In Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering (SEKE2009), Boston, USA, July 1-3, 2009.

Using Service-Oriented Architectures for Socio-Cultural Analysis

David Garlan, Kathleen M. Carley, Bradley Schmerl, Michael Bigrigg, and Orieta Celiku
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh PA 15213 USA
+1 412 268 5056
{garlan, carley, schmerl, bigrigg, orietac}@cs.cmu.edu

ABSTRACT

An emergent domain that raises some unique engineering challenges is that of software architectures to convert large bodies of unstructured data to human-usable knowledge, such as in the domain of socio-cultural information analysis. We report on an architecture based on Service-Oriented Architectures that we are applying in this domain. We list the requirements that such an architecture must support, describe our architecture for addressing them, and outline what we believe are the important engineering and research issues that must still be overcome.

1. INTRODUCTION

One of the most striking features of today's computing landscape is the exponentially increasing volume of information that is becoming electronically accessible. Finding ways to use this information effectively – to access it in its myriad forms and formats, to extract insight and knowledge, and to update those results as information changes – is a significant software engineering challenge. While systems such as search engines provide important capabilities for accessing and organizing some of this information, there remains a large gap between the huge corpus of largely-unstructured data and human-usable knowledge.

To address this problem a number of researchers are developing a new breed of powerful information analysis tools, called Dynamic Network Analysis, that include capabilities to do natural language processing on large volumes of data, techniques for extracting key relations between entities, and mechanisms for analyzing, filtering, forecasting and visualizing this information as an ecology of evolving networks including social, knowledge and activity networks [1] [2][5]. For example, as detailed later, such tools can be used by scientists to understand change in the Sudan, military or intelligence agencies to understand how to interact with allies, organizational analysts to examine changing connections among firms and products as evinced by news stories.

Unfortunately, as implemented today, such tools have a number of severe limitations.

Stovepiped systems: Current information analysis systems are often large, monolithic programs that make it difficult to compose their constituent capabilities with those of other systems.

Restricted models: Current systems can only work with a limited set of information models that make interchange and coordination problematic.

Lack of configurability: Current systems are often tuned to a specific class of analyses or information abstraction, and can only be tailored by users having detailed low-level knowledge of their parameters of operation.

Idiosyncratic interfaces: Each system adopts its own interface conventions, requiring users to learn different interaction conventions for each tool.

Platform restrictions: Current systems often make rigid assumptions about the specific platform that they can work on, making it difficult to use them in a distributed setting, or to balance the need for secure co-location with access to external capabilities.

Duplicated functionality: Current systems are often engineered to work in stand-alone fashion, requiring each system to duplicate functionality also required by others – for example, in support of graphical interfaces, data management, communication, security, etc.

In this paper we describe an approach that addresses these problems. The key idea is the use of a common integration architecture, based on service-oriented architectures, that handles the special requirements for flexible information analysis. Critical to the success of this approach is the strong involvement of the community of tool developers and tool users in identifying standard models and ontologies to support interoperability, within the service-oriented context. Focusing specifically on the domain of socio-cultural analysis, in the remainder of this paper we list those special requirements, describe our architecture for addressing them, and outline what are the important engineering and research issues that must still be overcome.

2. SOCIO-CULTURAL ANALYSIS

Socio-cultural analysis involves understanding, analyzing and predicting the relationships in large complex social systems. Complex social systems are typically represented as dynamic networks that relate entities in the system (e.g., people, knowledge, actions) to each other. The emergent field of dynamic network analysis (DNA) is centered on the collection, analysis, understanding and prediction of dynamic relations in and among networks, and the impact of such dynamics on individual and group behavior. DNA facilitates reasoning about real groups as complex dynamic systems that evolve over time. Within this field computational techniques, such as machine learning and artificial intelligence, are combined with traditional graph and social network theory, and empirical research on human behavior, groups, organizations, and societies to develop and test tools and theories of relational enabled and constrained action.

