ISSN 1816-6075 (Print), 1818-0523 (Online)
Journal of System and Management Sciences Vol. 9 (2019) No. 3, pp. 65-77
DOI:10.33168/JSMS.2019.0304

UML Sequence Diagram for Reusability of Data Components

Sang Young Lee

Namseoul University, South Korea
sylee@nsu.ac.kr

Abstract. Synchronized Multimedia Integration Language (SMIL) document supports various types of multimedia resources that exist on the Internet. This paper discussed the ultimate goal of reusing of component and service is to get varied advantages in the business. Specifically, first, it reduces work with improved productivity and shorter lead time. Second, the improved quality reduces the number of errors. Third, the improved productivity reduces maintenance costs. Fourth, it reduces later rework through analyzing overall requirements for reuse. In software engineering, component-based development is a way to develop a software by reusing components. In other words, the developed component is saved on the repository, and the saved component can be relocated and reused by a user. Accordingly, this paper proved that SMIL-based animation document can be provided in terms of viewpoint of reuse. In such viewpoint, SMIL document improves existing XML animation document. Also, SMIL can efficiently enhance its reusability in terms of an application that creates multimedia presentation using the Internet. Currently, relevant software such as parser, editor, player, etc. has been actively developed. Consequently, effective component for computer animation is required to manage SMIL documents. Thus, this paper suggests a system to program interactions over time by transforming temporal script of computer animation tool kit in RASP into SMIL document format. This type of system creates SMIL document by using the UML sequence diagram. Hence, it is important to confirm the performances with the sequence diagram generated for synchronizing SMIL documents.

Keywords: sequence diagram, reuse, components, synchronization.

1. Introduction

Synchronized Multimedia Integration Language (SMIL) document supports various types of multimedia resources on the Internet. In other word, it provides not only a systematic and structured way to present diverse multimedia contents but a certain thought on the Internet (Jourdan et al., 2000; Nanard, 2001). Also, the document is able to improve how to present common multimedia by utilizing its simple and easy-to-use tags and attributes. These development of multimedia languages and the spread of semantic web technology started from the necessity to provide a searching method for multimedia contents. In short, the need to generate more semantic tools has grown. By applying theoretical principles and standardized technology, these semantic web offers new ways to perform the tasks to search multimedia, considering the real meaning of things than the sentence structure (Mostafa, 2002; Jalender et al., 2010).

As main points of this paper, the ultimate goals of reusing components and services are as follows (Shireesha, 2010; Camara, 2016). First, it reduces the efforts with improved productivity and shorter lead time; second, the improved quality reduces the number of errors; third, the improved productivity reduces maintenance costs; and, fourth, it reduces later rework through analyzing overall requirements for reuse. With the business process modeling, this study aims to optimize process modeling of entire tasks and to effectively deal with a fast pace of changes in business processes. In the field of software engineering, component-based development indicates methodology to develop software by reusing components. The developed component is saved on the component repository, and the saved component can be relocated and reused by users.

This paper proposed a methodology to systemize SMIL documents to present multimedia contents. In addition, it also proposed how to generate SMIL with the UML sequence diagram. This paper consists of the following contents. In 2 chapter, it shows related researches. In 3 chapter, it provides multimedia contents, UML diagram, and synchronization in terms of viewpoint of reuse. In 4 chapter, it suggests system and algorithm to generate SMIL documents. And, in 5 chapter, it provides conclusions.

2. Related Works

Interest on reusing software in the field of computer graphic increases recently. If scenes of the animation are effectively visualized, the code and design are reused, thus reduces time, effort, and cost. In majority of cases, however, these reuse cause difficulties in understanding and applying interactions between animation components. Reusing components requires a way to reconfigure components. Accordingly, there have been various studies (Dhulipalla et al., 2016; Cheng et al., 2015).

SMIL is a standard to integrate synchronized multimedia with XML-based markup language. SMIL was developed as an open standard for video, audio, or text in the SML presentation. SMIL was adapted as the standard defined by W3C (World Wide Web Consortium). Moreover, W3C defined an SMIL subset called SMIL 2.1, which was designed for mobile devices and portable devices. It has been applied to mobile phone such as MMS (Multimedia Messaging Service). When mobile applications use the multimedia presentation, their utilizability would be improved. These applications include electronic learning (e-learning), mobile communication, mobile marketing, etc. Using SMIL, the interface between human and computer would be improved in those application (David, 2010; Kim, 2017; Garlan, 2014; Iftikhar, 2017).

