Software Engineering Institute

Emerging Technologies for Software-Reliant Systems of Systems

Grace A. Lewis

September 2010

TECHNICAL NOTE
CMU/SEI-2010-TN-019

Research, Technology, and System Solutions (RTSS) Program

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about SEI publications, please visit the library on the SEI website (www.sei.cmu.edu/library).

Table of Contents

List of Tables

Abstract

This report presents general computation trends and a particular set of emerging technologies to support the trends for software-reliant systems of systems (SoSs). Software-reliant SoSs now tend to be highly distributed software systems, formed from constituent software systems that are operated and managed by different organizations. These SoSs are moving from a directed management structure (in which constituent systems are integrated and built for a specific purpose) to a virtual one (in which there is no central authority or centrally agreed purpose). This shift is introducing a need for new technologies to deal with the lack of central authority or centrally agreed purpose.

1 Introduction

A system of systems (SoS) is "a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities" [OUSDATL 2008]. Maier underscores that an SoS is different from a very large and complex but monolithic system, defining these five characteristics of an SoS [Maier 1998]:

- operational independence of the constituent systems
- managerial independence of constituent systems
- evolutionary development
- emergent behavior
- geographic distribution

Maier also defines four types of SoS based on their management structure [Maier 1998]:

- **Directed**: Constituent systems are integrated and built to fulfill specific purposes.
- **Acknowledged**: The SoS has recognized objectives, a designated manager, and resources.
- **Collaborative**: Constituent systems voluntarily agree to fulfill central purposes.
- **Virtual**: No central authority exists and there is no centrally agreed purpose for the SoS.

A software-reliant SoS is an SoS that relies heavily on software to accomplish its goals. In accordance with Maier's characteristics, software-reliant SoSs tend to be highly distributed software systems formed from constituent software systems that are operated and managed by separate organizations.

As software-reliant SoSs move from a directed management structure and toward a virtual one, new technologies are necessary to deal with the lack of central authority or centrally agreed purpose. Two examples illustrate the challenges that today's technologies fall short of addressing:

- SoS developers need to seamlessly and rapidly integrate constituent SoSs without central direction.
- Constituent systems need to be able to seamlessly and rapidly join (and leave) an SoS.

Even though technologies such as service orientation, componentization, data warehousing, and user empowerment via tools such as mashups are steps in the right direction, they usually require upfront agreements between SoS integrators and constituent systems that make the process less rapid and seamless than desired, or neither rapid nor seamless at all.

The purpose of this report is to present an informal survey of technologies that are, or are likely to become, important for software-reliant SoSs in response to current computing trends. Section 2 of this report includes some general computing trends over the past few years. Section 3 includes a set of emerging technologies for meeting these trends. Section 4 presents some conclusions and final thoughts.

2 General Computing Trends

Against a backdrop of increased globalization, as well as a growing need for business agility and environmental awareness, several general computing trends are shaping the way that organizations are building SoSs to support their business and operational needs. These trends are discussed in the following sections in no particular order; for each trend, the implication for software-reliant SoS is given.

2.1 Loose Coupling

In software systems, coupling is the degree to which a system element relies on other system elements to perform its tasks. Loose coupling is a low degree of dependence between system elements that potentially leads to high modifiability, because changes are localized; and high interoperability, because elements are not constrained by dependencies. An abundance of technologies promote two types of loose coupling:

1. between capabilities and the consumers of those capabilities to ease integration
2. between system elements that contain capabilities and the interfaces exposed to consumers of those capabilities such that implementation details are hidden from consumers

From a software-reliant SoS perspective, this trend requires standardization of capability interfaces as well as ways to describe those capabilities.

2.2 Global Distribution of Hardware, Software, and People

Globalization is an essential part of software systems in many ways, including the following:

- Software systems are often built by multinational teams.
- Many organizations use offshoring as a way to reduce costs of software development.
- Large web-based systems often use distributed caching services for better response times.

From a software-reliant SoS perspective, this trend requires greater coordination of distributed hardware, software, and people—as well as better technologies for fault detection and recovery in distributed systems.

2.3 Horizontal Integration and Convergence

The computing industry's approach to integrating applications and platforms is moving from vertical to horizontal integration. In a vertical integration approach, a single manufacturer controls platform, middleware, and applications, bundling them into solutions for delivery to customers. Conversely, in horizontal integration, applications are expected to run on any middleware and middleware is expected to run on any platform. In addition, applications are expected to exchange data seamlessly. An example of horizontal integration is seen in the way that a SmartPhone user can provide address data that can be used to invoke a map application and the map application can then invoke a "restaurant finder" application.

This trend requires exposure of APIs at the middleware and platform levels in ways that permit software-reliant SoS developers to enable horizontal integration and convergence.

2.4 Virtualization

Virtualization in general is the abstraction of computing resources. Common forms of virtualization include

- **Network virtualization**

  Traditionally, network virtualization has referred to the division of available bandwidth into channels that can be assigned to a particular resource in real time. More recently, the term is being used to refer to the deployment and management of logical services instead of physical network resources (e.g., the logical separation of resources according to user roles or privileges).

- **Storage virtualization**

  This type of virtualization involves the combining of physical storage devices into what appears to be a single storage device (e.g., a SAN or storage area network).

- **Server virtualization**

  This type involves the hiding of server resources (number and identity of individual physical servers, processors, and operating systems) from server users (e.g., VMs or virtual machines).