The application of DNA techniques to a large complex social system, such as the US Army or gang networks, entails a series of procedures. First, one needs to gather the relational data. One approach for doing this is to extract relations from a corpus of texts such as public domain items like web pages, news articles, journal papers, stock holder reports, community rosters, and various forms of human and signals intelligence. Second, the extracted networks need to be analyzed. That is, given the relational data, identifying key actors and sub-groups, points of vulnerability, and so on. Third, given a set of vulnerabilities, we want to ask what would happen to the system were the vulnerabilities to be exploited. How might the networks change with and without strategic intervention?

The center for Computational Analysis of Social and Organizational Systems (CASOS) at Carnegie Mellon University has been engaged in developing methods and tools to achieve these activities. The tools are interoperable and can be organized as a chain to extract networks from texts, analyze these networks, and then engage in what-if reasoning. This tool suite takes into account multi-mode, multi-link, and multi-time period data including attributes of nodes and edges. This toolset contains the following tools: AutoMap [4] for extracting networks from natural language texts, ORA for analyzing the extracted networks [3], and Construct for what-if reasoning about the networks.

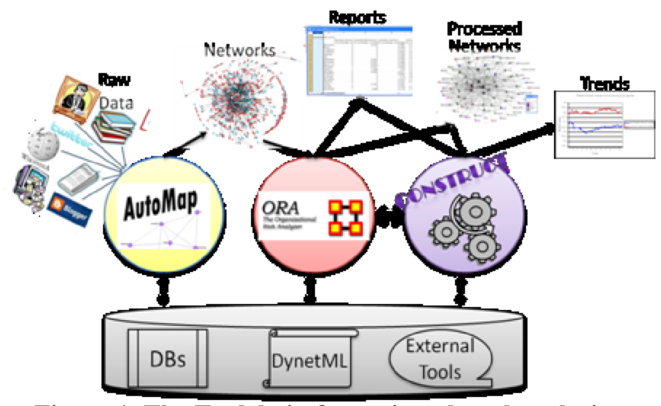

Figure 1. The Toolchain for socio-cultural analysis developed by the CASOS group at Carnegie Mellon.

Figure 1 provides an example of the way that these tools are integrated into a tool chain. Each of the tools (Automap, ORA, Construct) are monolithic programs. They are loosely integrated through an XML format called DyNetML, which is an interchange format for rich social network data.

While the existing tools are powerful, their interaction in terms of a tool chain is coarse-grained because the applications themselves are monolithic. Thus, expert knowledge is required to use each tool. The information shared amongst them in terms of traceability or reproducibility is impoverished, meaning that conducting analysis when new information becomes available, or on entirely new but related datasets, is difficult. Additionally, linking tools developed by other members of the DNA community is challenging.

3. ARCHITECTURAL DRIVERS

To overcome the limitations outlined above, we require a platform and architecture within which socio-cultural analysis tools can be integrated, configured, extended and programmed by end users, and tailored to specific domains without extensive low-level expertise.

Specifically, data collection, analysis and modeling tools must reside within an architecture that supports six key requirements [9].

Heterogeneity in data sources, analytical models, analysis mechanisms, and end-user needs. As the use of these systems expands, we can assume increasingly diverse sets of elements that will need to be integrated into future systems.

Flexible configuration to (a) assemble existing components (data sources, data coding tools, analysis tools, visualization tools, and simulation models) in new ways depending on the type of data available and the kind of analysis needed, (b) add components to support new capabilities, and (c) allow users to easily experiment with new analysis paths, workflows, and simulations without detailed technical knowledge of the tools and underlying technologies.

High performance processing and manipulation of large, diverse, and distributed sources of data to allow interactive exploration and analysis.

Traceability of analytic output to sources and intermediate models and records in order of processing, to allow analysts to compare results of analysis to ground and derived truth, and to adjust the fidelity and parameters of their models.

Security and privacy of potentially sensitive information that is used in the analyses.

What-if reasoning by enabling the analyst to change how data is coded, what data is coded, what virtual experiments in the simulations are run, track the impact of those decisions, set up multiple choice paths to run in parallel to facilitate rapid assessment making use of data-farming techniques, and replay facilities for desired procedures so that future data sets can be analyzed.

