Second LACCEI International Latin American and Caribbean Conference for Engineering and Technology (LACCET'2004)
"Challenges and Opportunities for Engineering Education, Research and Development"
2-4 June, 2004 Miami, Florida, USA

SCOPE: A Conceptual Framework of Ontology-based Program/Project Scope Control

Yimin Zhu, Ph.D.
Assistant Professor, Florida International University, Miami, FL 33174

Shu-Ching Chen, Ph.D.
Assistant Professor, Florida International University, Miami, FL 33174

Abstract

Program management is one of the major vehicles that many owners use to deliver large-scale capital projects such as airports and highway infrastructure. Such programs usually involve huge amount of investment and multiple years of planning, design and construction. Meanwhile, program management teams are often formed by various stakeholders with different background and interests. In such a dynamic environment, effective sharing and disseminating program information is critical to the success of program management.

This paper argues that one of the major challenges of program control is dealing with project changes within a program, which in many cases is related to project scope evolution and changes especially during the planning and the design phase. Unfortunately, most of the existing AEC (Architecture Engineering Construction) software tools do not well address the issue of managing project scopes and scope changes throughout the lifecycle of a program. In addition, due to the factor that exiting AEC software tools are highly fragmented, managing program resources and schedules in an ever changing environment becomes extremely difficult. Existing manual approaches are inefficient and error-prone. This paper discusses a method that can potentially improve computer-mediated program management tools by developing a program scope management model, which becomes a central point for program control. Through the program scope model, other important data such as program resources and schedules can be aligned for program control purpose.

Keywords

Program management, Scope control, Ontology

1. Introduction

The concept of program management for capital projects has been around for decades already. Although program management principles have been widely accepted and practiced, the formalization of program management as a distinct discipline is a relatively recent event. Lager-scale infrastructure programs usually consist of many concurrent projects, often involving a tremendous amount of investment and lasting for many years. Their economic and social impact is thus extremely significant. One of the major characteristics of program management is to deal with ever-changing program/project scopes, especially during the planning and the design phase since the objectives of those phases are to identify and define program, as well as project, requirements, and subsequently scopes as well. On the other hand, since program cost and durations are a function of program scopes, to finish a program within budget and on-time may simply mean to maintain and control program scope. In the meantime, mission critical data and knowledge about a program have to be aligned to the right scope in order to support successfully executive decision-making. Consequently, it is reasonable to claim that scope control has a critical role in program management.

Typically, a project is defined by two types of scopes, product scope and project scope. The project scope is theoretically a function of the product scope, as well as other factors such as the complexity of the product. Since a program is usually a collection of multiple projects, the success of program scope control thus relies on the effective control of product scopes, as well as other factors affecting the project scopes.

Scope variations take different forms, which include, but not limited to, scope changes and scope transfer. When any of those scope variations happens, project budget, funding, estimates, cost, schedules and resource allocation need to be adjusted accordingly. Program management relies on the data from all those data sources for decision-making, therefore it is imperative to be certain that the scopes of projects in various data sources are aligned.

When software applications are designed and developed, most of them have a narrow and specific focus such as supporting estimating or scheduling. To support specific management functions, different types of tools map user requirements differently and manually. For example, in schedules user requirements are translated into WBSs (Work Breakdown Structures), while in estimates the requirements are represented by SOW (Statement of Work). In design the requirements are not represented in a structured approach as in schedules and estimates. Drawings are more like "What-You-See-Are-What-You-Get." In other words, while there are software tools supporting specific management function, the capturing of scope and user requirements are outside the control of those tools (Figure 1), which relies on human interpretation and control. This separation in realty creates many scope control problems. For example, during the planning and the design phase of a program, scope changes and transfer happen frequently, it is daunting and inefficient for schedulers, estimator and other project professionals to keep track of the scope of each project manually.

Understanding and formalizing the process of deriving activities based on requirements is a challenging task. Currently the process of detailing project activities from owner goals, objectives and requirements is irreversible in that the human-reasoning process of translating requirements into activities and work packages is not a part of computer-mediated processes. For example, when developing schedules, only the activities will be entered into a computer while the reasoning and the rationale between the activities and the requirements/objectives are not understood by the computer. The problem of such reality is that in a complex and dynamic work environment such as program management where responsibilities are highly specialized and software tools are greatly fragmented, completely relying on human beings to coordinate and align program/project scopes in different format, i.e. schedules, designs, estimates and cost analysis and heterogeneous data sources is not an efficient and effective approach. It is thus important to develop a computer interpretable representation that links the details of project execution plan and high-level requirements. In this way, the reasoning process of instantiating activities from user requirements becomes traceable by software. In addition, the differences between various versions of project execution plans can potentially be identified thus the scope mis-match between different format of project execution plans such as those in schedules and estimates can be aligned more efficiently and effectively.

