Asit Dan
IBM

Priya Narasimhan
Carnegie Mellon University

Dependable Service-Oriented Computing

Distributed computing, in which an application runs over multiple independent computing nodes, has a higher risk of one or more nodes failing than a centralized, single-node environment. On the other hand, distributed computing can also make an overall system more dependable by detecting those faulty nodes — whether they’re due to an underlying hardware or software failure or to compromised security through malicious attacks — and then redistributing application components or coordinating them via predefined protocols to avoid such problems. So, traditional dependability studies focus on fault detection, protocols for redistributing application components and coordinating them across nodes, and even failure estimation using system and component characterization.

One key characteristic of traditional dependability research is that overall systems have a centralized design with predefined protocols for detecting, avoiding, and coordinating a set of homogeneous nodes. In service-oriented computing (SOC), on the other hand, an overall business solution comprises many reusable services, where each service is designed, developed, and deployed independently of the others. It can thus realize a business process via runtime orchestration of a set of loosely coupled services, making SOC-based solutions agile. However, because SOC solutions comprise such independent services, each with its own underlying platform, they raise new dependability issues. At the same time, SOC can extend the scope of developing dependable solutions beyond just runtime protocols — that is, by focusing on independently developed services’ entire life cycles (design, development, deployment, and runtime management). In the past few years, interest has been increasing in dependable SOC, both in industry and academia.

State of the Art in Dependability
Service-oriented applications often comprise numerous distributed components across independent administrative domains, spanning application components, databases, Web servers, compute nodes, storage nodes, and so on. An en-
tire SOC infrastructure must be self-managing and dependable. Developing high-confidence service-oriented architectures (SOAs) will require us to develop new tools and practices in the dependability field and to rethink how we currently construct distributed systems.

According to the IFIP Working Group on Dependable Computing and Fault Tolerance,7 “the notion of dependability, defined as the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers, enables these various concerns to be subsumed within a single conceptual framework.” So, many different aspects of a system’s design affect its dependability. A broad taxonomy on dependability is available elsewhere.8 Duncan Russell, Nik Looker, and Jie Xu characterize the study of dependability as focusing on three aspects:1

- a way to measure a system’s dependability;
- an understanding of what can affect system dependability; and
- ways to increase a system’s dependability.

These aspects are known, respectively, as attributes, threats, and means.

Typical dependability attributes that apply to most systems (and are generally agreed upon) include system availability and a set of closely related attributes, such as reliability (the ability to maintain quality of service); performability (which combines performance and availability); and security-related attributes such as confidentiality (protecting information access from unauthorized users) and integrity (absence of improper alterations). For systems that undergo modifications with time, maintainability is another key measure. Because SOC dynamically assembles services to compose solutions, we can define and measure many more attributes associated with a service life cycle to assure the overall solution’s dependability.

The state of the art in fault tolerance attempts to quantify dependability through fault injection in a single service or system. However, although we understand how to quantify dependability of a single system, we don’t know how to do the same for a distributed composition of independent, loosely coupled services. We don’t yet have a way of making quantitative but objective statements that encompass SOC, such as “service A is x percent dependable, service B is y percent dependable, and when the two are composed, the resulting service composition is z percent dependable.” Researchers haven’t examined how to extend or enhance existing reliability modeling techniques to address the loose coupling and intrinsic dependencies among services in the emerging SOC landscape.

Systematically understanding various aspects of system design that affect a solution’s overall dependability also leads to processes, policies, and best practices—that is, governance—for designing, developing, and managing a system to improve dependability. Decades of experience with enterprise systems have yielded reasonably good abstractions and technologies—such as transactions, discovery, naming, and event notification—for developing reusable middleware services. Additionally, some industrial standards support dependability; for instance, the fault-tolerant CORBA standard9 aims to provide strong replica consistency under faults via server redundancy techniques.

