Comput. Sci. Appl.
Volume 1, Number 6, 2014, pp. 349-358
Received: May 6, 2014; Published: December 25, 2014

Computer Science
and Applications
www.ethanpublishing.com

Service-Oriented Computing in a Cloud Computing Environment

Anatoly I. Petrenko

Department of System Design, Institute of Applied Sysmem Analysis, National Technical University of Ukraine "Kyiv Polytechnic Institute", Kiev 03056, Ukraine

Corresponding author: Anatoly I. Petrenko (tolja.petrenko@gmail.com)

Abstract: SOC (service-oriented computing) represents a new generation of distributed computing framework for applications development 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 an advanced distributed computing infrastructure for both intra- and cross-enterprise application integration and collaboration. The distinction between SOC and traditional computing is that application builders no longer construct software using a programming language. Instead, they specify the application logic in a high-level specification language, utilizing standard services as components. Services implement 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. The delivery of software as a set of distributed services 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. So any user community, regardless of its discipline, should be supplied with the technological approach to build their own distributed compute-intensive multidisciplinary applications rapidly. This paper contains the roadmap of development of the Engineering Design Platform, based on SOC and intended, in particular, for modeling and optimization of Nonlinear Dynamic Microsystems, being consisted of components of different physical nature and being widely spread in different scientific and engineering fields.

Key words: SOC (service-oriented computing), SOA (service-oriented architecture), web-services, services composition, cloud computing, engineering design platform, EDA for cloud.

1. Introduction

The SOC (service-oriented computing) paradigm refers to the set of concepts, principles, and methods that represent computing in SOA (service-oriented architecture) in which software applications are constructed based on independent component services with standard interfaces [1-9]. SOC represents a new approach in application development moving away from tightly coupled monolithic software towards software of loosely coupled, dynamically bound services. 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 [7-10]. So, SOC 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 [10-13].

The rest of this paper is organized as follows: Section 2 gives an overview of web-services as the best component of SOC; Section 3 describes Cloud computing advantages; Section 4 presents SOC in Engineering Design; Section 5 describes Going research in SOC for Engineering; Section 6 presents the performance of applied services in the SOC prototype; Section 7 concludes this paper.

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 WSDL (Web Services Definition Language). The UDDI (Universal Description, Discovery, and Integration) 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 BPEL4WS (Business Process Execution Language for Web Services) 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 do not offer user-customized management and monitoring mechanisms built into their infrastructure. Hence, it is 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 Refs. [13-16]. 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 WSDL (Web Services Description Language)-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 [17] 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, LIDS (LinkedDataServices) and WSMX (the prototype of Semantic Web-Services workflow).

(6) Demonstrating the effectiveness of the service-oriented computing in a cloud computing environment by developing Engineering Design Platform, in particular, for modeling and optimization of Nonlinear Dynamic Microsystems with components of different physical nature, 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.

5. 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.

Porting engineering applications to the grid and cloud platform can be based on the paradigm of SOC (service-oriented computing) which utilizes services as the basic constructs to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments. It requires updating the web-services Repository, which consists of the Platform Supporting Services and the Application support services. The Platform Supporting Services offer the standard operations for service management and hosting (e.g., cloud hosting, event processing and management, mediation and data services, services composition and workflow, security, connectivity, messaging, storage). The Application support services are created by investigating the generalized engineering design process and selecting its loosely coupled stages and procedures for subsequent their transferring to the forms of standardized web-services. It is possible also 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 repository of Application support services.

6. 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) [19].

