conceptual, but applications that run in this ecosystem typically use components across the full stack of layers (and more, as indicated by the ⋆). The highlighted components cover the minimum set of layers necessary for execution for the MapReduce and Pregel sub-ecosystems.

"learn models capturing knowledge about themselves and the environment", "reason using the models [...] enabling them to act [...] in accordance with higher-level goals, which may also be subject to change."

How do ecosystems appear? Computer ecosystems appear naturally[11], through a process of evolution that involves accumulation of technological artifacts in inter-communicating assemblies and hierarchies, and solving increasingly more sophisticated problems[12]. Real-world ecosystems are distributed or include distributed systems among their constituents [8], and are operated by and for multiple (competitive) stakeholders. Components often are heterogeneous, built by multiple developers, not using a verified reference architecture, and having to fit with one another despite not being designed end-to-end.

A simplified example of ecosystems, sub-ecosystems, and their constituents: Developing applications, and tuning, swapping, and adding or removing components requires deep understanding of the ecosystem. Figure 1 depicts the four-layer reference architecture of the big data ecosystem frequently used by the community. In this ecosystem, the programming model, e.g., MapReduce, or the execution engine, e.g., Hadoop, typically give name to an entire family of applications of the ecosystem, i.e., "We run Hadoop applications." Such families of applications, and the components needed to support them, form complex (sub-)ecosystems themselves; this is a common feature in technology [11, Ch. "Structural Deepening"]. To exemplify the big data ecosystem focusing on MapReduce, the figure emphasizes components in the bottom three layers, which are typically not under the control of the application developer but must nevertheless perform well to offer good non-functional properties, including performance, scalability, and reliability. This is due to *vicissitude* [22] in processing data in such ecosystem, that is, the presence of workflows of tasks that

11. Similar ecosystems appear in many areas of technology [11, Ch.2,7–9], and in many other kinds of systems [18, Ch.5–8].

12. The co-evolution of problems and solutions appears in all areas of design [19], [20] [21, Ch.I, S2–5].

are arbitrarily compute- and data-intensive, and of unseen dependencies and (non-)functional issues.

Examples of large-scale computer ecosystems: Unlike their constituents, ecosystems are difficult to identify precisely, because their limits have not been defined at design time, or shared in a single software repository or hardware blueprint. Large-scale examples of computer ecosystems include: (i) the over 1,000 Apache cloud and big data components published as open-source Apache-licensed software[13], (ii) the Amazon AWS cloud ecosystem, which is further populated by companies running exclusively on the AWS infrastructure, such as Netflix[14], (iii) the emerging ecosystem built around the MapReduce and Spark[15] big-data processing systems.

When is a system not an ecosystem? Under our definition, not every system can be an ecosystem, and even some advanced systems do not qualify as ecosystems (see [23]).

2.2. Fundamental Problems of Ecosystems

The first fundamental problem is that we lack the systematic laws and theories to explain and predict the large-scale, complex operation and evolution of computer ecosystems. For example, when an ecosystem under-performs or fails to meet increasingly more sophisticated non-functional requirements, customers stop using the service [24], [25], but currently we do not have the models to predict such underperforming situations, or the instruments to infer what could happen, even for simple ecosystems comprised of small combinations of arbitrary distributed systems.

The second fundamental problem is that we lack the comprehensive technology to maintain the current computer ecosystems. For example, we know from grid computing the damage that a failure can trigger in the entire computer ecosystem [26], [27], [28], and far all the large cloud operators (e.g., Amazon, Alibaba, Google, Microsoft), have suffered significant outages [29] and SLA issues [25] despite extensive site reliability teams and considerable intellectual abilities. In turn, these outages have correlated failures, as for example experienced when drafting this and other articles on the Amazon-based Overleaf. Moreover, we seem to have opened a Pandora's Box of poorly designed systems, which turned into targets and sources of cyberattacks (e.g., hacking[16], ransomware[17], malware [30], and botnets [31]).

The third fundamental problem is that we are not equipped to explore the future of computer ecosystems, and in particular we cannot now design, tune, and operate the computer ecosystems that can seamlessly support all the societally relevant application domains, to the point where multiple, possibly competitive, organizations and individuals can use computing as an utility (similarly to the electricity grid, including its local shifts toward decentralized smart grids) or as a service (as for the logistics and transportation industry). For example, sophisticated users are already demanding but not receiving detailed control over heterogeneous resources and services, the right to co-design services with functional requirements offered through everything as a service [32], and the opportunity to control detailed facets of non-functional characteristics such as risk management, performance isolation, and elasticity [33].

The fourth fundamental problem is that of peopleware, especially because the personnel designing, developing, and operating these computer ecosystems already numbers millions of people world-wide but is severely understaffed and with insufficient replacement available [34].

The fifth fundamental problem is participating in the emerging practice beyond mere technology. Compounding the other problems, the Distributed Systems community seems to focus excessively on technology, a separation of concerns that was perhaps justifiable but is becoming self-defeating. This focus has brought until now important advantages in producing rapidly many successful ecosystems, but is starting to have important drawbacks: (i) we have to answer difficult, interdisciplinary questions about how our systems influence the modern society and its most vulnerable individuals [35], and in general about the emergence of human factors such as (anti-)social behavior [36], (ii) we have to investigate general and specific questions about the evolution of systems, including how the knowledge and skill have concentrated in relatively few large-scale ecosystems?, and what to do, and with which interdisciplinary toolkit, to prevent this from hurting competition and future innovation [37]?

3. Massivizing Computer Systems (MCS)

In this section, we introduce the fundamental concepts and principles of MCS. We explain its background, give a definition of MCS, explain its goal and central premise, and focus on key aspects of this domain. We explain how MCS extends the focus of traditional Distributed Systems, and how it synthesizes research methods from other related domains.

3.1. What Is MCS?

We now define MCS as a use-inspired discipline [38]:

Definition: MCS focuses on the science, design, and engineering of ecosystems. It aims to understand ecosystems and to make them useful to the society.

Table 1 summarizes MCS: Who? What? How? Which other core issues? (all addressed in this section) and What are the related concepts MCS draws from? (addressed in Section 3.3). We now elaborate on each part, in turn.

Who? Stakeholders: MCS involves a large number of stakeholders, characteristic and necessary for a domain that applies to diverse problems with numerous users. We

13. https://github.com/apache
14. https://github.com/netflix
15. https://github.com/databricks
16. Examples: tinyurl.com/WaPo17IsraelKaspersky, tinyurl.com/NLVolkskrant18RussiaIntel, www.bbc.com/news/technology-42056555
17. Examples: tinyurl.com/ArsTechnica17WCryNKorea, tinyurl.com/TheVerge16SFRansomware

Massivizing Computer Systems (§3.1)		
Who?	Stakeholders	scientists, engineers, designers, others
What?	Central Paradigm	properties derived from ecosystem
	Focus	structure, organization, and dynamics
	Concerns	functional and non-functional properties
		emergence, evolution
How?	Design	design methods and processes
	Quantitative	measurement, observation
	Exper. & Sim.	methodology, TRL, benchmarking
	Empirical	correlation, causality iff. possible
	Instrumentation	experiment infrastructure
	Formal models	validated, calibrated, robust
Related (§3.3)	Computer science	Distrib.Sys., Sw.Eng., Perf.Eng.
	Systems/complexity	General Systems Theory, etc.
	Problem solving	computer-centric, human-centric

TABLE 1. AN OVERVIEW OF MCS.

consider explicitly the scientists, engineers, and designers of MCS systems involved in solving the numerous challenges of the field (discussed in Section 5) and in using results in practice, the industry clients and their diverse applications (Section 6), the governance and legal stakeholders, etc. We also consider as stakeholders the population: individuals at-large, as clients and as (life-long) students.