Server and storage virtualization are mostly adopted as an IT cost-savings strategy, so that resources can be better utilized. For example, an organization can have a small set of servers and assign virtual machines to projects, instead of buying a server for each project. Network virtualization is used mostly for easier network management but also IT savings. For example, multiple groups can be on the same physical network infrastructure but logically separated, instead of having separate physical networks for separate groups.

From a software-reliant SoS perspective, this trend requires the use of efficient virtualization strategies as well as improved resource hiding and interfaces to virtualized resources.

2.5 Commoditization of Technology

The price of technology is decreasing to a point that technology is ubiquitous. Most people have access to computers, many organizations offer online services, and advances in handheld devices are making it possible for people to have access to these services at any time.

In addition, because of commoditization,[1] it is becoming difficult for technology vendors to differentiate their products or to hold large market shares for a long period of time. To sustain market share, technology vendors have to add value through customizing their products or create new products to continually differentiate themselves from their competitors.

Technology commoditization requires software-reliant SoSs to be built in a way that minimizes the impact of changing technologies while making them accessible from a wide variety of devices.

[1] The definition of commodity being used to explain this trend is “a good or service whose wide availability typically leads to smaller profit margins and diminishes the importance of factors (as brand name) other than price” from the Merriam-Webster dictionary.

2.6 End-User Empowerment

End users want access to large amounts of information in real time. Because of technology commoditization, and because technology is getting easier to use, end users are also tending to be more competent with technology. End users want technologies that will help them get access to this information and process it without having to wait for developers to create the proper programs and reports.

From a software-reliant SoS perspective, this trend requires the awareness of what end users can and want to do now, even if they have not been trained as software developers.

2.7 Large-Scale Data Mining

Data is everywhere. There is more and more data to analyze, process, and transform into useful information in real time. Data warehouses and business intelligence are common products and technologies in industry. There is active research in this area for mining of business, scientific, and practically any other type of large heterogeneous data sets.

From a software-reliant SoS perspective, this trend requires the use of more efficient algorithms for pre-processing, processing, clustering, and analyzing large amounts of data, as well as the proper storage and computation power to do this in near real time. It will also require the use of data structures more efficient than relational databases, such as the ones being used in Facebook, Google, Twitter, and others [Bain 2009].

2.8 Low Energy Consumption

Research in low energy consumption is being driven by environmental concerns as well as the increased computing power in handheld devices. Related to low-energy computing, this trend is being advanced through work such as energy-efficient processor and computer architectures and energy-friendly computer and data centers [Intel 2010, DOE 2010].

From a software-reliant SoS perspective, this trend requires more research in energy efficiency, extending into algorithms and software that demand fewer computational cycles or take better advantage of existing computational resources.

2.9 Multi-Core and Parallelization

Computer processors were originally developed with only one core (the processing part of a CPU or central processing unit). Single-core processors process one instruction at a time. Multi-core processors have two or more independent cores in order to process multiple instructions in parallel. Multi-core processing seeks to improve performance through this parallelism, instead of by trying to make individual cores faster. However, the performance gained by use of multi-core processors highly depends on software algorithms and implementation that can be parallelized.

Therefore, from a software-reliant SoS perspective, this trend requires better software algorithms and implementation that can take advantage of having multiple cores.

3 Emerging Technologies

This section provides a list of technologies that are emerging to meet the computing trends described in Section 2. The list is simply in alphabetical order.

3.1 Cloud Computing

Cloud computing has sparked the interest of a wide range of organizations. In general, cloud computing is a distributed computing paradigm that focuses on providing users with access to scalable and virtualized hardware or software infrastructure over the internet.

Based on capabilities, there are three types of cloud computing implementations:

1. **Infrastructure as a Service (IaaS)**

   IaaS is mainly computational infrastructure available over the internet, such as compute cycles and storage, which can be utilized in the same way as internally owned resources. IaaS providers enforce minimal restrictions on their consumers to allow them maximum control and configuration of the resources. These resources typically provide a variety of interfaces to facilitate interaction, and there are usually additional services provided such as a query service for storage resources. Examples of commercial IaaS providers include Amazon Elastic Compute Cloud (EC2), Amazon Simple Storage Solution (S3), IBM Computing on Demand (CoD), and Microsoft Live Mesh [Amazon 2010a, Amazon 2010b, IBM 2010, and Microsoft 2010a].

2. **Platform as a Service (PaaS)**

   PaaS refers to application development platforms—hardware and software components—that enable developers to leverage the resources of established organizations in order to create and host applications of a larger scale than an individual or small organization would be able to handle. Services include, but are not limited to, software installation and configuration, resource scaling, platform maintenance and upgrading. Examples of commercial PaaS providers include Akamai EdgePlatform, Force.com, Google App Engine, Microsoft Azure Services Platform, and Yahoo! Open Strategy (Y!OS) [Akamai 2010, Salesforce 2010a, Google 2010a, Microsoft 2010b, and Yahoo 2010].

3. **Software as a Service (SaaS)**

   SaaS focuses on providing users with business-specific capabilities—hardware and software applications. In general, SaaS is a model of software deployment in which a provider licenses an application to customers for use as a service on demand. Examples of commercial SaaS providers include Google Apps, Salesforce.com, and Zoho [Google 2010b, Salesforce 2010b, and Zoho 2010].

