Service-oriented computing (SOC) in a cloud computing environment

A.I. Petrenko
National Technical University of Ukraine "Kyiv Polytechnic Institute"
tolja.petrenko@gmail.com

Abstact

Service-Oriented Computing (SOC) is an emerging paradigm for developing software systems that employ services. The distinction between SOC and traditional computing is that application builders no longer construct software from scratch using a programming language. Instead, they specify the application logic in a high-level specification language, utilizing standard services as components. This work presents the development of the ***Engineering Design Platform*** in a cloud computing environment**,** based on SOC and intended, in particular, for modeling and optimization of Nonlinear Dynamic Systems, based on components of different physical nature and being widely spread in different scientific and engineering fields.

1. Introduction

The ***Service-Oriented Computing*** (SOC) paradigm refers to the set of concepts, principles, and methods that represent computing in Service-Oriented Architecture (SOA) in which software applications are constructed based on independent component services with standard interfaces. SOC represents a new generation distributed computing platform for programming distributed applications by means of the composition of services. The visionary promise of SOC is a world-scale network of loosely coupled services that can be assembled with little effort in agile applications that may span organizations and computing platforms. Since services may be offered by different enterprises and communicate over the Internet, they provide a distributed computing infrastructure for both intra- and cross-enterprise application integration and collaboration. Service clients (end-user organizations that use some service) and service aggregators (organizations that consolidate multiple services into a new, single service offering) utilize service descriptions to achieve their objectives. So, Service-Oriented Computing is a paradigm and Service Oriented Architecture is an architectural model which allows interoperability, re-usability, loose-coupling of its components and provides mechanisms to describe publish and discover available services.

2. Web-services

The distinction between SOC and traditional computing is that application builders no longer construct software from scratch using a programming language. Instead, they specify the application logic in a high-level specification language, utilizing standard services as components.

The central definition being used here is a ***service***, the most important concept of the service-oriented paradigm. The definition of service for the W3C Working Group is: *"A service is an abstract resource that represents a capability of performing tasks that form a coherent functionality from the point of view of provider entities and requester entities. To be used, a service must be realized by a concrete provider agent."* This definition is correct but it is too abstract because too many things could be a service. There are various definitions of a service within the context of Service Oriented Computing in the literature, among of which there are the following: *"A service is a system function that is well defined, self-contained and does not depend on the context or state of other services"; "A service is a unit of work to be performed on behalf of some computing entity, such as a human user or another program".* Services perform functions that can range from answering simple requests to executing sophisticated business processes requiring peer-to-peer relationships between possibly multiple layers of service consumers and providers. Any piece of code and any application component deployed on a system can be reused and transformed into a network-available service. Services reflect a "service-oriented" approach to programming, based on the idea of composing applications by discovering and invoking network-available services rather than building new applications or by invoking available applications to accomplish some task. It is likely that in the future, all computing units, both hardware and software, both small (such embedded systems) and large (such as mainframes) will be organized as services, i.e., systems will be servicetized.

As of today the most prominent technology based on SOC is ***Web Services***, a set of open specifications that focuses on interoperability and compatibility with existing infrastructures. A Web service is a specific kind of service that is identified by a URI, whose service description and transport utilize open Internet standards. Interactions between Web services typically occur as SOAP calls carrying XML data content. Interface descriptions of the Web services are expressed using Web Services Definition Language (WSDL). The *Universal Description, Discovery, and Integration (UDDI)* standard defines a protocol for directory services that contain Web service descriptions. UDDI enables Web service clients to locate candidate services and discover their details. Service aggregators may use the Business Process Execution Language for Web Services (BPEL4WS) to create new Web services by defining corresponding compositions of the interfaces and internal processes of existing services. One of the most important aspects in SOC is *aggregation (composition)*. The public interfaces exposed by each service allow for the composition of the latter in complex workflows, in order to implement functionalities that reuse those that are already offered by the single services. At the present service composition can be done in two different approaches: ***orchestration*** and ***choreography***. In orchestration a single service, called orchestrator, is responsible for composing and coordinating the other services in order to complete the desired task. Choreography, instead, describes the interactions between the various services, which execute a global strategy in order to achieve the desired result without a single point of control. For these reasons it is said that orchestration offers a local viewpoint whereas choreography offers a global viewpoint. At the present the most credited language for dealing with service orchestration is WS-BPEL (BPEL for short). On the other hand, the reference language for choreography is WS-CDL. Service Oriented Computation deals with implementing the core services, and Service Oriented Composition/Management about managerial tasks (WS-BPEL, WS-CDL), and Service Oriented Communication would relate to message routing (*WS-Addressing, WS-Reliable Delivery, etc.***).**

