CIFE Center for Integrated Facility Engineering • Stanford University

CIFE Seed Proposal Summary Page For 2002 Awards

Proposal Number*: **Total Funds Requested:**

**Assigned by CIFE*

First Submission? _____

Extension? _____(If yes, please provide the previous proposal number & URL)

_____

Funding Start Date: **Funding Finish Date:**

Proposal Title: Composition of A/E/C Services

Stanford University Dept.: Civil and Environmental Engineering
Computer Science Department
Electrical Engineering Department

Principal Investigator: Kincho H. Law

Faculty/Staff Position:

CIFE Thrust Area :

Product & Process Modeling __ Visualization _____

Internet Collaboration _X___ Supply Chain Management & E-commerce _____

Facility Management _____ Management of Technology & Education _____

People Involved with the Project:

Faculty: Kincho H. Law, Armando Fox, and Gio Wiederhold

Research Associates:

Graduate Students: David Liu, and Laurence Melloul

Staff:

Abstract:

This interdisciplinary, joint research project seeks to address the issues that will allow the composition of services based on isolated large-scale software applications commonly employed in the A/E/C industry.

We expect the following results from the proposed research:

- A common data exchange model based on which information can be exchanged among collaborating software applications;
- A scalable infrastructure in which autonomous services can be integrated;
- Development of a high-level compositional language;
- A megaservice control module for coordinating the execution of megaservices; and
- Development of algorithms for distribution of megaservice data-flows.

We have successfully exchanged data among project management tools such as Vite, 4D modeling tool, Primavera and Microsoft Project. We plan to continue focusing on these software applications, which are to be adopted in the flagship project planned by CIFE. In addition, our plan also includes demonstrating the framework with supply chain management and on-line e-business applications.

Composition of A/E/C Services

David Liu, Laurence Melloul, Kincho H. Law, Armando Fox, and Gio Wiederhold

1. Engineering Problem and Proposed Research

As computation and communication technologies evolve, we are seeing a change in how software services are built. Rather than constructed from ground up, large software services are composed of cooperating components. Existing commercial off-the-shelf (COTS) software applications are utilized as building blocks that provide pieces of functionalities. The vision of software composition [13] is echoed in the megaprogramming framework [2, 24], which builds on software components called megamodules [18] that capture the functionality of autonomous services provided by large organizational units. Autonomous services are linked together according to composition specifications [20, 23] to form megaservices.

Impact of this approach is expected in numerous applications in the design and construction industries and holds promise to revolutionize the current practice based mostly on disjoint server computing. Currently, large software applications provide a large degree of automations for distinct functional groups. Multiple such applications must be invoked during the design and development process. However, an effective automation methodology for these functional groups to cooperate with each other is lacking. An infrastructure where all software services are integrated can improve productivity greatly. For instance, a construction personnel on-site could readily compare the as-built site condition with the planned design information, notify any inconsistency, and automatically trigger software services to conduct rescheduling of the project. In addition, an inquiry on the availability of materials and resources can be sent and the response can cause change orders to be issued.

This interdisciplinary, joint proposal seeks to address the issues that will allow the composition of services based on isolated large-scale software applications commonly employed in the A/E/C industry. Some of the technical challenges we must address include:

- Service integration infrastructure: To be able to integrate existing software applications together to perform coordinated tasks, a service integration infrastructure is required to tie together the distributed computing resources. Software applications need to be wrapped to form autonomous services so that their functionalities can be accessed by other components within the infrastructure. Runtime support needs to be provided so that autonomous services can be scheduled, data can be exchanged among autonomous services, and exceptions can be handled.
- Compositional language: The composition of autonomous services needs to be described in the form of a formal specification. A practical general-purpose compositional language needs to be provided to the application programmers for composing megaservices using autonomous services. The language provides the application programmers the necessary abstractions to describe the behaviors of their megaservices. It should be built around concepts and abstractions that do not correspond directly to the features of the underlying machine. Our goal is to design a language that can be used by people with minimum programming experience.
- Performance: As software applications and composed megaservice getting more complex, the system performance becomes a major issue. We will identify a few critical system parameters that correlate with the performance of megaservice executions. Techniques to

improve performance of service integration infrastructures will be introduced based on tuning of these parameters.

Figure 1 illustrates a sample service integration infrastructure for A/E/C software applications. Three groups of users originally relied on different sets of software tools, whose communications would have been rather ad-hoc. With a service integration infrastructure, existing software applications are wrapped to allow the autonomous services to be connected together through a communication network. To the users of different functional groups, the infrastructure appears as a single comprehensive software service that covers functionalities from project modeling, project planning to supply chain management.

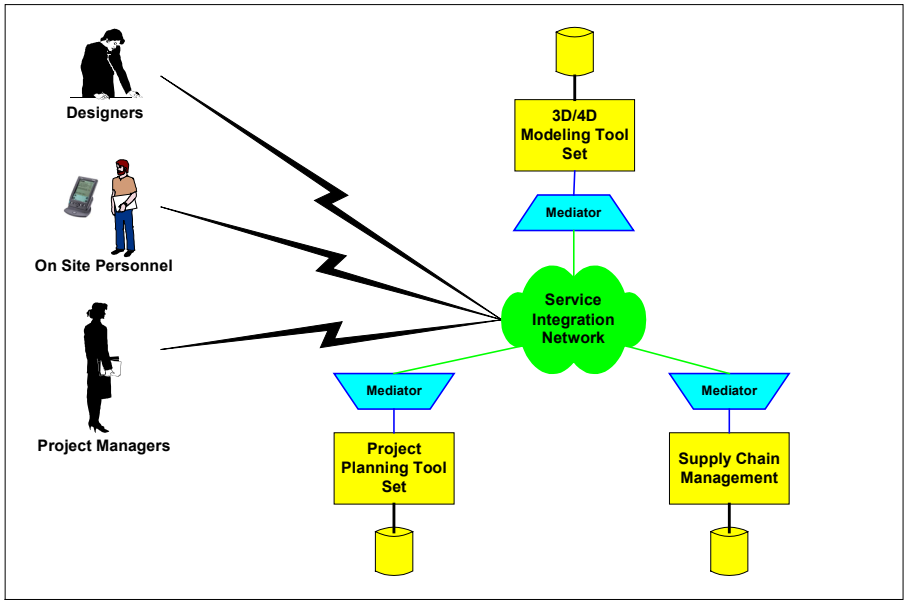

Figure 1: Sample Service Integration Infrastructure for A/E/C Services

1.1. Related Research and Commercial Efforts

1.1.1. Information Integration

Information integration has long been recognized as a central problem in enabling distributed and ubiquitous computing. Through integration, heterogeneous information sources can work together as a whole.

Many difficult problems need to be addressed before information sources can be integrated: (1) Legacy databases cannot be altered to support integrated applications; (2) Different databases have different meaning for the same terms, or use different terms for the same concept; (3) Information sources may have no fixed schema. Mediators [22, 23] are introduced into the integration architecture to address the above issues. Mediators deal with the issue of heterogeneity and present the information to the users with a homogeneous view of the underlying sources. Information sources are accessed via mediators, which translate between sources' local concepts and the global concepts shared by some or all of the sources.

Most information clients are best served by data in object form [17]. Object technology enables applications to use an infrastructure that aggregates details into meaningful units in many important domains. Much advancement has been made in technologies involving domain specific standards such as STEP, ifcXML, ebXML, and others.

Given a heterogeneous data model, integrating information from distributed sources can be thought of as constructing answers to queries using views that represent the capabilities of information sources. Two projects, Tsimmis [6, 16] and Infomaster [8], were carried out by the Stanford Infolab and Computational Logic group to address the issue of heterogeneous information integration. The key differences lie in their underlying database logic, their query-processing algorithms, and their approaches to semistructured data, etc.

In [10], we demonstrated the method of using the combination of relational database technology, active mediation [11], XML and PSL (Process Specification Language) to build intermediary data model for a variety of engineering service applications. As shown in Figure 2, PSL wrappers are built for such applications as Primavera P3, Microsoft Project, and 4D Viewer [9]. A translator is built between PSL and the relational data model so that applications using the relational data model may access the project data. XML and active mediation are used in combination to provide integrated data to online applications that reside on different types of accessing devices.

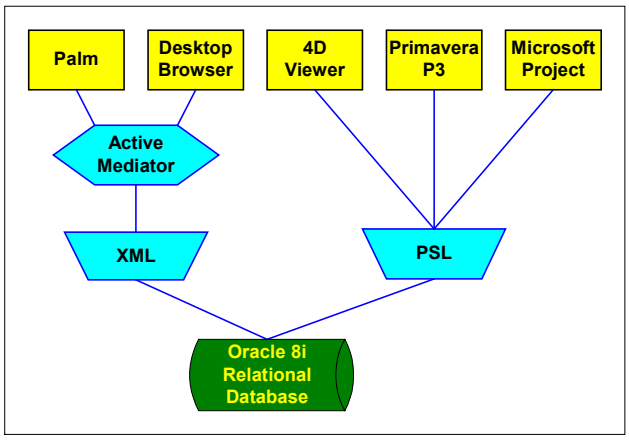

Figure 2: Data Integration for Sample Engineering Services

Being able to exchange data in a meaningful way between applications is the first step in conducting service integration. We plan to incorporate the relevant findings as one point-of-departure.

1.1.2. Service Integration

If information integration is composition of data, then service integration is composition of application programs. Service integration has been an active research area. Service integration technologies differ in their support of programming abstraction provided to the application programmers.

Object-oriented RPC is an extension of object-oriented procedure call to work across the network. The object-oriented structuring of distributed systems provides a basis for designing the interfaces for services. Services are accessed through one or more interfaces that define what is visible to the client, encapsulating the implementation behind the interfaces. It provides the low-level communication mechanism to utilize the services. Object-oriented RPC is used extensively in CORBA [4, 15], DCOM, and Java RMI [19].

An alternative interface and communication facility in service integration is to use message facility directly. The common model of messaging is a model of non-blocking asynchronous messages between processes using basic send/receive communication primitives. The application programs have explicit control over how services are synchronized. The implementation is closer to the hardware base, hence providing more control and flexibility to application programmers.

However, the messaging model has a significant drawback in that it is an awkward abstraction for application programmers who otherwise deal solely with procedures.

One can also conduct service integration using a shared memory facility for coordination and communication. A distributed shared memory system can be built to provide the abstraction of the shared memory across multiple network nodes. Operating as global communication buffers, Tuplespace [5, 7, 25] based systems have played the role of traffic cop for data flowing from one process to another in parallel and distributed systems. Tuplespace serves as a general data delivery medium. This shared medium also becomes a synchronization mechanism during the concurrent execution of the processes involved in computation. Distributed shared memory approach is data-driven. The control logic is implemented by placing and fetching data elements from the shared medium. Such programming model lacks separation between computation, communication and synchronization, where the computation code is usually interspersed throughout a program with communication, coordination and synchronization code.

1.1.3. Service Composition

The CHAIMS project [1, 23] has laid much foundation for autonomous service composition. CHAIMS is based on the concept of megaprogramming [24], which focuses on the composition of megamodules that are large, distributed components autonomously operated and maintained. Their concurrent nature and potentially long duration of service execution necessitates asynchronous invocation and collection of results.

One goal of CHAIMS is to provide a practical general-purpose compositional language. A purely compositional language CLAM is defined [20] to provide the application programmers the necessary abstractions to describe the behaviors of their megaprograms. CLAM extends the simple notion of composition by decomposing the traditional CALL statement. The decomposition provides parallelism for asynchronous composition of large-scale services. Another important characteristic of CLAM is its complete lack of support for computation. All arithmetic and many other convenient language features need to be implemented as megamodules. This is done to enforce the separation of composition from computation. Although the approach is theoretically feasible, it is sometimes impractical.

The run-time service integration platform [14] in CHAIMS is implemented using object-oriented RPC. The transfer of controls and data between the megaprogram and the megamodules is done via various RPC protocols, resulting in an integration model with centralized data-flows. As we analyzed in [12], there are many potential performance and scalability issues associated with centralized integration model, motivating us to explore a distributed data-flow integration model to conduct service composition.

1.2. Proposed Research

In this research, we seek to develop a scalable service composition framework that will facilitate the construction of large-scale software services typical to the A/E/C applications. Specifically, the proposed study includes four research and development efforts:

1. Scalable Service Integration Infrastructure
2. High-level Compositional Language
3. Megaservice Controller
4. Performance Enhancement for Megaservice Executions

For the first task, scalability is the key. Building a scalable runtime environment requires a mechanism to easily incorporate existing and new software applications. This will be achieved

by individually wrapping each software application into autonomous services that can be directly plugged onto the communication network. Since autonomous services are developed independently, a common protocol needs to be developed for accessing the services. The protocol needs to support (1) a homogeneous autonomous service metamodel; (2) asynchronous service invocations; and (3) both centralized and distributed data-flows. Research led by Professor Armando Fox and the Software Infrastructures Group will provide the expertise and experience to help our initial implementation of the prototype.

For the second task, our research involves designing a high-level compositional language. We will interact and adopt the research development in the CHAIMS project led by Professor Gio Wiederhold. We propose to refine the CLAM language by providing support for specifying complex computational logic. Active objects [11] can be used as computational components, while serving the role as light-weighted autonomous services that migrate among autonomous services.

For the third task, we plan to build the megaservice controller that serves as the centralized coordinator for megaservices. The controller takes the megaservice specifications as input and issues appropriate service invocation requests to the autonomous services. The megaservice controller schedules the execution of collaborating autonomous services. It is also responsible for directing data traffic among autonomous services and the megaservice. Previous research results in distributed computing and job scheduling will be used to build the megaservice controller.

For the fourth task, we plan to optimize the performance of megaservices by managing data-flows across a service integration network. As software applications become more complex and the volume of exchanged data increases, performance of megaservices becomes sensitive to the service integration model. As analyzed in [12], the distributed data-flow model is in general superior in performance compared to that of the centralized data-flow model. The analysis also provides recommendations for a couple of techniques to build high-performance, highly scalable service integration infrastructures. First, performance can be improved by cutting down the volume of data-flow between megaservice and autonomous services. This can be achieved by distributing data-flows among autonomous services. Secondly, we contemplate that code segments can be transferred from the megaservice to autonomous services so that data processing can be conducted remotely on the autonomous services to avoid the transferring of data between autonomous services and the megaservice. Technologies such as active mediation [11] and mobile agents [3] can be utilized for such purpose. Our investigation indicates that active mediation technology can potentially be incorporated into the integration infrastructure to provide the necessary run-time support.

1.3. Test Plan and Anticipated Results

We expect the following results from the proposed research:

- The first enabling factor of service composition is to have a common data exchange model based on which information can be exchanged among collaborating software applications. We expect to define a common data model that can be utilized by existing and future autonomous services. Mediators are employed to conduct conversions between proprietary data models used by individual software applications and the common data model.
- We expect to develop a scalable infrastructure in which autonomous services can be integrated. The immediate result is the development of an autonomous service access protocol (ASAP), based on which software applications can be wrapped to form autonomous services. Since much of the ASAP control logic and the internal structure of the wrappers are identical, we plan to develop common autonomous service mediator modules to simplify the task of constructing autonomous services.

- The development of the high-level compositional language will be an ongoing research effort. We intend to develop a deeper understanding of the critical features desired in large-scale software composition. We will extend the CLAM language to provide application programmer a convenient facility to specify complex computational logic, while keeping the philosophy of separating composition from computation.
- A megaservice controller will be built to carry out megaservices. However, we do not expect to focus our research effort in find new scheduling algorithms for parallel executions of autonomous services [21]. Rather, previous industrial and research result in the distributing computing arena will be utilized.
- One of our research focuses will be on developing algorithms for distribution of megaservice data-flows. Our mathematical analysis [12] indicates much can be achieved by optimizing the data-flows among collaborating autonomous services. Active mediation technology will also be incorporated to migrate computational logic in order to enhance megaservice performance.

As discussed earlier in our preliminary results, we have successfully exchanged data among project management tools such as Vite, 4D modeling tool, Primavera and Microsoft Project. We plan to continue focusing on these software applications, which are to be adopted in the flagship project planned by CIFE. In addition, our plan also includes demonstrating the framework with supply chain management and on-line e-business applications.

2. Support to CIFE Goals

The research presented in this proposal focuses on the thrust area of *Internet Collaboration*. Our goal is to build a realistic infrastructure that can interact with engineering services, including database access, design applications, project management tools and engineering simulations.

In addition, the research is also related to the following CIFE thrust areas:

Product and Process Modeling:

- Standards & practices, specifications, protocols
- Use of the Web for front-end integration
- Data flow modeling
- Service and maintenance modeling

Supply Chain Management and E-Commerce:

- Integration of legacy and emerging supply chain management systems
- Infrastructure needed to set up and maintain e-commerce
- Modeling construction supply chains
- Mobile information access/processing

Management of Technology and Education:

- Modeling and simulating project organizations – virtual organizations
- Understanding human factor issues in technology exploitation

3. Relation to Other CIFE Research

This research focuses on building a prototype system for a specific domain application in A/E/C so that fundamental research issues, ranging from product modeling, information mediation, specification of workflows, to scalable service integration can be thoroughly

investigated. The research results should also find broad applications to other telecommunication and manufacturing industries.

This area of research touches upon many aspects of past and current CIFE research projects:

- Mobile Computing and Active Mediation Technology in a Ubiquitous Computing Environment
- Agent-Based Project Scheduling and Control
- CIFE Interaction Workbench
- The iRoom to Go
- Leveraging Mobile Computing Device Technology in a Distributed Engineering Services Environment
- Designing and Evaluating Construction Information Workspaces

4. CIFE Industry Member Involvement

Many CIFE Industry Members have been working on data integration. Time is ripe for service integration. We seek to partner with companies that are interested in collaboration. Initial investigation includes detailed knowledge acquisition regarding how companies are currently integrating their legacy and new applications. We intend to investigate the current efforts by CIFE Industry Members in integration large-scale software applications. We will map out the use cases and potential desires of CIFE Industry Members.

We plan to participate with the CIFE research project with industry members' feedback. Specifically, we are interested to demonstrate the service composition framework with the application software used in the flagship project.

5. Milestones

We separate milestones into three categories:

Accomplished milestones:

- Identifying the critical features desired for the high-level compositional language.
- Analyzing the performance of service integration models in order to identify the critical performance bottlenecks in service composition.
- Defining the basic components in the autonomous service access protocol

Milestones to be accomplished before proposed project year:

- Development of algebra and algorithms to map legacy and relational data model into XML based object data model.
- Complete the definition of the compositional language. Build a compiler that conducts code checking and generates executable megaservice control sequences.

Milestones to be accomplished within the proposed project year:

- Complete the specification for the autonomous service access protocol. Build an autonomous service mediator, and demonstrate its effectiveness in wrapping software applications such as P3, Project and other applicable tools into autonomous services.
- Performance enhancement by developing algorithms for distributing data-flows among autonomous services.
- Incorporating active mediation technology into service integration network.

6. Risks

There are a couple of fundamental risks for this project:

- The adoption of a composition language is unpredictable. The research will focus on developing a pure-compositional language. The benefits are its simplicity and its separation of composition from computation. However the adoption of the language depends on other factors such as social acceptability, etc.
- Wrapping existing software application into autonomous services can be challenging. Some software applications only offer proprietary interactive interfaces to the users. "Hooks" need to be built into these applications to export the functionalities out.

All the risks are the result of current technology limitations. We see these risks to be alleviated with the continuing push of offering software applications in the form of autonomous services. We contemplate that the software applications in the future will have a more open interface or "protocol" so that their functionalities can be utilized more conveniently, sometimes beyond the original expectation of the application designers. However, the current limitations will focus our research on developing a methodology in wrapping legacy software application into autonomous services so that they can be more widely used.

7. Anticipated Papers and Presentations

We anticipate several technical presentations and papers to be published as CIFE reports, conference proceedings, and journal articles. We expect to submit publications to information technology related journals in the pertinent domains.

8. Potential External Funding Sources

Upon the completion of the seed phase of this research, we will explore government funding opportunities such as those at National Science Foundation (NSF), NIST and DARPA but we believe that demonstration of the framework will also attract funding from industry participants that will benefit from the fruition of this technology.

9. Reference

[1] D. Beringer, C. Tornabene, P. Jain, and G. Wiederhold, "A Language and System for Composing Autonomous, Heterogeneous and Distributed Megamodules", Proceedings of DEXA International Workshop on Large-Scale Software Composition, Vienna Austria, August 1998.

[2] B. Boehm and B. Scherlis, "Megaprogramming", Proceedings of DARPA Software Technology Conference, Los Angeles, April 1992, pp. 68-82.

[3] D. Chess, B. Grosof, C. Harrison, D. Levine, C. Parris, and G. Tsudik, "Itinerant Agents for Mobile Computing", *IEEE Personal Communications*, vol. 2(5), October 1995, pp. 34-49.

[4] "The Common Object Request Broker: Architecture and Specification Version 2.0", *Object Management Group*, Report, July 1995.

[5] A. Fox, B. Johanson, P. Hanrahan, and T. Winograd, "Integrating Information Appliances into an Interactive Workspace", *IEEE Computer Graphics & Applications*, May 2000.

[6] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, and J. Widom, "The TSIMMIS approach to mediation: Data models and Languages", *Journal of Intelligent Information Systems*, 1997.

[7] D. Gelernter and A. J. Bernstein, "Distributed Communication via Global Buffer", Proceedings of ACM Principles of Distributed Computing Conference, 1982, pp. 10-18.

[8] M. R. Genesereth, A. M. Keller, and O. M. Duschka, "Infomaster: An Information Integration System", Proceedings of ACM SIGMOD Conference, May 1997.

[9] B. Koo and M. Fischer, "Feasibility Study of 4D CAD in Commercial Construction", *Journal of Construction Engineering and Management, ASCE*, vol. 126(4), 2000, pp. 251-260.

[10] D. Liu, J. Cheng, K. Law, and G. Wiederhold, "Ubiquitous Computing Environment for Engineering Services", Unpublished manuscript, 2002.

[11] D. Liu, K. Law, and G. Wiederhold, "CHAOS: An Active Security Mediation System", Proceedings of International Conference on Advanced Information Systems Engineering, LNCS, vol.1789, B. Wangler and L. Bergman (eds.), Springer-Verlag, 2000, pp. 232-246.

[12] D. Liu, K. Law, and G. Wiederhold, "Analysis of Integration Models for Service Composition", Proceedings of Third International Workshop on Software and Performance, Rome, Italy, July 2002.

[13] M. D. McIlroy, "Mass Produced Software Components", *Software Engineering, NATO Science Committee*, January 1969, pp. 138-150.

[14] L. Melloul, D. Beringer, N. Sample, and G. Wiederhold, "CPAM, a Protocol for Software Composition", Proceedings of Conference on Advanced Information Systems Engineering '99, Heidelberg, Germany, June 1999.

[15] R. Otte, P. Patrick, and M. Roy, "Understanding CORBA", Upper Saddle River, New Jersey, Prentice Hall, 1996.

[16] Y. Papakonstantinou, "Query processing in heterogeneous information sources", Stanford University, PhD thesis, 1996.

[17] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina, "Object Fusion in Mediator Systems", Proceedings of VLDB Conference, 1996.

[18] D. L. Parnas, "On the Criteria to be Used in Decomposing Systems into Modules", *P. Freeman and A. I. Wasserman*, Tutorial on Software Design Techniques, IEEE Computer Society Press, 1983.

[19] E. Pitt and K. McNiff, "java.rmi: The Remote Method Invocation Guide", Addison Wesley, 2001.

[20] N. Sample, D. Beringer, L. Melloul, and G. Wiederhold, "CLAM: Composition Language for Autonomous Megamodules", Proceedings of Third International Conference on Coordination Models and Languages, Amsterdam, April 1999.

[21] N. Sample, P. Keyani, and G. Wiederhold, "Scheduling Under Uncertainty: Planning for the Ubiquitous Grid", Proceedings of Fifth International Conference on Coordination Models and Languages, 2002.

[22] G. Wiederhold, "Mediators in the Architecture of Future Information Systems", *IEEE Computer*, March 1992, pp. 38-49.

[23] G. Wiederhold, D. Beringer, N. Sample, and L. Melloul, "Composition of Multi-site Services", Proceedings of IDPT'99, Kusadasi, Turkey, June 1999.

[24] G. Wiederhold, P. Wegner, and S. Ceri, "Towards Megaprogramming", *Comm. ACM*, vol. 35(11), Nov 1992, pp. 89-99.

[25] P. Wyckoff, S. W. McLaughry, T. J. Lehman, and D. A. Ford, "TSpaces", *IBM Systems Journal*, vol. 37(3), August 1998, pp. 454-474.