Goal: The goal of MCS is to understand and eventually control complex ecosystems and, recursively, their constituent parts, thus satisficing possibly dynamic requirements and turning ecosystems into efficient utilities. To this end, MCS must explain how and why the ecosystem differs, functionally and non-functionally, from mere composition of its constituents.

What? The Central Premise: MCS starts from the premise that the interaction between systems in an ecosystem, and the way the ecosystems stakeholders interact with the ecosystem (and among themselves), drives to a large extent the operation and characteristics of the ecosystem. Thus, MCS focuses explicitly on the *structure, organization, and dynamics* of systems when operating in assemblies, hierarchies, and larger ecosystems, rather than understanding and building single systems working in isolation. Both the *functional and the non-functional properties* of these ecosystems, and recursively of their constituent systems, are central to understanding and engineering ecosystems. Over periods of time both that are short (seconds to days) or long (weeks to years), ecosystems may experience various forms of *emergent and chaotic behavior*, and of *evolution* (discussed in the following). Understanding emergent and evolutionary behavior, and controlling it subject to efficiency[18] considerations, is also central to MCS.

How? A general approach and methodology: To begin work on MCS, we consider the following elements that will need to be adapted, extended, and created for computer ecosystems, and ultimately will result in new approaches and methodologies: (i) methods and processes characteristic to design [19], [20], and design science applied to information systems [39] and to the design of (computer) systems; (ii) quantitative research, in particular collection of data through measurement and (longitudinal) observation, statistical modeling of workloads [40], failures [27], [28], and reaching formal (analytical) models; (iii) experimental research, including real-world experimentation through prototypes, and simulation, both under realistic workload conditions and even under community-wide benchmarking settings; (iv) empirical and phenomenological research, including qualitative research resulting in comprehensive surveys [41] and field surveys; (v) modern system evaluation, using instrumentation beyond what is needed to test typical Distributed Systems (e.g., large-scale infrastructure comparable with medium-scale industry infrastructure [42]), focusing on an extended array of indicators and metrics (e.g., performance, availability, cost, risk, various forms of elasticity [33]), and developing approaches for meaningful comparison across many alternatives for the same component [43] or policy point [44]. We expand on these in Section 3.2.

How? Other issues: We envision several other core issues important for MCS: (i) peopleware: processes for training, educating, engaging people, especially the next generation of scientists, designers, and engineers, (ii) making available free and open-access artifacts, both open-source software and common-format data, (iii) ensuring a balance of recognition between scientific, design, and engineering outcomes, across the community, and (iv) ethics and other interdisciplinary issues.

3.2. More on the General Approach

Design: By definition, MCS employs a diverse body of knowledge and skill typical to modern *science* and *engineering*, from which we further distinguish *design*[19]. The work we conduct in this field aims to go beyond random walks, and direct application or replication of prior work, We aim to establish *design methods and processes*, based on principles and on instruments, that meet the goal of MCS. We envision here, as a first step, adapting and extending techniques from the design of information systems [39] and of computer systems, and also from design not related to computers [20], [45] or even to technology [19].

Quantitative results: Obtain quantitative, predictive, actionable understanding about the sophisticated functional and non-functional properties of ecosystems, and about their dynamics. It is here that advances in Performance Engineering, especially *measurement* and statistically sound *observation*, can help the domain of MCS get started. Specifically, collecting data from running ecosystems and from experimental settings, both real-world and simulated (see following heading), we can start accumulating *knowledge*. These would lead to observational models, and, later,

18. Although process economics is better equipped than MCS to address costing, pricing, and utility functions, in practice designers and engineers are expected to conduct or at least provide quantitative input for these tasks.

19. We adopt here the argument made by Cross in the 1970s, and extended by Lawson [19, Ch.8, loc.2414, and Ch.16, loc.4988], that design is a distinct way of thinking about real-world problems with high degree of uncertainty, and of solving them: problems and solutions *co-evolve*.

possibly also to calibrated mechanistic models and full-system (weakly emergent [18, p.171]) models.

Experimentation and simulation: MCS depends on methodologically sound real-world[20] and simulation-based[21] experiments, which have complementary strengths and weaknesses but combined can provide essential feedback to scientists, engineers, and designers. Experimentation is valuable in validating and demonstrating the *technology-readiness level* (TRL)of various concepts and theories, using *prototypes* or even higher-TRL artifacts running preferably in real-world environments[22], in providing calibration and measurement data, in revealing aspects that we have not considered before, etc. *Benchmarking*, a subfield of experimentation, focuses the community on a set of common processes, knowledge, and instrumentation. Simulation is useful in investigating and comparing known and new designs, and dynamics including non-deterministic behavior, over long periods of simulated-time. Simulation can also be used to replay interesting conditions from the past, giving the human in the loop more time and more instruments to understand.

Empirical (correlation), and if possible also phenomenological (causal), research is necessary[23], if we are to understand and control especially the emergent properties of ecosystems. Observation and measurement, and experimentation and simulation of ecosystems are the main empirical methods. Additionally, MCS must also study empirically the highly variable, possibly non-deterministic processes that include humans: their use of ecosystems and their new (practical) problems with using ecosystems, and their study, design, and engineering of ecosystems. This latter part is much less developed in Distributed Systems, but a rise in empirical methods in Software Engineering [47], [53] and in design sciences [20, Ch.1] already employs: studying the artifacts themselves (e.g., with static code analysis), interviews with designers, observations and case studies of one or several design projects, experimental studies typically of synthetic projects, simulation by letting computers try to design and observing the results, and reflecting and thinking about own experience. The benefits of using these methods include deeper, including practical, understanding. The dangers include relying on "soft methods" [47] and ignoring the "threats to validity" [53].

Instrumentation: Similarly to other Big Science domains, such as astrophysics, high-energy physics, genomics and systems biology, and many other domains reliant today on e-Science, MCS requires significant instrumentation. It needs adequate environments to experiment in, for example, the DAS-5 in the Netherlands [42] and Grid'5000 in France. As in the other natural sciences, creating these instruments can lead to numerous advances in science and engineering; moreover, these instruments are ecosystems themselves and thus an *endogenous* object of study for MCS. MCS also needs the infrastructure needed to complement the human mind in the task of understanding the data collected about ecosystems, to generate hypotheses automatically, and to preserve this data for future generations.

