การวางซ้อนโดยนัยยะภายในโดเมนเฉพาะกิจ: แนวทางเชิงบูรณาการสำหรับการพัฒนาการสร้างแบบจำลอง

กิตติ เศรษฐวรพันธุ์[1]

บทคัดย่อ

การที่จะสร้างสิ่งแวดล้อมสำหรับการสร้างแบบจำลองภายในโดเมนเฉพาะกิจจากแบบจำลองทางความคิดนั้น หนึ่งในทางออกที่ง่ายและมีประสิทธิภาพที่สุดก็คือ การวางซ้อนแบบจำลองทางความคิดกับสิ่งแวดล้อมสำหรับการสร้างแบบจำลองที่มีอยู่แล้วหรือภาษาหลักในการสร้างแบบจำลอง โดยพื้นฐานความคิดนี้ นักพัฒนาแบบจำลองสามารถใช้ทรัพยากรที่มีอยู่ในรูปแบบต่าง ๆ อาทิเช่น กล่องสร้างแบบจำลอง ชุดคำสั่งในโปรแกรมสร้างแบบจำลอง ซึ่งมีลักษณะคล้ายองค์ประกอบต่าง ๆ ในแบบจำลองทางความคิดเพื่อที่จะนำไปใช้ในการพัฒนาสิ่งแวดล้อมสำหรับการสร้างแบบจำลองภายในโดเมนเฉพาะกิจที่ตรงตามความต้องการของนักพัฒนาและสามารถนำกลับมาใช้ใหม่ได้ อย่างไรก็ตามแนวทางการวางซ้อนเช่นนี้ก็ไม่ได้ง่ายสักทีเดียว เนื่องจากไม่ได้อาศัยเพียงแค่กระบวนการแลกเปลี่ยนและเปรียบเทียบข้อมูลเท่านั้น ยังต้องอาศัยกรอบแนวคิดและรูปแบบที่ชัดเจนตามแนวทางของการวางซ้อนโดยนัยยะ การออกแบบกล่องสร้างแบบจำลอง รวมทั้งการสร้างแบบจำลองเชิงซ้อน ซึ่งแนวทางทั้งหมดเหล่านี้หล่อหลอมรวมเป็นแนวทางเชิงบูรณาการสำหรับการพัฒนาการสร้างแบบจำลอง

คำสำคัญ: การวางซ้อนโดยนัยยะ การออกแบบกล่องสร้างแบบจำลอง การสร้างแบบจำลองเชิงซ้อน

[1] อาจารย์ คณะบริหารธุรกิจ สถาบันการจัดการปัญญาภิวัฒน์ โทรศัพท์ 0-2832-0230 อีเมล: kittiset@pim.ac.th

รับเมื่อ 12 พฤศจิกายน 2555 ตอบรับเมื่อ 28 พฤษภาคม 2556

Domain Specific Ontological Mapping: an Integrated Approach for Modeling and Simulation Development

Kitti Setavoraphan[1]

Abstract

To establish a domain-specific simulation environment (DSSE) from conceptual simulation models (CSMs), one of the most efficient and easiest solutions is to map CSMs onto an existing simulation environment or a host simulation language. Based on this idea, a simulation developer can exploit available resources (e.g., building blocks or callable functions) similar to the CSMs to develop one's own domain specific simulation environments that provide, e.g., reusable model constructs/functions and their callable libraries. However, this mapping is not as easily done as it may seem. This is because mapping requires not only a common layer for information/knowledge exchange but also a framework and pattern for mapping. Methodologies such as ontology mapping, simulation block building, and visual subnetwork modeling have been applied to develop an integrated approach that facilitates mapping in the modeling and simulation development.

Keywords: Ontological Mapping, Simulation Building Blocks, Subnetwork Modeling

[1] Lecturer, Faculty of Business Administration, Panyapiwat Institute of Management, Tel. 0-2832-0230, E-mail: kittiset@pim.ac.th

Received 12 November 2012; Accepted 28 May 2013

1. Introduction

Recently, domain specific simulation environment (DSSE) approach has played a key role in enhancing modeling and simulation development performances in terms of reusability and accessibility [1]. Its foundation consists of three major aspects: structure, content, and simulation environment application. First, the structure of the DSSE can be laid out by using the conceptual simulation modeling (CSM) approach to generate a blueprint, describing physical and behavioral characteristics of both reality and simulation domain. CSM also delivers a communication tool for domain experts and simulation developers to support their collaborations via its products such as static/dynamic components, functional layers, and knowledge representations (as well seen in ISAP - developed by Setavoraphan and Grant) [2]. However, CSM itself is not complete enough to develop DSSE because all the details of needed simulation components cannot be included within this kind of knowledge-based representations–using only symbols, notations, and diagrams.

To avoid inconsistency and invalidity in using conceptual simulation models, those symbols, notations, and diagrams need to be transformed into a contextualized document. This process aims to retrieve the appropriate simulation contents from CSMs - considered as the second major aspect of the DSSE development. Making contextualized documentation helps not only eliminate irrelevance but also improve the semantics of the contents [3]. To conduct such a transformational document, the Semantic Web terminology has been used with the exploitation of an ontological analysis approach, including a common language (e.g., XML) and an integrated descriptive and behavioral language (e.g., SRML). This contextualized documentation later plays a critical role together with CSMs in developing DSSE.