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 GridALLTED (Grid ALL Technology Designer) and it is devoted for schematic

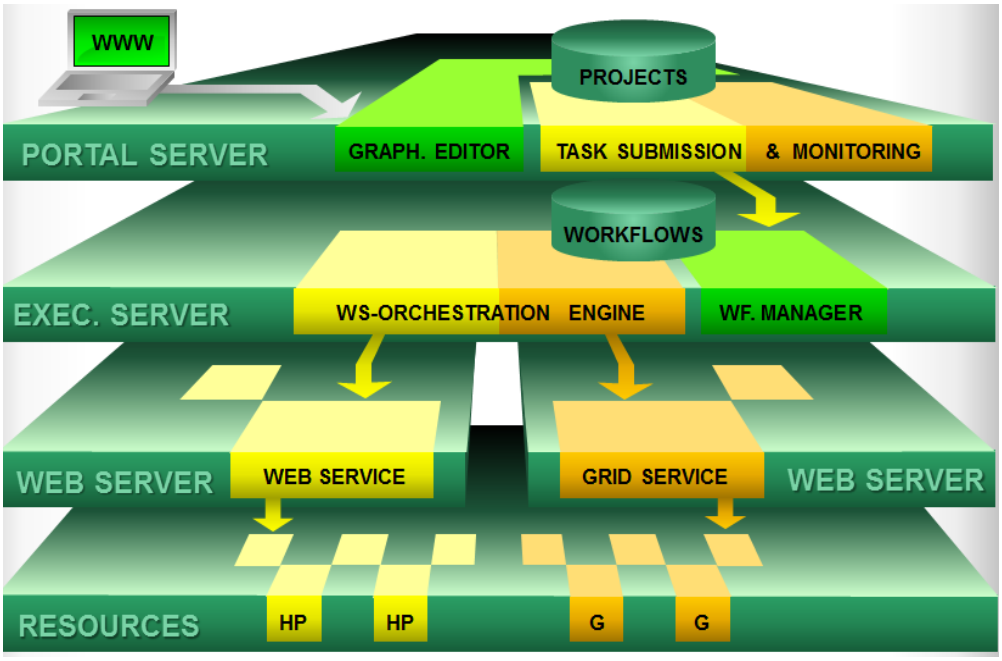


Fig. 1 GridALLED general architecture.

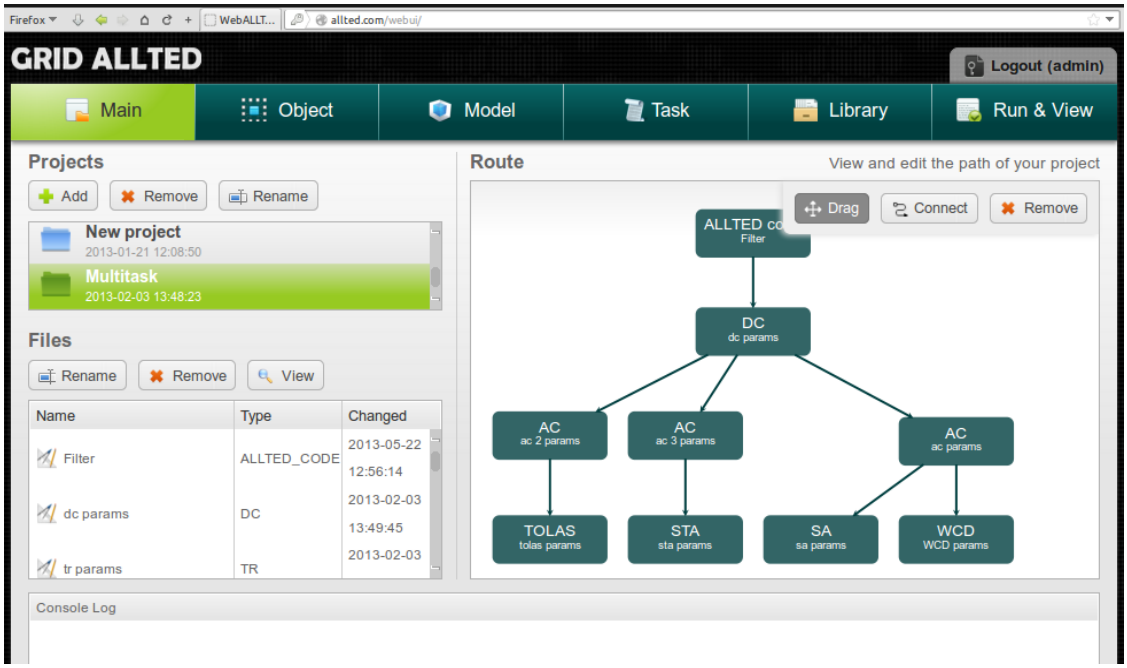


Fig. 2 User interface workflow editor.