Formal (analytical) models: We envision that a complex set of formal mathematical models, validated and calibrated with long-term data, robust and with explanatory power beyond past data, will emerge over time to support MCS. Such models will likely be hierarchical, componentized, unlike the first-order approximations of classical physics, and will require computers to manipulate. The key challenge to overcome will be to support the dynamic, deeply hierarchical, emergent nature of modern ecosystems, possibly lacking a steady-state. This could be achieved through a combination of ODEs and PDEs (multiple independent control-variables), time-dependent evolution and events, discrete states and Boolean logic, stochastic properties for each component and behavior, etc.

3.3. How Far Are We Already?

To understand the extent of progress we have made in MCS, we need to understand both what techniques and processes the field is comprised of already (discussed in this section), and what applications it can have (in Section 6). We see Massivizing Computer Systems as derived from Distributed Systems, which in turn are derived from core Computer Systems. Additionally, Massivizing Computer Systems aims to synthesize interdisciplinary knowledge and skills primarily from Software Engineering and Performance Engineering. This matches Snir's view that computer science is "one broad discipline, with strong interactions between its various components" [38], under which subdisciplines reinforce each other, and multidisciplinary and interdisciplinary research and practice further enable the profession.

We have compiled [23] a non-exhaustive list of principles and concepts MCS can import from: (i) established domains, such as Distributed Systems, Computer Systems, and the complementary fields of Software Engineering and Performance Engineering; (ii) Complex Adaptive Systems, and the related domains of General Systems Theory, Chaos Theory, Catastrophe Theory, Hierarchical Theory, etc., albeit, we are also aware that much distance must be covered between theory and practice, related to these fields, and (iii) generalized theories and techniques of problem solving and satisficing[24], both *computer-* and *human-centric* [54], [55].

20. MCS follows the multi-decade tradition of experimental computer science [46], [47], [48] and Distributed Systems [42], [49].

21. Simon makes a compelling case that simulation can lead to new understanding, of both computer systems about which we know much and about which we do not [18, Section "Understanding by Simulating"]. He refutes that a simulator is "no better than the assumptions built into it", that they cannot reveal unexpected aspects, that they only apply for systems whose laws of operation we already know.

22. Like Tichy [47], we disagree that mere demonstrations and proof-of-concepts can replace experimentation and simulation, even if they prove valuable for engineering products and educating stakeholders.

23. As a matter of pragmatism, our empirical research may need to be data-driven (that is, *discovery science* [50]), instead of hypothesis-driven, simply because the complexity of the problems seems to exceed the capabilities of the unaided human mind. Systems Biology [50], [51] [52, Ch.1] and other sciences argue similarly, since mid-2000s.

24. Satisficing [18, p.28] is about finding a solution that meets a set of requirements based on a threshold ("better than X"), instead of the goal of optimization to find an optimum ("the absolute best").

	Principle	
Type	Index	Key aspects
Systems (§4.1)	P1	The Age of Ecosystems
	P2	software-defined everything
	P3	non-functional requirements
	P4	management, scheduling, self-awareness
	P5	super-distributed
Peopleware (§4.2)	P6	fundamental rights
	P7	professional privilege
Methodology (§4.3)	P8	science, practice, and culture of MCS
	P9	evolution and emergence
	P10	ethics and transparency

All principles summarized in this work, and detailed in [23].

TABLE 2. THE 10 KEY PRINCIPLES OF MCS.

4. Ten Core Principles of MCS

We introduce in this section ten core principles of MCS, on which we expand in [23]. Our highest principle is that:

P1: This is the Age of Computer Ecosystems.

Derived from its goal and as stated in its central premise (see Section 3.1), MCS aims to understand and design computer ecosystems, working efficiently at any scale, to benefit the society. This requires a science of pragmatic, predictable, accountable computer systems that can be composed in nearly infinite ways, be controlled and understood despite the presence of complexity, emergence, and evolution, and whose core operative skills can be taught to all people. Overall, this leads to the principles summarized by Table 2.

4.1. Systems Principles

P2: Software-defined everything, but humans can still shape and control the loop.

The ecosystem is comprised of software and software-defined (virtual) hardware, which allow for advanced control capabilities and for extreme flexibility. "Software is eating the world", but under human and other control.

P3: Non-functional properties are first-class concerns, composable and portable, whose relative importance and target values are dynamic.

Non-functional requirements (e.g., security, trust, privacy, scalability, elasticity, availability, performance, cost, limited risks) are first-class concerns, and *the importance and the characteristics of each may be fluid over time, and depends on stakeholders, clients, and applications*.

In contrast to today's Distributed Systems, we envision guarantees of both functional and non-functional properties, however assemblies are composed, even when complexity, emergence, and evolution manifest. Guarantees include not only specialized service objectives/targets (SLOs) and overall agreements (SLAs), but also general, ecosystem-wide guarantees such as tolerance to performance variability [33], vicissitude [22]), correlated failures, security attacks, etc.

P4: Resource Management and Scheduling, and their combination with other capabilities to achieve local and global Self-Awareness, are key to ensure non-functional properties at runtime.

Resource Management and Scheduling is a key building block without which MCS is not sustainable or often even achievable. Consequently also of the scale and complexity of modern ecosystems, *disaggregation* and *re-aggregation of software and software-defined hardware* become key operations that are not controlled in today's Distributed Systems. *Self-awareness* [56] is another key building block.

P5: Ecosystems are super-distributed.

MCS ecosysems are *super-distributed*: components in MCS are often distributed, and ecosystems in MCS are recursively distributed: distributed ecosystems comprised of distributed ecosystems, in turn comprised of distributed ecosystems, etc. Beyond the traditional concerns of Distributed Systems, super-distribution is also concerned with many desirable *super*-properties: super-flexibility and super-scalability (discussed in the following), multiple ownership of components and federation, multi-tenancy, disaggregation and re-aggregation of systems and workloads, interoperability including third-party systems, etc.

Extending a term from management theory [57, Ch.2], we define *super-flexibility* as the ability of an ecosystem to ensure *both* the functional and non-functional properties associated with stability and closed systems (e.g., correctness, high performance, scalability, reliability, and security), *and* those associated with dynamic and open systems (e.g., elasticity, streaming and event-driven, composability and portability). Super-flexibility also introduces a framework for managing product mergers and break-ups (e.g., due to technical reasons, but also due to legal reasons such as anti-monopoly/anti-trust law) on short-notice and quickly.

Similarly to super-flexibility, *super-scalability* combines the properties of closed systems (e.g., weak and strong scalability) and of open systems (e.g., the many faces of elasticity [33]). Inspired by Gray [58], we see this new form of scalability as a grand challenge in computer science.

4.2. Peopleware Principles

MCS provides services to hundreds of millions of people, through ecosystems created by a large number of amateurs and professionals. Inspired by the software industry's struggle to manage and develop its human resources, we explicitly set principles about peopleware.

P6: People have a fundamental right to learn and to use ICT, and to understand their own use.

MCS must lead to teachable technology: in our vision, all stakeholders of all public computer ecosystems can be taught basic ecosystems-related skills. The technical level requires to use and read the consumption meters of MCS cannot exclude systematically any part of the population.

