Integrating Software Services for Preproject-Planning

Edward L DIVITA M.Sc., Ph.D. Candidate
divita@stanford.edu Stanford University Stanford, CA 94305-4020
Martin FISCHER Associate Professor
fischer@ce.stanford.edu
CIFE, Stanford University
Stanford, CA 94305-4020
John KUNZ
Senior Research Scientist
kunz@stanford.edu
CIFE, Stanford University
Stanford, CA 94305-4020

Edward Divita is s Ph.D. candidate in the Department of Civil and Environmental Engineering, Stanford University, whose research interests include integrating and automating Preproject-Planning

Martin Fischer is a professor at CIFE whose research interest lies in formalizing construction knowledge within a 4D (3D plus time) framework.

John Kunz is a Senior Research Scientist at CIFE whose research interests include Non-Numeric (Symbolic) Modeling of Engineering Products and Processes.

Summary

Sharing computer models among project participants and their software applications (services) can improve the quality and reduce the time to perform Pre-Project Planning (PPP) for construction projects. However, sharing knowledge and data among independent services for PPP requires a formal computer interpretable vocabulary (ontology), and managing the automated interoperation of services requires control architecture and reasoning mechanisms. This research extends PPP process modeling and *Circle Integration*. To date, PPP process modeling has been limited to generic representations of the PPP process. We present a more formal representation of PPP variables and relationships from which the PPP workflow process can be derived in real time while performing integrated and automated PPP. We refer to the specification of PPP variables and relations as the PPP ontology. The PPP ontology provides a foundation from which developers of PPP software services can create PPP analysis services. Such services can be joined together in any combination and will work together to perform multi-disciplinary PPP tasks. Circle integration is an architecture and methodology for integrating software services by linking them together to form a unified analysis system. We extend circle integration architecture by automatically relating services to each other at run time and by operationalizing circle integration control mechanisms that plan, execute, and re-plan the PPP process at run time based upon the requirements of the set of services that happen to be connected to the system. In this paper, we describe the structure of the PPP ontology and the function of the circle integration mechanisms using a simplified example of golf resort development.

1. Background and Motivation

In the past, execution of PPP has been time consuming and costly because it requires that many technical specialists work both independently and in coordination with other project team members. PPP involves organizing, analyzing project alternatives, and developing a project definition package that provides sufficient information for owners to consider risk/return and decide whether or not to commit resources to continue the project. In current practice, process models guide team members through PPP [1]. The limitation of these models is that they are not computer integrated and automated. The result is a protracted and costly PPP process. A recent approach to advancing PPP is Collaborative Desktop Engineering [2] utilizing Circle Integration [3] to link software tools (services) together to form an integrated analysis system. This approach involves creating individual software services that support specialty analyses (e.g., market analysis, cost analysis, schedule analysis, financial analysis) and linking those services together to form a system of services capable of working together to perform integrated and automated PPP. Divita et al. [4] describe FACT (Facility Alternative Creation Tool), a prototype circle integration system for fast food restaurant PPP.

Comparing human specialists working together and software services working together exposes an engineering problem. When human specialists work together during PPP they generally communicate using an informal vocabulary to describe and share knowledge and data about facilities and facility behaviors.

However, software services require a more formal computer interpretable vocabulary to support sharing of knowledge and data. An ontology is a formal specification describing the concepts and relationships that can exist for an agent or community of agents in a domain of discourse to enable knowledge sharing and reuse [5]. Musen [6] discusses the role of domain ontologies in software engineering. An ontology can be specified in computer interpretable form (e.g., object classes with attributes and relations). Continuing standardization efforts such as STEP [7] and Industry Foundation Classes [8] endeavor to specify standard product and project modeling components to support interoperability of building industry software, especially for the exchange of data. However, there currently is no standard vocabulary for computer exchange of PPP knowledge and data. In our research, we create a PPP ontology from observation of fast food restaurant, residential housing, and golf resort PPP. This approach allows us to extend the current standards for product/project modeling to include the domain of PPP. Further, the motivation for our PPP ontology goes beyond the exchange of data. We are pursuing a representation scheme that informs automation of the PPP workflow process.