Based on access, there are two types of cloud computing implementations or deployment models:[2]

1. **Public clouds**

   In public clouds, resources are offered as a service, usually over an internet connection, for a monthly or a pay-per-usage fee. Users can scale on-demand and do not need to purchase hardware. Cloud providers manage the infrastructure and pool resources into capacity required by consumers.

2. **Private clouds**

   Private clouds are typically deployed inside a firewall and managed by the user organization. In this case, the user organization owns the software and hardware running in the cloud, manages the cloud, and provides virtualized cloud resources. These resources are typically not shared outside the organization, and full control is retained by the organization.

Adoption drivers for cloud computing include scalability, lower infrastructure costs, and risk reduction. Barriers include challenges for meeting system quality attributes such as security, interoperability, and reliability [Strowd 2010].

From a software-reliant SoS perspective, constituent systems may reside in the cloud. Therefore it is necessary for SoS engineers to understand not only the economies of scale that are inherent in cloud computing but also the implications of using resources from the cloud from a quality attribute perspective.

Related Terms and Technologies

- **Grid Computing**

  Grid computing is a form of distributed computing based on "a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" [Foster 2008]. It is very similar to IaaS implementations of cloud computing. The main difference is that cloud computing adds an on-demand provisioning aspect and greater resource management capabilities.

- **Utility Computing**

  Utility computing is a service provisioning model in which consumers use services on a pay-per-use basis. Utility computing is also similar to IaaS implementations of cloud computing [Strickland 2008]. However, the main difference is that utility computing is simply a "resources for rent" model as opposed to the much broader approach defined by cloud computing for designing, building, deploying, and running applications in the cloud.

- **On-Demand Computing**

  On-Demand Computing is simply another term used to refer the on-demand characteristics of cloud computing and utility computing.

---

[2] The National Institute of Standards and Technology (NIST) defines two additional types of cloud deployment models: (1) community clouds that are shared by multiple organizations and support specific needs and concerns of a community and (2) hybrid clouds that are the combination of two or more public, private, and community clouds. However, both community and hybrid cloud are specialties of public and private clouds and are therefore not included in the discussion. Additional information is available at http://csrc.nist.gov/groups/SNS/cloud-computing/.

- **Containerized Data Centers**

  Containerized data centers are portable data centers that contain all the power and cooling equipment to run a data center in an energy-efficient manner. Some industry players in this growing market are Google, HP, IBM, Rackable Systems, Sun, and Verari Systems [Miller 2009].

3.2 Complex Event Processing (CEP)

Event processing refers to computing that operates on events, involving their production, transformation, detection, and consumption. Complex Event Processing (CEP) is a special form of event processing which operates on complex events. A complex event is "an event that is an abstraction of other events called its members" [Luckham 2008]. These complex events are composed or derived from a set of events related by time, causality, abstraction, or other relationships.

CEP systems find patterns in events to detect certain business opportunities or threats [Chandy 2007]. CEP solutions are commonly found in financial analysis, network security, manufacturing, asset management, management dashboard, and power grid monitoring applications. The commonality across these applications is that large numbers of events are monitored and analyzed to discern patterns that require real-time responses.

In a software-reliant SoS setting, CEP systems may monitor and operate on events produced from multiple, heterogeneous constituent systems. SoS engineers in this setting will need to focus on setting standards or mediation platforms for events that can produce results in real time.

Related Terms and Technologies

- **Event-Driven Architecture (EDA)**

  EDA is an architectural style in which some components are event-driven and communicate by means of events [Luckham 2008]. CEP is an event-processing style that can be used in an EDA.

- **Event Stream Processing (ESP)**

  ESP is performed on event streams—linearly ordered sequences of events [Luckham 2008]. CEP engines are common components of ESP solutions for building event-driven information systems.

3.3 Data Intelligence

Data intelligence is the mining, aggregation, fusion, selection, search, and exploitation of huge volumes of disparate data coming from diverse sources such as databases, events, sensor networks, human observation, human judgment, RSS (or really simple syndication) feeds, and GPS (global positioning system) data. Just as the structuring of data in context is considered to yield useful information, data intelligence can lead to a next step—knowledge.

Data intelligence relies on large-scale data mining in which significant amounts of heterogeneous, raw data go through a pre-processing stage, a transformation stage, and finally a pattern recognition stage that produces knowledge.

Data-centric software-reliant SoSs may rely on data intelligence techniques to operate on data from heterogeneous, constituent systems. SoS engineers, in addition to concentrating on data

processing algorithm complexity, will have to focus on data transformation and mediation algorithms that can be just as complex, especially when dealing with disparate data models.

Related Terms and Technologies

- **Information Superiority**

  The term used by the U.S Department of Defense (DoD) for data intelligence is information superiority: "The capability to collect, process, and disseminate an uninterrupted flow of information while exploiting or denying an adversary's ability to do the same" [DoD 2000].

- **MapReduce**

  MapReduce is a software framework that was made popular by Google to support distributed computing on large data sets on clusters of computers [Dean 2004]. An extremely simple explanation of MapReduce is that it enables a big task (such as data processing) to be divided into discrete tasks that can be done in parallel, by means of *map* and *reduce* functions that are applied to the data. The key to MapReduce is that both *map* and *reduce* functions can be distributed and run in parallel. The runtime infrastructure takes care of partitioning the input data as well as scheduling, coordinating, and managing the program's execution across a set of machines. Even though not directly related to data intelligence, MapReduce is an emerging technology to process large amounts of data and solve complex data analytics problems [MapReduce 2010].