P7: **Experimenting, creating, and operating ecosystems are professional privileges, granted through provable professional competence and integrity.**

To limit damage to the society and to the profession itself, everyone who experiments with, creates, or operates ecosystems that others rely on must be subject to professional checks and balances. As a community, we are no longer in position to argue technology in general, and especially ecosystems reaching many people, is only beneficial and thus creating and operating such technology should be done without restriction [59]. This puts our field in line with medical and legal professions, but with the added pressure resulting from the increase of contract work in our field [60].

As has been argued about the profession of computing in general [61], we need to establish a profession of Massivizing Computer Systems. This requires establishing the core roles that stakeholders can play, including the services *professionals* can provide to *clients*. Clients have the right to be protected "from [...] own ignorance by such a professional" [19, loc.4338-4339]. The profession sanctions, through the guidelines of a professional society, the *body of knowledge* and the *skills* used in practice, and the *code of ethics* of the profession. Bodies of knowledge expand through organized (scientific) disciplines, whereas skills expand through the practice of organized trades. Professional (accredited) education provides training for both, and higher education also provides training into the processes of expanding both. Trained professionals are certified.

4.3. Methodological Principles

As a field of computer systems, itself a field of computer science, MCS leverages their scientific principles, including the list compiled by Denning [15, p.32]: (i) focusing on a pervasive phenomenon, which it tries to understand, use, and control (MCS focuses on computer ecosystems); (ii) spans both artificial and natural processes regarding the phenomenon (MCS both designs and studies its artifacts at-large); (iii) aims to provide meaningful and non-trivial understanding of the phenomenon; (iv) aims to achieve reproducibility, and is concerned with the falsifiability of its proposed theories and models; etc. MCS also includes in its methodological principles a broader principle, related to the ethics of the profession (linked also with Principle 7).

P8: **We understand and create together a science, practice, and culture of computer ecosystems.**

Addressing a reproducibility crisis that goes beyond computer systems [62], [63], we envision a domain of MCS where *everything we develop is tested and benchmarked, reproducibly*, and provide five steps toward this goal [23].

P9: **We are aware of the evolution and emergent behavior of computer ecosystems, and control and nurture them. This also requires debate and interdisciplinary expertise.**

Short- and long-term evolution, and short-term emergent behavior, can shape the use of current and future ecosystems. Practitioners in MCS must be aware of the evolution of system properties, requirements, and stakeholders, and strive to be aware of emergent behavior. We must study existing principles [64] and revisit periodically what is valuable in our and related fields, engage the experts in debate through Dagstuhl-like workshops [65], and involve the society at-large in discussing the ethics and practice of the field [66]. **Corrolary:** this principle also requires to revisit periodically the principles of MCS proposed in Section 4.

P10: **We consider and help develop the ethics of computer ecosystems, and inform and educate all stakeholders about them.**

Focusing exclusively on technology exposes the community to various ethical risks. We envision for MCS *an ethical imperative to actually solve societal problems*, which means our focus must broaden and become more interdisciplinary, and MCS must *develop a body of ethics to complement the body of knowledge*. As an added benefit, new functional and non-functional requirements will thus emerge.

5. Twenty Research Challenges for MCS

Although we see well the challenges raised by the proliferation of ecosystems and especially their constituents, we are just beginning to understand the difficulties of working with ecosystems instead of merely systems. Known difficulties include, but are not limited to the issues captured by our principles (see Section 4): sheer scale, group and hierarchical behavior under multiple ownership and multi-tenancy, interplay and combined action of multiple adaptive technique, super-distributed properties, etc.

C1: Ecosystems instead of systems. (From **P1**)

We see as the grand challenge of MCS re-focusing on entire ecosystems: How to take ecosystem-wide views? How to understand, design, implement, deploy, and operate ecosystems? How to balance so many needs and capabilities? How to support so many types of stakeholders? How do the challenges raised by ecosystems co-evolve with their solutions? What new properties will emerge in ecosystems at-large and how to address them? These and similar questions raise numerous challenges related to systems (Section 5.1), peopleware (Section 5.2), and methodology (Section 5.3). Table 3 summarizes this non-exhaustive list of challenges.

5.1. Systems Challenges

C2: ($\star$) Make ecosystems fully software-defined, and cope with legacy and partially software-defined systems. (From **P2**)

Type	Challenge		
	Index	Key aspects	Princip.
Systems (§5.1)	C1	Ecosystems, overall	P1
	C2	(⋆) Software-defined everything	P2
	C3	(⋆) Non-functional requirements	P3, P5
	C4	(∼) Extreme heterogeneity	P4
	C5	(⋆) Socially aware	P4
	C6	(⋆) Adaptation, self-awareness	P4
	C7	(⋆) Scheduling, the dual problem	P4, P5
	C8	(∼) Sophisticated services	P4
	C9	(⋆) Ecosystem Navigation	P2–5
	C10	(⋆) Interoperability, federation	P4, P5
Peopleware (§5.2)	C11	(⋆) Community engagement	P6
	C12	(⋆) Curriculum, BOKMCS	P6
	C13	(⋆) Explaining to all stakeholders	P4, P6
	C14	(∼) The Design of Design	P6, P7
Methodology (§5.3)	C15	(⋆) Simulation and Real-world experimentation	P7, P8
	C16	(∼) Reproducibility and benchmarking	P7, P8
	C17	(⋆) Testing, validation, verification	P8
	C18	(∼) A science of MCS	P8, P9
	C19	(⋆) The New World	P8, P9
	C20	(⋆) The ethics of MCS	P10

(⋆) Appears in the extended version of this document [23].
(∼) Only summarized here, details in [23].

TABLE 3. A SHORTLIST OF THE CHALLENGES RAISED BY MCS.

C3: (⋆) Make non-functional requirements first-class considerations, understand key trade-offs between them, and enable ways to specify targets (dynamically) with minimal (specialist) input. (From **P3, P5**.)

C4: (∼) Manage extreme heterogeneity. (From **P4**)

Large-scale computer ecosystems exhibit unprecedented, extreme heterogeneity, which we characterize mainly as (i) workload heterogenity, (ii) infrastructure, and (iii) peopleware (addressed in C5). We discuss these in [23].

Heterogeneity in both applications and infrastructure creates new research challenges. How to program applications easily, out of several heterogeneous components? How to exploit the heterogeneity of infrastructure for optimal performance, cost savings, energy efficiency, etc.? How to control the trade-off between efficiency, and various non-functional and functional requirements? There is already some work in this area, albeit preliminary. It includes languages and tools to program once and run on heterogeneous hardware [67], which extends the large body of previous work on middleware to abstract storage devices and services.

C5: (⋆) Socially aware systems, with the human in the control loop. (From **P4**)

C6: (⋆) Make use of adaptation approaches, from simple feedback loops to self-awareness [56], to respond automatically to anomalies and to changes in requirements. (From **P4**)

C7: (⋆) Scheduling, consisting of both provisioning and allocation, on behalf of different, possibly delegating stakeholders. (From **P4, P5**.)