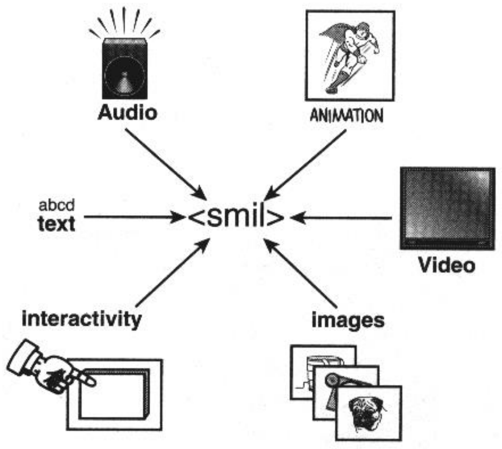

Fig. 1: SMIL multimedia sources

In general, SMIL documents can be applied to a wide range of applications and perform different functions. The documents allow applications to collect presentation contents in a varied of ways. It also enables an application to create a very small sized document. Due to the simple text structure of SMIL, the volume of a multimedia presentation would be reduced. In addition to its functions, the user can present and manipulate the content by inserting control events in the SMIL document (Jamshidi, 2018; Kim, 2016; Henrik, 2008; Pu et al., 2003).

The SMIL animation prefers declarative manner based on correct route-oriented animation model. It consists of basic XML animation elements including Animate Motion, Animate Color and Set. These elements deliver timing attributes (Begin, End and Dur) of time layouts expressing the media objects. Thus, these elements are included into other languages such as XHTML + Time, SVG or CSS, which provides an improved animation. Furthermore, use case diagram and sequence diagram provide a way to present a document for time synchronization (Lee, 2019; Islam, 2016; Jiang et al., 2015).

Meanwhile, the UML sequence diagram explains the interaction in the web-based system having relationship between objects and shows the scenario about system operation. A simple UML sequence diagram explains exactly one scenario. If a diagram explains many scenarios and describes real-time system, it has to systematically consider the configuration of UML sequence diagram.

3. Sequence Diagram and Reusing Software

The advantages of the sequence diagram are to use it when reusing software and to visually express and understand component. Usually, the diagram is organized over time, showing the flow of the time of messages. It also provides a way to describe the relationship between objects and messages. The sequence diagram has two dimensions: horizontal one to describe the relationship between vertical dimension and objects and vertical one to describe the relationship between horizontal dimension and objects. The messages in both dimensions are indicated by a horizontal line arrow from the sender and the recipient. Separating each layer horizontally is a typical way to configure the web-based system. It can be effectively described by the sequence diagram, which is appropriate to stratify the web-based system. Normally, in the system, messages are exchanged in the layer to invoke programs. The root program element performs such exchanges, which is the most fundamental and important element in the sequence diagram. And, the elements of the node program that exist in subprogram are displayed in front of the elements of leaf and isolated program.

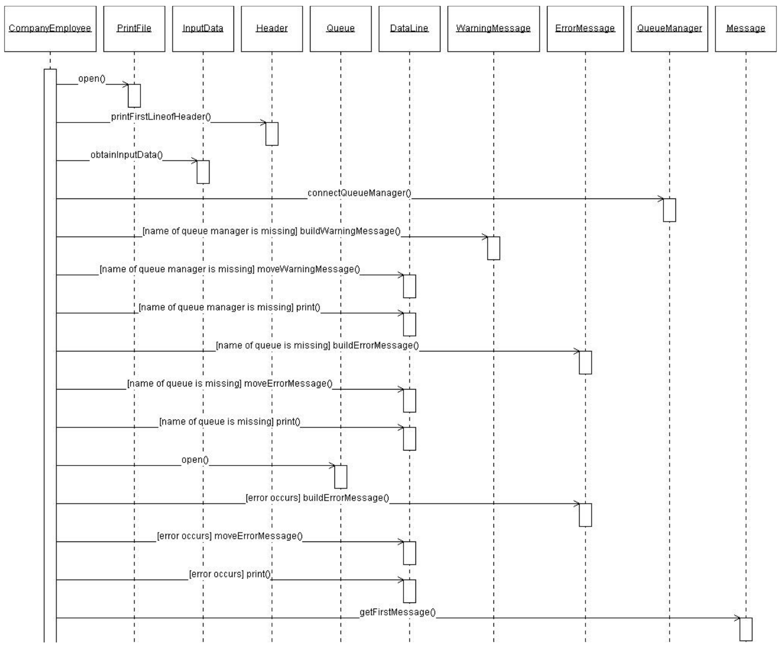
CompanyEmployee
PrintFile
InputData
Header
Queue
DataLine
WarningMessage
ErrorMessage
QueueManager
Message
open()
printFirstLineofHeader()
obtainInputData()
connectQueueManager()
[name of queue manager is missing] buildWarningMessage()
[name of queue manager is missing] moveWarningMessage()
[name of queue manager is missing] print()
[name of queue is missing] buildErrorMessage()
[name of queue is missing] moveErrorMessage()
[name of queue is missing] print()
open()
[error occurs] buildErrorMessage()
[error occurs] moveErrorMessage()
[error occurs] print()
getFirstMessage()