This paper argues that the fact that requirements and scope are managed and interpreted manually has significant negative impact on the effectiveness of data integration and the quality of the integrated data. In the meantime, this project acknowledges that scope is the most elusive and the most dynamic concept in program/project management. A formal and explicit definition of scope has not been given by previous studies, which is another indicator that alternatives need to be researched instead of creating ontology for scope. Therefore, this research will investigate the idea of developing ontology for scoping, a process that links the requirements and the execution details, rather than developing the ontology of scope itself (Figure 2).

This paper further acknowledges that the semantic representation of program/project scoping is just part of a large framework, scoping is a process, involving various program management functions and different stakeholders at different stages. In this paper, the scoping framework, called SCOPE (Scoping Construction Ontology for Program/Project Execution) includes the structural representation of scoping, which represents attributes of a reasoning process, and also the processes involved in scoping and scope evolution. This paper will also discuss methodology of developing SCOPE and use it for scope alignment during the early planning and design phases in program management environment.

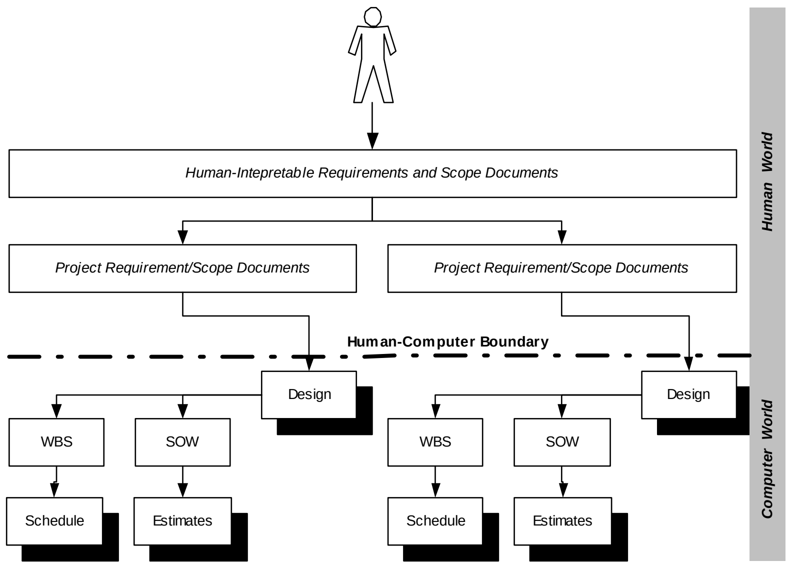
Human-Intepretable Requirements and Scope Documents
Project Requirement/Scope Documents
Project Requirement/Scope Documents
Human-Computer Boundary
Design
Design
WBS
SOW
WBS
SOW
Schedule
Estimates
Schedule
Estimates
Human World
Computer World