3. Cloud Computing

Advancements in Cloud Computing have raised the potential of realizing service-orientation to unprecedented heights. Cloud Computing is an emerging paradigm for consumption and delivery of IT based services, based on concepts derived from consumer internet services, like self-service, apparently unlimited or elastic resources and flexible services options like IaaS (*Infrastructure as a Service*),PaaS (*Platform as a Service*) and SaaS (*Software as a Service*) . The delivery of software as a set of distributed services that can be configured and bound can help to solve problems like software reuse, deployment and evolution. The "*software as a service model*" will open the way to the rapid creation of new value-added composite services based on existing ones. Although service-oriented computing in cloud computing environments presents a new set of research challenges, their combination provides potentially transformative opportunities. With cloud computing, new Internet services can be developed and deployed without capital acquisitions of hardware or large human integration expenses.

Semantic processing of service-based interfaces, semantic service discovery, service composition and consumption, and quality of service are the current leading research topics. What happens when computing itself is the service? Essentially, this is what cloud computing provides: applications, platforms, network capabilities, and storage, all as services. Over time, it will be interesting to see how cloud-based services will enhance sharing and thus improve this web of possibilities.

SOC requires the management of loosely coupled services to maintain its working condition. Furthermore, each service within a workflow could reside with unique service providers. This is a challenge to service discovery because current service repositories are decentralized and not well advertised. In a cloud computing environment the challenge of service management and monitoring is extended. Current cloud computing providers don't offer user-customized management and monitoring mechanisms built into their infrastructure. Hence, it's still the service developer's responsibility to provide programs and utilities to manage and monitor services.'

4. SOC in Engineering Design

The IASA (Institute of Applied System Analysis) of NTUU "Kiev Polytechnic Institute" is conducting the following research and development activities in the domain of SOC target:

1). Investigating Engineering Design procedures together with partners as possible services in distributed environments instead of present attempts to migrate monolithic large CAE/CAD software systems into the grid/cloud infrastructure as it is done in [1-4]. To get this it is necessary:

▪ *to investigate* the generalized engineering design process and to select its loosely coupled stages and procedures for subsequent their transferring to the forms of standardized web-services;

▪ *to analyze* the existing mathematical modeling and optimal design software for the possible re-use of the best algorithms and design procedures implementations in the creating the depository of applied web services;

▪ *to develop* a container with interfaces for standardized individual web-services based on international standards and protocols which allow building compositions from these web-services as design (calculations) workflows.

▪ *to implement* novel service-oriented design paradigm in Engineering according to which all levels of design including components, circuit and system levels are divided into separate loosely coupled stages and procedures for their subsequent transfer to the form of standardized web-services.

2). Extending of ***service** management and monitoring facilities* in a cloud computing environment by making these services to be more centralized and allowing them to use interconnected multiple distributed services databases. To realize this opportunity, it is necessary to incorporate in a cloud the service-based information similar to the type of information captured in UDDI directory services and provide cross-cloud connectivity to facilitate the ability to openly discover the services residing within distributed databases. Standard cloud APIs will let service providers deploy their services seamlessly to multiple clouds computing providers and cloud computing providers should add features to their cloud infrastructures to enable management and monitoring for deployed services.

3). Using of *service metadata* for service. Inference of machine-interpretable information about what the service can do and what it can provide remains an open issue. Syntactic interpretation of service-based information lacks the confidence to perform this function well because the meaning of underlying information is missing. *Semantic approaches* that allow meaningful definitions of information in cloud environments offer solutions for many service providers who may reside within the same infrastructure by agreement on linked ontology. Third-party software agents operating within a cloud might be able to derive ontological information from the stored data and operations. Service-oriented and cloud computing combined will indeed begin to challenge the way of enterprise computing development. Thanks to ontology it becomes possible to create service-oriented applications even by orchestrating legacy applications that do not support the Web Services specifications.