Much less consensus exists, however, when it comes to appropriate abstractions and technologies for fundamental system services that would provide end-to-end dependable SOC. Little insight or experience is available to tell us how we can accomplish assembly at the framework, library, or architectural levels to support a range of dependable services. For instance, such systems will demand and require “multi-ility” services, such as secure, reliable, and predictable replica-based fault detection and fault containment. However, developing multi-ility frameworks that must operate across independent, potentially mutually distrustful environments is a relatively unexplored area with myriad research challenges that require us to revisit today’s abstractions and dependable technology solutions. Ideally, future SOAs should let us assemble distributed services quickly, dependably, and with assured functional behavior, quality of service (QoS), and concrete ways to verify, validate, and quantify properties (or “-ilities”).

New Challenges and Opportunities in SOC

The emerging SOC paradigm is changing how enterprises architect, develop, deliver, and use distributed software systems. As SOC gains momentum, dependability is likely to become an important driving factor and also a key competitive differentiator for the effective, 24/7,
highly available deployment of real-world services that meet business requirements. SOC's new set of challenges require us to revisit dependable distributed computing principles and understand how to apply, transform, or revolutionize dependability practices for use in the emerging SOC world.

SOC research must address various knowledge barriers. Much of a deployed SOA's administrative cost is likely to arise from manually finding and fixing various problems over a system's lifetime. To provide high-confidence SOC-based platforms, and to mitigate administrative burdens for supported applications, systems require automated self-management and troubleshooting to determine a problem's root cause. Such systems must then perform recovery that targets the root cause appropriately, rather than distracting administrators' attention with potential red herrings.

Current software systems often guarantee or test dependability only for individual services and not for an aggregate set of services that compose a system. When services must work together in a SOA context, their interdependencies make it difficult to provide effective dependability. Failures in one service can propagate to others, leading to dependent failure modes and emergent runtime behavior that might obscure the root cause. Although small problems might be individually recoverable, they could "change shape" and escalate into a larger problem in SOC's aggregate context. The dynamic or late binding of services can also make it difficult to anticipate every possible dependency and path of fault propagation ahead of runtime. Furthermore, multiple root causes can influence how a problem manifests, and multiple (possibly distributed) observed manifestations can exist for the same underlying root cause.

So, numerous significant research challenges arise for supporting automated self-managed SOAs. What information should we monitor, and how often, to accurately and rapidly diagnose the source of failures, attacks, or problems? How do we provide effective fault-containment and root-cause analysis in the face of failure propagation and coupling between services, particularly in large-scale SOC-based systems? How do we incorporate cost models for service-oriented systems when engineering large-scale, service-oriented systems? How do we perform distributed service provisioning, resource management, scheduling, and federation across (independent administrative) domains?

Another research challenge is our lack of understanding in how to construct dependable, self-tuning SOC-based infrastructures to provide the right amount and right type of fault tolerance for any given application. Dependability has many building blocks, including group communication protocols, state-machine replication, primary-backup replication, and transaction processing. Each option comes with its own configuration parameters — for example, check-pointing frequency or heartbeat timeouts. Presented with so many options, the average software developer who wants fault tolerance for his or her system is ill-equipped to decide which option is best or how to properly configure that option. Furthermore, we have no way to map a high-level service-level agreement (SLA) (for example, x 9's of availability, downtime not to exceed y minutes per month, and so on) into these plumbing-level options. A fault-tolerance expert must often work with service developers and profile the service's behavior for a long time period before jointly making any fault-tolerance decisions. The barrier here

---

Dependable SOA Resources

You can find numerous resources on dependable systems and service-oriented computing.

Conferences, Organizations, and Standards

- International Conference on Service-Oriented Computing, www.icsoc.org
- The Open Group, www.opengroup.org
- OASIS Web Services Reliable Messaging (WSRM), www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsm

Newsletters and News Updates

- Fault-tolerant systems and services, www.cs.cmu.edu/~priya/fault-tolerance-resources.html
is that we’re not sure how to mix and match or configure various fault-tolerance options to suit diverse services’ needs in both the individual and the aggregate, or how to translate high-level reliability service-level objectives into appropriate infrastructural capabilities.