In the M&S terminology, DSSE is determined as a simulation environment application that provides reusable simulation model constructs to represent domain specific system elements. Basically, a DSSE application can be developed under: a) an original simulation environment where everything is uniquely created; or b) an existing simulation environment where the availability of resources exists for accessing and utilizing by the simulation developers. Both conditions have advantages and disadvantages. To satisfy the final aspect of the development of the DSSE application, the choice of selection should be based on not only maximizing productivity and efficiency but also minimizing cost and time consuming. This statement can be implied with the assumption that the simulation developers have already owned any of existing simulation environment applications for generic uses such as AweSim [4]. Theoretically, it is possible to transform the simulation environment application to be a DSSE application by mapping the structure and simulation contents into those available resources, so that the resources are caused to function as defined in specifications. Mapping, thus, is seen as the key solution in developing a DSSE application under the existing simulation environment.

Mapping is taken into consideration to enable individuals to keep their own world views and at the same time to share knowledge across domains [5]. By a means of sharing knowledge, mapping is required to deal with semantic interoperability, the issue of allowing the exchange meaningful information/ knowledge between applications/ domains [6]. However, the problem here is that the representations of information/knowledge in each domain are depicted in different ways. To solve this problem, one needs to view the information/ knowledge representations in the framework of an ontology which provides a joint terminology and frame of reference of specifications and semantics of conceptualization. The use of ontologies helps create a common layer where the conceptualization of information/knowledge can be transformed and categorized into a set of standard representation elements such as entities, properties, and relations. When the information/knowledge from alternative domains is agreed in such a sense of semantic similarities, it allows the individuals to recognize what to be mapped. However, to have the efficient and correct exchange of information/knowledge between domains through mapping, the framework and pattern of mapping must be clearly specified.

As described above, mapping by a means of ontology mapping is seen as a solution to facilitate the exchange information/knowledge between domains. There are a number of articles in the literature focusing on research in ontology mapping in different areas to provide definitions, techniques, algorithms, and representations of mapping [5],[7],[8]. The concepts retrieved from the literature are used to support creating a specific framework and pattern of mapping to match the characteristics and requirements of the materials at hand (e.g., CSMs, contextualized documentation, and simulation environment application). This means that mapping is not just about sharing information/ knowledge between different ontologies, but it also includes the levels of similarities in many aspects (e.g., structure, contents, and representations). Therefore, the methodologies of simulation block building and visual subnetwork modeling have been integrated into ontology mapping to construct a solid and robust framework and pattern of mapping between conceptualization and a simulation environment application-to generate a DSSE application.

2. Key Concepts

This section is aimed to provide a survey of the key concepts found in a number of articles in the literature related to ontology mapping, simulation block building, and visual subnetwork modeling. There is a particular purpose lying within each of these concepts. Ontology mapping is important for transferring simulation contents between domains. Simulation block building is applied to design the structure of the mapping destination that will encapsulate the simulation contents being used in a simulation environment application. Finally, visual subnetwork modeling provided in Visual SLAM [9] and AweSim is used as a demonstration how to configure and enforce the simulation building blocks to function.

Moreover, the selection of definitions, methods, techniques, and tools provided for the methodologies has been made based upon the overall framework designed for the development of a DSSE application. This framework is expected to arrange and control the similarities of structure, simulation contents, and simulation environment applications for both conceptualization and application. The idea behind the framework of similarity is to eliminate complexity, irrelevance, and inconsistency in transferring semantics of information/knowledge during the transitions of representation formats from one to another.

2.1 Ontology Mapping

"An ontology is an explicit, formal specification of a shared conceptualization of a domain of interest" [10]. The purpose of using a terminology of ontology is to *"reduce or eliminate conceptual and terminological confusion among the members of a user community who need to share various kinds of (electronic) documents and information"* [11]. To accomplish this purpose, a set of relevant concepts (e.g., entities, instances, relations, and properties) that characterize a given domain needs to be identified and defined properly, which later becomes a mutual point of interest for mapping of two ontologies.

According to Rahm and Bernstein [12], an ontology mapping process is defined as a set of activities required to transform instances of a source ontology into instances of a target ontology. For a clearer picture, Ehrig and Staab [13] define the term of ontology mapping: *"Given two ontologies O_1 and O_2, mapping one ontology onto another means that for each entity (concept C, relation R, or instance I) in ontology O_1, we try to find a correspond entity, which has the same intended meaning, in ontology O_2."* Also, ontology mapping can be defined in different terms such as alignment, merging, articulation, fusion, integration, morphism, and so on, depending on the application and intended outcome [8].

Recently, there are numerous frameworks that provide a methodological approach to ontology mapping. For example, a cooperative framework for integrating ontologies described by Breis and Bejar [14] is a system having the algorithm that supports the integration by using taxonomic features and synonymous concepts in the two ontologies. Madhavan et al. [15] develop a framework that enables mapping between ontologies in different representation languages without first translating the ontologies into a common language. A framework for ontological structures to support ontology sharing, namely IFF, is proposed by Kent [16], which represents ontologies as logics and ontology sharing as a specifiable ontology extension hierarchy. Among the frameworks available up to date, there exists a set of commonalities in their approaches and processes to ontology mapping. These commonalities are assembled and identified in the MAFRA conceptual framework [17], providing the critical clues that lead to the possibilities for mapping between conceptualization and a simulation environment application.