Evaluating current integrated systems of software services reveals a second engineering problem. Combine 2 [9], COKE [10], IRTMM [11], and OPIS [12] all integrate component applications together to form unified systems for AEC project analysis. However, all are closed systems in that the applications are part of a fixed suite and the relationships among them are pre-determined. We foresee that specialty software developers will compete to provide component software services that users can download on demand from the internet. Said users will benefit from being able to link together groups of such software services to perform computer integrated multi-disciplinary tasks. Therefore, we are pursuing a theory and method for linking software services together on demand and supporting their working together in a coordinated manner. FACT allows services to be chosen freely and allied to form a virtual organization of services that can work together to perform an integrated multi-disciplinary task such as PPP.

To summarize, this research is intended to solve the following problems: 1) There is no standard vocabulary for computer exchange of PPP knowledge and data; and 2) Current integrated systems are closed and their component software services can not be freely chosen, linked, and expected to work in a cooperative/coordinated manner to perform multidisciplinary tasks such as PPP.

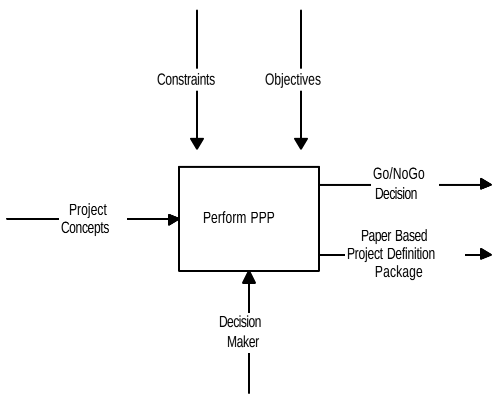
Our approach is founded upon 2 basic principles: 1) Software services exchange knowledge and data well if they utilize a shared project model (SPM) [13]. SPMs can be built using formal computer interpretable vocabularies known as ontologies. 2) A plan is an organized sequence of tasks that represents a complex process. Plans can be generated for tasks that have relationships from which precedence relationships can be derived. A good plan is informed by current actual requirements. As requirements change, the plan has to be verified (and, if necessary, updated).

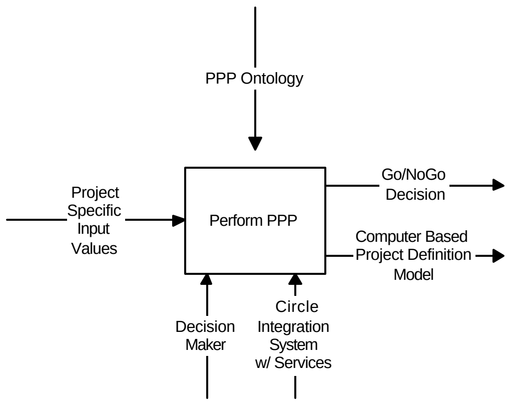
The PPP ontology will extend PPP process modeling and current project modeling standards by providing PPP domain specific variable and relation specifications with form and content that goes beyond enabling data exchange by supporting automation of PPP workflow. It does not attempt to be complete in identifying all possible PPP variables and relations. Nonetheless, it provides a powerful framework for defining such variables and relations that can be extended as time and effort allow and/or requirements demand.

FACT extends circle integration by contributing reasoning mechanisms that can generate a PPP workflow plan from the requirements that are inherent to any set of linked services that were developed with commitment to the PPP ontology. When services join the circle, relationships are established between the services and the shared project model. These relationships inform the precedence of tasks from which a PPP workflow plan is derived. The mechanisms also update the plan during execution of the PPP workflow process. In essence, the PPP workflow plan informs the linked software services how to work together to perform PPP.

2. Pre-Project Planning

Gibson et al. [14] found that pre-project planning is a complex process that can be standardized by breaking the procedure down into discrete tasks that can be organized in a structured manner. Figure 1 part A is an IDEF0 diagram that describes the context of current state of the art PPP process modeling [14]. In the IDEF0 diagramming method, each box represents a process.