Figure 1

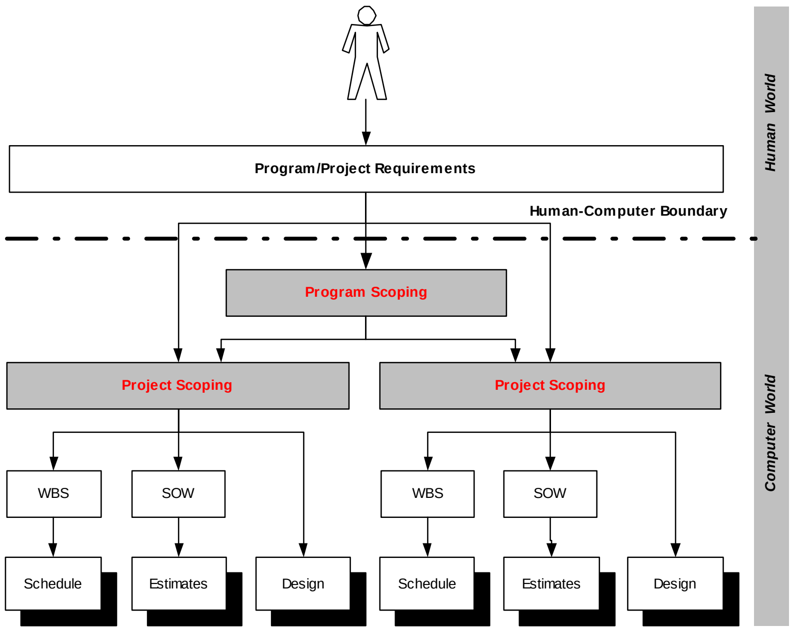

Figure 2

3. Related Studies

3.1 Scope and Scope Management

What is scope? The Merriam-Webster Online Dictionary define the word "scope" as,

1. intention, object
2. space or opportunity for unhampered motion, activity, or thought

3. extent of treatment, activity, or influence
4. range of operation: as a): the range of a logical operator: a string in predicate calculus that is governed by a quantifier b): a grammatical constituent that determines the interpretation of a predicate or quantifier

The definition very subtly indicates two different ways of interpreting scope, i.e. requirements (1) and activities (2 and 3), which may lead to confusions as whether scope is equivalent to requirements or activities that to be implemented to meet the requirements. Such dichotomy has a clear reflection in project scope definitions, i.e. the activities-thinking and the requirements-thinking.

In the construction domain, scope definition has different perspectives (Uppal, 2002)

- Functional scope – product to be made or capacity to be provided
- Technical scope – design philosophy, design standards, and specifications to be to be used
- Physical scope – what is left standing when the project is completed
- Activity scope – the division and responsibility among all entities doing the work, for example the statement that hookup is, but commissioning is not, included in the scope of work.

"Of these conditions, scope is the most pervasive because it influences all of the other categories. It is also the most difficult to measure since it must be expressed in terms completely external to itself…Part of the problem with scope is that it continually redefines itself as one progresses up and down the project tree…" (McCue, 1997) Perhaps the most useful tool at management's disposal is that of "accountability."

3.2 Scope Management

Traditionally, scope management includes processes to define a project that include work required, and only the work required, therefore its main concern is to clearly define and effective control what is or is not included in the project (PMI, 2002).

As shown in Figure 3, the entire process defined by PMBOK includes five stages, initiation, scope planning, scope definition, scope verification and scope change control.

- Initiation - committing the organization to begin the next phase of the project.
- Scope Planning - developing a written scope statement as the basis for future project decisions.
- Scope Definition - subdividing the major project deliverables into smaller and more manageable components.
- Scope Verification - formalizing acceptance of the project scope.
- Scope Change Control - controlling changes to project scope.

The Construction Industry Institute publication report on scope definitions and control is a well-recognized, industry standard document on scope issues. It provides estimators with a background on scope definition problems and addresses specific project areas heavily affected by poor scope definition. This includes project areas such as budget estimate accuracy, bulk quantity control, and Change management (CII, 1986).

Defining project scope is a process of identifying project information, in the format of project scope package, for execution (Dumont et al, 1997). Practitioners have already recognized that the success of the detailed design and engineering heavily depends on the completeness of the scope definition documents. PDRI (Project Definition Rating Index), focusing on the quality and the completeness of a project scope, is a tool created to address the issue of adequately defining project scopes. The tool includes a checklist of seventy weighted scope definition elements allowing users to measure and manage the level of scope definitions as project planning progresses.

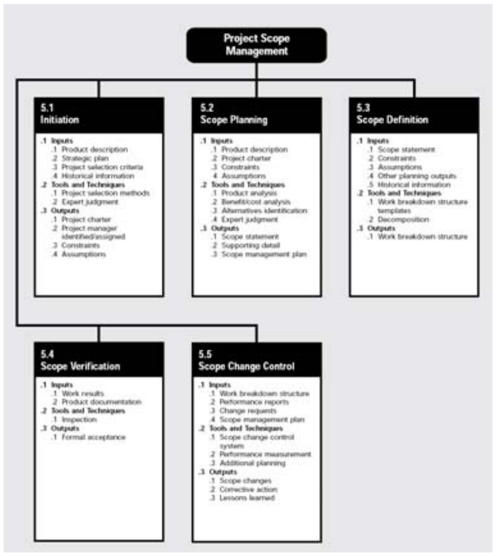

Figure 3

Some other studies focused on developing a methodology for defining the scope of work for large environmental restoration projects (Poldy et al, 1992). Such methodology is based upon an anchored rating scale and scope definition checklists for the assessment and cleanup phases of environmental restoration projects. The scale is a simple and rapid method for measuring the scope of large numbers of projects. The methodology is intended for use at the program management level in developing a better understanding of project scope definition and as a means for calibrating uncertainty surrounding project estimates.

