3-1-2009

Implementing a SOA Using Scrum: Achieving Improved Productivity using Service Trees

VamshiKiran Gireddy
vgireddy@gmail.com

Andrew P. Ciganek

IMPLEMENTING A SOA USING SCRUM: ACHIEVING IMPROVED PRODUCTIVITY USING SERVICE TREES

Vamshi Kiran Gireddy
Jacksonville State University
vgireddy@gmail.com

Andrew P. Ciganek
Jacksonville State University
aciganek@jsu.edu

ABSTRACT

Service Oriented Architecture (SOA) is spreading widely today in a variety of development environments. Many organizations pursing a SOA, however, are unaware of the impact that this approach has on their development methodology and implementation strategies, which may be inappropriate and inefficient. Some issues caused by an inappropriately managed SOA implementation include dependency conflicts, inefficient time utilization, and ineffective utilization of resources resulting in a backlog of tasks. The individuality of services creates a gap that makes service implementation a relevant topic to pursue which can be addressed through the granular development of these services. Consequently, this research-in-progress examines SOA-based services development by focusing on service generation and proposing the concept of service trees whose branches serve as nodes of SOA-based services. Further, implementing the tree's branches utilizing the scrum development methodology produces an approach where multiple services can be developed in parallel, offering a means to improve the productivity of a SOA implementation.

Keywords

SOA, development methodology, scrum, service trees

INTRODUCTION

SOA is gaining wide acceptance in industry today as a systems development approach where functionality is grouped around business processes and packaged as interoperable services. SOA also describes an information technology (IT) infrastructure, which allows different applications to exchange data with one another as they participate in business processes. The aim is for the loose coupling of services with operating systems, programming languages and other technologies which underlie applications (Lawson, 2007).

SOA separates functions into distinct units or services, which are made accessible over a network so that they can be combined and reused in the production of business applications. These services communicate with each other by passing data from one service to another, or by coordinating an activity between two or more services. A SOA assumes an architecture oriented towards services, but this may not necessarily be reflected in the design of a SOA – a Service-Oriented Design (SOD) (Erl, 2007). The research addresses SOA development by focusing on SOD through the concept of "service trees" and the scrum development methodology.

LITERATURE REVIEW

Service Tree

A "service tree" is a tree whose nodes represent the services of a SOA. Service tree design is based on two features of general IT systems, dependency and priority. A service tree starts with the most independent service node and grows as a tree with its dependent nodes as its children, grandchildren and so on. For example, if service B depends on service A, then service A is the root node and should be developed first followed by the development of service B. If multiple parents exist for a service node, then a new tree should be started and that node be represented as a child for the parent level, which makes analysis of tree structure easier. This priority for the development of services demands that highly required services receive preference through the development of its own tree. The same is true when dependency is difficult to establish.

A service tree can be divided into levels according to the dependencies and priorities of services. Services belonging to the same level can be divided further into tasks and grouped to form sets of tasks with the same priorities and dependencies. Each set of tasks are then ready for development and can be organized according to a descending order of task priorities and dependencies. Since these sets of tasks do not have any dependencies between them, they can be developed in parallel.

Scrum

Scrum is an agile development methodology where small teams consisting of a maximum six to eight people divide their work into "mini projects" lasting about one month and have a limited number of detailed tasks to be solved. Whereas traditional development methods focus on delivering a product, the aim of scrum is to deliver business value. Scrum's development period is divided into an equal set of small time periods called "sprints". During each sprint, the team creates an increment of *potential shippable* or usable software. The set of features that are incorporated into each sprint is derived from the *product backlog*, which is a prioritized set of high level requirements of work to be done. During a *sprint planning meeting*, the product owner informs the team of the items in the product backlog to be completed (Vizdos, 2008).

The product backlog contains broad descriptions of all required features and "wish-list" items. A sprint backlog is a document with a significant amount of detail containing information about how the team is going to implement the requirements for the upcoming sprint. After each sprint, a sprint retrospective is held where all team members reflect about the past sprint. Scrum adopts an empirical approach, that is, accepting that the problem cannot be fully understood or defined and thus focusing instead on maximizing the team's ability to deliver quickly and respond to emerging requirements. When applied with scrum, service tree iterations become sprints and in each sprint, a service level is implemented producing the maximum possible services as developed in the tree. This result of this approach can improve both productivity and the efficient utilization of time. Figure 1 depicts a service tree using the scrum development principles where development of the tree starts with service S_1 and is followed by its dependent services or low priority services.