design of complicated technical systems (including Nonlinear Dynamic Systems) composed of either/and electronic, hydraulic, pneumatic, mechanical, electromagnetic, and other subsystems.

In our project a set of web and grid services, been provided by SOC tools, will offer with the following functionality for Engineering Design:

- Automatic forming of mathematical model of an object (or a process) from description of its structure and component properties as algebra-differential or differential equations resulted to the format which other subsystems of the workflow can work with;
- Reducing the dimension of the formed mathematical model of an object (or a process) by transformation of structure of object (a triangle to a star) and developing the macromodel of an object;
- Introducing new models for objects by customers, changing their parameter values and statistics;
- DC analysis of an object (or a process) based on automatically formed model by the use of different methods: Newton-Raphson, continuation of solving with a changeable parameter, the search for curve of decision;
- Application of the diagonal modification method for solving ill-conditional systems of linear equations, which excludes necessity of equations reordering in the cases of zero pilot elements of matrix;
- Frequency analysis of an object (or a process) based on automatically formed model by solving the linear systems of equations with complex coefficients and automatic determination of corresponding design parameters (frequency band, resonance frequencies and values and similar);
- Dynamic analysis of an object (or a process) in time domain based on automatically formed model by the use of implicit methods of variable order (1-6) and variable step as well as automatic determination of corresponding design parameters (delay time, rise and fall times, etc.);
- Analysis of sensitivities of design parameters of an object (or a process) based on automatically formed model in time or frequency domains to changes of parameters of internal components;
- Parametrical optimization of p characteristics of an object (or a process) based on its model in time or frequency domains by using the newest method of variable order (1st-4th), which covers the gradient methods of 1st order and the quasi- Newton methods of variable metric of 2nd order as particular cases;

• Statistical analysis of parameters and characteristics of an object (or a process) based on automatically formed model in time or frequency domains by the Monte-Carlo method with possibility to optimize the coefficient of output (yields);

• Visualization of calculation results in a graphical form.

GridALLTED is based on the original numerical algorithms for all the stages of design [20]: 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.

GridALLTED 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 in Fig. 3.

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 (Table 1) [21].

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).

In comparison with SPICE-like programs GridALLTED 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 [22];

• Dynamically configurable software architecture due to composited services and executing in grid nodes.