C8: (∼) Sophisticated components in the ecosystem offered as services. (From **P4**)

We are entering a period of Everything-as-a-Service (XaaS, e.g., Science-, Backend-, and Function-as-a-Service), where any product or technology can be supplied as a service and delivered to the consumer through the Internet [32]. Enterprises such as Google, Microsoft, and Oracle, are moving away from the traditional license models and are offering their products on-demand, through a consumption-based model. Many companies and organizations have transitioned (parts of) their operation to cloud-based services [68], shifting their operation model to take advantage of the elasticity, availability, security (albeit, possibly not also privacy), and pay-as-you-go pricing model of the cloud [69]. Interesting new challenges arise, that relate traditional to new non-functional requirements (e.g., cost, resource waste, manageability). We discuss the challenges of serverless/FaaS operation in [23, §6.5] and in [70].

C9: (⋆) The Ecosystem Navigation challenge: solving problems of comparison, selection, composition, replacement, adaptation, and operation of components (and assemblies) on behalf of the user. (From **P2–5**.)

C10: (⋆) Interoperate assemblies, dynamically: geo-distributed, federated, multi-DC operation, and service delegation. (From **P4, P5**.)

5.2. Peopleware Challenges

C11: (⋆) Create communities and environments for people to engage with the design and operation of ecosystems. (From **P6**)

C12: (⋆) Create a teachable common body of knowledge for MCS (BOKMCS). (From **P6**)

C13: (⋆) Support for showing and explaining the operation of the ecosystem to all stakeholders, continuously. (From **P4, P6**.)

C14: (∼) The Design of Design. (From **P6, P7**.)

We see design as a major challenge for the field of MCS: not only good designs are difficult to achieve for the level of complexity posed by computer ecosystems, but also students and later practitioners in the field do not have prior training in Design Thinking and sometimes even Systems Thinking (see also C12). society also benefits when the general public understands the basic principles of design and is able to enjoy them as art (see also P6). We envision design processes that trade-off the rigor and precision needed to make software ecosystems run, and creativity and innovation.

5.3. Methodological Challenges

C15: (⋆) Simulation-based calibrated approaches and real-world experimentation with methodology that ensures reproducibility as key instruments for problem exploration and solving, and for evaluating and comparing ecosystems. (From **P8**)

C16: (∼) Reproducibility of analysis results regarding functional and non-functional properties of systems, including through a new generation of evolving benchmarks, and through processes and instruments for preserving and sharing benchmarking results. (From **P8**)

Reproducibility is a key concern for MCS, which follows the concerns raised by large-scale systems [71], [72], [73], [74]. To improve reproducibility of experiments, the SPEC RG Cloud group[25] is developing new methodologies for (cloud) experimentation: guidelines on reporting metrics and values, specifying the aspects of the environment that can lead to reproducibility, sharing the software and data artifacts used during experimentation, etc.

C17: (⋆) Testing, validation, verification in this new world. Manage the trade-offs between accuracy and time to results. (From **P8**)

C18: (∼) Build a science of Massivizing Computer Systems. Revisit in the process the principles of Distributed Systems, Software Engineering, Performance Engineering. (From **P9**)

Overall, as Systems Biology experienced [52, p.3–9], the process of trying to define an independent field of science can be more important than actually seeing it recognized as such. We propose a pragmatic approach for MCS to meet the high threshold conjectured by Denning [15] and become a field of science: (i) studying the novel natural and artificial processes that appear in computer ecosystems (the "Central Premise" in Section 3.1), which we argue are pervasive in the modern digital markets and critical to knowledge-based societies; (ii) defining a body of knowledge and skills that relate to computer ecosystems, as explained in Section 3.1, based on sound and far-reaching principles (Section 4), starting from the already large existing body of knowledge identified in Section 3.3; (iii) experimenting with ecosystems and simulating them, to enable discovery and validation, and meet reproducibility and falsifiability principles (Section 4, especially P8); (iv) contributing to codifying and teaching the body of knowledge and skills developed at point (ii); (v) complementing (ii), and jointly with all stakeholders, defining, codifying, and teaching a body of ethics relevant for work in computer ecosystems. Other sciences have taken this pragmatic approach (see Section 7.2).

25. https://research.spec.org/working-groups/rg-cloud.html

§	Description	Key aspects
	Endogenous applications	
(∼) 6.1	Datacenter management	RM&S, XaaS, ref.archi.
(⋆) [23, §6.2]	Emerging application structures	serverless MCS
(⋆) [23, §6.6]	Generalized graph processing	full MCS challenges
	Exogenous applications	
(∼) 6.2	Future science	e-, democratized science
(⋆) [23, §6.3]	Online gaming	multi-functional MCS
(⋆) [23, §6.4]	Future banking	regulated MCS

(⋆) Appears in the extended version of this document [23].
(∼) Only summarized here, details in [23].

TABLE 4. SELECTED USE-CASES FOR MCS.

To start addressing this challenge, we ask: Can we build a science of MCS from first-principles? Considering the large body of existing related knowledge (see Section 3.3), which existing laws, theories, and concepts still apply to MCS? Which do not? What abstractions can we reuse, e.g., can there be an operating system for massivized computer systems? What new abstractions are needed?

C19: (⋆) The New World challenge: understanding and explaining new modes of use, including new, realistic, accurate, yet tractable models of workloads and environments. (From **P9**)

C20: (⋆) Understand challenges in the ethics of MCS, and evolve our instruments to support ethics in this context. (From **P10**)

6. Massivizing Computer Systems: Use Cases

We envision that computer ecosystems built on the principles of MCS will lead to significant benefits over the current approaches, and in some cases to technology disruption: achieving economies of scale (e.g., reducing resource waste and cost), ensuring better and more diverse non-functional properties of systems, lowering the barrier of expertise needed for use, removing the most tedious tasks from the daily tasks of engineers, etc. This can have immediate impact in many application domains. Table 4 summarizes six such application domains.

We distinguish two directions of application: (i) *endogenous*, that is, the computer science and in particular the computer systems areas using the concepts and technologies developed within the science of MCS, such as cloud computing and big data in general, and datacenter management (Section 6.1), as specific application domains, and (ii) *exogenous*, that is, domains of application that use ICT and in particular computer systems technology to augment or expand their capabilities, such as science (Section 6.2).

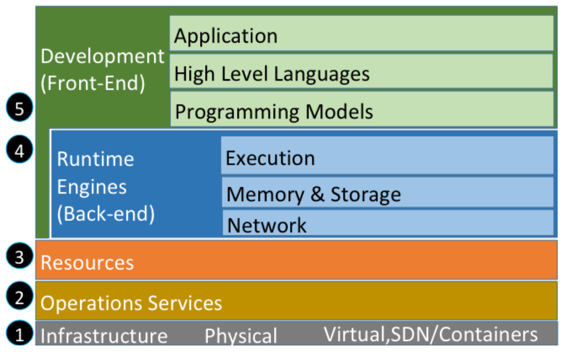

Figure 2. Reference architecture for datacenters (2 levels of depth).

