УДК 004.415.2

I. Sydorov

DOMAIN ANALYSIS METHOD AS A SERVICE OF CLOUD-ENABELED SERVICE ORIENTED PLATFORM

The problems of building PaaS and SaaS applications are covered. Reference architecture is offered for building cloud-enabled platform to host a number of SaaS applications and support general characteristics of a cloud such as rapid elasticity, on-demand self-service, multi-tenancy, resource pooling and measured service. General directions of adopting cloud computing in aviation industry are mentioned. Domain analysis as a service of cloud-enabled service oriented platform is presented. The method is realized with the help of two controllers. The task - to create software patten for object communication device of the legacy aircraft simulator is decided. Formalization of processes description is carried out by means of Real Time Process Algebra (RTPA). Description of processes in RTPA is input for the mean that creates the service (domain analysis controller) for cloud-enable service oriented platform. Domain analysis service can be use not only for reengineering aviation simulator software.

Keywords: software as a service, cloud-enabled platform, domain analysis, RTPA, aircraft simulator.

During recent years the term "cloud" just exploded the internet and became a very popular, not to say fashionable, trend in IT industry. Most influential vendors like Microsoft, Google and Amazon presented their platforms for building cloud application. In this article discussed and proposed a pattern for building cloud enabled applications built on concepts of Service Oriented Architecture.

The term "cloud" is used as a metaphor for the Internet, based on the cloud drawing used in the past to represent the telephone network, and later to depict the Internet in computer network diagrams as an abstraction of the underlying infrastructure it represents. Today cloud is widely used to represent an abstraction of technology, resources and its location used build integrated computer infrastructure, including networks, systems and applications. In 80s and 90s during the pre-web enterprise era, applications where very monolithic. Companies had a number of very complex and self-reliable systems; any interfacing between those was basically done via batch dumps.

Seems that web changed it all, but not really. Yes most enterprises are now using web based application, but still they are inherently monolithic and very tricky to change, extend and integrate with each other. The answer to that challenge is Service Oriented Architecture, which is about building an application based on set of services that can be easily extended, reorganized to meet ever-changing business requirements. As we know the maintenance stage of a software development lifecycle takes up to 60 % of effort and SOA that provides atomic enterprise building blocks can significantly reduce future cost on software adaptation.

So as for Cloud applications, most of them are fundamentally built with the same "monolithic" architectural principles with a little concern of Enterprise Architectural patterns.

One of the key characteristics for cloud application is the ability to manage resources efficiently. Such efficiency should be translated into natural elasticity of the resources, where in a given moment you will need to scale up or down, based on unplanned peak of work, but minimizing investments in infrastructure.

Traditional capacity planning assumes that customer has a certain infrastructure capacity. During most of the day the service may remain sub-utilized, wasting capacity because demand is less than requirements. In any given moment capacity could be exceeded, ending up with frustration for end users because of slowness and application errors. Typically to solve this, more resources are added to the data center, which means more cost and waste of valuable resources.

In a cloud deployment scenario a minimum resources are provisioned to the customer. As the demand increases additional on-demand resources are added and as soon as the need in resources decreases the data center capacity decreased.

Cloud deployment in turn can be classified as [1, p. 75]: private – deployed in a customer's own datacenter; public – deployed on a public cloud provider; hybrid – is a combination of both public and private with a secure bridge between them.

No matter the type of deployment the company decides on for the future, the cloud services platform should enforce the core characteristics of cloud architecture. Those characteristics are defined by the National Institute of Standards and Technology:

− on-demand self-service: A consumer can unilaterally provision computing capabilities such as server time, network storage, as needed automatically without requiring human interaction with a service provider;

− broad network access: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick clients such as workstations, laptops, slates, mobile phones etc. as well as other traditional or cloud-based software services;

− resource pooling: The provider's computing resources are pooled to serve multiple consumers using multi-tenant model. Different physical and virtual resources are dynamically assigned and reassigned according to demand. There is a certain degree of location independence so customer is often has no control over exact location of resources but may have an ability to specify location at a higher level of abstraction, for example county, region or datacenter;

− rapid elasticity: capabilities can be rapidly and elastically provisioned (in some cases automatically) to quickly scale up and shrink down;

− measured service: Cloud systems automatically control and optimize resources usage by leveraging a metering capability at some level of abstraction appropriate to the type of a service (storage, processing, bandwidth, or active users). Resources usage can be monitored, controlled and reported, providing transparency for both provider and consumer;

− multi-tenancy: Many organizations (tenants) can use cloud with mechanisms to protect and isolate each tenant from all others securing company's sensitive information.