3.4 End-User Programming (EUP)

End-user programming (EUP) describes the practice where end users write computer programs to satisfy a specific need, even though they have not necessarily been taught how to write code in conventional programming languages [EUSES 2010].

EUP has been around for a while, in the form of shell scripts and Excel spreadsheets that allow users to quickly automate tasks. However, the advent of the internet, and the recent explosion in the availability of web technologies, has made it much easier for end users to produce and customize software. From the end-user perspective, the construction of these applications can be done simply through a set of drag-and-drop operations that pull together capabilities from different sources to build a desired functionality, as in mashups and dashboard tools.

EUP in a software-reliant SoS setting may involve end users pulling together data and capabilities from different constituent systems. Not only does EUP capability have to exist at the SoS level, but also constituent systems have to be able to expose data and capabilities to be used by EUP tools at the SoS level. In addition, SoS engineers will have to balance the flexibility that EUP provides with protection against the problems that might arise from that flexibility. For example, EUP accomplished through the internet has vastly increased the use of shared code and shared data; the accompanying risk is that users are more exposed to code and data that might be of poor quality or malicious.

Related Terms and Technologies

- **Intentional Programming**

  Intentional programming is a concept that was introduced by Microsoft Research in the early 1990s [Czarnecki 2000]. The basic idea is that a software designer or programmer represents the elements of a particular domain as "intents" that correspond to high-level programming

constructs. The designer or programmer then sits with the domain experts (end users), and together they define an application's behavior using these intents in a WYSIWYG-like manner.[3] Finally, there are tools that take these composed intents and translate them into lower-level programming languages.

- **Edge Programming**

  Edge programming refers to the decentralized programming of complex systems. The main concept is that programming happens at the *edge* of these complex systems: "in multiple, separate, decentralized units that manage their own software development processes … where the knowledge and other resources needed for effective decision-making are located" [Sullivan 2007]. In this case, even though the programming is done by software professionals, edge programming addresses the decentralized characteristics of SoS.

- **Gesture Programming**

  Gesture Programming is a form of programming by human demonstration. The main concept is the capture of gestures that are translated via tools into code. A gesture could be a particular hand movement, the movement of a mouse, or the pressing an area of a touch screen [Voyles 1999]. As gesture-based input devices evolve, the potential for gesture-based programming increases [Johnson 2010].

3.5 Green Computing

Green computing refers to "the study and practice of designing, manufacturing, using, and disposing of computers, servers, and associated subsystems—such as monitors, printers, storage devices, and networking and communications systems—efficiently and effectively with minimal or no impact on the environment [Murugesan 2008]." Software-related green practices that lead to reduced power and cooling include

- Algorithmic efficiency, which translates into code that optimizes computational resource usage such as memory consumption and execution speed
- Platform virtualization, which allows the replacement of multiple physical systems with multiple virtual systems running on a single machine
- Thin clients, which consume between 6 and 50 watts, compared to the 150 to 350 watts used by typical desktops [Davis 2008]

In software-reliant SoS environments, it will become common for thinner, smarter clients to interact with virtualized, smarter systems.

- Systems and their infrastructures will monitor usage in order to regulate power consumption.
- Systems will need to serve fat and thin clients in addition to a growing community of mobile users.
- Building management systems (BAS) and energy management systems (EMS) will become common constituents of SoS.

[3] WYSIWYG is an acronym for *What You See Is What You Get*.

Related Terms and Technologies

- **Energy-Efficient Computing**

  This concept refers to the design, development, and use of computers and computer components targeted at optimized energy consumption. This approach takes advantage of energy-efficient hardware in system development and promotes day-to-day computer-related practices to reduce energy consumption (e.g., disabling screensavers and printing less).

- **Smart Grid**

  The Smart Grid is a modernization project for the United States electricity grid. The main idea is the overlay of the electricity distribution grid with an information and metering system [DOE 2008]. An advantage of the Smart Grid approach is two-way digital communication between the grid and its consumers. Two examples of how two-way communication promotes energy efficiency are

  - The grid can better understand consumer demand.
  - Smart consumer appliances can monitor energy prices in real time and make better decisions.

  Even though it is a specific initiative, the Smart Grid is representative of the direction that technology related to energy consumption is headed.

3.6 Mobile Computing

Mobile computing is a generic term that describes the possibility to use computing technology "on the go" through devices such as SmartPhones, PDAs (personal digital assistants), portable computers, and wearable computers. The mobile market today has nearly four billion subscribers [Johnson 2010]. Mobile users expect seamless access to information anytime, anywhere, and from any device.

This trend has a large implication for software-reliant SoSs because capabilities will need to be provided to mobile as well as "fixed" users. The concept of application stores or "app stores" will become a mixed-approach to the delivery of capabilities where application logic will be downloaded and installed on mobile devices with reach-back capabilities into other constituent systems such as enterprise systems. This mobility will also introduce new risks in SoS environments, mostly related to security:

- **Data leakage**

  Mobile devices such as SmartPhones can store a lot of data. If a mobile device is stolen or misplaced, any data that has been downloaded onto this device is compromised.