PPP Ontology
Project Specific Input Values
Perform PPP
Go/NoGo Decision
Computer Based Project Definition Model
Decision Maker
Circle Integration System w/ Services

A. Gibson [14] B. FACT [4]

FIG. 1. IDEF0 Diagram – Perform Pre-project Planning [14]

Arrows entering the left of the box are inputs that are transformed by the process. Arrows entering the top of the box are controls or constraints on the process. Arrows leaving the right are outputs of the process. Arrows entering the bottom are mechanisms needed for the process to occur. This [14] PPP process model (including the expanded diagrams with greater detail that are presented in the referenced paper) provides a useful generic representation of the PPP process. However, it does not specify essential variables and relationships that can be used by software developers to create PPP services. Figure 1 part B is an IDEF0 diagram that describes the context of PPP using FACT. The following are key distinctions between the two models.

In the Gibson [14] model (Figure 1 part A), the inputs are project concepts. At the most detailed level, the inputs defined for this model are general concepts that are not sufficiently specific for direct application to computer-based analysis. In the FACT model (Figure 1 part B), the inputs are project specific input values. During automated PPP using FACT, variable parameters will be assigned values for inputs to each task of the PPP process. The Gibson model identifies constraints and objectives as controls for the process. Such controls are vague and never specified in detail within the model. The FACT model suggests that the vocabulary of variables and relations, the PPP ontology, controls the process. The Gibson model proposes that a decision maker is the mechanism needed to make the process occur. In the FACT model, a decision maker together with the circle integration system (suite of services linked together working as a unified analysis system) is the mechanism. Lastly, the Gibson model proposes that the output of the PPP process is a go/no go decision and a paper-based project definition package. In the FACT model, while a go/no go decision is a key output, equally important is the computer based project definition model. In addition to containing record values for all key variables, the computer based project definition model is dynamic in that users can continue to make changes and analyze the model. Further, the model is ready (computer interpretable) for further analysis by the project team during later stages of the project life cycle should the decision to proceed with the project be affirmed.

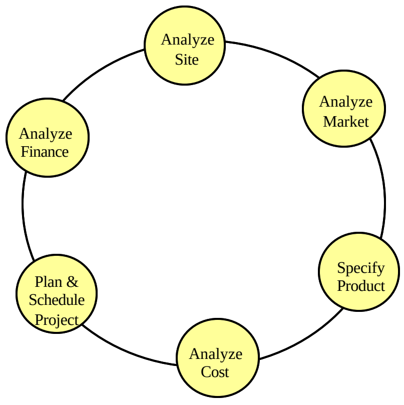


FIG. 2. Basic Circle Integration Architecture

FACT extends PPP modeling by deriving essential generic variables and relationships required for PPP and specifying those as a computer interpretable reference ontology. The PPP ontology will allow software service developers to create PPP services that can be linked on demand and will work together to perform integrated and automated PPP in a general way and share PPP knowledge and data in an unforced manner. The PPP ontology, though, provides only the shared computer interpretable vocabulary that provides a basis for communication. Operationalizing a system of linked services requires system architecture and general reasoning methods for controlling cooperative function of the unified system.

Circle Integration

Figure 2 describes basic circle integration architecture [3]. In the basic circle architecture, services for each activity (nodes on the circle e.g., Analyze Finance) have a single predecessor (e.g., Plan & Schedule Project) and a single successor (e.g., Analyze Site). Thereby, integration control involves passing information around a circle of applications. Circle integration defines a complete design version as an iteration through the circle that is accepted by the complete team. Circle integration characteristic s include explicit and clear feedback loops; simple and effective version management; rapid feedback; and stable designs. Fischer and Kunz [3] conceptualized circle integration but did not operationalize and test the concepts. This research operationalizes the circle integration model for PPP and extends it by adding a PPP ontology and circle integration reasoning mechanisms. Figure 3 shows the FACT architecture. The key components of the FACT architecture that extend basic circle architecture are the PPP Ontology and the Circle Integration Mechanisms.