4). Performing semantic approach with help of novel *RESTful Web services* which are alternative to SOAP- and Web Services Description Language (WSDL)-based Web services. REST (*Representational State Transfer)* defines a set of architectural principles by which Web services can be designed with focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages. A concrete implementation of a REST Web service follows four basic design principles: (a) Use HTTP methods explicitly; (b) Be stateless; (c) Expose directory structure-like URIs (Uniform Resource Identifiers); (d) Transfer XML, JavaScript Object Notation (JSON), or both. Use only the standard HTTP messages -- GET,PUT, POST and DELETE -- to provide the full capabilities of the application Exposing a system's resources through a RESTful API is a flexible way to provide different kinds of applications with data formatted in a standard way. It helps to meet integration requirements that are critical to building systems where data can be easily combined (*mashups*) and to extend or build on a set of base, RESTful services into something much bigger. REST supports intermediaries (proxies and gateways) as data transformation and caching components and enables transfer of data in streams of unlimited size and type.

It is planned to analyze recent trends in field of web-services and its semantic annotations, to compare and integrate the procedure-oriented and resource-oriented services taking account their advantages and constraints and using *LinkedData technology* [5] for combining Web services, RESTful services and Semantic Web-Services on the base of known SPARQL, RDF and other standards.

5). Re-engineering the existing *service workflow tools* (*Taverna, Kepler* or *Askalon*) for cases of orchestrating web-services of different types, including RESTful services, Semantic web-services and traditional WS* services by using orchestration capabilities of standardized WS-BPEL engines, LinkedDataServices (LIDS) and WSMX (the prototype of Semantic Web-Services workflow).

6). Demonstrating the effectiveness of the Service-oriented computing (SOC) in a cloud computing environment by developing *Engineering Design Platform***,** in particular, for modeling and optimization of Nonlinear Dynamic Systems, based on components of different physical nature and being widely spread in different scientific and engineering fields. It is the cross-disciplinary application for distributed computing in the form of a network of collaborative components functioning within or across organization borders. it seems to be very useful for people who have needs to use applications composed by SOC, as well as for the people who can design sophisticated applications using services.

6. Going research

For building a prototype of the *Engineering Design Platform* based on SOC we are looking partners for submitting *Horison-2020* project which are agree to participate in :

▪ developing a *distributed web-services repository* which provides the access to autonomous, platform-independent Design procedures of CAE / CAD tools, say, for *MEMS* design (operations with large-scale mathematical models, steady state analysis, transient and frequency domain analysis, sensitivity and statistical analysis, parametric optimization and optimal tolerances assignment, solution centering, etc.) and supporting procedures (cross-domain mathematical model description translation, data formats translation etc.) based on innovative original numerical methods; Algorithms proposed for many design web-services are novel and unique (multi-criterion optimization, optimal tolerances assignment, yield maximization, stiff- and ill-conditional tasks solving, etc.).

▪ providing possibilities for different research teams to contribute in web-services repository development using different programming languages and planning to implement different data from distributed sources. Due to loosely coupled web-services feathers users can modify and adapt a composed application which is preserved when some web-services are changed. Design in Engineering becomes personalized and customized because users can build and adjust their design scenario and workflow by selecting the necessary web-services (as calculation procedures) to be executed on grid/cloud resources. A user can also introduce new component models and their parameters, which is absent in any existing SPICE-like simulation software.

▪ creating a *service workflow tools* for composition and orchestration of heterogeneous web-services into a user defined computing scenario or a Design route, which comprises a set of ontologies, domain-specific heuristics, and a knowledge base to support the semi-automatic workflow composition. In particular, the ontology will cover various aspects of Engineering Design and the composition will be based knowledge advanced matchmaking algorithms based on the assumption of concept types, the properties of inputs, outputs, and data. The workflow composition tool will adapt and extend existing knowledge base solutions. The IASA will demonstrate how to semi-automatically create Engineering Design workflows using a knowledge-based approach and how to improve their composition by elaborating different workflow versions to increase the potential for optimising non-functional parameters (different workflow versions may expose different potential for improving execution times, energy consumption, or computing costs).

▪ transferring *Engineering Design Platform* in cloud environment and execute it there on computing resources being selected by a broker of cloud infrastructure middleware.

7. Networked Optimal Mircosystems Designer

The main idea of SOC was proved experimentally by development of the multi-layered architecture of the grid-enabled computer simulation software (fig.1)[6] This architecture is characterized by the following:
▪ functionality is distributed across the ecosystem of both web services and grid services (enabling utilization of grid computing resources)
▪ it is compatible with adopted standards and protocols
▪ it supports custom user analysis scenario development and execution
functionality is accessible with lightweight web interface
▪ it hides the complexity of web-service interaction from user with abstract workflow concept and simple graphical workflow editor (fig.2).