Maedche and Staab [18] develop MAFRA as a mapping framework for distributed ontologies in the Semantic Web. MAFRA is built on the idea that mapping existing ontology will be easier than creating a common ontology. This is because only a smaller community is involved in the mapping process. Also, this framework aims to detect similarities of entities contained in two different ontologies-being the critical mechanisms of this mapping framework. Thus, the framework of MAFRA discovery reveals the essential modules that support the exploitation of semantic similarities in ontology mapping, as described in follows:

• *Lift and Normalization*: The main purpose of this module is to raise all data to be mapped onto the same representation level, which copes with syntactical and structural language heterogeneity [19]. Maedche and Staab [18] states that *"both ontologies must be normalized to a uniform representation, ..., thus eliminating syntax differences and making semantics differences between the source and the target ontology more apparent."*

• *Similarity*: This module aims to support mapping discovery by establishing similarities between entities from the source and target ontology. The mapping approach is based on different similarity measures, which have been proposed in the literature by Rahm and Bernstein [12], Doan et al. [20] and Maedche and Staab [18].

• *Semantic Bridging*: This module is responsible for establishing correspondence between entities from the source and target ontology based on the similarities found between them. The goal in specifying the semantic bridge ontology is to maintain and exploit the existent constructs and minimize extra constructs [18].

• Referring to the concepts of these modules, it can be concluded that the key role in conducting ontology mapping is the ability to establish similarities between the source and target ontology and to specify a common representation framework (or space) for mapping these similarities. In Section 3, a work of ontology mapping based on the similarity-centric approach is demonstrated.

2.2 Simulation Block Building

The BETADE program at Delft University of Technology, Netherland, has been applied to support distributed working, designing, and modeling in order to construct distributed applications or models entirely out of reusable building blocks. The working definition used in the BETADE research program is [21]:

"A building block is a self-contained, interoperable, reusable and replaceable unit, encapsulating its internal structure and providing useful services or functionality to its environment through precisely defined interfaces. A building block may be customized in order to match the specific requirements of the environment in which it is 'plugged' or used."

To apply a building block in such an environment, it needs to clarify the relationship between a building block and components or other related terms. Verbraeck et al. [21] states: *"A component is the implementation of a building block in a software environment. The interface (functionality) of the building block and the component are therefore different presentations of the same thing."* The authors also argue that a building block on its own-without domain specific context, communicating, co-operating or even competing with other building block-cannot provide the functionality a user requires.

In order to provide functionality, an application is constructed by an aggregate of more than one building block, where lower level building blocks are combined into new higher level building blocks and interact each other to function as the user specifies. The aggregation of building blocks is based on the object-oriented paradigm in order to support reusability of building blocks in applications or models, which can be illustrated as in Figure 1. It is seen that a set of building blocks consists of model building blocks (1st level) that are constructed of building block elements (2nd level) [22]. Each building block element communicates using a standard interface used for formal entries for messages and entity passing and represents a specific functionality. Therefore, different kinds of model building blocks can be designed and constructed by using different building block elements.

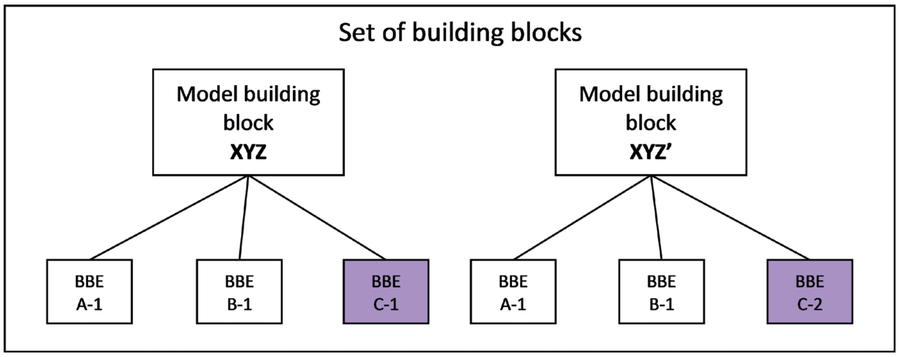

Figure 1 Example of a set of model building blocks using building block elements [23].

Recently, building blocks have been applied in simulation studies to support the development of discrete event simulation models (see examples in Verbraeck and Versteegt [24], Valentin [25], and Saanen [26]. Similar model constructs are used over and over again, especially when developing different simulation models within the same domain. In this case, it seems very logical to structure and package the repetitive model constructs into building blocks and make them available for the modeler for repeated use [25]. Moreover, these building blocks can be collected together in categorized libraries that emerge a set of specific vocabularies for future callable references in simulation modeling-which later possibly turns into a domain specific simulation language (DSSL).

The main purpose in establishing a DSSL is to define and specify semantics, relations and constraints associated with those domain concepts in a representation format of domain specific simulation elements–being used as interfaces of communication. Like a natural language, when the popularity of using and creating domain specific simulation elements within DSSL increases, it is able to create its own simulation environment (community) to support the development of simulation models (communication) using those elements (vocabularies). This kind of simulation environment can be determined as a domain specific simulation environment (DSSE). Following this logical reasoning, it can be concluded that building blocks play a critical role in the development of DSSE by a means of structuring, defining semantics, and configuring functionalities.