Although this seems to be straightforward, this requires a different application architecture paradigm that can leverage these characteristics. Proposed architecture option is based on Event-Driven architecture framework, as well as use of Message-Oriented middleware to interconnect processes and ensure:

− extreme loose coupling and well distributed workload;
− horizontal scalability;
− shared-nothing processes.

The component of the architecture that enables this capability is named Controller and has the following characteristics (Fig. 1):

− is able to post and receive notifications;
− is able to receive remote configuration parameters;
− has embedded security components;
− is able to notify every event or transaction for auditing purposes;
− is able to run multiple processes at the same time, without conflicting with the management of the requirements (self-balanced);
− is able to support multi-tenancy;

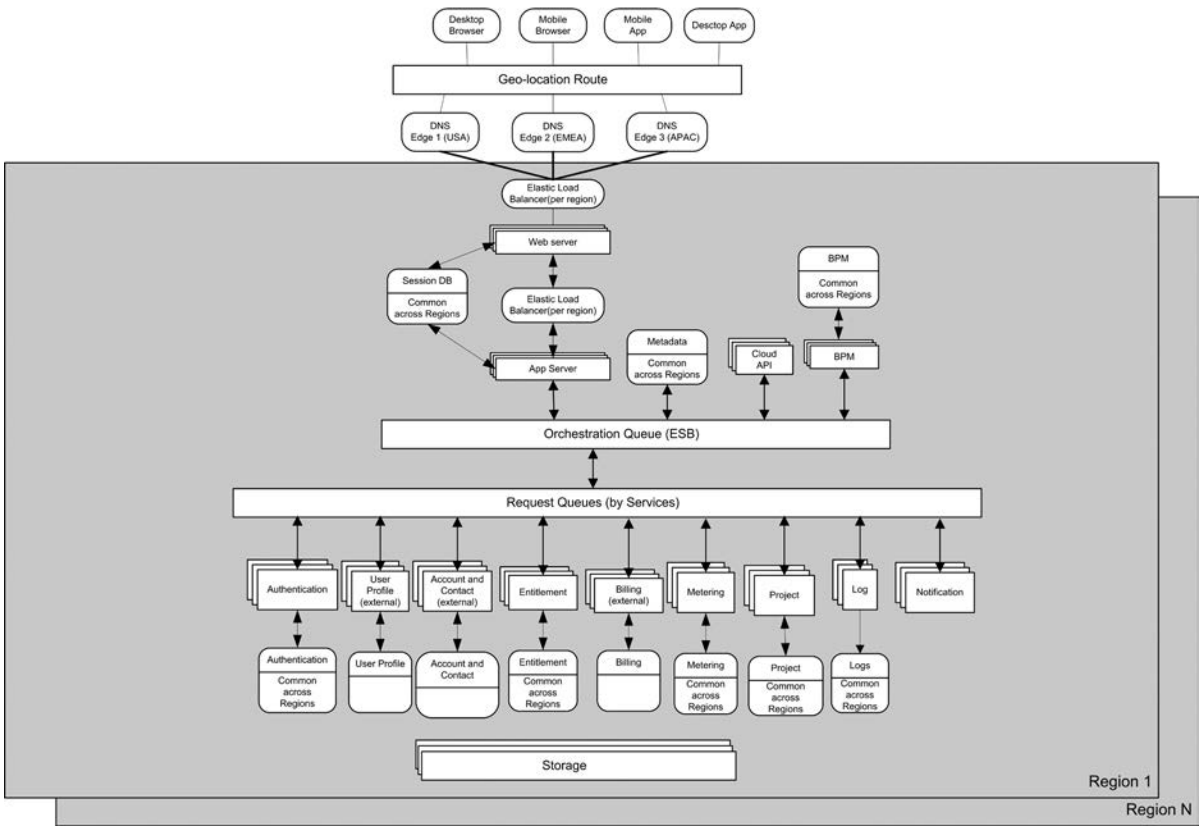

Fig. 1. Platform architecture

– if needed is able to interact with external applications in their native form or based on the provided integration mechanisms.

So the platform consists from small atomic business components – controllers. Each controller is responsible for a subset of business operations (e.g. user management, logging, billing, etc.). Interaction between controllers is done via message bus. Message bus implementation can be different according particular project needs. It can be privately deployed Apache ActiveMQ or public cloud services can be used such as Amazon Simple Queue Service or Windows Azure Queue Storage. An orchestration layer makes it possible to apply custom business rules to enrich application capabilities. Application and web servers hidden behind elastic load balancers provide access to the system by heterogeneous clients such as web browsers, mobile phones, thick and thin clients. Unified message format for the bus and abstracted API for interaction with the bus makes it possible to develop controllers in a various programming languages (such as C#, Java, C++, NodeJs) host them on various platforms (Windows, Linux). This flexibility is also very beneficial in integration scenarios so the best approach can be chosen for a particular system.