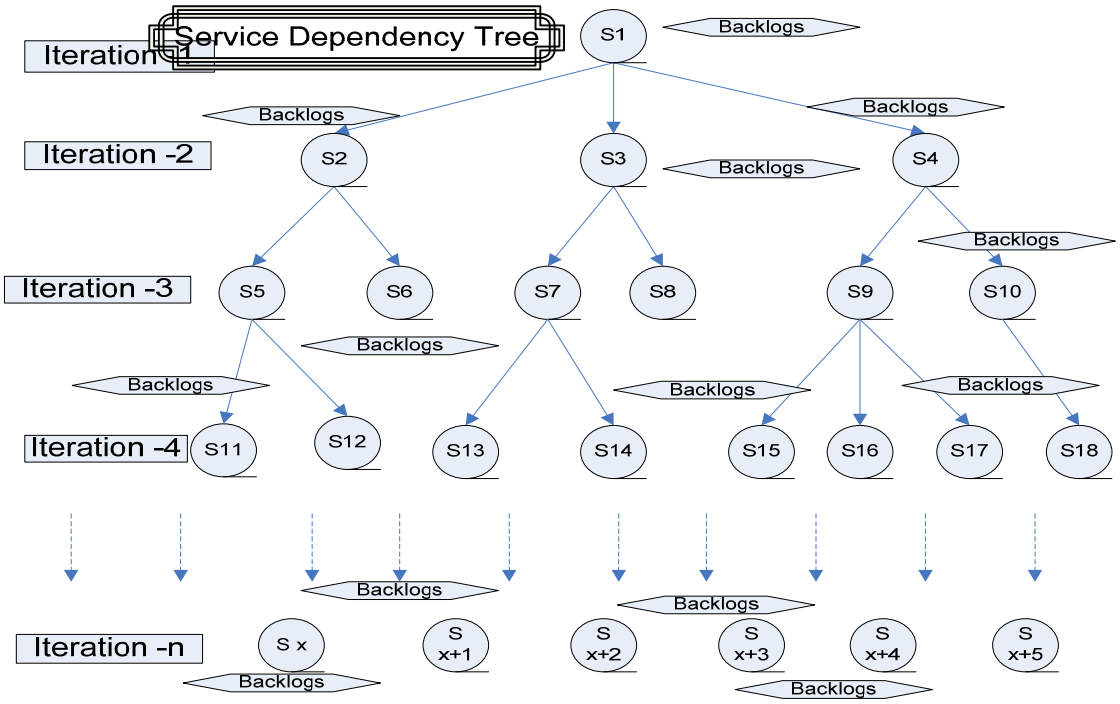

Figure 1. Service Tree Development Using Scrum

APPLICATION OF SCRUM

Each level of a service tree can be assigned to a sprint and backlogs of that level can be considered sprint backlogs. Each task of that level can be assigned up to sixteen hours, which is a common standard. The service tree (product backlog) can then be assigned to a collection of sprints. The subsequent figures depict service tree development utilizing scrum. In Figure 2, a set of unorganized tasks of the same level are present. Figure 3 depicts the organization that takes place where dependencies or priorities are present at this task level (e.g., a priority index of I_x: Iteration X; T_{mn}: Node M, Task N; sample dependencies of $I_1T_{21} > I_1T_{32} > I_1 T_{23}$, $I_1 T_{31} > I_1T_{22} > I_1T_{42} > I_1T_{43}$, and I_1T_{41}). Consequently, applying scrum to each organized level allows nodes to be developed in parallel, which can improve both productivity and the efficient utilization of time. Utilizing the dependencies between the service nodes, a service dependency matrix (see Table 1) can be generated to form a tree structure and identify relationships with dependencies and priorities as well as represent the tree in a single unit (see Figure 4).