The PPP Ontology specifies project variables and relationships. As seen in figure 3, Internal Rate of Return (IRR) is a project variable. The Analyze Finance service has an “Observes” relationship with the variables *Golf Course Cost* and *Golf Course Timing.* The Analyze Finance service has a “Predicts” relationship with *Project IRR*. These relationships describe how each service relates to the PPP ontology. The Analyze Finance service “Observes” the value of the variables *Golf Course Cost* and *Golf Course Timing* as inputs prior to performing analysis. By performing analysis, the Analyze Finance service “Predicts” the value of *Project IRR* as an output. The circle integration mechanisms make the integrated and automated PPP process occur. The *Select Services!* method enables the user to add services to and delete services from the circle. The *Plan!* method derives precedence relations from the “Predicts” and “Observes” relationships and arranges the services around the circle in sequence creating a PPP process workflow plan. The Execute! method prompts the function of each service in sequence. Finally, the Re-Plan! method verifies and, if necessary, updates the PPP process workflow plan when value(s) of variables that have been computed (“predicted”) by one service are “changed” by another.

4. Circle Integration Mechanisms

We will now illustrate the function of the circle integration mechanisms by relating to a simplified portion of a golf project case. The overall goal of this simplified fragment of PPP is to perform financial analysis, cost analysis, and plan & schedule analysis using specialty services for each task. Further, we want to choose the specialty services from independent sources and join them together to form an integrated analysis system so they can work together and exchange data easily.

Select Services!

Let us assume that a user selects three services (Analyze Finance, Analyze Cost, and Plan & Schedule Project). When the services are linked into the system (execution of Select Services! method), liaison relationships are established between the services and the Shared Project Model. Figure 4 illustrates the types of relationships that are established at the time of linking. The Analyze Finance Service establishes a “Predicts” relationship with IRR and an “Observes” relationship with Cost and Timing. In simple terms, the Analyze Finance service must observe the value(s) of Cost and Timing in order to predict the value of IRR.