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

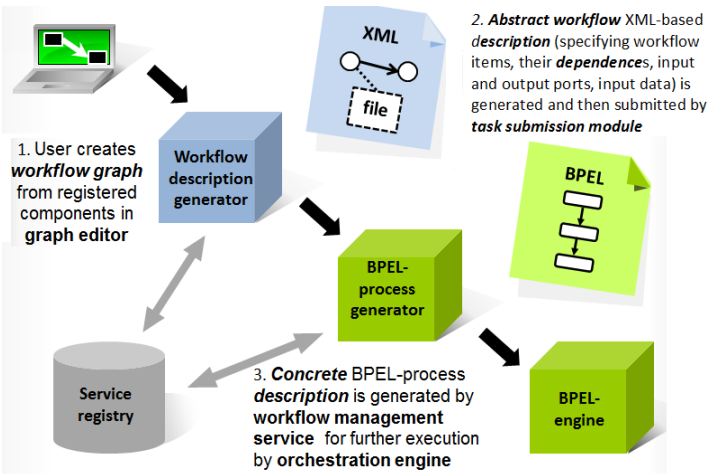
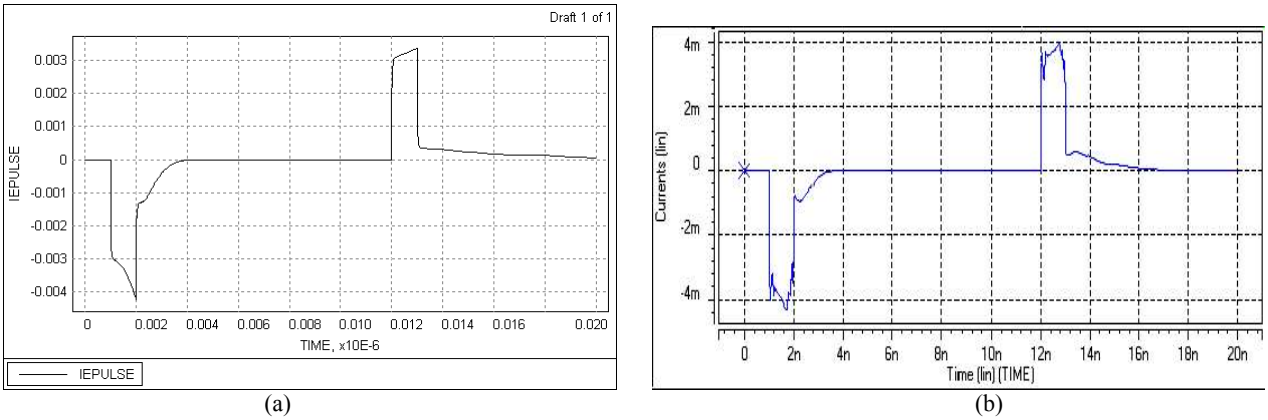


Fig. 3 GridALLTED task description processing.

Table 1 Comparative simulation results of GridALLTED and HSPICE.

Circuit	Schmitt trigger	Bjtinv	Gm3	Make2
DC Iteration number GridALLTED/HSPICE	88/67	95/96	80/185	12/10
TR Iteration number GridALLTED/HSPICE	146/537	1340/3239	149/219	527 (for 256 steps)/327 (outputs are distorted)



(a) (b)

Fig. 4 The simulation results obtained by (a) GridALLTED and (b) HSPISE.

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 GridALLTED were developed and they run now on the university cluster with 5.83 Tflops productivity. At present time GridALLTED is shifted to Cloud resources.

7. Conclusions

The analysis of a current state of Engineering simulation and design software proves the urgent need of these systems re-engineering to enable their operation in distributed computing environments. It requires reorganizing such systems in the form of a set of separate interacting modules or services. In that case, the designer's work is to compose a scenario for service interacting (a computational experiment route).

Service-oriented application is a software system which involves the architecture and distributed networks of interoperating services. The interoperability of these services provides developers with flexibility and allows them to easily integrate one service with another via well-defined interfaces.

Service-oriented application introduces the idea of giving opportunity to users to be the providers of the content and composed services where users could generate the content and make decisions. It also introduced mash-ups, which is a term that refers to web applications consisting of some independent services working together to form a large service or van application.

The prototype of the service-oriented Engineering Design Platform was developed on a base of the proposed architecture for EDA (Electronic Design Automation) domain which migration to cloud computing still remains slow. Beside EDA the simulation, analysis and design can be done using GridALLTED for different control systems and dynamic systems composed of electronic, hydraulic, pneumatic, mechanical, electrical, electromagnetic, and other physical phenomena elements. Complex non-linear systems of this type are widely used in modern aerospace, robotics, NC machine tools and test equipment, highway engineering, agricultural and other applications. Of course, if provided set of web-services will not be fully comprehensive, the interested users will be able to develop and add additional own web services as well as incorporate existing web services into their own complex applications and the Repository of services.