Aviation industry is slow to change, very risk averse. Software solutions are complex and not easily tackled [2, p. 68]. Therefore some companies started adoption of a cloud computing for the need of aviation industry. Xerox is launching its cloud platform to help airlines to seamlessly share important data and key transmissions from airline to other carriers and flight authorities. In areas where data sharing, instant and effective distribution is important cloud platforms and applications can be very beneficial. Seamless integration between systems, B2B interaction, scalability and fault tolerance can help cloud computing to make its way into aviation industry.

Domain analysis method as a service

Domain analysis is the effective method when legacy software is rebuilded [3, p. 239]. We have case study domain analysis method for legacy software of aviation simulator on the base of cloud-enable service oriented platform [4.1.10.36]. The domain analysis method is realized with the help of two controllers (Fig. 2).

The first controller by name builder is predefined. The second controller by name domain analiser is

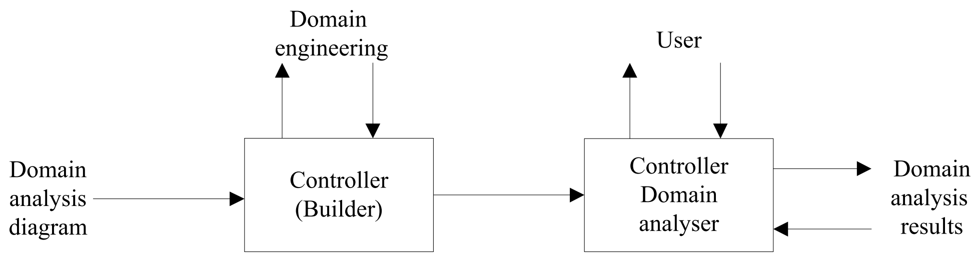

Fig. 2. Using controllers

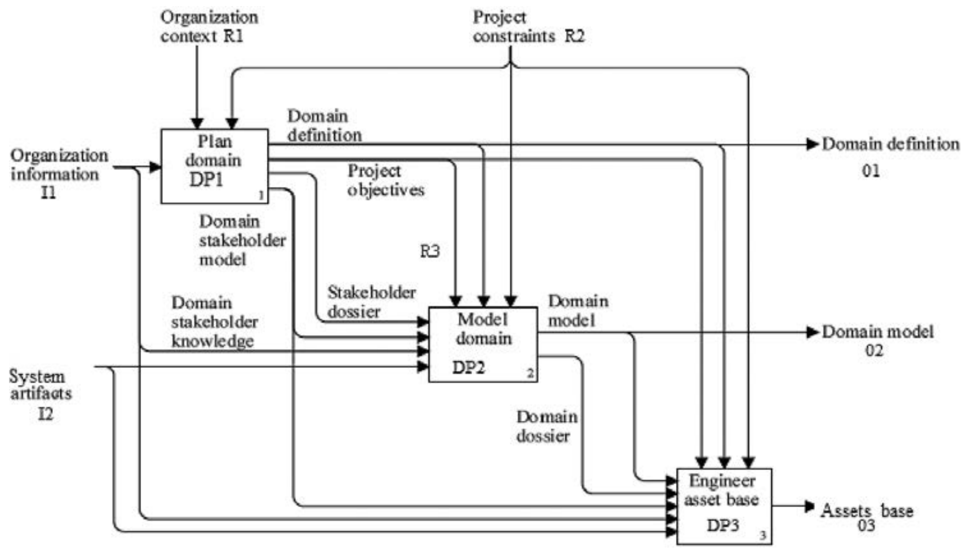

Fig. 3. Domain analysis detailed diagram

built. The first controller builds domain analysis controller by domain analysis diagram. The user utilizes the second controller for domain analysis.

There is the following task: create software pattern for object communication device of the legacy aircraft simulator.

The example of domain analysis for builder consists of three processes (DP1, DP2, DP3). Diagram is presented on Fig. 3.

Inputs for this processes are the following: organization information is the trainer center information (I1); system artifacts are the simulator components definition, communication device information (I2). Outputs are the following: domain definition ontology (O1); domain model is communication device architecture (O2); assets base consists of reusable components (O3). Control constraints policies are the following: organization context – R1; project constraints – R2.