6.1. Datacenters: Managing Digital Factories

In the Digital Economy, datacenters serve the role of modern factories, producing efficient, dependable services. Their clients range from scientists running complex simulations and data processing pipelines (further explored in Section 6.2), to consumers playing online games and meta-gaming (see [23, §6.3]). To achieve their promise of efficiency and flexibility, datacenters must both use their resources near-optimally, and cover a broad range of scales and designs: from the large multi-cluster deployments typical to IaaS clouds such as Amazon EC2 and Microsoft Azure, to the cloud-edge [75] micro-datacenters used for video transcoding and streaming [76]. This raises numerous scientific, designerly [20], and engineering challenges.

In our previous work [77], we have identified simulation as a key instrument to explore efficient and controllable datacenter ecosystems (see C1 and C15) and, as a key challenge, a fully automated resource management and scheduling system for datacenters, able to address: (i) the core principles of MCS, and (ii) in particular the challenges introduced in Section 4.1. We address here several of these issues.

We envision datacenters are increasingly equipped with a (fully) software-defined stack (C2), supporting complex services (C8) while making nearly optimal use of available resources (C7), managing dynamically (C6) the workload, infrastructure, and peopleware heterogeneity (C4 and C5). Thus, live-teams of engineers could spend less time on relatively trivial decisions, and more on: (i) monitoring, diagnosing, and controlling new and particularly complex workflows and dataflows on behalf of users, (ii) navigating the ecosystem (C9), (iii) designing ecosystems, constructing and exploring what-if scenarios, etc.

Datacenters will support increasingly more sophisticated non-functional requirements (C3), emerging in MCS, e.g., super-scalability and super-flexibility (see P5), or in specialized classes of datacenters, e.g., trust and personalized control in edge-centric (micro-)datacenters [7]. We envision a guiding, non-mandatory reference architecture for datacenter ecosystems to capture and help manage the diversity of offered services and underlying software layers. For example, Figure 2 depicts our reference architecture for datacenters, comprised of 5 core layers, *Front-end* for the application-level functionality, *Back-end* for task, resource, and service management on behalf of the application, *Resources* for task, resource, and service management on behalf of the cloud operator, *Operations Service* for basic services that are typically associated with (distributed) operating systems, and *Infrastructure* for managing physical and virtual resources; a 6th layer, *DevOps* covers functions essential to operating the datacenter but orthogonal to the service provided to customers, such as monitoring, logging, and benchmarking. Layers 3–5 emphasize the intense focus of the community on simplifying the development of cloud-based applications. Further refining the scheduling components that appear in each layer, we envision a reference architecture for scheduling in datacenters; this would mean for example expanding on previous work on grid scheduling [78].

6.2. Science ↻ MCS, Virtuous Cycle: The Future of Big Science, Democratic Science, and e-Science

We see the future of science as forming a virtuous cycle with that of technology. Science is increasingly interwoven with the technology that enables it [11, Ch.3, loc.903]. Modern science requires experimentation, observation, and reasoning that are possible only through modern technology, and modern technology increasingly complements its own capabilities with the findings of modern science [11, Ch.3]. Unsurprisingly, this requires computer systems with increased the sophistication and scale. We discuss in this section three scientific drivers for MCS, in turn, Big Science, Democratic Science, and, in-between, e-Science. For each, we see MCS as a disruptive technology.

Big Science[26] pushes the limits of current computer ecosystems and raises many of the challenges we raise in Section 5, e.g., massive projects requiring software-defined everything (C2), diverse non-functional properties including efficiency and trust (C3), various forms of system and peopleware heterogeneity (C4 and 5, respectively), etc. For example, large scientific experiments rely on federated infrastructure to perform data collection, filtering, and analytics, and especially their storage and processing infrastructure spans federated, geo-distributed data centers. Possibly the best modern example of Big Science is the Large Hadron Collider [81], which recently reached the 200 petabyte milestone[27]. Analyzing such volume of data is already strenuous for modern computer systems. However, upcoming Big Science projects are expected to deliver even larger volumes and vicissitudes. Such projects include the Square Kilometer Array [82], the KM3NeT cubic-kilometer telescope searching for neutrinos, or the new Large Hadron Collider that is expected to be three times as large as its predecessor. We envision MCS to be the enabler of the computing technology and infrastructure behind such challenging projects, which should provide a sustainable and efficient data processing and storage layers.

26. Massive scientific endeavors working as industrial-scale research [80, p.1]: large teams, large scientific apparatus, big budgets.

27. https://tinyurl.com/CERN200PB

	Emergence		Epistemological Characteristics*			
Field (Decade Emerging)	Crisis	Continues	Objectives	Object	Methodology	Character
Modern Ecology (1990s)	Biodiversity loss	Ecology and Evolution	DS	Biosphere	ADHS	AC
Modern Chem. Process (1990s)	Process complexity	Chemical Engineering	DE	Chemical proc.	ADHSP	ACEM
Systems Biology (2000s)	Systems complexity	Molecular biology	S	Biological sys.	AHS	ACEMTU
Modern Mech. Design (2000s)	Process sustainability	Technical Design	DE	Mechanical sys.	DHSP	ACEM
Modern Optoelectronics (2010s)	Artificial media	Microwave technology	S	Metamaterials	DHSP	ACEMTU
MCS **(this work)**	**Systems complexity**	**Distributed Systems**	**DES**	**Ecosystems**	**ADHSP**	**ACES**

Acronyms follow the framework of Ropohl [79, p.4–7]: Objectives: D = Design, E = Engineering, S = Scientific. Methodology: A = abstraction, D = design (abductive creation), H = hierarchy, I = idealization, S = simulation, P = prototyping.

Character: A = applicability, C = approved by the scientific/design/engineering community, E = empirically accurate, H = harmony between results, M = mathematically detailed, S = simplicity, T = truth, U = universality.

TABLE 5. Comparison of fields. The row for MCS is envisioned.

Democratic science is also a scientific driver for MCS. Recent advances in hardware technology have made the market more accessible than ever: storing one gigabyte of data costs below \$0.05[28], performing 1 GFLOP costs below \$0.1[29], etc. Through XaaS (C8), this gives unprecedented access to science-grade facilities to an increasing number of small research groups around the world, democratizing access to (virtual) computing infrastructure and accelerating scientific discovery. We envision such infrastructure will enable (and disrupt) large-scale scientific experimentation and discovery, with the hardware resources of today and following MCS principles, instead of earlier approaches [83].

7. Related Work

In this section, we compare MCS with other paradigms, of computer science, and of (technical) sciences.

7.1. Other Paradigms of Distributed Systems

In the large field of Distributed Systems, we identify three major paradigms that MCS builds upon: cluster, grid, and more recently, cloud computing and edge computing [7]. We have explored in Section 1 some of the shortcomings of previous paradigms, such as grid, cloud, peer-to-peer, and edge computing, when addressing challenges of computer ecosystems. Overall, in contrast with these paradigms, MCS focuses on new problems and challenges (i.e., related to ecosystems, considering peopleware, and the combined spectrum science-engineering-design), for which it offers new views (e.g., ecosystems-first), new and powerful (predictive) concepts and techniques including a synthesis of techniques across Distributed Systems, Software Engineering, and Performance Engineering, and new and existing but improved technologies and instruments (e.g., ecosystems studied *in silico* or with full-stack simulation). Furthermore, MCS brings computing closer to the users, empowering them to control how computing ecosystems behave by means of expressive, modern non-functional requirements (such as elasticity and security) and by considering universal access to services to also include less sophisticated users.