Fig. 2: Sequence diagrams of application program

The major actors in the web-based system can be seen in the upper-left side of the sequence diagram, while other actors are assigned by time and importance. Basically, the object in the sequence diagram receives messages and invokes time-ordered operations. The next messages can be created in the next dimension of time only if running the only one message. Hence, the sequence diagram has an advantage to display correct time a message runs, by using the object.

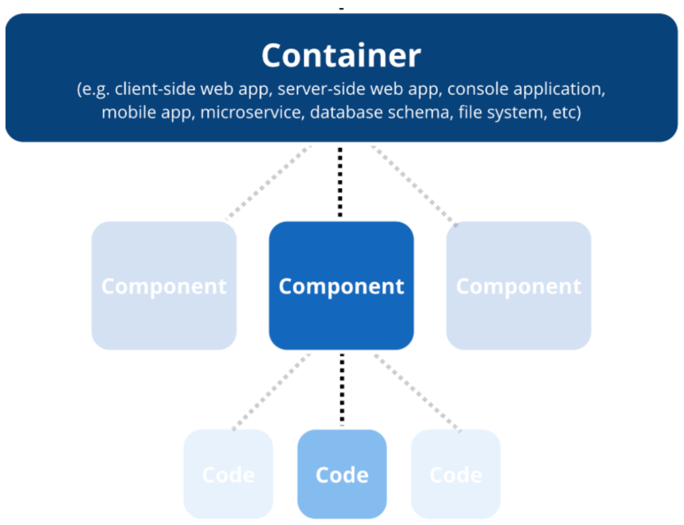
Container
(e.g. client-side web app, server-side web app, console application, mobile app, microservice, database schema, file system, etc)
Component
Component
Component
Code
Code
Code

Fig. 3: Sequence diagrams and reusing software and components

This paper presents that reusing of software improves productivity while maintaining the quality of the software. Reusing makes the new software easy to build from the existing software or knowledge on software. The figure 3 shows that components linking a container are connected with a code. Those components indicate component objects that are able to express applications with graphic contents. The users have quick access to the components exactly. This type of access can be modified.

Reusable software components are possible to reuse in another domain, and they can be interchangeable. Reusing software provides the following benefits.
- It provides a foundation for improving quality.
- It improves reliability for software development and maintenance and reduces long-term cost.

In software engineering, its focus is the functions of component itself. The component is treated as a system where it performs a certain task specified in advance, which consists of a number of components. In other word, the software component is a building block to perform a certain function. It is able to communicate with each other through a standard interface.

Reusing components drives a shift in developer's point of view from software programing to system configuration of software system. Such component reuse saves development time and cost and improves product quality. The definition of component to be reused is as follows: the functions of each component are defined in advance; communication methods are standardized; and communication protocols are set in advance.
- In the same system, it is easy to integrate and communicate with a different component.

- In another system outside of the same domain, it is possible to integrate and communicate with a different component.
- It is possible to integrate and communicate with an application in the external domain.

To be a huge system, it should be formed with reusable components, and that they already exist to facilitate developing the system. The components must be easily searched and used. It is required to develop new methodologies and tools that allow the component to create and access the library. Therefore, it will be considered in developing reusable software components and maintaining them in the future.