4. RELATED WORK

A number of modern integration frameworks make services accessible. One of these is Web Services technologies that provide standards for interaction, including SOAP [8] and REST [6]. Although standards for Internet-base invocation are a first step towards service integration web service infrastructure does not support ways to define workflows of services, web service lifecycle issues, or dynamically locating of services – capabilities necessary for our domain..

Service-oriented architecture (SOA) [10][11] aims to address some of these issues by defining standards for workflows (called *orchestrations*), policies for governance, and facilities for service discovery. Many definitions and implementations of SOAs aim to be applicable for general business domains. While SOAs provide important capabilities for service coordination, by themselves they have limitations that must be overcome to be applicable to our domain: (a) orchestration scripts define low level coordination, and are not appropriate for use by non-technical users; (b) support for agile and dynamic workflows is often impoverished in existing technologies; and (c) existing technologies have performance-related issues that make them difficult to use in context (such as ours) where large flows of data must be efficiently processed.

As an example of these limitations, consider the standard methods for defining SOA workflows: the Business Process Execution Language (BPEL) [12] and Business Process Modeling Notation (BPMN) [15]. BPEL and BPMN are graphical programming languages that allow specification of general business processes. BPEL especially is intended to be interpreted and therefore requires the detail of a programming language and the skill of a programmer. To address this, the SOA community has introduced a more abstract notation for defining orchestrations called BPMN. The goal is that orchestrations defined in BPMN can be understood by all business users. However, business analysts are still required to define orchestrations in BPMN, rather than non-technical users.

Taking these limitations into consideration, it is necessary to augment SOA technology and concepts to particular domains. For socio-cultural analysis, this is particularly relevant because it is necessary that services should be ultimately assembled by non-technical field analysts who have expertise in the domain they are trying to analyze, but little expertise in programming. Thus, one of the challenges is identifying the abstractions and protocols that should be built on top of SOAs, but that are tailored to the needs of the socio-engineering analysis domain. Furthermore, we require an easy-to-user approach for service assembly.

Among the other technologies that attempt to provide general-to-use workflow definition in other domains are Yahoo! Pipes [18] for defining mashups on the Internet and uDesign for defining activities in pervasive computing environments [17]. Our work is similar in spirit to these efforts, but specialized for socio-cultural analysts.

There are also several implementations of infrastructures that provide an extensible framework for socio-cultural analysis, particularly in the intelligence analysis domain. For example, COMPOEX [7] provides an integration architecture for assisting military commanders and civilian leaders in selecting models and analyses to plan and execute military campaigns. The goal of our approach is to develop a framework that is targeted more generally at socio-cultural analysis (not limited to military and intelligence activities). Furthermore, COMPOEX is focused on simulation once models have been developed, whereas our approach also includes the ingestion of raw data to produce the models.

5. ARCHITECTURE DESCRIPTION

From a functional perspective, information analysis systems have a common flow of processing: Data is input into the system originating from many sources. These sources, including public news reports and intelligence reports, are typically written in natural languages. These inputs need to be processed and marked up to identify key concepts that are needed for intelligence analysis, and output in a form that is suitable for simulation and analysis. The concepts, or *entities*, of interest in the inputs are mostly fixed for the domain, and include knowledge and agents. The output from processing is a model represented as a graph with relationships among agents and knowledge. Models can then be analyzed and viewed in a variety of ways. Further, simulation or what-if analysis may be performed to create a set of related models. Insight gained is then used to refine the data processing and analysis. Finally, reports of various kinds can be generated and stored.

Our architecture naturally follows this decomposition of activities, while building on best practices in the engineering of service-oriented architectures and new techniques to support end-user programming and system configuration. The basis for the architectures is (a) the use of a multi-layered system capturing the essential flows of information and processing, and (b) support for flexible orchestration, coordination, and transformation.