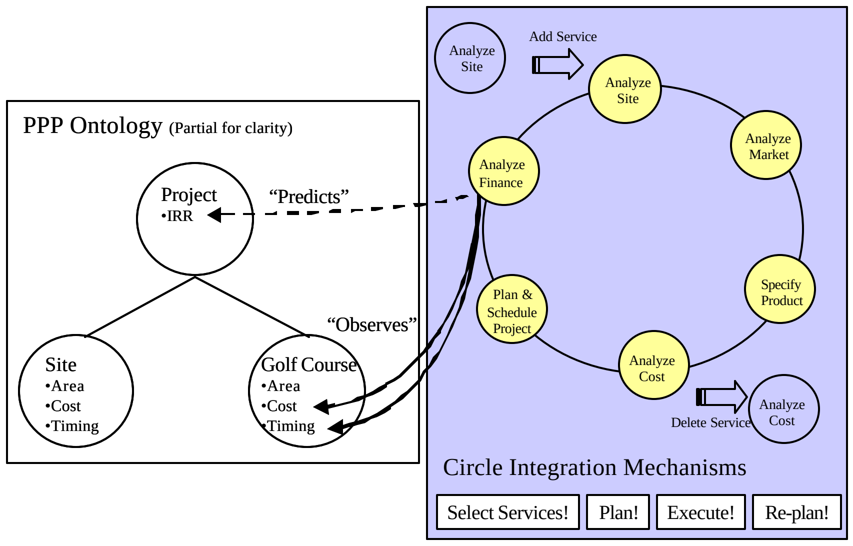


FIG. 3. FACT Architecture with Circle Integration

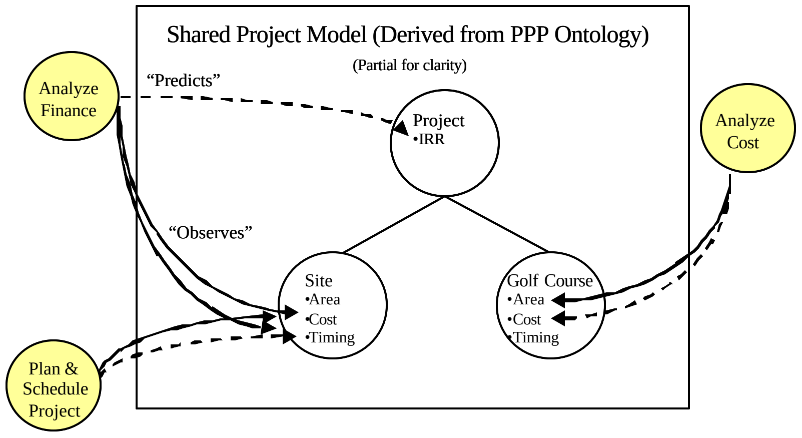


FIG. 4. "Predicts" and "Observes" Relationships between Services and Components of Ontology

By committing to the PPP Ontology, developers of PPP services can precisely specify how they use relationships and project variables that may be shared by other PPP services. In addition, the FACT circle integration engine can derive precedence from the knowledge encoded in services that commit to the PPP Ontology.

Plan! and Execute!

After the services are linked and the relationships are established, FACT applies the Plan! method which executes general sequencing logic to generate a PPP service workflow plan. To generate the plan, the sequencing mechanism considers each service individually (e.g., Analyze Finance service in figure 4). Then it considers each variable that the service observes (e.g., Site Cost in figure 4). If another service predicts that variable, then the predicting service becomes the predecessor. In figure 4, Cost is predicted by the Analyze Cost service. Therefore, the Analyze Cost service precedes the Analyze Finance service (see figure 5 for derived sequence of services). Continuing this stream of logic through all of the variables observed by the Analyze Finance service reveals that the Plan & Schedule Project service is also a predecessor of the Analyze Finance service. Applying the general sequencing logic for all of the services that have been linked produces the system of precedence illustrated in figure 5. The Execute! method simply follows the plan. In the simplified case example, golf course and site costs are determined first. Then, the schedule duration and timing is determined. Lastly, a cash flow array is created using cost and timing data and the IRR is calculated by discounting the cash flow. This above logic can be applied to a much larger system of services with many more variables. Regardless of the number of services and variables, the general sequencing logic applied to the liaison relationships will always produce an initial workflow plan that informs the services how to work together. However, as the design process progresses during integrated and automated PPP, earlier assumptions may be superseded by a decision to revise the value of a variable. When changes of this nature arise, the workflow plan must be verified and/or updated.

Re-Plan!

Now let us consider a scenario that requires the system to deal with change. Assume that when at the Analyze Finance node, a decision maker decides that the cost is too high and he wants to revise the value. During integrated and automated PPP, when a variable is changed at a service, the Re-Plan! method is invoked. This circle integration mechanism takes into consideration the variable whose value has been changed and begins to iterate through the services in sequence. First, the mechanism checks at each service to see if the service predicts the variable that was changed. If so, the user will be advised that the value of the variable was predicted by another service. The user will be given an opportunity to return to the predicting service to re-consider the prior analysis. The iteration will continue until all services are checked for prediction. Then, the mechanism will iterate again through the services in sequence checking to see if any service observes the variable that was changed. If so, the observed value is replaced with the changed value and analysis is again performed by that service. The system continues on to the next service in sequence. The process continues until the user returns to the service where the change was initiated. A complete design version is now defined as completing the Execute! method or in the case of change(s) the Re-Plan! method.