Reusing software components is divided into four levels.
- Code level component (module, procedure, subroutine, library, etc.)
- Entire applications
- Product in the analysis level
- Product in the design level

The most frequently used level of reusing components is the code level. The component of code level is able to be reused by using the standard library. Through the process, it is easy to extend component, and it can increase performance for the reuse. However, in this case, reusing components has the lower level of abstraction and the level is lowered than expectation. Actually, in most such domains, reusing an entire application provides better reusability than code level. This method is used when making commercial products or customized ones.

4. System and Algorithm to Generate SMIL Documents

SMIL is an XML-based language, which can describe temporal relation in synchronization between multimedia data that is not handled with HTML. SMIL document is configured with <head> element in <smil> and </Smil> tags and <body> element. What follows is a description of the SMIL tag.

(1) <head> tag

It is a tag about document information presented according to sequence. It includes meta element and <switch> or <layout> tag.

<layout> tag

It decides what form element is placed in the <body> of document.

<region> tag

It controls the position, size, and other attributes of media object element.

<switch> tag

It prescribes a set of preferential elements and selects one of several elements. This tag is available in the <body>.

(2) <body> tag

It is an element which includes information connected with the temporal connection behavior of the document.

<media> tag

This is an element that presents the media objects (e.g. <ref>, <animation>, <audio>, <img>, <video>, <text>, <text stream> etc.). belonging to the media tag. This creates an <animation> tag.

<synchronization> tag

It is used for temporal synchronization. The <par> tag makes the child of element overlapping at the same time and <seq> tag makes the child of an element have a temporal sequence.

<hyperlink> tag

The <anchor> and <a> tags are connected with hyperlink. The <a> tag is similar to the <a> tag of HTML and fixes the whole media sources from start to ending time. The <anchor> tag, hyperlink element such as <a> tag, can be designated anywhere on the screen during the defined period.

This chapter explains a system to generate SMIL document, and its algorithm, and application results when temporal script related in animation components is entered. First, the UML sequence diagram is created from the temporal script about the animation script. Second, SMIL document is generated from UML sequence diagram. The figure below indicates the sequence diagram of temporal scripts, tag table, and SMIL document. Temporal script determines how to connect tasks and when the task is created.

Figure 4 presents the configuration of the whole system and the functions of each element.

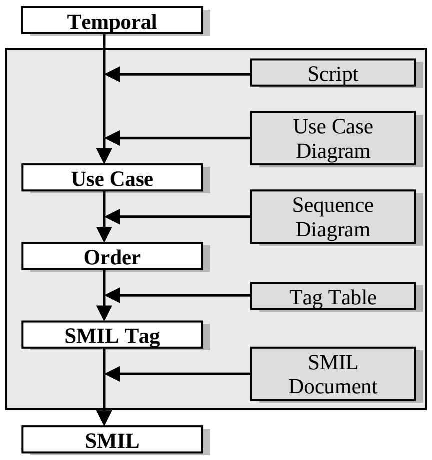

Fig. 4: SMIL document generating system configuration

- Script Analyzer

The script analyzer transforms the temporal relation according to the message mapping table.

Table 1: Sequence diagram message mapping

Relation	**message**
MEETS	seq ()
DELIMITS	par ()
STARTS	par ()
STOPS	{eiend-ejend=0} only, ei, ej are event identifier
TimingAct(Object,begin,end)	{eiend -eibegin}
TimingAct(Objecti,begin,end) TimingAct(Objectj,begin,end)	{ejbegin-eiend}

The analysis results of the temporal script in the table above are as follows. Example 1) and Example 2) are examples connected with the message.

Example 1) ObjectB→setRel(MEETS, ObjectC);

Table 2: Script analyzation (Example 1)

Object		**Message**
Begin	**End**	
Object B	Object C	seq ()

The following is an example of par ().

Example 2) ObjectC→setRel(DELIMITS, ObjectE);

Table 3: Script analyzation (Example 2)

Object		**Message**
Begin	**End**	
Object C	Object E	par ()

Example 3 presents an example of constraints. The constraint to set the ending time between two objects are as follows.

Example 3) ObjectF→setRel(STOPS, ObjectD);

Table 4: Script analyzation (Example 3)

Object	**Event identifier**		**Constraints**
	Begin	**End**	
Object F	e_{1begin},	e_{1end}	$\{e_{1end}-e_{2end}=0\}$
Object D	e_{2begin},	e_{2end}	

Below is a constraint, which presents the live time of the objects.