Facility scope is defined as "the performance, time, and cost parameters of the equipment, materials, services, etc. that are acquired by an owner to support the specific objective of a facility project and refer the definition, revision, refinement and control of facility project scope through different project phases as facility scope management." (Huang et al, 1993) The paper discussed how to use knowledge mapping to develop standard scope templates for facility scope management.

4. The SCOPE Framework

SCOPE (Scoping Construction Ontology for Program/Project Execution) is ontology to capture the characteristics of the processes that instantiate activities or work-items to meet the user requirements. Currently, such processes are manual in that they rely on human beings to develop the details of activities and to reason about the scope based on the details. This research intends to facilitate the implementation of computer mediated scope control by developing the ontology so that software tools can keep track of the development of scope along with the detailing of project definitions.

4.1 Ontology

Studies on ontology became increasingly widespread in the field of information systems science. Ontology provides different styles of specifications, i.e. informal or formal, and computationally tractable standardized definitions of the terms used in specific domains, in order to maximize inter-communicability with other domains. The importance of ontology has been recognized in fields such as e-commerce, enterprise and information integration, natural language processing, knowledge engineering, database design, geographic information science, and intelligent information access.

There are many different definitions of "ontology". In philosophy, the word "ontology" means a systematic explanation of Existence. The concept has been used widely in AI (Artificial Intelligence) fields, where ontology is defined as the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary (Neches, et al 1991). This definition bears ambiguity in that ontology includes not only terms that are explicitly defined but also inferred by using the rules. Probably, the most widely used definition of ontology is "ontology is an explicit specification of conceptualization."(Gruber, 1993) The major issue of this definition is that it is based on the formal notion of "Conceptualization" (Genesereth and Nilson, 1987), which uses linguistic terms to denote the relevant relations. Those terms cannot be thought of as mere comments, informal extra-information. Rather, the formal structure used for a conceptualization should somehow account for their *meaning*.

Gruber later explained that "ontologies are agreements about *shared* conceptualizations. Shared conceptualizations include conceptual frameworks for modeling domain knowledge; content-specific protocols for communication among inter-operating agents; and agreements about the representation of particular domain theories. In the knowledge sharing context, ontology is specified in the form of definitions of representational vocabulary. A very simple case would be a type hierarchy, specifying classes and their subsumption relationships. Relational database schemata also serve as ontology by specifying the relations that can exist in some shared database and the integrity constraints that must hold for them."(Gruber, 1994)

In general, "An ontology may take a variety of forms, but necessarily it will include a *vocabulary of terms*, and some *specification of their meaning*. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constrain the possible interpretations of terms." (Uschold, 1998)

Knowledge in ontology is formalized using five kinds of components: classes, relations, functions, axioms and instances (Gruber, 93). Classes in ontology are usually organized in taxonomies, which sometimes are used interchangeably with full ontology (Studer et al., 98)

Principles in developing ontology include

- Clarity and Objective (Gruber, 93) – the ontology should provide the meaning of defined terms by providing objective definitions and natural language documentation
- Completeness (Gruber, 93) – a definition expressed by a necessary and sufficient condition is preferred over a partial definition (defined only by a necessary or sufficient condition).
- Coherence (Gruber, 93) – to permit inferences that are consistent with the definitions.
- Maximize monotonic extendibility (Gruber, 93) – new general or specialized terms should be included in ontology in such a way that the addition does not require the revision of existing definitions.
- Minimal ontological commitments (Gruber, 93) – there should be as few claims as possible about the world being modeled, which means that the ontology should specify as little as possible about the meaning of its terms, giving the parties committed to the ontology freedom to specialize and instantiate the ontology as required.
- Ontological distinction principle (Gruber, 93) – classes in an ontology should be disjoint. The criterion used to isolate the core of properties considered to be invariant for an instance of a class is called the Identity Criterion.
- Diversification of hierarchies (Arpirez et al, 98) – it is to increase the power provided by multiple inheritance mechanisms.
- Modularity (Bernaras et al., 96) – it is to minimize coupling between modules.
- Minimize the semantic distance between sibling concepts (Aprirez et al, 98) – Similar concepts are grouped and represented as subclasses of on class and should be defined using the same primitives whereas concepts which are less similar are represented further apart in the hierarchy.
- Standardize names whenever possible (Arpirez et al, 98)