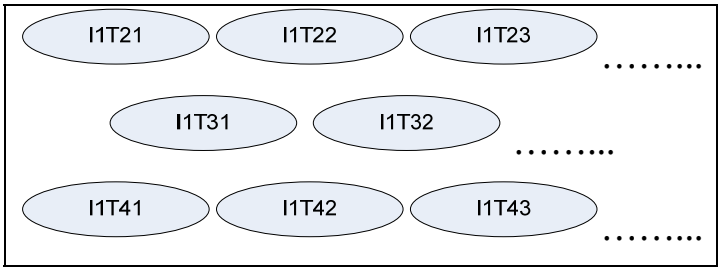

Figure 2. Unorganized Tasks of Level-1

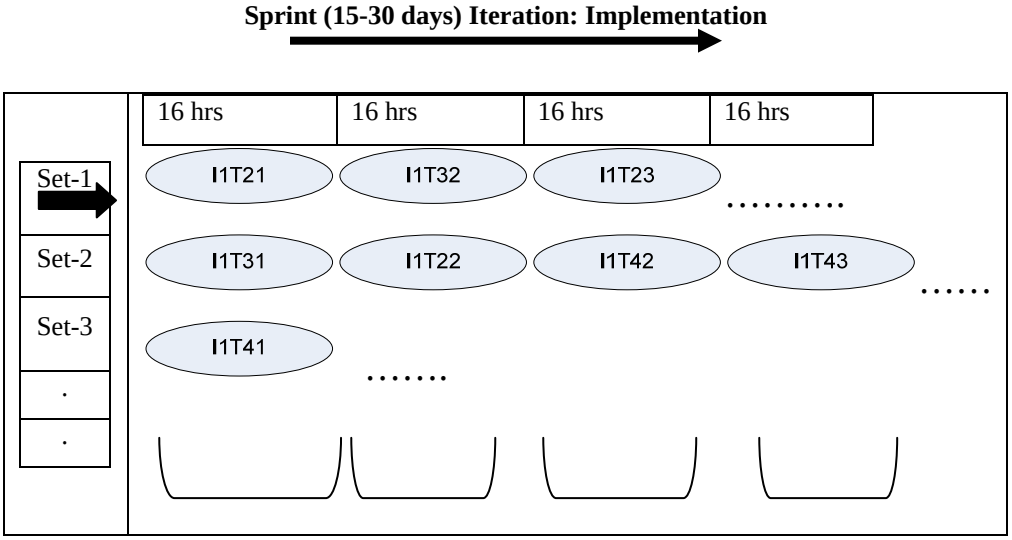

Figure 3. Organized Tasks of Level-1

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
S_1	0	1	1	1	2	2	2	2	2	2
S_2	-1	0	0	0	1	1	0	0	0	0
S_3	-1	0	0	0	0	0	1	1	0	0
S_4	-1	0	0	0	0	0	0	0	1	1
S_5	-2	-1	0	0	0	0	0	0	0	0
S_6	-2	-1	0	0	0	0	0	0	0	0
S_7	-2	0	-1	0	0	0	0	0	0	0
S_8	-2	0	-1	0	0	0	0	0	0	0
S_9	-2	0	0	-1	0	0	0	0	0	0
S_{10}	-2	0	0	-1	0	0	0	0	0	0