Example 4) proc→addTimingAct(ObjectA,1,10);

Below is the time delay between two objects.

Example 5) proc→addTimingAct(ObjectA,1,10);

proc→addTimingAct(ObjectB,30,40);

- Use Case Diagram Generator

The Use-Case Diagram Generator generates algorithm and the use case as well as the users and the SMIL document.

- Sequence Diagram Loader

The Sequence Diagram Loader is a script that inserts a sequence diagram of the use case diagram using sentence messages created by a script analyzer.

- Tag Table Generator

The Tag Table Generator creates SMIL tag table that extracts tags in the sequence diagram. It synchronizes a tag table and provides media and hyperlink tags.

- SMIL Document Generator

The SMIL Document Generator creates SMIL document instance when inputting tags on the tag table. Basically, <smil>, declarative text, comment text, start tag, and end tag are created on the document instance.

The followings are the sequence diagram of a temporal script input, tag table, and SMIL document. To transform these temporal scripts to SMIL document format, a diagram format is required. Therefore, in this paper, UML use case and sequence diagrams are used.

- Use Case Diagram

Figure 5 presents an actor interacting with a use case of SMIL document generation.

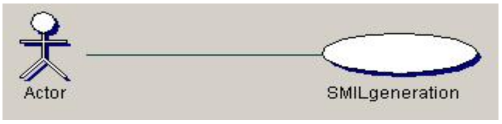

Fig. 5: SMIL document generation use case diagram

- Sequence diagram

Figure 6 presents a sequence diagram inserted into a use case of SMIL document generation. Sequence diagram presents objects, messages between objects, and constraints, etc.

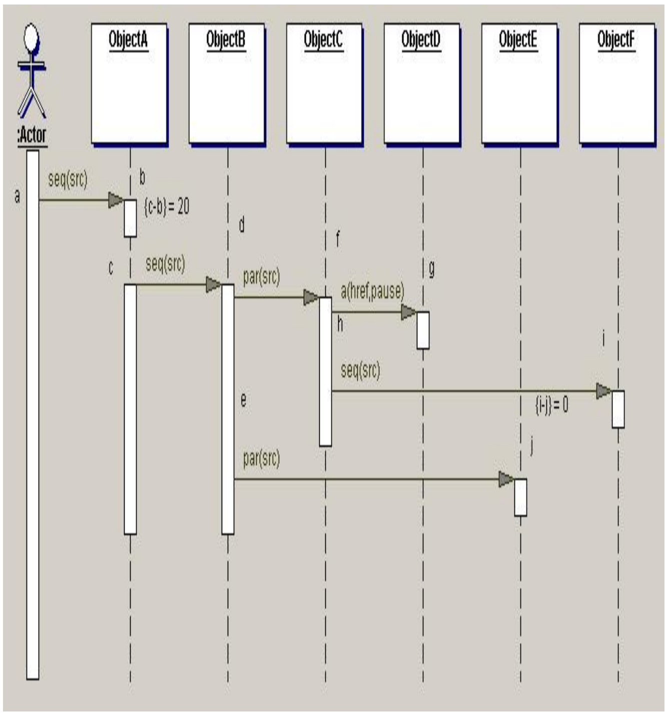

Fig. 6: SMIL document generation sequence diagram

A tag table is created for the synchronization tag, the hyperlink tag, and the SMIL tag. Therefore, SMIL document is created from the tag table and the sequence diagram.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD
  SMIL1.0//EN" "http://www.w3.org/TR/REC-
  smil/SMIL10.dtd">
  <smil>
  <body>
  <seq>
   <animation src="ObjectA", begin=1, end=10>
   <seq>
   <animation src="ObjectB" begin=30, end=40>
   <par>
   <seq>
   <par>
   <a href="D", show="pause">
   <animation src="ObjectC">
   </a>
   <animation src="ObjectE">
   </par>
   <animation src="ObjectF">
   </seq>
```