4.2 Framework Description

Program planning and design sometimes can last for many years. The continuity of the reasoning process for planning and design is significant. Currently, such knowledge is normally kept in professionals' minds. The major characteristics of the scoping processes are managed manually. Figure 4 lists a conceptual framework illustrating the concept of SCOPE. Each SCOPE includes scope initiation, planning, definition, verification and change control. As program progresses, SCOPE evolves. The characteristics of the evolution will also be captured in SCOPE. In such a way, the SCOPE can provide users with scoping information and rationale not just for one phase of a program, but also the history.

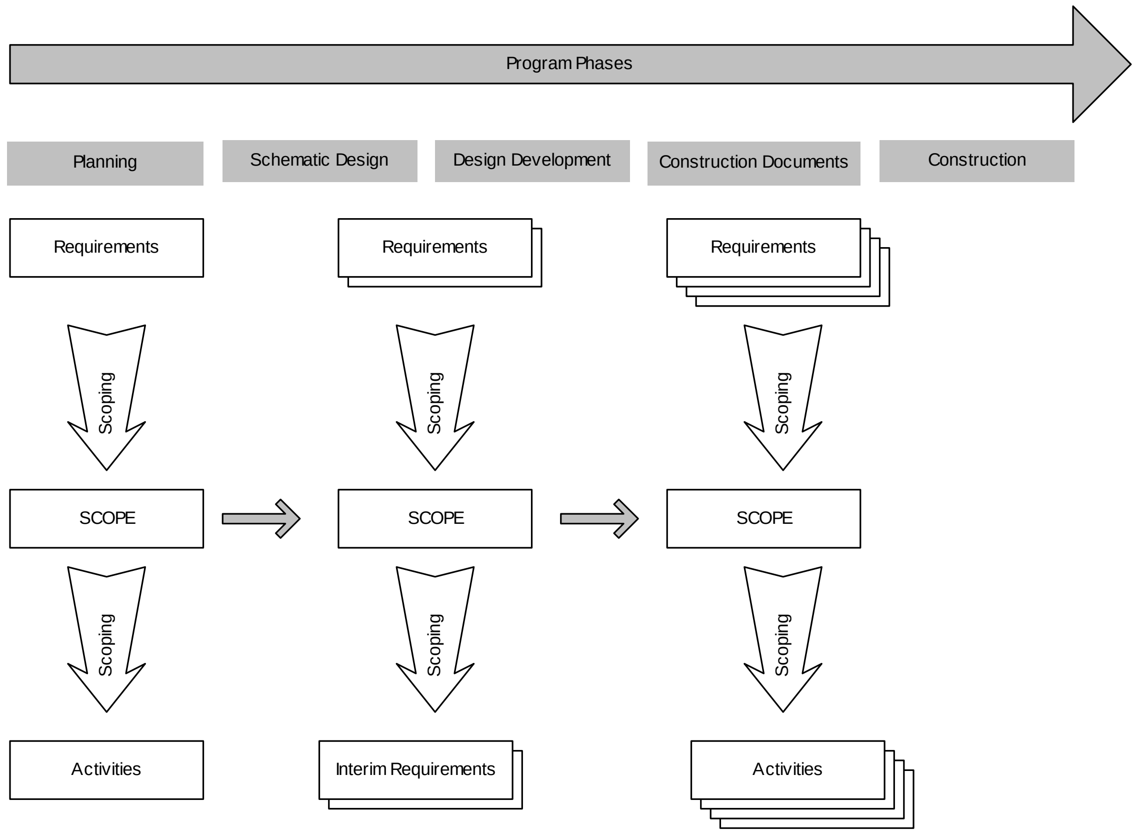

Figure 4

In many cases, the scoping processes of design, scheduling and costing may be different in terms of representations and processes as the final results of those functional activities are different. In addition, in a typical program management environment, the data sources for design, scheduling and costing are often heterogeneous. To resolve this issue, the SCOPE will be developed based on contemporary ontology development technology to ensure interoperability and the sharing of knowledge. Figure 5 demonstrates the conceptual architecture for implementation.