- **Network security**

  Wireless networks are typically less secure than wired networks. In addition, network security is harder in mobile environments mostly because of wireless broadband internet access and hotspots that enable mobile users to access enterprise systems via mobile devices.

- **Device protection**

  The protection of mobile devices should be on the top of security agendas due to the threat they pose to confidential information [Jhingan 2010]. Even if a device is not stolen or lost, hackers can gain temporary access to unprotected devices and steal data or install malicious software for capturing keystrokes or passwords.

- **Malware and viruses**

  Antivirus software is not built with mobile devices in mind [Kwang 2010]. Threats to these devices so far are still low. However, as mobile device usage increases, the cost of unlimited data plans will decrease and storage and computation capabilities will increase. It is certain that threats will grow accordingly.

Security issues can be addressed with proper governance, but enforcement is difficult unless there are specific guidelines, means to automate governance process and compliance, incentives, and penalties for non-compliance.

Related Terms and Technologies

- **Location-Based Services**

  Location-based services take advantage of capabilities of mobile devices and the mobile network to determine a user's location in order to deliver services that are tailored to the user's location. Location-based services are used in applications such as social applications for finding close friends or restaurants, transportation applications to track vehicles or parcels, and e-commerce applications to recommend stores or coupons in the area or emergency systems to inform of problems in the area.

- **Physical Computing**

  Physical computing refers to systems that combine hardware and software such that systems can sense and respond to the physical world. Common elements in physical computing are sensors, microcontrollers, and electro-mechanical control devices. Other elements include the support for computer vision, motion detection, and voice recognition capabilities [O'Sullivan 2004]. Even though the concept has existed for many years, physical computing opportunities as a complement to mobile computing have increased with the growing complexity and creativity of input and output devices, as well as the availability of programming platforms such as Arduino [Arduino 2010].

3.7 Opportunistic Networks

Opportunistic networks, or oppnets, are different from traditional networks because they are not pre-designed in terms of number and location of nodes. Link performance in oppnets is often highly variable [Huang 2008].

In general, oppnets start with the deployment of a small, pre-designed *seed oppnet*. The oppnet then starts *growing* by detecting diverse systems present in its vicinity. Each detected system is evaluated and those with the best "scores" evaluations are invited (or ordered) by an oppnet to become its helpers. These helpers are then employed to execute tasks such as communications, computing, storage, and sensing, in support of the oppnet's goals [Lilien 2006]. Each one of these steps poses significant technical challenges, in addition to privacy and security challenges for constituent systems.

Related Terms and Technologies

- **Mobile Ad-Hoc Network (MANET)**

  A MANET is a self-configuring network in which nodes are mobile devices connected by wireless links [Conti 2007]. Oppnets are considered specializations of MANETs. The main differences between oppnets and MANETs are that in MANETs

  - Communication is usually synchronous.
  - An assumption is that every node wants to contribute (help).

- **Vehicular Ad-Hoc Network (VANET)**

  VANETs are a type of MANET in which nodes are vehicles and roadside equipment.

- **Mesh Network**

  A mesh network is a type of ad-hoc network in which it is possible to get from one node to another via one or more *hops*. The main benefit of mesh networks is that the network can continue to operate after problems with nodes, or with connections between nodes, because there is usually more than one path between nodes. Mesh networks are commonly wireless networks, but can be wired as well.

- **Unstructured Peer-to-Peer (P2P) Network**

  P2P networks are distributed networks in which each node shares resources such as CPU cycles, storage, and bandwidth with other nodes in the network, without the need for a central coordinator. All nodes are both suppliers and consumers of resources. Typically, an overlay layer is created on top of the network layer for connectivity, routing, and messaging. In an unstructured P2P network, the overlay layer is not well defined, and node connections are random. However, over time, the network becomes self-organized as many nodes and their content become known to the rest of the network [Buford 2009].

- **Wireless Sensor Network**

  A wireless sensor network is another form of ad-hoc network in which nodes cooperate to monitor physical or environmental conditions, such as temperature, sound, vibration, light intensity, motion, or proximity to objects [Raghavendra 2006].

- **Cognitive Network**

  A cognitive network is a form of ad-hoc, self-organizing network that has a cognitive process at the node and network levels. The cognitive process is used for perceiving current network conditions and then planning, acting, and deciding on those conditions in order to meet certain goals [Mahmoud 2007]. Cognitive networks therefore have the capability to adapt in response to certain conditions, based on prior reasoning and acquired knowledge.

3.8 Self-* Computing

Self-* computing refers to systems that are aware of their environment and adaptable to changing characteristics of the environment. Some terms or capabilities associated to such systems include

- self-adaptation
- self-awareness
- self-configuration and reconfiguration
- self-healing

- self-knowledge of components
- self-optimization
- self-protection

The primary goals of these systems are to reduce

- response time to changes in the environment
- the amount of human intervention in management tasks

These goals apply to software-reliant SoSs, especially as they tend toward collaborative and virtual SoS types in which there is less management and control of constituent systems.

Related Terms and Technologies

- **Autonomic Computing**

  Autonomic computing, an initiative started by IBM in 2001, refers to computer systems that can manage themselves given high-level goals from administrators [Kephart 2003]. The four main aspects of autonomic computing are self-configuration, self-optimization, self-healing and self-protection.