7.2. Parallels with and Other Fields of Science

We see the emergence of MCS from Distributed Systems as a process similar to the emergence of other science domains, which we have witnessed in the past three decades. Table 5 uses the framework of Ropohl [79, p.4–7] to summarize how MCS matches emergent sub-fields of other science domains. These emerging fields have started humbly, part of a broader paradigm, then developed into domains themselves [38].

Among the fields we survey, closest to MCS is Systems Biology. In contrast to it, which has a distinctly scientific orientation and thus a character focused especially toward universality and mathematical formulation of results, MCS focuses explicitly on design and synthesis (engineering) in its objectives, and for now on empirical results.

28. https://www.backblaze.com/blog/hard-drive-cost-per-gigabyte/
29. AMD Radeon Vega 64 costs ∼\$2,000, for 13.7 TFLOPS (single).

8. Conclusion

Responding to the needs of an increasingly digital and knowledge-based society, we envision ever-larger roles for vast and complex combinations of distributed systems that serve individuals and human-centered organizations. However, current technology seems ill-equipped to achieve this vision: we identify an ongoing *ecosystems crisis* hampering not only evolving toward the vision, but even the current operation of modern distributed systems. In this work, we propose as alternative *Massivizing Computer Systems*: (1) we define MCS to focus on a distinctive central paradigm, built around computer ecosystems, (2) we propose a set of core principles for MCS, (3) we propose a diverse set of challenges focusing on systems, peopleware, and methodological aspects derived from the core principles; (4) we identify and explore various benefits we envision MCS can bring to the future of several application domains; (5) we contrast MCS with both paradigms of modern computer science fields such as Distributed Systems, and emerging fields of science and technical science.

We have started to address the research agenda formulated in this article, both as a single research group and, through the SPEC RG Cloud group, in collaboration with numerous academic and industry partners. We hope our vision will stimulate a larger community to join us in addressing these complex yet rewarding challenges.

Acknowledgments

Work supported by the projects Vidi MagnaData and Commit.

References

[1] E. W. Dijkstra, "Computing Science: Achievements and Challenges," in *SAC*, 1999, p. 1.

[2] D. Tapscott, *The digital economy: Promise and peril in the age of networked intelligence*. McGraw-Hill New York, 1996, vol. 1.

[3] P. Mell *et al.*, "The NIST definition of cloud computing," 2011.

[4] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 2016.

[5] V. Mayer-Schnberger and K. Cukier, *Big data: A revolution that will transform how we live, work, and think*. Taylor & Francis, 2014.

[6] A. Roy *et al.*, "Inside the Social Network's (Datacenter) Network," in *SIGCOMM*, 2015, pp. 123–137.

[7] P. G. López *et al.*, "Edge-centric Computing: Vision and Challenges," *Computer Communication Review*, vol. 45, pp. 37–42, 2015.

[8] M. van Steen *et al.*, "A brief introduction to distributed systems," *Computing*, vol. 98, pp. 967–1009, 2016.

[9] E. W. Dijkstra, "The Humble Programmer," *Commun. ACM*, vol. 15, pp. 859–66, 1972.

[10] N. Wirth, "A Brief History of Software Engineering," *IEEE Annals of the History of Computing*, vol. 30, pp. 32–39, 2008.

[11] W. B. Arthur, *The Nature of Technology*. Free Press, 2009.

[12] T. DeMarco *et al.*, *Peopleware*. Dorset House, 2012.

[13] A. Andrzejak, M. Arlitt, and J. Rolia, "Bounding the resource savings of utility computing models," HP Laboratories Palo Alto technical report HPL-2002-339, Dec 2002.

[14] R. Buyya, "The Gridbus Toolkit for Grid and Utility Computing," in *CLUSTER*, 2003.

[15] P. J. Denning, "The science in computer science," *Commun. ACM*, vol. 56, pp. 35–38, 2013.

[16] D. Meadows, *Thinking in Systems*. Chelsea Green Publishing, 2008.

[17] S. Kounev *et al.*, "The Notion of Self-aware Computing," in *Self-Aware Computing Systems.*, 2017, pp. 3–16.

[18] H. A. Simon, *The Sciences of the Artificial*. MIT Pess, 1996.

[19] B. Lawson, *How Designers Think*. Taylor and Francis, 2004.

[20] N. Cross, *Design Thinking*. Berg, 2011.

[21] F. P. Brooks, *The Design of Design*. Addison-Wesley / Pearson Education, 2010.

[22] B. Ghit *et al.*, "V for Vicissitude: The Challenge of Scaling Complex Big Data Workflows," in *CCGrid*, 2014, pp. 927–932.

[23] A. Iosup, A. Uta, L. Versluis, G. Andreadis, E. van Eyk, T. Hegeman, S. Talluri, V. van Beek, and L. Toader, "Massivizing computer systems: a vision to understand, design, and engineer computer ecosystems through and beyond modern distributed systems," *CoRR*, vol. abs/1802.05465, 2018. [Online]. Available: http://arxiv.org/abs/1802.05465

[24] F. F. Nah, "A study on tolerable waiting time: how long are Web users willing to wait?" *Behaviour & IT*, vol. 23, pp. 153–163, 2004.

[25] R. Taylor *et al.*, "Death by a thousand SLAs: a short study of commercial suicide pacts," *Forschungsbericht, HPLabs*, 2005.

[26] Iosup *et al.*, "On the dynamic resource availability in grids," in *GRID*, 2007, pp. 26–33.

[27] Gallet *et al.*, "A Model for Space-Correlated Failures in Large-Scale Distributed Systems," in *Euro-Par*, 2010, pp. 88–100.

[28] Yigitbasi *et al.*, "Analysis and modeling of time-correlated failures in large-scale distributed systems," in *GRID*, 2010, pp. 65–72.

[29] H. S. Gunawi *et al.*, "Why does the cloud stop computing? lessons from hundreds of service outages," in *SoCC*, 2016, pp. 1–16.

[30] C. Yang *et al.*, "Understanding the market-level and network-level behaviors of the Android malware ecosystem," in *ICDCS*, 2017.

[31] G. Gu *et al.*, "BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection," in *USENIX Security Symposium*, 2008, pp. 139–154.

[32] Y. Duan *et al.*, "Everything as a Service (XaaS) on the Cloud: Origins, Current and Future Trends," in *CLOUD*, 2015.

[33] N. Herbst *et al.*, "Ready for Rain? A View from SPEC Research on the Future of Cloud Metrics," *CoRR*, vol. abs/1604.03470, 2016.

[34] European Commission, "Uptake of cloud in europe. digital agenda for europe report," Digital Agenda for Europe report. Publications Office of the European Union, Luxembourg., Sep 2014.

[35] S. Chakrabarti, "Hard questions: What effect does social media have on democracy?" Facebook Online Newsroom, https://newsroom.fb.com/news/2018/01/effect-social-media-democracy/, Jan 2018.

[36] M. Märtens *et al.*, "Toxicity detection in multiplayer online games," in *NETGAMES*, 2015, pp. 1–6.