Using the architecture shown in Figure 5, different applications are not required to share a common representation of SCOPE or to map the SCOPE between them. Rather the generation of the SCOPE is based on the MMM (*Markov Model Mediator*) mechanism (Shyu et al, 2000, 2001, 2003a and 2003b). The proposed MMM mechanism is suitable to serve as the conceptual representation for the data sources in the program management environment with respect to the following aspects:

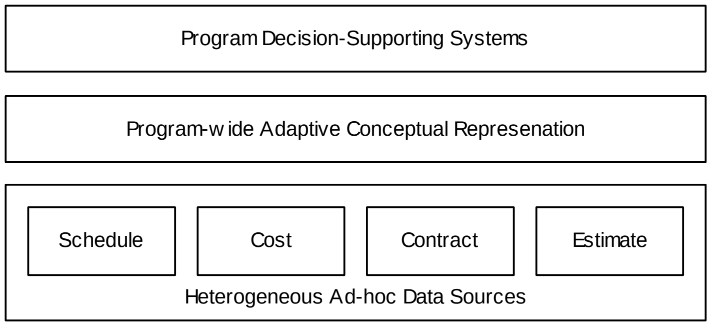
Program Decision-Supporting Systems
Program-w ide Adaptive Conceptual Represenation
Schedule
Cost
Contract
Estimate
Heterogeneous Ad-hoc Data Sources

Figure 5

1. Schema Independence Support: No matter what the underlying schemas of the data sources are, the integrated MMM can still provide querying and browsing of the information as well as control of data capture and presentation without the schema translation processes. The only information required from each data source is its objects and their attributes/features. Since things in the world have defining attributes, each data source can be represented by a set of objects and attributes/features. A local MMM is constructed for each data source and the semantic relationships among the objects within a data source are captured in that local MMM. An integrated MMM is then constructed from the local MMMs. As long as the relationships among the objects in each local MMM are captured, the same object relationships are also retained in an integrated MMM. Hence, as long as each data model of a data source provides the defining objects and their corresponding sets of features (attributes), the semantic relationships can be maintained in an integrated MMM without mapping between schemas. In other words, schema autonomy can be preserved as much as possible and schema independence can be supported by using the MMM mechanism.
2. Data Source Autonomy Preservation: One of the important characteristics of the conceptual representation is the ability to preserve the autonomy of each data source; that is, the data in a data source can be created and manipulated independently of other data sources. Database autonomy can be easily maintained by using the proposed MMM mechanism. No data source needs to expose all of its information to its integrated schema; instead, only higher level information such as the objects and their corresponding set of attributes/features need to be provided. Data in a data source can be manipulated via its own schema without any data conversion.

There are two types of MMMs. Each data source is modeled as a local MMM and all the data sources in the program management environment can be modeled as an integrated MMM. The compact notion δ=(S, F, A, B, π) is adopted for the MMM mechanism, where

1. S is a set of objects called states: An MMM consists of a sequence of states which represent the objects (in S) in the data source(s). The states are connected by directed arcs (transitions) which contain probabilistic and other data used to determine which state should be selected next. All transitions $S_i \rightarrow S_j$ such that $\Pr(S_j \mid S_i) > 0$ are said to be allowed, the rest are prohibited.
2. F is a set of attributes/features: A class of attributes or features, associated with an object, is to characterize the object and to represent the information pertaining to the database available to the application queries. Each object has its own set of attributes/features.
3. A is the state transition probability distribution: The state transition probability denotes the probability that a traversal choice to node *j* given the current node is in *i*. Here, the node represents a state in an MMM.
4. B is the observation symbol probability distribution: The observation symbol probability denotes the probability of observing an output symbol from a state. Here, the observed output symbols represent the attributes and the states represent the objects. Since an object has one or more attributes and an attribute can appear in multiple objects, the observation symbol probabilities shows the probabilities an attribute is observed from a set of objects.

5. π is the initial state probability distribution: For any object in a data source, the initial state probability is defined as the fraction of the number of occurrences of this object with respect to the total number of occurrences for all the objects in that data source.