- **Biomimetics**

  Biomimetics refers to the study and imitation of nature's methods, design, and processes in order to solve human problems [Bar-Cohen 2005]. From a computing perspective, biomimetics is being applied in systems such as autonomous robots and vehicles, machine vision systems, machine hearing systems, and navigational systems.

- **Sociomimetics**

  Still at a very early stage of development and application, sociomimetics refers to the study and imitation of social behavior patterns in electronic information systems as a technique to enhance their effectiveness [Cross 2006].

3.9 Social Computing

Social computing is a "general term for an area of computer science that is concerned with the intersection of social behavior and computational systems."[4] There is a wide range of examples of social computing. The better-known side of social computing is related to social software such as wikis, blogs, instant messaging, and collaboration tools. However, the lesser-known side of social computing, which is of greater interest and application to software-reliant SoSs, is that of socially inspired computation. In socially inspired computing, groups of people carry out the computation; examples of this type of social computing include

- **Collaborative Filtering**

  Collaborative filtering refers to mechanisms for making automated predictions for a user based on similar data from other users. The most common application of collaborative filtering is a recommendation system for products or websites based on what other users have bought or links they have followed.

[4] This generic definition is from Wikipedia: http://en.wikipedia.org/wiki/Social_computing.

- **Online Auctions**

  Online auctions enable the electronic selling and buying of products and services via auction sites such as eBay (http://www.ebay.com). In online actions, buyers and sellers give transaction feedback that is aggregated and published as a rating. This rating can be used to make decisions on whether to buy from or sell to a particular user. Buyer and seller networks are also studied to detect fraudulent ratings.

- **Prediction Markets**

  Prediction markets are specialized, small-scale financial markets operated to predict future events. In theory, the market price used for prediction is based on the collaborative knowledge of the participants. Based on knowledge or inside information about a particular situation, each participant can *bet* on a particular outcome, using shares or contracts purchased for that purpose (usually not with real money). Current uses of prediction markets include election outcomes, disease outbreaks, sports scores, sales predictions, and public opinion. Examples of prediction market sites are InTrade (http://www.intrade.com), Iowa Electronics Market (http://tippie.uiowa.edu/iem) and simExchange (http://www.simexchange.com).

- **Social Tagging**

  Tagging is an activity in which end users add their own metadata or keywords to resources. A resource can have many tags that are generated by many different users. These tags are managed by a classification software or system that will provide links to other items with the same or related tags. The benefit of social tagging is that items are classified from the perspective of the end users as opposed to that of the resource creator or owner. The information tagging systems created in the context are often called folksonomies. Examples of social tagging sites are Delicious (http://www.delicious.com) and Digg (http://www.digg.com).

Related Terms and Technologies

- **Enterprise 2.0**

  Enterprise 2.0 refers to the use of social computing within the enterprise. The scope of *enterprise* in this context incorporates business partners as well as customers and the public [Hinchcliffe 2010].

- **Social Information Processing**

  Social Information Processing is another term that refers to the collective creation, annotation, evaluation, and sharing of content via social computing tools and technologies. The concept extends beyond blogging and tagging to collective problem solving such as that promoted by Amazon's Mechanical Turk, a marketplace for people that need to solve hard problems (https://www.mturk.com/mturk/welcome).

4 Conclusions

This report discusses computing trends and emerging technologies, as they relate to software-reliant SoS environments. Table 1 provides a rough mapping of the emerging technologies to the general computing trends.

Table 1: Mapping of Emerging Technologies to Computing Trends

Technologies	Loose Coupling	Global Distribution of Hardware, Software and People	Horizontal Integration and Convergence	Virtualization	Commoditization of Technology	End-User Empowerment	Large-Scale Data Mining	Low Energy Consumption	Multi-Core and Parallelization
Cloud Computing	X	X	X	X	X		X	X	X
Complex Event Processing	X	X				X	X		X
Data Intelligence	X	X		X		X	X		X
End-User Programming			X		X	X			
Green Computing			X	X				X	X
Mobile Computing	X	X	X		X	X		X	
Opportunistic Networks	X	X	X		X				
Self-* Computing	X	X		X	X			X	X
Social Computing		X	X		X	X			

As the table shows, not all technologies apply to each trend. Also emerging technologies can be used in combination (as well as in conjunction with existing technologies) to enable the highly distributed, heterogeneous, loosely coupled characteristics of SoSs and their constituent systems. Some examples of technologies used together include

- Service-orientation is used to provide standardized service interfaces to cloud resources and for the management of those resources.
- The availability of pervasive and mobile devices coupled with opportunistic network technologies and self-* computing mechanisms will enable more robust mobile applications in areas with poor bandwidth and connectivity.
- Data intelligence and complex event processing will work together to make sense of disparate sources of data and support decision makers that need answers in near real time.
- Advances in mobile devices and green computing technologies will enable the moving of computation to the mobile devices instead of energy-consuming servers.

- Social computing, mobile computing, and data intelligence will support ubiquitous, informal recording of data that can be classified and processed into valuable, on-time information.

In addition, while extensive, the list of emerging technologies presented in this report is not meant to be inclusive and will change over time. Gartner's yearly report on emerging technologies describes why a list like this one will change. Technologies, Gartner states, go through a "hype cycle" that includes a peak of inflated expectations, a trough of disillusionment, and a slope of enlightenment [Gartner 2009]. Further, as software-reliant SoSs move from a directed to a virtual management structure, more complex technologies will be necessary to deal with problems stemming from the lack of central authority or centrally agreed purpose. However, not every SoS must be a fully-virtual SoS, and not every SoS must be fully-directed to be successful.