According to Snowdon et al. [27], building blocks must support easy model development, which means a set of building blocks can be viewed as a conceptual model that can be reused and directly/easily transferred to the simulation model. The statement leads this research study to find out a connection between conceptualization and simulation, in which a building block becomes a data bridge of two different domains. In addition to the characteristics as of DSSL, the building block is also capable of communicating and translating concepts either implicitly or explicitly of its environment. This allows the building block to merge with other existing DSSLs by customization/configuration to match the specific requirements of the new environments. As the result, it is not necessary to think of the development of building blocks in a traditional way-which starts from a scratch.

The literature by Verbraeck and Valentin [28] shows that building blocks can be developed in existing simulation environments such as Arena, Automod, eM-plant and Taylor ED, whose system architecture is based on object orientation. The authors also define the characteristics of simulation building blocks used in the existing simulation environments: *"Building blocks can range from just one very basic functionality (like executing a simple function) till very complex building blocks with hundreds of functionalities, no matter what kind of problem, and no matter how large the set of available building blocks is."* Their statement substantiates that such an object-oriented simulation environment is fit best with the characteristics of simulation building blocks since it provides the most important mechanisms-composition A simulation building block can be composed of either basic simulation building block elements (e.g., create, queue and resource) or customized/specialized simulation building block elements (e.g., conveyor, crane, and AGV) in a specific way. However, configuration of the building block is still needed to have well defined interface for connecting it to other building blocks and to fit in its environments. Therefore, experiences and knowledge in employing specific simulation environments are critical to succeed in composing simulation building block elements and connecting them together to function.

Based on the concepts described through this subsection, the author is able to make a hypothesis that we have explored the successful development of simulation building blocks using an existing object-oriented simulation environment. Visual SLAM and AweSim, thus, have been chosen as a simulation modeling language/environment. A brief discussion of Visual SLAM and AweSim, including its feature that supports building simulation blocks, is given in the next subsection.

2.3 Visual Subnetwork Modeling

Prior to have a better understanding why Visual SLAM and AweSim have been selected for this demonstration, it is necessary to recognize the idea behind the development of this simulation modeling language/environment. Pritsker and O' Reilly [29] provides an explanation related to their perspectives and essential concepts for building blocks:

"For many years, it has been desired to develop a modeling language that is modular and hierarchical. Modularity would allow submodels to be developed and used as building blocks for a total systems model. Hierarchical models would display the aggregate features of a model and allow the details to be viewed by driving the view to less aggregate displays of the model. These properties would allow for the building of submodels by team members which then could be integrated into a system model. To achieve these capabilities, modeling languages were designed using object-oriented concepts."

This idea, thus, becomes a fact that the object orientation is taken in Visual SLAM. It aims to employ object-oriented concepts and coding within the network worldview (or objects) of the Visual SLAM simulation language. In the meantime, AweSim is a simulation problem-solving environment for Visual SLAM, providing extensive input, output, and integration capabilities to facilitate the use of Visual SLAM by users. For the development of simulation building blocks, both Visual SLAM (modeling language) and AweSim (mechanisms/environment) are needed.

Since Visual SLAM employs object-oriented concepts, it allows for defining a subnetwork as an object class. An entity is routed to the subnetwork for a particular instance of that object class. For example, different machines that perform similarly to process parts can be modeled as a subnetwork. The subnetwork for the processing to be done by the specific machine is modeled by Visual SLAM network elements and by passing parameters to define the node and activity characteristics for the subnetwork instance. Because of the object nature of a subnetwork, it can be referred to as a visual subnetwork or VSN.

Subnetworks always contain two important modeling aspects: modularity and hierarchy. Each subnetwork encapsulates the data (e.g., subnetwork variables, entity attributes and parameters) in such a way that a self-contained block/module is created and is able to connect with other subnetworks Modularity allows for the subnetwork to be reused in different locations within a large network model and to be built for use by other modelers. Moreover, for the hierarchical modeling aspect, entities are transferred from a calling network to a subnetwork. The calling network can be the main network or a subnetwork that is one level higher than the subnetwork that it calls. The hierarchy is also related to the different levels of detail specified in each subnetwork.

When looking at these capabilities, there appear similarities between VSNs and simulation building blocks. The similarities by a means of the object-oriented world view facilitate not only information mapping but also physical and behavioral modeling to develop and use simulation building blocks through VSNs. Therefore, in practice, a VSN can be generated as a model building block, whereas a network node/branch in Visual SLAM network model can be used as a building block element.

Moreover, the construction of VSNs is supported by mechanisms and tools available in the simulation environment, for example, AweSim. As a result, the simulation developers do not need to worry whether the VSNs match with the requirements of their environment. In another case, the simulation developers are also able to customize a VSN in order to function as they require by creating user-written functions (user-codes) and use them via the interface points (in AweSim, which are called EVENT and ENTER nodes). This available feature provides complete modeling flexibility for the configurations of VSNs. The simulation building blocks created in Visual SLAM will be maintained in AweSim's libraries, which allow the simulation developers to reuse, modify, add, and delete those for the future simulation projects under the same domain. Later, the network libraries become vocabularies that are used only in a domain specific simulation, which automatically creates a simulation environment that supports modeling specific problems.

For the details of the syntax and semantics associated with VSNs, network nodes, and user-written functions, see Simulation with Visual SLAM and AweSim, The User Manual Guide, and Visual SLAM Quick Reference Manual by Pritsker and O' Reilly [29].