Multi-layered system. The domain of dynamic network analysis naturally lends itself to a four-tiered system shown in Figure 2:

1. Data Layer: a set of heterogeneous data sources. These include databases, wire feeds, intelligence streams, email corpuses, web sites, historical documents, etc. These form the raw inputs to the system, and may be relatively stable

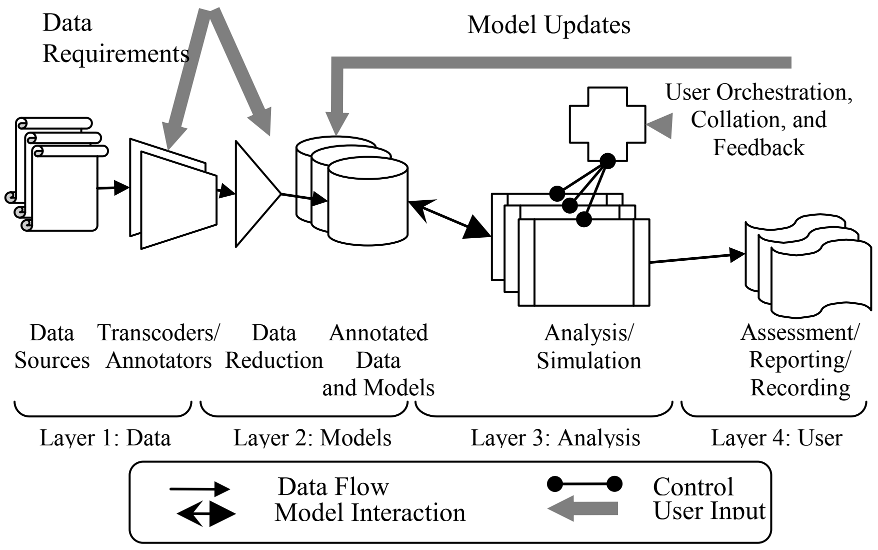

Figure 2. High Level View of SORACS Architecture.

(as in the case of historical databases), changeable (as in the case of web sites), or highly dynamic (as in the case of wire feeds and intelligence streams).

2. Model Layer: a set of high-level models, which represent information extracted from the first layer. This layer will be populated by a variety of models including annotated documents (e.g., as a result of natural language processing) and network models (e.g., ORA Meta-Networks 0) representing relationships between key entities in the domain of discourse. Bridging these two layers is a set of model extractors (e.g., Automap [4], CEMap) that effect the transformation of raw data into theoretically richer forms for analysis.

3. *Analysis Layer:* populated by a collection of analysis tools (including ORA, UCINET, Pythia). Such tools will reside as semi-independent components, interacting with models in the second layer and generating input for and analyzing results from tools in the fourth layer through a standard set of protocols. This layer also includes simulations, such as Construct, for forecasting and exploring alternative histories and futures. Simulation and analysis tools will have well-defined interfaces, and be integrated into a service-oriented framework that enables registry and lookup in support of dynamic configuration and incremental reconfiguration.

4. User Layer: the end-user layer, which provides an interface for users to interactively view the analysis results, configure new analyses, trace analysis to sources, and generate reports. Capabilities in this layer fall into two categories. One is output of analyses and simulations from the lower layer (such as ORA reports); this also allows the user to fine-tune the parameters of these (e.g., specifying whether reports will be generated for the entire network or key entities). The other supports orchestration, allowing users to put together new combinations of processing that determine both the nature of model generation and the way in which analysis/simulation services are assembled.

Orchestration, Coordination, and Transformation. To enable dynamic integration and configuration of the components into the four layers requires a number of mechanisms for orchestration, coordination and transformation.

First is the ability to automatically produce full analysis pipelines from end-user descriptions that specify at a high level of abstraction what kinds of processing is needed to produce a particular kind of analysis. Using a combination of graphical and textual inputs, users will be able to specify and configure a collection of data transformation and analyses services to support their needs. The system automatically assembles these parts, providing the "glue" for connecting the parts.

Second is the ability to select the appropriate transcoders to bridge data-mismatch assumptions between components. Building on earlier research in document transformation, the platform will include a registry of transcoders and filters, together with algorithms that find an optimal chain of transformations (based on information fidelity metrics) [13][14]. While manual fine-tuning of these transformations may be necessary in some cases, we expect that the majority can be done automatically.

Third is the use of a standards-based service-oriented architecture for the analysis tools. Specifically, the architecture contains a registry of the services provided by the suite of existing and newly-developed tools. As users compose dataflow paths for analysis, services are automatically selected and composed in appropriate ways (in some cases requiring the automatic interposition of data transformers).