Service-oriented applications governance involves knowledge about services, providers and users. Development is divided between a service provider and an application builder. This separation enables application builders to focus on business logic of his application while leaving the technical details to service providers. If a large multidisciplinary and multinational Repository of Application support services is created, the end-users can tailor the services to their own personal requirements and expectations by incorporating functionalities of available services into large-scale Internet-based distributed application software.

Resulting in an application design 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 and users themselves are transferred from software users to software developers.

References

[1] M.N. Huhns, M.P. Singh, Service-oriented computing: Key concepts and principles, IEEE Internet Computing 9 (1) (2005) 75-81.

[2] Y. Chen, W.-T. Tsai, Distributed Service-Oriented Software Development, Kendall Hunt Publishing, 2008, p. 467.

[3] Y. Chen, W.-T. Tsai, Service-Oriented Computing and Web Software Integration, 4th ed., Kendall Hunt Publishing, 2014, p. 400.

[4] Y. Wei, M.B. Blake, Service-oriented computing and cloud computing: Challenges and opportunities, IEEE Internet Computing 14 (6) (2010) 72-75.

[5] M.P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, Service-oriented computing: A research roadmap, International Journal of Cooperative Information Systems 17 (2) (2008) 223-255.

[6] M.P. Papazoglou, W.-J. van den Heuvel, Service-oriented design and development methodology, Int. J. of Web Engineering and Technology 2 (2006) 412-442.

[7] G. Rains, Cloud computing and SOA, MITRE, White Paper, Oct. 2009 [Online], http://www.mitre.org/work/tech_papers/tech_papers_09/ 09_074309_0743.pdf/.

[8] A.I. Petrenko, Service-oriented computing (SOC) in a cloud computing environment, http://www.krput.edu.pl/files/swrutu/prof_petrenko/EWTD-13.pdf.

[9] W.T. Tsai, X. Sun, Y. Chen, Q. Huang, G. Bitter, M. White, Teaching service-oriented computing and STEM topics via robotic games, in: Proc. of IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC 2008), pp. 131-137.

[10] Program of the 12th International Conference on Service Oriented Computing, Paris, Nov. 3-6, 2014, http://events.telecom-sudparis.eu/icsoc/Program/ICSOC2014-full-program.pdf.

[11] M.B. Blake, Decomposing composition: Service-oriented software engineers, IEEE Software 24 (6) (2007) 68-77.

[12] Y. Chen, W.-T. Tsai, Service-orientation in computing curriculum, in: Proc. of IEEE 6th International Symposium on Service Oriented System Engineering (SOSE 2011), pp.122-132.

[13] S. Chari, Smart IBM solutions for high performance engineering clouds: Advanced performance for complex engineering problems delivered in a flexible and secure mode through private and private-hosted clouds, 2011, http://www.cabotpartners.com/Downloads/HPC_Cloud_Engineering_June_2011.pdf.

[14] TINACloud Project, http://www.tina.com /English/ tina/.

[15] PartSim Project, http://www.partsim.com/ examples.

[16] RT-LAB Project, http://www.opal-rt.com/company/company-profile/.

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

[18] LinkedData Technology Project, http://linkeddatabook.com/editions/1.0/.

[19] M. Zgurovsky, A. Petrenko, V. Ladogubets, O. Finogenov, B. Bulakh, WebALLTED: Interdisciplinary simulation in grid and cloud, Computer Science 14 (2) (2013) 295-306.

[20] A. Petrenko, V. Ladogubets, V. Tchkalov, Z. Pudlowski, ALLTED: A Computer-Aided System for Electronic Circuit Design, UICEE, UNESCO, Melbourne, 1997, p. 204.

[21] CircuitSim90 Benchmark Standards, http://www.intusoft.com/benchmarks.htm.

[22] A. Petrenko, Macromodels of micro-electro-mechanical systems (MEMS), in: N. Islam (Ed.), Microelectro-Mechanical Systems and Devices, pp. 155-190 [Online], http://www.intechopen.com/books/microelectromechanical-systems-and-devices).