3. Mapping CSM with Visual SLAM

The concepts of ontology mapping, simulation block building, and visual subnetwork modeling are integrated as a paramount approach for the development of a domain specific simulation environment (DSSE). The approach is later implied with the approaches of conceptual simulation modeling and contextualized documentation. This implication turns into an output of mapping a conceptual model collaborated with its contextualized document into components available in an existing simulation environment application like AweSim with respect to its simulation requirements and constraints. In this paper, the demonstration is expected to briefly provide an overview of mapping ontologies and building simulation blocks, using VSNs available in AweSim.

As discussed earlier, this study employs a similarity-centric approach to perform ontology mapping between conceptualization and simulation. Using this approach, the simulation developers must be able to establish similarities between the source ontology (e.g., CSMs) and the target ontology (e.g., Visual SLAM) and specify a common representation framework/space for mapping these similarities However, it must be understood by simulation developers that similarity mapping concerns not only the concepts to be mapped but also the structure to be generated (for encapsulating the concepts). The mapping between CSM and Visual SLAM, thus, can be referred to the transformation of both structure and semantics of Simulation Modeling Units, SMUs [30] into VSNs (by a means of simulation building blocks).

Both SMUs and VSNs are considered as objects having the aspects of modularity and hierarchy, which can be constructed as building blocks at different levels of detail. Therefore, it is critical to limit the detailed levels of their structures before mapping their concepts. Refer to the structure of building blocks, there are only two levels: model building blocks (1st level) and building block elements (2nd level). Technically, model building blocks should be a direct translation of the defined instances from CSM (e.g., SMUs) because the model building blocks represent the world-view of the domain expert in terms of standard functionalities used/reused in reality. Meanwhile, building block elements are deterministically used as either internal or external functionalities of the model building blocks, providing the semantics of their construction and interconnection to match the user requirements.

When considering and comparing the structures of SMUs and VSNs, it seems possible to perform one-to-one mapping between them to generate model building blocks and building block elements with respect to the levels of detail. In general, an SMU can be viewed as a VSN as well as a model building block, whereas the operations within the SMU can be transformed into a set of Visual SLAM network nodes that perform as building block elements. For the structure mapping, however, it is natural to refine the structure of CSMs by adding details (e.g., tools or structures) for implementation in order to link them together for testing in a host simulation language. In this case, it is not always necessary that the results of mapping between CSMs and Visual SLAM will follow the same pattern previously described. This is because the status of being either a model building block or a building block element of SMUs/VSNs is depended on the correctness of implementation. A set of SMUs are only used as a core design for the development of simulation building blocks, while adding the details for implementation to create VSNs or network nodes is relied on the determination of the simulation developers. The structure mapping, therefore, becomes a more or less abstraction issue for future discussion.

To handle the problem caused by the structure mapping, a critical support role in this similarity-centric approach is taken by ontology mapping. This study is set to apply the framework of MAFRA (Maedche and Staab [17] to deal with semantic similarities of the concepts/ontologies between two domains. Focusing on similarities of semantics rather than those of structures is helpful for making decision not only in proposing candidates for mapping but also in adding details for implementation (if necessary). As a result, the simulation developers are able to determine which candidates can be used, what levels of detail (e.g., model building block or building block element) they can be constructed and how they can be linked together for testing. This provides flexibility in building simulation blocks on the host simulation language like Visual SLAM. However, the MAFRA mapping framework contains some requirements prior to map semantic similarities between ontologies.

The requirements have been described in terms of modules which include lift and normalization, similarity and semantic bridge. The main objective of these modules is to facilitate defining and establishing a specific framework that fits the surroundings of both source ontology and target ontology. There are three critical conditions to keep in mind. First, both ontologies must be normalized to a uniform representation (or the same representation level). Second, similarities between entities from the source and target ontology must apparently be established. Third, there must be a space for the similarities of two ontologies to be mapped. Following these conditions helps the author to develop a tool, called Similarity Mapping Plane (SMP), to support ontology mapping between conceptualization and simulation.

Prior to exploit SMP, it is important for the simulation developers to be able to specify what source (ontology) to be mapped. On the other hand, it means how to exchange information/data between ontologies. To deal with the information/data exchange, the first step is to categorize the information/data into generic formats-which are object and content formats-since the target of mapping, obviously, is simulation instances and simulation contents. The object format includes SMUs, VSNs, and network nodes, whereas the content format contains descriptions, properties and so on.

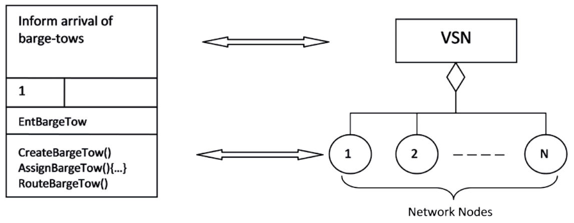

Figure 2 An example of mapping between an SMU and a VSN.

The next step is to select mapping objects. For example, SMU Inform Arrival of Barge-tows is selected as an object to be constructed as a VSN. Meanwhile, its operations such as CreateBargeTow(), AssignBargeTow() and RouteBargeTow() are also set as objects to be transformed into network nodes (as shown in Figure 2). In practice, building a VSN begins with defining and specifying a set of network nodes and all the required parsing parameters. This allows the simulation developers to perform mapping at the second level (operations ↔ network nodes) prior to reach the first level (SMUs ↔ VSNs).