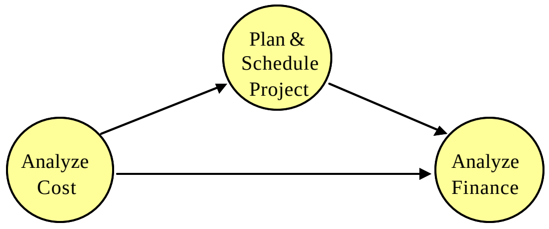


FIG. 5. Derived Sequence of Services

5. Conclusions

FACT's novelty in contrast to other service integration control mechanisms is its flexibility. Specifically, this flexibility is evidenced by its ability to generate a workflow plan for services in real time when services are joined together and also its ability to re-plan during process execution in response to changes that occur as a part of the PPP process. As such, FACT makes it possible for a user to select independent software services and join them together to form a unified system that functions in an integrated manner.

A key formalism that makes FACT possible is the PPP ontology. The PPP ontology extends PPP process modeling and current project modeling standards by providing a useful framework and some precise specifications for computer-interpretable variables and relations specific for the PPP domain.

The underlying objective of this research is to enable improving the PPP process. Charrette testing with the FACT prototype involving graduate students and industry professionals suggests that integrated and automated PPP is faster, more accurate, and less variable than manual methods [2]. The current version of FACT is limited to relatively simple test cases. The challenges associated with scaling up the PPP Ontology and applying the circle integration mechanisms to very detailed complex cases is currently being considered.

6. References

[1] Construction Industry Institute (CII), (1995), *Special Publication 39-2: Pre-Project Planning Handbook*, Rep., Austin, Texas

[2] Divita, Edward L., Kunz, John C., and Fischer, Martin A. (1998-A). "Collaborative Desktop Engineering."*Artificial Intelligence in Structural Engineering: Information Technology for Design, Collaboration, Maintenance, and Monitoring*, Smith, Ian (Ed.), Springer, 69-85.

[3] Fischer, M., & Kunz, J. (1995), "The Circle: Architecture for Integrating Software," *Journal of Computing in Civil Engineering*., ASCE, 9(2), 122-133.

[4] Divita, Edward, Jr., Fischer, Martin A., and Kunz, John (1998-B). "Integration and Automation of Pre-Project Planning through Circle Integration with a Shared Project Model." Technical Report, Nr. 115, Center for Integrated Facility Engineering, Stanford.

[5] Gruber, T. R., (1993), "A translation approach to portable ontologies," *Knowledge Acquisition*, 5(2):199-220.

[6] Musen, Mark, A., (1997), "Domain Ontologies in Software Engineering: Use of Protege with the EON Architecture," *IMIA Working Group 6 Conference on Natural Language and Medical Concept Representation*, Jacksonville, Florida.

[7] ISO 10303-1 (1994), ISO-TC184/SC4 ISO, Industrial automation systems and integration --Product data representation and exchange -- Part 1: Overview and fundamental principles.

[8] Industry Alliance for Interoperability (IAI), (1999), *Industry Foundation Classes IFC Project Model Specifications*, Ver. 2.0

[9] Combine 2 (1995). Computer Models for the Building Industry in Europe. Final Report, Godfried Augenbroe Ed., TU Delft.

[10] Fischer, Martin A. (1991). "Constructibility Input to Preliminary Design of Reinforced Concrete Structures." Technical Report, Nr. 64, Center for Integrated Facility Engineering, Stanford.

[11] Kunz, J.C., Jin, Y., Levitt, R.E., Lin, S., Teicholz, P.A., (1995), "The Intelligent Real-Time Maintenance Management (IRTMM) System: Support for Integrated Value-Based Maintenance Planning," *CIFE Technical Report 100*, Stanford University, January 1995

[12] Froese, T., (1992), *Integrating project management software through object-oriented project models*, PhD thesis, Dept. of Civ. Engrg., Stanford Univ., Stanford, CA

[13] Fischer, M. & Froese, T., (1996). "Examples and Characteristics of Shared Project Models," *Journal of Computing in Civil Engineering*, ASCE, Vol. 10, No. 3, 174-182.

[14] Gibson, G.E., Kacxmarowski, J.H., and Lore Jr., H.E. (1995), "Preproject-Planning Process for Capital Facilities," *Journal of Construction Engineering and Management*, Vol. 121, No. 3, pp 312-318, September, 1995