Other research challenges for dependability involve service composition. If we are to assemble distributed systems out of key building-block services, then we must consider how we can assemble these services quickly, dependably, and without significantly compromising each service’s capabilities. How do we decompose an existing or future system into the “right” services from dependability perspectives? Because services might themselves be composed of other (sub)services, what is the right granularity at which to consider service composition from a dependability perspective? How do we layer coordination, orchestration, assembly, and processing on top of independent services? How do we capture anticipated and emergent or unanticipated dependencies and interactions between services in a semantically rich way? What new language formalisms will be required to articulate and analyze services and cyber–physical systems, both at service-development and service-assembly time?

A well-established SLA between a service consumer (that is, a business process or composite service) and a service provider can assure both parties of the anticipated workload and QoS to be supported. Similarly, precertifying services can assure code quality for business process assembly and avoid the need for full-scale testing of an overall solution. Finally, how do we achieve correct-by-construction service composition without impacting the flexibility needed to add or modify services on the fly?

In summary, making SOC dependable requires additional focus on the following areas:

- SOA governance in modeling, assembly, deployment, and management for making service-based solutions dependable;
- life-cycle management for evolving existing services to meet new requirements;
- service customization and assurance using SLAs;
- identifying and addressing emerging threats in the SOC world;
- fault tolerance, fault containment, and failure diagnosis;
- experiences with real-world dependable service deployments; and
- architecting or quantifying dependability.

The articles in this special issue address some of these challenges by focusing on new methodologies, tools, and runtime systems.

**In this Issue**

In SOC, the loose coupling of services in composing end-to-end business processes and the underlying runtime middleware for orchestrating, routing, mediating, and transforming service requests, provide ample opportunities for monitoring execution and collecting detailed information to pinpoint problems as they arise and subsequently avoiding faulty services by invoking alternate services. “Building Accountability Middleware to Support Dependable SOA,” by Kwei-Jay Lin, Mark Panahi, Yue Zhang, Jing Zhang, and Soo-Ho Chang, proposes using embedded intelligent agents to gather detailed online information and then using efficient algorithms to analyze this data and diagnose any problems for subsequent execution reconfiguration. The article describes an Intelligent Accountability Middleware Architecture (Llama) and demonstrates, through laboratory experiments, the feasibility of this approach as well as the modest performance overheads it incurs.

The second article, “A Dependable ESB Framework for Service Integration,” by Jianwei Yin, Hanwei Chen, Shuiguang Deng, Zhaoxui Wu, and Calton Pu, reports on a real-world, large-scale deployment of a dependable enterprise service bus (ESB) that ensures a service is both secure and available. In deploying this solution over the unreliable and insecure Internet, this approach uses efficient end-to-end security via encryption, signatures, authentication, and authorization, meeting stringent security needs in the healthcare field, as well as an error-tolerant dynamic service invocation.

Although both articles focus on the SOC runtime environment, we expect that future SOC research will lead to similar innovations for design, assembly, and deployment phases—that is, during a service’s entire life cycle.

**References**

1. D.J. Russell, N. Looker, and J.Xu, “SOA, Dependability,

Asit Dan is a SOA innovation leader in IBM’s SOA Advanced Technology Software Group, driving adoption of innovative SOA-based solutions using emerging technologies across IBM. Prior to this position, he was with IBM Research since 1990. His research interests include information as a service (IaaS), SOA governance, and other aspects of service-oriented computing. Dan has a PhD in electrical and computer engineering from the University of Massachusetts. Contact him at asit@us.ibm.com.

Priya Narasimhan is an associate professor of electrical and computer engineering at Carnegie Mellon University and an Alfred P. Sloan research fellow. Her research interests include various aspects of dependable systems, middleware for embedded systems and sensor networks, and service-oriented computing. Narasimhan has a PhD in computer engineering from the University of California, Santa Barbara. Contact her at priya@cs.cmu.edu.