Finally, the focus on emerging technologies in this report should not be taken to imply that solving problems begins with choosing a technology. It must be acknowledged that many problems are behavioral and not solvable with technology. Where technology adoption can provide a solution in a software-reliant SoS environment (really any system environment), the key is to start with the problem and then seek technologies that match it and fit the organizational context—not the other way around.

References

URLs are valid as of the publication date of this document.

[Akamai 2010]
Akamai Technologies. *Akamai Edge Platform: Application Acceleration that Delivers Content and Applications Quickly*. http://www.akamai.com/html/technology/edgeplatform.html (2010).

[Amazon 2010a]
Amazon Web Services. *Amazon Elastic Compute Cloud (Amazon EC2)*. http://aws.amazon.com/ec2/ (2010).

[Amazon 2010b]
Amazon Web Services. *Amazon Simple Storage Service (Amazon S3)*. http://aws.amazon.com/s3/ (2010).

[Arduino 2010]
Arduino. http://www.arduino.cc/ (2010).

[Bain 2009]
Bain, T. *Is the Relational Database Doomed?* ReadWrite Enterprise. http://www.readwriteweb.com/enterprise/2009/02/is-the-relational-database-doomed.php (2009).

[Bar-Cohen 2005]
Bar-Cohen, Y. *Biomimetics: Biologically Inspired Technologies.* CRC Press, 2005.

[Buford 2009]
Buford, J., Yu, H., & Lua, E. *P2P Networking and Applications.* Morgan Kaufmann, 2009.

[Chandy 2007]
Chandy, K. & Schulte, R. *The Role of Event Processing in Modern Business*. ebizQ. http://www.ebizq.net/hot_topics/cep/features/8303.html?page=1 (2007).

[Conti 2007]
Conti, M. & Giordano, S. "Multihop Ad Hoc Networking: The Reality." *IEEE Communications Magazine 45*, 4 (April 2007): 88-95.

[Cross 2006]
Cross, R. *Recovering Lost Group Communications Skills*. Bioteams. http://www.bioteams.com/2006/04/25/recovering_lost_group.html (2006).

[Czarnecki 2000]
Czarnecki, K. & Eisenecker, U. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, 2000.

[Davis 2008]
Davis, E. *Green Benefits Put Thin-Client Computing Back On The Desktop Hardware Agenda*. Forrester Research, 2008.

[Dean 2004]
Dean, J. & Gamawhat, S. "MapReduce: Simplified Data Processing on Large Clusters." *Sixth Symposium on Operating System Design and Implementation (OSDI'04)*. San Francisco, CA, December, 2004. http://labs.google.com/papers/mapreduce-osdi04.pdf

[DoD 2000]
U.S. Department of Defense (DoD). *Joint Vision 2020*. http://www.fs.fed.us/fire/doctrine/genesis_and_evolution/source_materials/joint_vision_2020.pdf (2000).

[DOE 2008]
U.S. Department of Energy (DoE). *The SMART GRID: An Introduction*. DoE, 2008.

[DOE 2010]
U.S. Department of Energy. *Energy Star Data Center Energy Efficiency Initiatives*. http://www.energystar.gov/index.cfm?c=prod_development.server_efficiency (2010).

[EUSES 2010]
EUSES Consortium. *EUSES: Home*. http://eusesconsortium.org/ (2010).

[Foster 2008]
Foster, I., Yong, Zhau, Ioan, R. & Lu, S. "Cloud Computing and Grid Computing 360-Degree Compared," 1-10. *Proceedings of the Grid Computing Environments Workshop (GCE '08)*. Austin, TX (USA), November 12-16, 2008. DOI 10.1109/GCE.2008.4738445.

[Gartner 2009]
Gartner. *Hype Cycle for Emerging Technologies*. Gartner, 2009.

[Google 2010a]
Google. *Google App Engine*. http://code.google.com/appengine/ (2010).

[Google 2010b]
Google. *Google Apps for Business | Official Website*. http://www.google.com/apps/intl/en/business/index.html (2010).

[Hinchcliff 2010]
Hinchclife, D. *Ten Emerging Enterprise 2.0 Technologies to Watch*. ZDNet. http://www.zdnet.com/blog/hinchcliffe/ten-emerging-enterprise-20-technologies-to-watch/1224?tag=mantle_skin;content (2010).

[Huang 2008]
Huang, C., Lan, K., & Tsai, C. "A Survey of Opportunistic Networks," 1672-1677. *22nd International Conference Advanced Information Networking and Applications – Workshops (AINAW 2008).* March 25-28, 2008. IEEE, 2008.

[IBM 2010]
IBM. *IBM Computing on Demand*. http://www-03.ibm.com/systems/deepcomputing/cod/index.html (2010).

[Intel 2010]
Intel Corporation. *Intel and the Environment – Eco-Smart Computing Inside*. http://www.intel.com/about/corporateresponsibility/environment/ecosmart.htm (2010).

[Jhingan 2010]
Jhingan, H. "Mobile Device Protection Should Move to Top of Security Agenda." *Voice & Data*. http://voicendata.ciol.com/content/news/110080402.asp (2010).