After the mapping objects have been selected, one or more candidates must be nominated from the target of mapping per a selected mapping object by briefly scanning if there is any semantic similarity by a means of functionalities. This would be the easiest way to obtain a number of candidates. It seems to be a time-consuming activity and to require experiencing in the host simulation language-to do manually searching. At this rate of searching, each candidate is placed onto SMP as well as the selected mapping object. However, both of them cannot be completely mapped until their contents are provided in the same representation level of similarities to be supportive decisions.

In order to normalize the contents of two ontologies into a uniform representation, a framework of similarities must be drawn from the contents available in the target of mapping. The contents of the Visual SLAM network nodes, including Visual SLAM (e.g., support, user-written, and user-callable) functions, are represented in terms of descriptions, statement forms, input listings and specifications within documentation (as seen in the documents for Visual SLAM and AweSim by Pritsker and O’ Reilly) All the contents described in the documentation are determined as entities of the target ontology (as stated by Maedche and Staab [17] With these entities, the simulation developers are able to establish a scope of the similarity framework to seek for similar entities in the source ontology. However, it needs to be noted here that the representation of those entities in the source ontology must be on the same level as well as documentation.

When considering the source ontology to be mapped, it is found out that the entities retrieved from CSMs are also represented in a context of documentation as well. For example, there appear the entities like description, property, input statement, and function for CreateBargeTow() provided in descriptive tables, network statements, and its contextualized documentation. The entities from the source and target ontology, therefore, become normalized on the documentation basis and ready for selection. The selection of entities is made on the following criteria:

• The entities must clearly represent the main contents of both ontologies.

• The entities must be included in both ontologies and can be matched up directly.

• The entities must provide details that facilitate the simulation developers to measure or weigh the degrees of similarity between entities.

For the above example, there is only one candidate to be mapped with CreateBargeTow(), which is the CREATE node. Table 1 shows how to construct SMP and to weigh the degrees of similarity between entities from the source and target ontology.

Apparently, the degrees of similarity can be weighed in a sense of scoring numeric evaluation. The range of evaluation might be varied, depending on individuals’ judging criteria and experiences. However, to make this mapping example easy to understand, the evaluation is set at three different degree levels of similarity with numeric scores in (): none (0), likely similar (1), and similar (2). The scores given for each entity are then summed up at the bottom of SMP. The more total score is; the higher possibility of mapping is.

Moreover, the total score can be used as an indicator to consider if the target ontology needs to be added by any other details for implementation. If so, there are three methods for the simulation developers to add those details. The first method is to edit the structure of the target ontology by adding partial specific functionalities to the origin (e.g., modification of the network nodes). The second method is to recreate the target ontology by making a new

Table 1 Similarity Mapping Plane for CreateBargeTow()

Source	**Weight**	**Target**
<u>Instance</u> Name: CreateBargeTow()	1	<u>Node</u> Name: CREATE
Description A barge-tow entity is created by a mean of containing a set of barges and a tow boat.	2	Description Entities are generated within the network.
Properties :First arrival :Arrival rate :Current time :Max# entities	1	<u>*Inputs*</u> :Time between creations (TBC) :Time of first creation (TF) :Maximum creations (MC) :Mark variable which will store the time of creation (MV) :Number of branches (M)
<u>*Input statement*</u> CreateBargeTow, First arrival, Arrival rate, Current time, Max# of signal entities;	1	<u>*Input format*</u> CREATE, TBC, TF, MV, MC, M;

<u>*Explanation*</u>
function CreateBargeTow()
{
  var FirstArrival = 0;
  var ArrivalRate;
  var CurrentTime = TNOW;
  var MaxEntities;
 	
 //Create a new entity
 	
  Set NewEntBargeTow = IP.NewEntity();
  Set NewEntBargeTow.ArrivalTime = IP.TNOW;
  IP.Schedule("FirstArrival", NewEntBargeTow, (NewEnt-BargeTow.ArrivalTime+ArrivalRate);
 	
 //Schedule the next entities
 	
  for (i=1; i<=Max# entities; i++)
  Set NextEntBargeTow = IP.CloneEntity();
  Set NextEntBargeTow = IP.TNOW;
  IP.Schedule("NextArrival", NextEntBargeTow, (NextEntBargeTow.ArrivalTime+ArrivalRate));
 	
}
	1	<u>*Explanation*</u>
		CREATE NODE
		:The first entity is created at a time specified by the value of TF;
		:The time between creations of entities after the first is specified by the variable TBC;
		:The time at which the entity is created can be assigned to a variable MV;
		:Entities will continue to be created until a limit is reached, specified by MC

Total scores	6/10	Likely similar

Weight by degrees of similarity (score): None (0); Likely similar (1); and Similar (2).

complete set of specific functionalities (e.g., simulation programming for the functionalities). Finally, the third method is to add one or more extra extensions to the target ontology for being used as supporting roles (or surroundings). This happens when the target ontology cannot be self-contained enough to correspondence of the contents of the source ontology to be mapped or to developing itself into a stand-alone instance (e.g.,