5. Conclusion

Scope control and management is a critical issue for the success of a program, or a project. Existing scope control and management approaches are most manual, which is not efficient and effective. To achieve the overall goal of creating a digital society, this paper discusses a solution, called SCOPE, of using computer to mediate the process of scoping and scope evolution. The discussion is conceptual. Future work of developing SCOPE and testing the MMM based architecture are required in order to determine the feasibility of the framework.

References

1. Arpirez, J., Gomez-Perez, A., Lozano, A. and Pinto, S., An Ontology-based WWW Borker to Select Ontologies," Workshop on Applications of Ontologies and PSMs, Brighton, England, pp. 16-24, August, 1998.
2. Bernaras, A., Laresgoiti, I. and Corera, J., "Building and Reusing Ontologies for Electrical Network Applications," Proceedings of the 12th ECAT, pp. 298-302, 1996.
3. Construction Industry Institute, Cost/Schedule Controls Task Force. July 1986. Construction Industry Institute Scope Definition and Control Publication 6-2.
4. Dumont, P. R., Gibson, G. E. Jr. and Fish, J. R., "Scope Management Using Project Definition Rating Index", Journal of Management in Engineering, vol. 13 no. 5, pp 54-60, 1997
5. Genesereth, M. R. and Nilsson, N. J., *Logical Foundation of Artificial Intelligence.* Morgan Kaufmann, Los Altos, California, 1987
6. Gruber, T., "A Translation Approach to Portable Ontology Specification", Knowledge Acquisition, vol: 5, pp. 199 – 220, 1993
7. Gruber, T., "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", *IJHCS*, 43(5/6): 907-928, 1994
8. Huang, Y. L., Ibbs, C. W. and Yamazaki, Y., "Knowledge Mapping for Facility Scope Management", Proceedings of the 5th International Conference on Computing in Civil and Building Engineering - V-ICCCBE, pp. 409-417, Anaheim, CA, 1993
9. McCue, D., "Scope Definition is a Company Business", 1997 AACE International Transactions, pp. IT.01.1-IT.01.6, 1997.
10. Neches, R., Fikes, R. E., Finin, T., Gruber, T. R., Senator, T. and Swartout, W. R., "Enabling Technology for Knowledge Sharing", AI Magazine, vol. 12, no. 3, pp. 36-56, 1991.
11. Poldy, J. L., Shangraw, W. R. and Shangraw, R. F, Jr., "A Project Definition Methodology for Environmental Restoration Projects," 1992 AACE International Transactions, pp. H.5.1-H.5.8, 1992
12. Project Management Institute, "A Guide to the Project Management Body of Knowledge", Electronic Imaging Services, Inc., Evanston, IL, 2002
13. Shyu, M.-L., Chen, S.-C., and Kashyap, R. L., "A Probabilistic-Based Mechanism for Video Database Management Systems," Proc. of IEEE Intl. Conf. on Multimedia and Expo (ICME'00), New York City, USA, 467-470, 2000.
14. Shyu, M.-L., Chen, S.-C., Haruechaiyasak, C., "Mining User Access Behavior on the WWW," Proc. of IEEE International Conference on Systems, Man, and Cybernetics, pp. 1717-1722, Tucson, Arizona, USA, October 7-10, 2001
15. Shyu, M.-L., Chen, S.-C., Chen, M., Zhang, C., and Sarinnapakorn, K., "Image Database Retrieval Utilizing Affinity Relationships," Proceedings of the First ACM International Workshop on Multimedia Databases (ACM MMDB'03), pp. 78-85, New Orleans, Louisiana, USA, November 7, 2003a.
16. Shyu, M.-L., Chen, S.-C., Chen, M., Zhang, C., and Shu, C., "MMM: A Stochastic Mechanism for Image Database Queries," Proceedings of the IEEE Fifth International

Symposium on Multimedia Software Engineering (MSE2003), pp. 188-195, Taichung, Taiwan, ROC, December 10-12, 2003b.

17. Studer, R., Benjamines, V. R., Fensel, D., “Knowledge Engineering: Principles and Methods”, Data and Knowledge Engineering, vol. 25, pp 161-197, 1998.
18. Uppal, B. U., “Cost Engineering and Scope of Work,” AACE International Transactions, pp. ES61, 2002.
19. Uschold, M. E., Knowledge level modelling: Concepts and terminology. *Knowledge Engineering Review*, 13(1). Also available as AIAI-TR-196 from AIAI, The University of Edinburgh, 1998