0: No dependency 1: Immediate child -1: Parent 2: Grand Child -2: Grand Parent

Table 1. Service Dependency Matrix

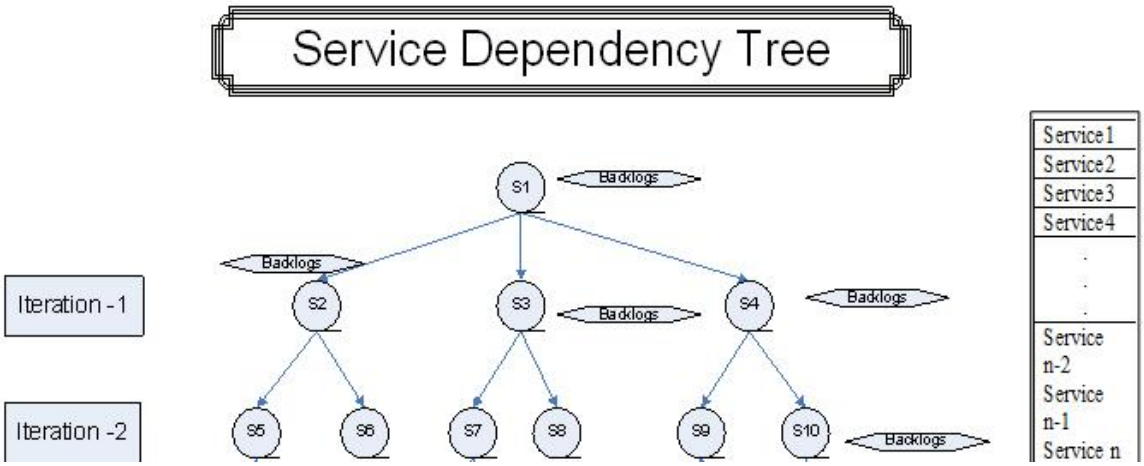

Figure 4. Sample Service Tree

Figure 4 depicts a model service tree structure and allocation of services in iterative periods with a service backlog task box. Figure 5 depicts a variety of SOA-based services available to develop a user profiler application. A "user profiler" is a set of tasks that involves in a user's interaction with an enterprise Web services, like user-registration, authentication, authorization, and user subscription services, among others. Figure 6 illustrates how the service tree and scrum methodology can be applied to the user profiler application. User authentication, authorization and other services require a user to be first registered, so the registration service first needs to be developed. Figure 6 represents how a user profiler service tree would be developed based on its dependencies.

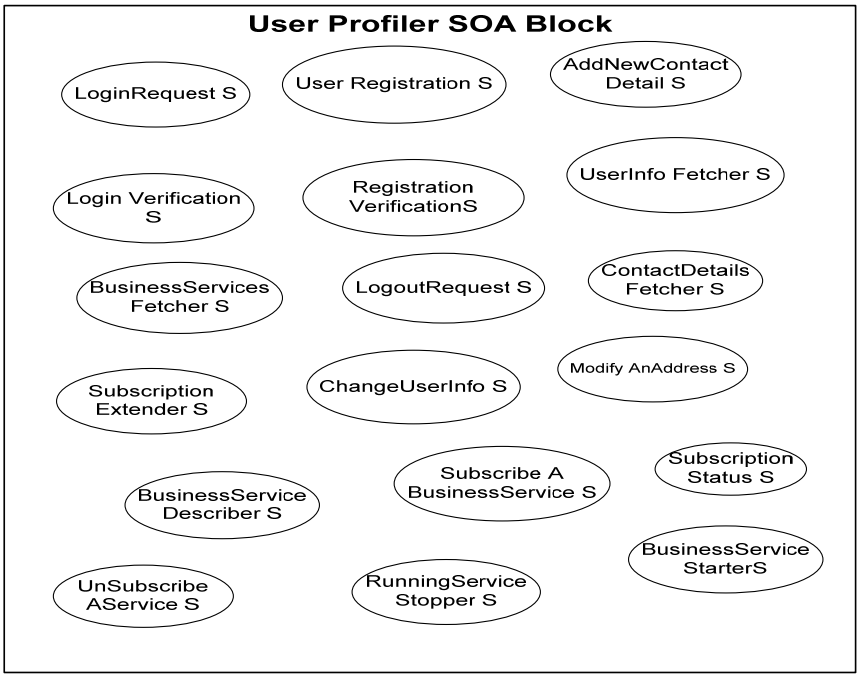

Figure 5. Sample SOA Block, User Profiler

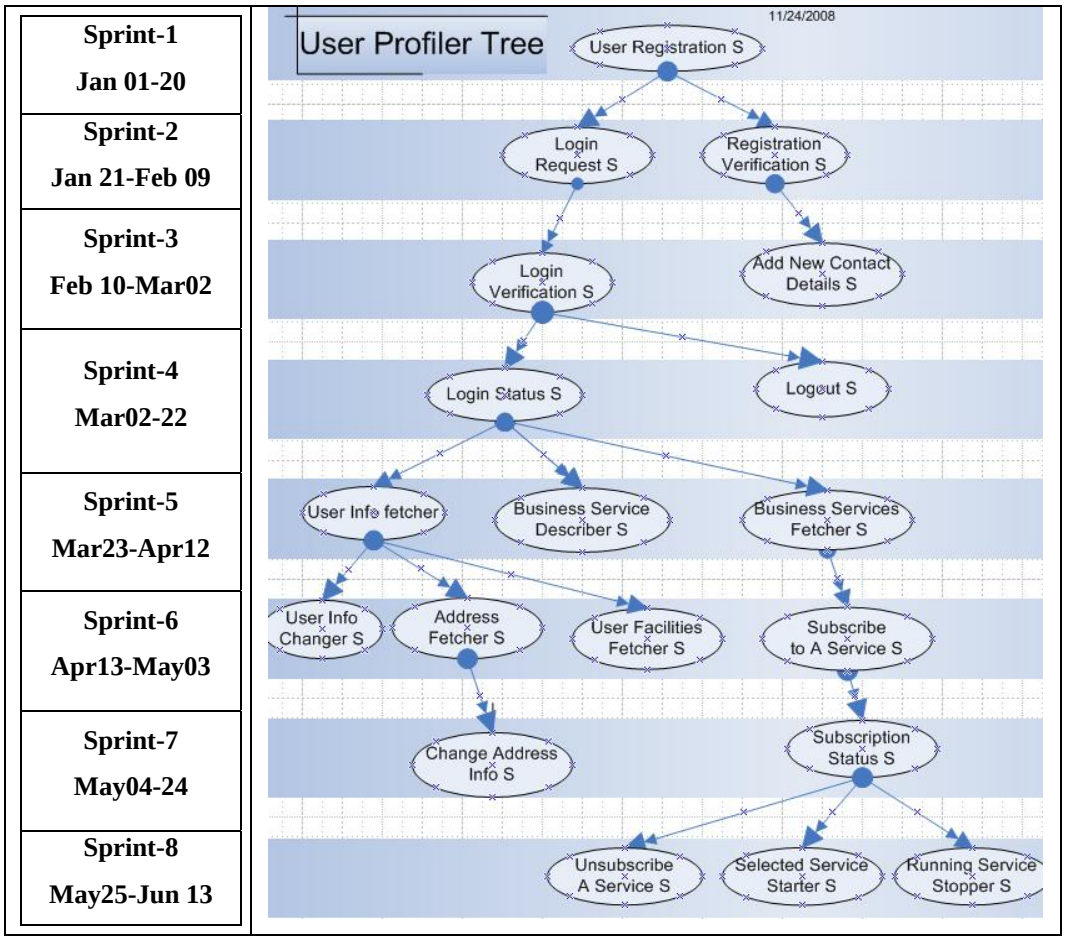

Figure 6. Service Tree and Scrum Applied To User Profiler Scenario

CONCLUSION

An inappropriately managed SOA implementation may lead to dependency conflicts, inefficient time utilization, and ineffective utilization of resources resulting in a backlog of tasks. In this research-in-progress, a service tree approach utilizing the scrum methodology for a SOA implementation was presented. This approach can impact SOA implementations by improving both the productivity and the efficient utilization of time. Future research will also constitute mathematical and programmatic representations for arranging the service nodes while incorporating external influence, for example from a project lead. Such influences could also priorities to be set among backlog tasks and services (e.g., mathematical and programmatic representations on transferring nodes of a tree between its different available levels and branches). Future research will also illustrate the service tree approach in additional examples, develop metrics to quantify the benefits of this approach, and illustrate the effectiveness of this design.

REFERENCES

1. Erl, T. (2007) SOA Principles of Service Design, Prentice Hall, Crafordsville, IN.
2. Lawson, L. (2007) Core Problems with SOA, Viewed on October 17, 2008, http://www.itbusinessedge.com/blogs/mia/?p=141
3. Vizdos, M. (2008) Implementing Scrum, Viewed on October 17, 2008, http://www.implementingscrum.com/