Formalization of processes description is carried out by means of Real Time Process Algebra (RTPA) is the following:

Static description

(Process Schema DOMAIN ENGENEERING = PN 0
{Process ID: DOMAIN ENGENEERING ({I:I1,I2};
{O:O1, O2,O3}; {R: R1,R2,R3})} {Detailed Processes:
DP1, DP2, DP3}
(Process Schema PLAN DOMAIN = PN 1
|| {ProcessID: PLAN DOMAIN ({I: Organization information};{O: Domain definition, Project objectives, Stakeholder dossier, Domain stakeholder model}; {R: Organization context, Project constrains})}
|| {DetailedProcesses: DP1, DP2, DP3, DP4}
)
(Process Schema MODEL DOMAIN = PN 2
|| {ProcessID: MODEL DOMAIN ({I: Stakeholder dossier, Domain stakeholder model, Domain stakeholder knowledge, Exemplar system artifacts};{O: Domain model, Domain dossier};{R:
Domain definition, Project constraints, Project objectives})}
|| {DetailedProcesses: DP1, DP2, DP3}
)
(Process Schema ENGINEER ASSET BASE = PN 3
|| {ProcessID: ENGINEER ASSET BASE ({I: Domain model, Domain dossier, Domain stakeholder model, Organization information, Exemplar system artifacts};{O: Asset base};{R:
Domain definition, Project constraints, Project objectives})}
|| {DetailedProcesses: DP1, DP2, DP3 }
)
(Process DOMAIN ENGINEERING Relations = PN1:O1→PN2:R1; PN1:O2→PN2:R3;
PN1:O3→PN2:I1; PN1:O3→PN2:I2; PN1:O1→PN3:R1; PN1:O2→PN3:R3; PN2:O1→PN3:I1;
PN2:O2P→N3:I2
)

Dynamic description

Process Dispatch = § →
(@Event1 DOMAIN ENGINEERING → {PN1:DP1,DP2,DP3}
| @Event2 DOMAIN ENGINEERING → {PN2:DP1,DP2,DP3}
| @Event3 DOMAIN ENGINEERING →{PN3:DP1,DP2,DP3}
)

Conclusion

Description of processes in RTPA is input for the mean that creates the service (domain analysis controller) for cloud-enable service oriented platform. Domain analysis service can be use not only for reengineering aviation simulator software.

References

1. Сидоров Н. А. Метод построения средств доменного анализа на основе формальных спецификаций в RTP A. / Н. А.Сидоров, Н. Б. Мендзебровский, Ю. Н. Рябоконь // Вісник східноукраїнського нац. ун-ту ім. В. Даля. – 2012. – № 12. – С. 75–80.
2. Реинженерия наследуемого программного обеспечения информационно-моделирующих тренажеров / Н. А. Сидоров, В. Г. Недоведеев, И. П. Сердюк, В. А. Хоменко, Е. Н. Сидоров // УСиМ. – 2008. – № 4. – С. 68–75.
3. Хоменко В. А. Шаблон программного обеспечения устройств связи с объектом / В. А. Хоменко, Е. Н. Сидоров, И. Б. Мендзебровский // Проблеми програмування. – 2008. – № 2–3. – С. 239–248.
4. Architecture of a cloud-enabled service oriented platform The Fifth world congress “Aviation in the XXI-st century” : Матеріали конгресу [“Авіація в 21 столітті”], Київ, 25–27 вер. / National Aviation University ; I. Sydorov. – K. : National Aviation University, – 2012. – С. 1. 10. 36 – 1. 10. 38.

Сидоров Є. М.

МЕТОД ДОМЕННОГО АНАЛІЗУ ЯК СЕРВІС ХМАРНОЇ СЕРВІС-ОРІЄНТОВАНОЇ ПЛАТФОРМИ

У статті розглянуто розв’язання завдання побудови PaaS and SaaS застосувань. Наведено архітектуру для побудови хмарно-орієнтованої платформи SaaS застосувань, яка забезпечує основні характеристики сервісів. Архітектуру продемонстровано в контексті застосування хмарних обчислень. Доменний аналіз заявлено як сервіс хмарно-орієнтованої платформи. Метод доменного аналізу реалізовано за допомогою двох контролерів платформ. Розв’язується завдання – створити програмний шаблон для пристрою комунікації успадкованого авіаційного транспорту. Формалізація опису процесів виконання із застосуванням алгебри процесів реального часу (RTPA). Опис процесів у RTPA є входом для засобу, який створює service (контролер доменного аналізу) хмарно-орієнтованої платформи. Сервіс доменного аналізу може бути застосовано не тільки для реінженерії програмного забезпечення авіаційних тренажерів.

Ключові слова: програмне забезпечення як сервіс, хмарно-орієнтована платформа, доменний аналіз, RTPA, авіаційний тренажер.

Матеріал надійшов 15.05.2014