```
    </par>
    </seq>
    </seq>
  </body>
</smil>
```

5. Conclusion

This paper presents the reuse of component and service to get varied advantages in a business. First, it reduces work with improved productivity and shorter lead time. Second, the improved quality reduces the number of errors. Third, the improved productivity reduces maintenance costs. Fourth, it reduces later rework through analyzing overall requirements for reuse. In software engineering, component-based development is a way to develop a software by reusing components. The developed component is saved on the repository, and the saved component can be relocated and reused by a user.

Accordingly, this paper proved that SMIL-based animation document can be provided in terms of viewpoint of reuse. As such, SMIL document improves existing XML animation document. Also, SMIL can efficiently enhance its reusability in terms of an application that creates multimedia presentation using the Internet. Currently, parser, editor, player, and other relevant software has been actively developed. Consequently, effective component for computer animation is required to manage SMIL documents. Thus, this paper suggests a system to program interactions over time by transforming temporal script of computer animation tool kit in RASP into SMIL document format. This type of system creates SMIL document by using the UML sequence diagram. Hence, it is important to confirm the performances with the sequence diagram generated for synchronizing SMIL documents. This allows a user to reuse components and services saved on the repository effectively. If this implements service and business process, it can save development time and cost. Furthermore, it can be used as a business model help a user to adopt himself/herself in the work environment, dynamically changing the business process.

6. Acknowledgments

Funding for this paper was provided by Namseoul university.

7. References

Camara, J. (2016). Incorporating architecture-based self-adaptation into an adaptive industrial software system. *Journal of Systems and Software*, 122(2), 507-523.

Cheng, D., Garlan, & Schmerl, B. (2015). Evaluating the effectiveness of the rainbow self-adaptive system. *In Proc. of the ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, IEEE, 132-141.

David, F. & Jez, H. (2010). Continuous delivery: reliable software releases through build, test and deployment automation. *Addison-Wesley Professional*.

Dhulipalla, V. K. & Praveen, K. (2016). Student performance metrics in Android. *International Journal of Digital Contents and Applications for Smart Devices*, 3(1), 1-8.

Garlan, D. (2004). Architecture-based self-adaptation with reusable infrastructure. *Computer*, 37(10), 46-54.

Henrik, S. & Martin E. R. (2008). Potentials of SMIL applications, for mobile devices. *Journal of Mobile Multimedia*, 3(1), 88-100.

Iftikhar, M. U. & Weyns, D. (2017). A runtime environment for architecture-based adaptation with guarantees. *In 2017 IEEE Int. Conf. Software Architecture Workshops*, IEEE, 278-281.

Islam M. & Ahmed, G. M. (2016). Online analytical processing for the application of data cubes in business data visualization. *International Journal of Computer Graphics*, 7(2), 1-8.

Jalender, B., Govardhan, A. & Premchand P. (2010). A pragmatic approach to software reuse. *Journal of Theoretical and Applied Information Technology*, 14(2), 87-96.

Jamshidi, P. (2018). Microservices: the journey so far and challenges ahead. *IEEE Software*, 35(3), 24-35.

Jiang, B., Zhang, C., Wang, C. & Wang, X. (2015). Video compression algorithm based on all phase biorthogonal transform and MPEG-2. *International Journal of Hybrid Information Technology*, 8(3), 133-144.

Jourdan, M. C. & Tardif, R. (2000). A scalable toolkit for designing multimedia authoring environments. *Multimedia Tools and Applications Kluwer Academic Publishers*, 12(3), 257-279.

Kim, J. H. & Bhadauria, S. S. (2016). Inventory management to secure software system. *Asia-Pacific Journal of Convergent Research Interchange*, 2(3), 11-20.

Kim, T. (2017). A study on access control technology for a multimedia-based e-learning system: a learning intervention. *Asia-pacific Journal of Convergent Research Interchange, SoCoRI*, 3(4), 25-33.

Lee, S. (2019). UML sequence diagram for reusability of data components. *2019 1st Domestic and International Integration Conference, Asia-pacific Society of Convergent Research Interchange,* Jeju Island, Korea on August, 18-20.

Mostafa, M. E. (2002). MMS-The modern wireless solution for multimedia messaging. *Proceedings of the 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 5(2), 2466–2472.

Nanard & Nanard, J. (2001). Towards multimedia computing, lessons learned from a half century of computer science. *International Conference on Media Futures,* Florence, Italy, May 8-9.

Pu, J., Millham, R. & Yang, H. (2003). Acquiring domain knowledge in reverse engineering the web-based system into UML. *Conference of IASTED Software Engineering and Application*.

Shireesha, P. & Sharma, S. (2010). Building reusable software component for optimization check in ABAP coding. *International Journal of Software Engineering & Applications*, 1(3), 120-132.