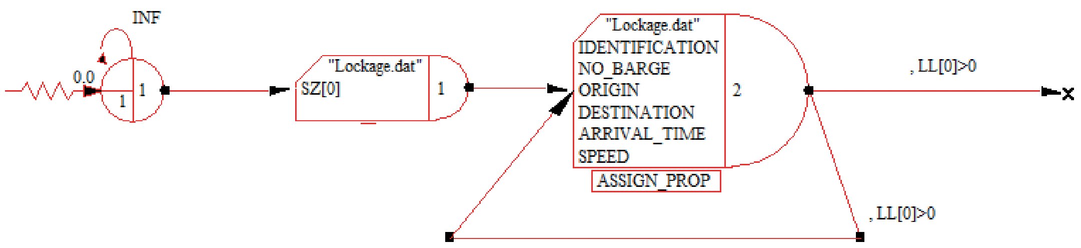

Figure 3 A set of the Visual SLAM network nodes that represent SMU *Inform Arrival of Barge-tow.*

network node or functional module). It can be seen in many cases that to represent a true semantic of a specific function requires a composition of a set of detailed functions; for instance, a PROCESS function is composed of SEIZE, DELAY and RELEASE function. Adding the details for implementation, the simulation developers need to closely collaborate with the simulation experts for advising and revising in their works.

In addition to facilitate the similarity mapping, SMP also provides an extra joint entity, called Explanation, to support revealing the true semantics of simulation functionalities for both ontologies. Explanation entity in the source ontology is derived from its contextualized documentation, whereas the one in the target ontology may be retrieved either from the documents for the host simulation language or from the simulation developers' understanding. Since most of the target ontologies are the Visual SLAM network nodes which lack detailed explanation how they function, the simulation developers are required to test each of them to have recognition of their functionalities and usage.

As a result, the simulation developers can describe these Visual SLAM network nodes in terms of basic function procedures (or processing steps). For other cases such as having simulation functionalities already described by the documents or specifically created by the simulation developers, they can be directly put onto SMP for comparing similarities with those descriptive functions from the contextualized documentation. Providing the Explanation entity is very helpful not only for completing the ontology mapping between conceptualization and simulation but also for making a final decision whether to utilize the target ontology in developing a simulation building block. Figure 3 illustrates the output of ontology mapping from SMP.

4. Conclusions

This study is aimed to propose the integrated approach that facilitates the development of a domain specific simulation environment (DSSE). Use of the DSSE may be for one time use (application) or multiple uses (language like). The integrated approach consists of the methodologies of ontology mapping, simulation block building and visual subnetwork modeling, which also collaborates with the conceptual simulation modeling (CSM) and transformation approach. The idea behind the integrated approach is to design a framework and pattern of mapping between the structures and contents derived from conceptualization and an existing simulation environment application/host simulation language. This brings the simulation developers the abilities to create their own simulation environments to support simulation modeling for a specific problem domain. Not only flexibility in modeling will their DSSEs provide but also reusability in generating simulation models related to the domain. The integrated approach has been applied into the development of, for example, a DSSE for an inland lockage operations system on the McClellan-Kerr Arkansas River, USA [1] Although it practically woks in the pilot case-study, the evaluation of the approach has yet been seriously done by public at the moment. The levels of user satisfaction and effective collaboration, thus, are mainly focused in the future case studies.

Basically, the idea of mapping and simulation block building has not received extensive attentions from most simulation language developers. It seems difficult and complicated for them to begin with laying out the structures by using conceptual simulation models, defining and specifying the simulation contents by making contextualized documentation, and encapsulating and mapping those information/data into a module by constructing simulation building blocks. Often, they prefer to use logical models and jump right away to develop simulation models. Programming is also another choice of their preferences for modeling simulation for their particular problems. It might be a comfort zone for them to deal with simulation modeling.

Another reason is that to obtain good production from using the integrated approach is depended on how detailed the source ontology (e.g., CSMs and contextualized documentation) can be and how much expertise in simulation modeling (including simulation environment applications and languages) the simulation developers have. This is a big barrier that not only blocks them from using the approach effectively and efficiently but also enforces them to deny involving with it at the end. Moreover, there is still room for improvement of the representations and applications to gain more insights and effectiveness to the approach in terms of details for implementation. It is found out that the simulation developers are lost in translation and unable to link the elements for testing. To reach the optimum goal for this research study; therefore, it is critical to have collaborations from individuals in different major areas such as domain experts, simulation experts, software engineers and computer programmers. The author personally believes that this approach can be developed as a fundamental applied for the Modeling and Simulation (M&S) communities.

References

[1] K. Setavoraphan, "An Application of Domain Specific Simulation Environment for An Inland Lockage Operations System," in *Proceedings of the 11th International Conference on Industrial Management*, 2012.

[2] K. Setavoraphan and F. H. Grant, "Conceptual simulation modeling: The structure of domain specific simulation environment," in *Proceedings of the 2008 Winter Simulation Conference*, 2008, pp. 975-986.

[3] K. Setavoraphan and F. D. Grant, "Contextualized Documentation: A Method for Transformation of Conceptual Simulation Modeling," in *Proceedings of The Third International Conference on Information and Communication Technology for Embedded Systems*, 2012.

[4] AweSim: Total Simulation Project Support,

User's guide. Version 3.0. Symix Systems, 1999.

[5] M. Ehrig and Y. Sure, "Ontology mapping-an integrated approach," *In ESWS 2004*, pp.76-91, 2004.

[6] P. Bouquet, B. Magnini, L. Serafini and S. Zanobini, "A SAT-based algorithm for context matching," in *IV International and Interdisciplinary Conference on Modeling and Using Context*, Standford University, California, USA, 2003.