[Johnson 2010]
Johnson, L., Levine, A., Smith, R., & Stone, S. *The 2010 Horizon Report*. Austin, Texas: The New Media Consortium. 2010. http://www.nmc.org/pdf/2010-Horizon-Report.pdf

[Kephart 2003]
Kephart, J. & Chess D. "The Vision of Autonomic Computing." *Computer 36*, 1 (January 2003): 41-50.

[Kwang 2010]
Kwang, K. *Mobile Security Risks Persist But Limited for Now*. ZDNet Asia. http://www.zdnetasia.com/mobile-security-risks-persist-but-limited-for-now-62201532.htm (2010).

[Lilien 2006]
Leszek Lilien, Z., Kamal, Huma, & Gupta, Ajay. "Opportunistic Networks: Challenges in Specializing the P2P Paradigm," 722-726. *17th International Conference on Database and Expert Systems Applications (DEXA'06)*. Krakow, Poland, September 4-8, 2006. IEEE, 2006.

[Luckham 2008]
Luckham, D. & Schulte, R. (editors). *Event Processing Glossary - Version 1.1*. Event Processing Technical Society. 2008. http://complexevents.com/wp-content/uploads/2008/08/epts-glossary-v11.pdf

[Mahmoud 2007]
Mahmoud, Q. (Ed.) *Cognitive Networks: Towards Self-Aware Networks*. Wiley, 2007.

[Maier 1998]
Maier, M. "Architecting Principles for Systems-of-Systems." *Systems Engineering, 1*, 4 (1998): 267-284.

[MapReduce 2010]
MapReduce.org. *MapReduce.org*. http://www.mapreduce.org/ (2010).

[Microsoft 2010a]
Microsoft Corporation. *Live Mesh Beta*. http://www.mesh.com/ (2010).

[Microsoft 2010b]
Microsoft Corporation. *Windows Azure Platform.* http://www.microsoft.com/azure/ (2010).

[Miller 2009]
Miller, R. *Google Unveils Its Container Data Center*. Data Center Knowledge. 2009. http://www.datacenterknowledge.com/archives/2009/04/01/google-unveils-its-container-data-center/

[Murugesan 2008]
Murugesan, S. "Harnessing Green IT: Principles and Practices." *IEEE IT Professional* (January-February 2008): 24-33.

[O'Sullivan 2004]
O'Sullivan, D. & Igoe, T. "Physical Computing: Sensing and Controlling the Physical World with Computers." Course Technology PTR. 2004.

[OUSDATL 2008]
Office of the Undersecretary of Defense for Acquisition, Technology, and Logistics. *DoD Systems Engineering Guide for Systems of Systems*. January, 2008. Office of the Undersecretary of Defense for ATL, Washington, DC.

[Raghavendra 2006]
Raghavendra, C. S., Sivalingam, K., & Znati, T. (Eds.). *Wireless Sensor Networks*. Springer, 2006.

[Salesforce 2010a]
Salesforce.com. *Cloud Computing – salesforce.com.* http://www.salesforce.com/platform/ (2010).

[Salesforce 2010b]
Salesforce.com. *salesforce.com*. http://www.salesforce.com/crm/products.jsp (2010).

[Strickland 2008]
Strickland, Jonathan. *How Utility Computing Works.* HowStuffWorks.com. 2008. http://communication.howstuffworks.com/utility-computing.htm

[Strowd 2010]
Strowd, H. & Lewis, G. *T-Check in System of Systems Technologies: Cloud Computing* (CMU/SEI-2010-TN-009). Software Engineering Institute, Carnegie Mellon University, 2010. http://www.sei.cmu.edu/library/abstracts/reports/10tn009.cfm

[Sullivan 2007]
Sullivan, K. "Edge Programming." *Proceedings of the 29th International Conference on Software Engineering Workshops*. IEEE, 2007. http://www.cs.virginia.edu/~sullivan/ULS1/ULS07/sullivan.pdf

[Voyles 1999]
Voyles, R. & Khosla, P. "Gesture-Based Programming: A Preliminary Demonstration," 708-713. *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*. Detroit, Michigan, May 10-15, 1999. IEEE, 1999.

[Yahoo 2010]
Yahoo. *Introducing the Yahoo! Open Strategy*. http://developer.yahoo.com/yos/intro/ (2010).

[Zoho 2010]
Zoho Corporation. *Email, Hosting, CRM, Project Management, Office Suite, Document Management, Remote Support*. http://www.zoho.com/ (2010).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	September 2010	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Emerging Technologies for Software-Reliant Systems of Systems	FA8721-05-C-0003

6. AUTHOR(S)

Grace A. Lewis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER

CMU/SEI-2010-TN-019

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

HQ ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2116

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

This report presents general computation trends and a particular set of emerging technologies to support the trends for software-reliant systems of systems (SoSs). Software-reliant SoSs now tend to be highly distributed software systems, formed from constituent software systems that are operated and managed by different organizations. These SoSs are moving from a directed management structure (in which constituent systems are integrated and built for a specific purpose) to a virtual one (in which there is no central authority or centrally agreed purpose). This shift is introducing a need for new technologies to deal with the lack of central authority or centrally agreed purpose.

14. SUBJECT TERMS	15. NUMBER OF PAGES
SoS, system of systems, cloud computing, green computing, end user programming	29

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL