IMPACT: International Journal of Research in Engineering & Technology (IMPACT: IJRET)
ISSN(E): 2321-8843; ISSN(P): 2347-4599
Vol. 2, Issue 6, Jun 2014, 175-180

STATE-OF-THE ART DIFFERENCES BETWEEN TRADITIONAL SOFTWARE ENGINEERING (TSE) AND SERVICE-ORIENTED SOFTWARE ENGINEERING (SOSE)

MUSA M. AHMED

Department of Science Education, Modibbo Adama University of Technology, Yola, Nigeria

ABSTRACT

To gain insight into the challenges of SOSE issues, a good understanding of its features is imperative, this paper reports a systematic study of the difference between the Traditional Software Engineering, TSE, and Service-Oriented Software Engineering, SOSE. While TSE follows the SDLC formally, SOSE uses services as building blocks. In TSE, group of programmers work together through "Design-Development-Testing" circles, SOSE's roles are divided into service consumer, service broker and service provider and employs "Discovery-Decomposition-Evaluation" circles. Services are dynamically discovered and composed in SOSE whereas requirements are set at the beginning and delivered at the end of the project in the case of TSE. Seven of such differences are identified in tabular format and followed by some recommendations.

KEYWORDS: Differences, Service-Oriented Software Engineering, Software Engineering, Traditional Software Engineering

INTRODUCTION

The inspiration of software development methodologies is engineering discipline such as civil or mechanical engineering, such disciplines put a lot of emphasis on planning before you build. Such engineers will work on a series of drawings that precisely indicate what needs to be built and how these things need to be put together. Many design decisions, such as how to deal with load on a bridge are made as the drawings are produced. The drawings are then handed over to a different group, often a different company to be built. It's assumed that the construction processes will follow the drawings. In practice, the constructor will run into some problems but these are usually small. Since the drawings specify the pieces and how they need to be put together, they act as the foundation for a detailed construction plan. Such plans figure out the tasks that need to be done and dependencies exist between these tasks. This allows for a reasonable predictable schedule and budget for construction. It also says in detail how the people doing the construction work should do their work. This allows the construction to be less skilled intellectually, although they are often very skilled manually, (Fowler, 2000)

There are two fundamentally different activities in engineering processes. Design which is difficult to predict and requires expensive and creative people, and construction which is easier to predict. Once we have design, we can deal with the construction in a much more predictable way. Therefore, we need to figure out how the design for software so that the construction can be straight forward. In software engineering, Different stakeholders in the development process must assume different roles. As such, the development process can benefit from a separation of concerns that acknowledges the difference between the two key activities software engineers perform in Software development. During conceptualization

and analysis, software engineers elicit requirements that clarify the business needs. Design, development, and testing is an iterative set of phases in traditional software engineering while Service software engineers discover services, decomposes and evaluate software. Service-centric system-management life cycle is modular. You can split it into three aggregate phases: Business-process management, design-time software engineering, and runtime software engineering, and different stakeholders can manage these aggregate phases (Blake, 2007).

Most SOSE methodologies have been proposed in both academia and industry aiming at providing approaches, methods and tools for researchers and practitioners to engineer SBAs. However, without being fully understood, a methodology is less valuable no matter how perfect it is. This is particularly relevant to SOSE methodologies as they are more complex than TSE ones, having to deal with new challenges while keeping the principles of TSE. With the aim of understanding SOSE features, this paper presents in tabular format the difference between SOSE and TSE.

Traditional Software Engineering, An Overview

As a results of the new paradigm for software development gain favor at the NATO conference in 1968, (Randel, 1968). It considered software development as a form of engineering. The hope is that if software development can develop as an engineering discipline then it would be possible to develop complex and large software development projects on-time, on-budget, with fewer bugs and to meet most(if not all) of the requirements, (Aitken and Ilango, 2013). Engineering is generally thought of as processes for using knowledge to achieve objectives, usually in building (or at least designing) complex systems or structures. Engineers puts a great deal of effort into predicting and planning their work and generally work with fixed requirements, that is, they know what they are supposed to build at the commencement of the projects.

Software Engineering according to Wikipedia (2014) is the establishment and use of sound engineering principles in order to economically obtain software that is reliable and work efficiently on real machines. In other words, software should be maintainable, dependable and acceptable. Software engineering is the application of a systematic and disciplined approach to the development, testing and maintenance of a program, (Wu, 2004). Software engineering presents a broad perspective on software systems engineering, concentrating on widely used techniques for developing large-scale systems.

Software Engineering may be defined as the systematic design and development of software products and the management of the software process (Mills, 1980). Software Engineering has as one of its primary objectives the production of programmes that meet specifications, and are demonstrably accurate, produced on time and within budget. Software Engineering is the branch of system engineering concerned with the development of large and complex software intensive system (Finkelstein and Kramer, 2000). It focuses on:

- A real-world goals for services provided by and constraints on such systems;
- The precise specification of system structure and behavior, and the implementation of these specifications;
- The activities required in order to develop an assurance that the specifications and real-world goals have been met;
- The evolution of such systems over time and across system families.
- It is also concerned with the:

- Process
- Method
- Tools

For development of software intensive systems in an economics and timely manner.

Service-Oriented Software Engineering, An Overview

Service-Oriented software Engineering is a software engineering methodology focused on the development of software systems by composition of reusable services often provided by other service providers, the essential elements is the dynamic nature of the connection between the service users to the service providers to provide a leading edge IT solution. Service engineering generalizes the development of so-called valued-added services, the traditional services bridging the gap between technology and the end user known from telecommunication and reactive systems. Recently, the trend in software development has shifted from developing software systems to developing service-oriented systems that are composed of ready to use services. The service-oriented architecture (SOA) architectural style has been widely adopted in industries thanks to its ability to providing seamless integration among software services (Erl, 2005). If the services are well-specified, loosely coupled, and coherent, implementing a SOA can bring many benefits to an enterprise, including: reduced IT costs, and increased organizational agility (Erl, 2005).

This new paradigm computing utilizes services as the basic construct to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments. The visionary promise of service-oriented computing is a world of cooperating services where application components are assembled with little effort into a network of services that can be loosely coupled to create flexible dynamic business processes and agile applications that may span organizations and computing platforms, (Papazolou, et al, 2003). The subject of service-oriented computing is vast and enormously complex, spanning many concepts and technology that find their origins in diverse disciplines that are woven together in an intricate need to merge technology with understanding of business processes and organizational structures, a combination of recognizing an enterprise's pain points and the potential solutions that can be applied to correct them. The ability to layer solutions and support heterogeneity allows for gradual migration to service-based solutions.

The relevant key concepts in SOSE (Munro et al, 2000) include but not limited to the following:

- An open market place for services,
- Dynamic provision of software in response to changing requests,
- The potential for one-time execution followed by unbinding,
- A services supply network where service providers may subcontract to provide their services,
- Delivery transparency to software users, whose interest lies in its use.

The term service-oriented software has now been applied to the older technologies of DCOM and CORBA, More recently to J2EE and .NET deployments and of course to Web services. There's no reason why the technology has to

be a discriminating feature in SOA. Standards such as SOAP for web services help to ensure that heterogeneity of solutions poses no problems (Munro et al, 2000).

Traditional Software Engineering verses Service-Oriented Software Engineering

Below are the Differences between Traditional Software Engineering and Service-Oriented Software Engineering

Table 1

S/No	Traditional Software Engineering	Service-Oriented Software Engineering
1.	Process and tools driven	Services are the building blocks
2.	Does not well-come changing requirements during projects progress	Open services driven, architecture of service-oriented system can be changed or even determined at run time since service are the building blocks.
3.	Follow the phases of SDLC formally, encourages formal analysis and planning.	Additional development roles are involved in development; these roles are rather split into three essential roles: Service consumer, Service provider and Service broker.
4.	All requirements decided in the requirements gathering phases should be delivered by the final software built.	Services are engineered with multiple sets of requirements to fulfill different groups of potentials consumers with different quality requirements.
5.	Documents driven, every activity is measured by intensive documentation or deliverances.	It provides software solutions dynamically in response to changing requests.
6.	Software engineering life-cycles involves “Design-Development-Testing” repetitions.	Service-Oriented Software engineering life-cycles involves “Discovery-Decomposition-Evaluation” repetitions.
7.	Requirements are set at the beginning and delivered at the end of the Project.	Requirements are dynamically generated; as such changes can easily be made and composed.

CONCLUSIONS

One of the major advantages of SOSE is the ease to making changes; the flexibility that gives this approach the potential to ease the evolution problem creates new difficulties in software understanding. The main contribution of this paper come from an overview of both the TSE and SOSE being recognized in the research community and their differences classified in tabular format. In contrast to TSE’s SDLC formality, SOSE methodology uses services as its building blocks, and development roles are categories into three: Service consumer, service broker and service providers (sub-provider). To improve the understandability of SOSE methodology, a number of its service life cycles are compared with that of TSE and the differences are highlighted.

RECOMMENDATION

- Service providers should develop user-friendly services that are easily discoverable and understandable.
- Managing trust within the automated procurement process of SOSE will be more difficult, however, automated methods for negotiating such non numerical and human-oriented concepts will require further research before they are sufficiently mature to be incorporated into everyday business practices.
- A legal framework within which to form contracts is another major challenge for an automated and global solution, setting a legal framework within which to form contracts is crucial.

REFERENCES

1. Finkelstein and J. Kramer (2000), “Software Engineering: A Road map” in “the future of software Engineering”. ACM press.

2. Aitken, A. and Ilango, V. (2013), A comparative Analysis of traditional Software engineering and Agile Software Engineering.

3. T. Wu (2004), An introduction to object-oriented programming with java, updated third edition. Mc Graw Hill Higher Education.

4. H. D. Mills (1980), The management of Software Engineering; Part I: Principles of Software Engineering. IBM systems journal vol 19 issue 4 pg 414-420

5. M. B. Blake (2007), Decomposing Composition: Service-Oriented Software Engineers. IEEE Computer Society Vol. 24 No 6

6. M. Fowler (2000), The New Methodology. www.martinfowler.com/articles/newmethodologyoriginal.html

7. M. P. Papazolou, P. Traverso, S. Dustdar and F. Leymann (2003), service –oriented computing. Communication of the ACM VOL 46 Pg 25-2.

8. N. Gold, A. Mohan, C. Knight and M. Munro (2000), Understanding Service-oriented Software. www.computer.org/publications/dlib

9. Nano, P. and Raindell, B. (1968), Software Engineering: Report of a conference sponsored by the NATO science committee (2-11 Oct. 1968). Brussels, scientific Affairs Division, NATO.

10. T. Erl(2005), Service-oriented architecture: concepts, technology and design. Prentice Hall Wikipedia (2014), Software Engineering. Wikimedia Foundation Inc. http://en.wikipedia.org/wiki/softwareengineering