[37] The Economist, "Taming the titans," Jan 20–26, 2018, p.11–12, https://www.economist.com/printedition/2018-01-20, Jan 2018.

[38] M. Snir, "Computer and information science and engineering: one discipline, many specialties," *Commun. ACM*, vol. 54, no. 3, 2011.

[39] A. R. Hevner *et al.*, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, 2004.

[40] A. Iosup *et al.*, "Grid Computing Workloads," *IEEE Internet Computing*, vol. 15, pp. 19–26, 2011.

[41] P. Brereton *et al.*, "Lessons from applying the systematic literature review process within the software engineering domain," *J. of Sys. and Softw.*, vol. 80, no. 4, 2007.

[42] H. E. Bal *et al.*, "A Medium-Scale Distributed System for Computer Science Research: Infrastructure for the Long Term," *IEEE Computer*, vol. 49, pp. 54–63, 2016.

[43] A. Iosup *et al.*, "LDBC Graphalytics: A Benchmark for Large-Scale Graph Analysis on Parallel and Distributed Platforms," *PVLDB*, vol. 9, pp. 1317–1328, 2016.

[44] A. Ilyushkin *et al.*, "An Experimental Performance Evaluation of Autoscalers for Complex Workflows," in *TOMPECS (Best Paper Nominations from ICPE'17, revised and extended versions)*, 2018.

[45] W. G. Vincenti, *What Engineers Know and How They Know It*. JohnS Hopkins University Press, 1990.

[46] J. Plaice, "Computer Science Is an Experimental Science," *ACM Comput. Surv.*, vol. 27, p. 33, 1995.

[47] W. F. Tichy, "Should Computer Scientists Experiment More?" *IEEE Computer*, vol. 31, pp. 32–40, 1998.

[48] D. Feitelson, "Experimental computer science: The need for a cultural change," Technical Report, http://www.cs.huji.ac.il/~feit/exp/, Dec 2006.

[49] D. Epema, "Computer science as an experimental science," Quadraad magazine, https://www.cs.vu.nl/das4/Quadraad_2011-2-DAS4.pdf, Aug 2011.

[50] T. Ideker *et al.*, "A new approach to decoding life: Systems Biology," *Annu. Rev. Genomics Hum. Genet.*, vol. 2, pp. 343–72, 2001.

[51] H. Kitano, "Systems Biology: a brief overview," *Science*, vol. 295, pp. 1662–4, 2002.

[52] L. Alberghina *et al.*, *Systems Biology*. Springer, 2005.

[53] B. Meyer, "Towards empirical answers to important software engineering questions," Blog, https://bertrandmeyer.com/2018/01/26/towards-empirical-answers-important-software-engineering-questions/, Jan 2018.

[54] G. Pahl *et al.*, *Engineering Design*. Springer-Verlag, 2007.

[55] J. J. Shah *et al.*, "Metrics for Measuring Ideation Effectiveness," *Design Studies*, vol. 24, no. 2, 2003.

[56] A. Iosup *et al.*, "Self-awareness of Cloud Applications," in *Self-Aware Computing Systems.*, 2017, pp. 575–610.

[57] H. Bahrami *et al.*, *Super-Flexibility for Knowledge Enterprises*. Springer Verlag, 2005.

[58] J. Gray, "What next?: A dozen information-technology research goals," *J. ACM*, vol. 50, pp. 41–57, 2003.

[59] M. Y. Vardi, "Computer professionals for social responsibility," *Commun. ACM*, vol. 61, no. 9, 2018.

[60] Y. Noguchi, "Will work for no benefits: The challenges of being in the new contract workforce," NPR Series on the Rise of the Contract Workers, https://tinyurl.com/NPR18ContractWork, Jan 2018.

[61] P. J. Denning *et al.*, "The profession of IT - Who are we - now?" *Commun. ACM*, vol. 54, pp. 25–27, 2011.

[62] D. G. Feitelson, "Experimental computer science: The need for a cultural change," The Hebrew University of Jerusalem, Tech. Rep., 2005.

[63] M. Baker, "Is there a reproducibility crisis?" *Nature*, vol. 533, no. 7604, 2016.

[64] P. J. Denning *et al.*, *Great Principles of Computing*. MIT Press, 2015.

[65] M. Y. Vardi, "Where have all the workshops gone?" *Commun. ACM*, vol. 54, p. 5, 2011.

[66] ——, "More debate, please!" *Commun. ACM*, vol. 53, no. 5, 2010.

[67] S. Palkar *et al.*, "Weld: Rethinking the Interface Between Data-Intensive Applications," *CoRR*, vol. abs/1709.06416, 2017.

[68] F. Gröne *et al.*, "Zero infrastructure anything-as-a-service: A technology operating model for the cloud-centric era," https://www.strategyand.pwc.com/reports/zero-infrastructure, Nov 2015.

[69] Y. Jararweh *et al.*, "Software defined cloud: Survey, system and evaluation," *FGCS*, vol. 58, pp. 56–74, 2016.

[70] E. van Eyk *et al.*, "The spec cloud group's research vision on faas and serverless architectures," in *WoSC at Middleware 2017*, 2017.

[71] R. D. Peng, "Reproducible research in computational science," *Science*, vol. 334, pp. 1226–1227, 2011.

[72] D. C. Ince *et al.*, "The case for open computer programs," *Nature*, vol. 482, no. 7386, 2012.

[73] B. Howe, "Virtual appliances, cloud computing, and reproducible research," *Comput. in Sci. & Eng.*, vol. 14, pp. 36–41, 2012.

[74] J. Vitek *et al.*, "R3: Repeatability, reproducibility and rigor," *ACM SIGPLAN Notices*, vol. 47, no. 4a, 2012.

[75] M. Satyanarayanan, "The Emergence of Edge Computing," *IEEE Computer*, vol. 50, pp. 30–39, 2017.

[76] G. Ananthanarayanan *et al.*, "Real-time video analytics: The killer app for edge computing," *IEEE Computer*, vol. 50, 2017.

[77] A. Iosup *et al.*, "The OpenDC Vision: Towards Collaborative Datacenter Simulation and Exploration for Everybody," in *ISPDC*, 2017.

[78] J. Schopf, "Ten actions when grid scheduling," in *Grid Resource Management: State of the Art and Future Trends*, 2004, pp. 15–23.

[79] M. Boon, "Computer Science Is an Experimental Science," *Int'l. Studies in Phil. Sci.*, vol. 30, pp. 27–48, 2006.

[80] M. Hiltzik, *Big Science: Ernest Lawrence and the Invention that Launched the Military-industrial Complex*. Simon&Schuster, 2016.

[81] L. Evans, "The large hadron collider," *New Journal of Physics*, vol. 9, no. 9, 2007.

[82] C. Carilli *et al.*, "Science with the Square Kilometer Array: motivation, key science projects, standards and assumptions," *arXiv preprint astro-ph/0409274*, 2004.

[83] W. Cirne *et al.*, "Labs of the World, Unite!!!" *J. Grid Comput.*, vol. 4, pp. 225–246, 2006.