[7] D. Marques, (2008, October 5). "A survey of recent research in ontology mapping." 2005. [Online]. Available: http://sfu.ca/~mhatala/iat881/2005/DM-OntologyMapping.pdf>

[8] Y. Kalfoglou and M. Schorlemmer, "Ontology mapping: the state of the art," *The Knowledge Engineering Review*, vol.18, no.1, pp.1-31, 2003.

[9] Visual SLAM, *Quick reference manual*, Version 2.0. Pritsker Corporation, 1997.

[10] T. R. Gruber, "Towards principles for the design of ontologies used for knowledge sharing," in *Formal Ontology n Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, Kluwer Acadmic Publishers, 1993.

[11] R. Navigli, P. Velardi, and A. Gangemi, "Ontology learning and its application to automated terminology translation," *IEEE Intelligent Systems*, vol.18, no.1, pp.22-31, 2003.

[12] A. Rahm and A. Bernstein, "A survey of approaches to automatic schema matching," *The Very Large Databases Journal*, vol.10, no.4, pp.334-350, 2001.

[13] M. Ehrig and S. Staab, "QOM-Quick Ontology Mapping," in *International Semantic Web Conference*, 2004, pp.683-697.

[14] J. F. Breis and R. M. Bejar, "A cooperative framework for integrating ontologies," *International Journal of Human-Computer Studies*, vol.46, no.6, pp.707-728, 2002.

[15] J. Madhavan, P. A. Bernstein, P. Domingos, and A. Halevy, "Representing and reasoning about mappings between domain models," in *Proceedings of the 18th National Conference on Artificial Intelligence*, Edmonton, Alberta, Canada, 2002.

[16] R. Kent, "The information flow foundation for conceptual knowledge organization," in *Proceedings of the 6th International Conference of the International Society for Knowledge Organization (ISKO)*, Toronto, Canada, 2000.

[17] A. Maedche, B. Motik, N. Silva, and R. Volz, "Mafra: A mapping framework for distributed ontologies," in *Proceedings of the 13th European Conference and Knowledge Engineering and Knowledge Management*, Madrid, Spain, 2002.

[18] A. Maedche and S. Staab, "Semi-automatic engineering of ontologies from texts," in *Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering*, Chicago, IL, USA, 2000.

[19] P. R. S. Visser, D. M. Jones, T. J. M. Bench-Capon, and M. J. R. Shave, "An analysis of ontology mismatches: Heterogeneity versus interoperability," in *AAAI 1997 Spring Symposium on Ontological Engineering*, Stanford, CA, USA, 1997.

[20] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, "Learning to map between ontologies on the semantic web," in *Proceedings of WWW-2002, 11th International WWW Conference*, Hawaii, 2002.

[21] A. Verbraeck, Y. Saanen, Z. Stojanovic, E. Valentin, K. van der Meer, A. Meijer, and B. Shishkov, "Chapter 2: What are building blocks", in *Building blocks for effective telematics application development and evaluation*, A. Verbraeck, A. Dahanayake (Eds.),Delft University of Technology, ISBN 90-5638-092-3, 2002, pp. 8-21.

[22] A. Verbraeck and E. Valentin, "Simulation building blocks for airport terminal modeling," in *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yucesan, C. –H. Chen, J. L. Snowdon, and J. M. Charnes. pp. 563-571. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc., 2002.

[23] E. C. Valentin and A. Verbraeck, "Guidelines for designing simulation building blocks," in *Proceedings of the 2002 Winter Simulation Conference,* ed. E. Yucesan, C. –H. Chen, J. L. Snowdon, and J. M. Charnes. pp. 563-571. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc., 2002.

[24] A. Verbraeck and C. Versteegt, "A bridge between the design and implementation of complex transportation systems-Linking simulation models and physical systems," in *ESS 2000-Simulation in Industry*, Dietmar P. F. Moller (Ed.), Ghent: SCS Publications, 2000, pp. 238-243.

[25] E. C. Valentin, "Chapter15: Building blocks for modeling of passengers at airports," in *Building blocks for effective telematics application development and evaluation*, Delft University of Technology, Netherlands, 2002.

[26] Y. A. Saanen, "Chapter16: The application of advanced simulations for the engineering of logistic control systems," in *Building blocks for effective telematics application development and evaluation*, A. Verbraeck, A. Dahanayake (Eds.), Delft University of Technology, Natherlands, 2002.

[27] J. L. Snowdon, S. El-Taji, M. Montevecchi, E. MacNair, C. A. Callery, and S. Miller, in *Proceedings of the 1998 Winter Simulation Conference*, 1998.

[28] A. Verbraeck and E. Valentin, "The use of building blocks to enhance flexibility and maintainability of simulation models and simulation libraries," in *Proceedings ESS'2001-13th European Simulation Symposium 2001*,N. Giambiasi, C. Frydman (Eds.), 2001, pp. 973-979.

[29] A. A. B. Pritsker and J. J. O'Reilly, *Simulation with Visual SLAM and AweSim*, 2nd ed. New York: John Wiley & Sons, 1999.

[30] K. Setavoraphan, "Simulation Modeling Unit: A Bridge Cross-Domain Communication in Building and Developing Simulation," *Panyapiwat Journal*, vol.3, no.2, 2012.