This toolkit is named by WebALLTED (Web ALL Technology Designer) and it is devoted for schematic design of complicated technical systems (including Nonlinear Dynamic Systems) composed of either/and electronic, hydraulic, pneumatic, mechanical, electromagnetic, and other subsystems.

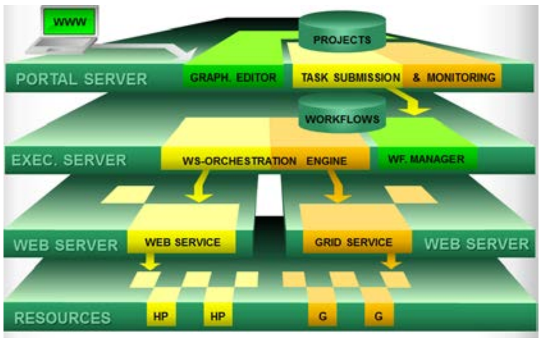

Fig.1. WebALLRD general architecture

WebALLTED is based on the original numerical algorithms for all the stages of design [7]: starting from steady state, frequency and transient analyses till parametrical optimization of a designed device output characteristics, optimal component tolerances assignment, centering of solution, and Yield maximization.

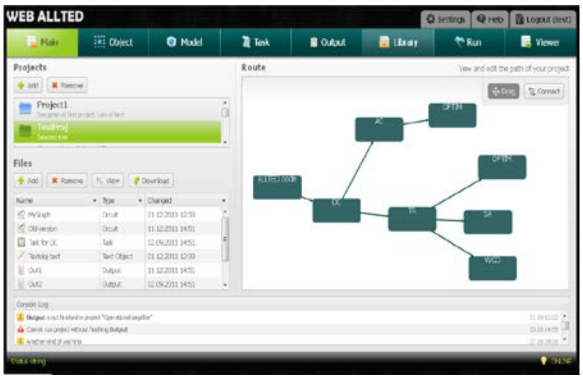

Fig.2. User Interface Workflow Editor

WebALLTED can be used in distributed Grid environment or it can be embedded in every Intranet domain with client-server base configuration. The typical sequence of design procedures is shown at fig.3.

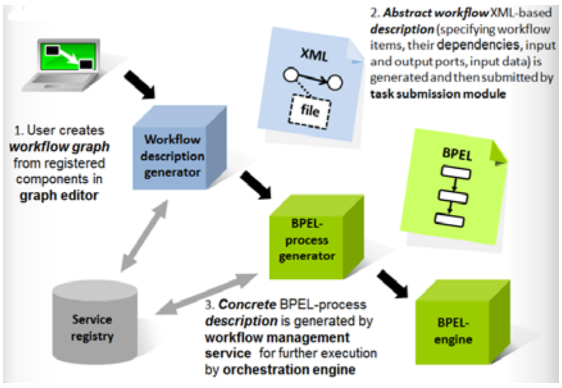
1. User creates workflow graph from registered components in graph editor
Workflow description generator
2. Abstract workflow XML-based description (specifying workflow items, their dependencies, input and output ports, input data) is generated and then submitted by task submission module
XML
file
BPEL
BPEL-process generator
Service registry
3. Concrete BPEL-process description is generated by workflow management service for further execution by orchestration engine
BPEL-engine

Fig.3. WebALLTED Task Description Processing

Advantages of above methods in comparison with numerical methods used in SPICE-like simulators are illustrated by the example of simulation of benchmark circuits set being proposed by North Carolina Microelectronics Centre (tabl. 1).

There are visible false oscillations at the plot (Fig.4), when simulating the circuit Make2 with default conditions in HSPICE due to used numerical integration methods (2-nd order in HSPICE and variable order (till 6-st) in WEbALLTED).

The toolkit in hand was used for selection of optimal ratio W/L (width to lenth) for CMOS in DEC Alpha AXP 64-bit microprocessor resulting in twice increasing its run frequency. The toolkit was used also for General Electric Ultrasonic Transducer simulation.

Table 1. Comparative simulation results of WebALLTED and HSPICE

Circuit	***DC*** Iteration number, ***ALLTED***	***DC*** Iteration number, ***HSPICE***	***TR*** Iteration number, ***ALLTED***	***TR*** Iteration number, ***HSPICE***
Schmitt trigger	88	67	146	537
Bjtinv	95	96	1340	3239
Gm3	80	185	149	219
Make2	12	10	527 but only 256 steps	327 but outputs are distorted

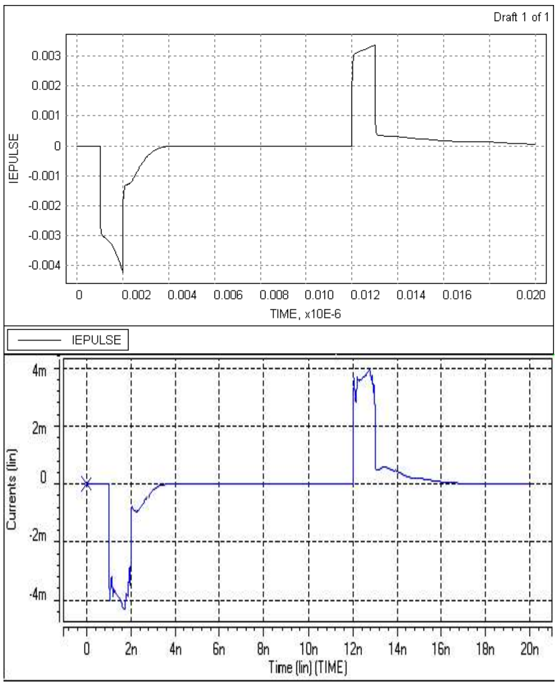

Fig.3. The simulation results obtained by ALLTED (a) and by HSPISE (b)

In comparison with SPICE-like programs WebALLTED offers:

■ Faster simulation speed and improved numerical convergence;
■ Sensitivity analysis for frequency and transient analyses;
■ Comprehensive optimization procedure and optimal tolerances assignment;
■ Alternative approach to the secondary response parameters determination (delays, rise and fall times, etc.);
■ Powerful user-defined modeling capability,
■ Original way of generating a system-level model of MEMS from FEM component equations (being received, for example, by means of ANSYS) when these equations with boundary conditions are

transformed into the equivalent equations of a schematic model, which consists of L, C and G components, and then are simplified by means of Y-Δ transformation.

The reduced MEMS model has a circuit character unlike the existing approaches (for example, in CoventorWare), where it is represented by the reduced system of differential equations, that allows using directly the input interface of WebALLTED domestic circuit simulation package and making use of its unique power optimization and tolerance procedures.

The parallel algorithms of numeral integration of the promoted reliability and exactness for dynamic analysis tasks and their implementation into the NetALLTED were developed and they run now on the university cluster with 5.83 Tflops productivity .

8. Conclusion and future work

Solution in hand is designed primarily to meet the needs of small and medium enterprises in the modern toolkit design of complex technical objects and technological processes, as well as the small research laboratories to perform complex computational experiments. A long-term strategy for the Engineering Design is to create flexible networked simulation and modelling tools for “bottom-up” or “top-down/ bottom-up”. This original conception of Engineering SOC with Design procedures as web-services has no complete competitor worldwide.

Achieved results *can be shifted to other subject areas* if necessary additional web-services are developed together with applied engineers with the possibility of their replenishment and editing.

References

[1] TINACloud project home: http://www.tina.com /English/ tina/

[2] PartSim project home: http://www.partsim.com/ examples

[3] RT-LAB project home: http://www.opal-rt.com/company/company-profile/

[4] FineSim Pro project home: http://www.automation.com/ content/magmas-latest-version-of-finesim-pro-delivers-3x-faster-runtime/

[4] FineSim Pro project home: http://www.automation.com/ content/magmas-latest-version-of-finesim-pro-delivers-3x-faster-runtime/

[5] LinkedData technology project home: http://linkeddatabook.com/editions/1.0/

[6] Petrenko A.I., V.Ladogubets, O. Finogenov, B. Bulakh. WebALLTED: Interdisciplinary Simulator Based on Grid Services .-// Proc. of East-West Design and Test Conference (EWDT-12), Knarkiv, 15-18 Sept.2012

[7] Petrenko A.,Ladogubets V., Tchkalov V., Pudlowski Z. ”ALLTED- a Computer-Aided System for Electronic Circuit Design”, UICEE.(UNESCO), Melbourne, 1997.-204 p.