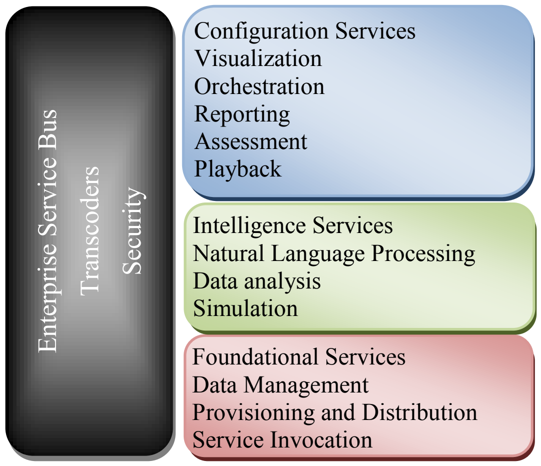

Figure 3. Layered Service Oriented Architecture.

In addition the integration framework will provide a set of common services for communication, security, provenance, and mismatch avoidance.

The organization of these services follows a fairly standard approach used by modern SOAs. As illustrated in Figure 3, we organize the services into three groups. At the bottom are foundational services, including data management, provisioning and distribution, and service invocation. Next are services that provide the meat of the processing. This is where tools specific to the data transformation, analysis, and simulation for understanding and interpreting information. At the top are configuration service, which support the specification and tailoring of computations, as well as certain visualization services for presenting information to a user. Communication and coordination is handled by an Enterprise Service Bus, which supports service discovery, look-up, enlistment, and interaction.

6. IMPLEMENTATION

The tools described in Section 2 encompass a wide range of activities required by socio-cultural analysts. They are therefore good candidates for the initial investigation of an architecture in this domain. In this section we outline our current implementation of an initial version of the above architecture using the existing tools developed by CASOS. The initial step in our investigation is identifying and implementing the individual services that can be derived from these tools from which orchestrations can be derived.

Automap analyses textual data. It can process data lexically (e.g., by removing extraneous white space, splitting sentences) and grammatically (e.g., by identifying and extracting parts of speech, resolving pronouns). Services derived from Automap can be considered lexical services and grammatical services or simply a combined textual service. Service-able functionality also exists within ORA. ORA contains many different common network science metrics and grouping algorithms (e.g., CONCOR, Newman, FOG, Johnson Hierarchical, Attribute based). It also has facilities for generating, editing, visualizing, and detecting changes in networks. Construct includes services such as experimental design construction, report generation and simulation. In all cases, the services can be provided at a fine or coarse grain level in which analysis operations are provided as a graph analysis service.

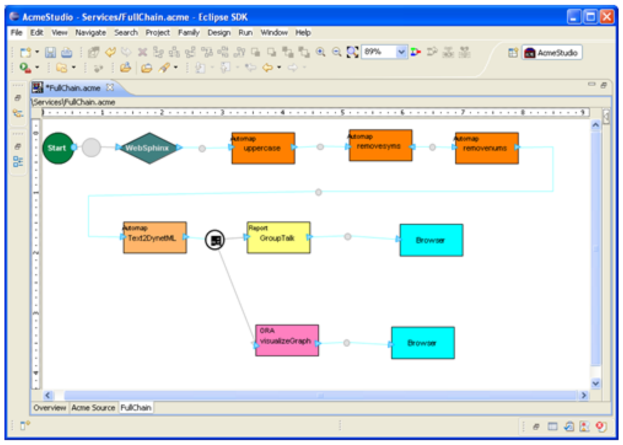

Figure 4. The AcmeStudio orchestration interface.

Once the services were identified, we entered a rearchitecting phase that involved making the services more decoupled in the tools. For Automap, this involved making available each of the lexical analysis components as independent components decoupled from the existing user interface; for ORA it involved decoupling the analyses and reports desired from the user interface. This process is ongoing, and we present a discussion of the challenges in the following section.

Once the functions that could be used as services were identified and isolated in the code, we then implemented them as services using the standard approach to web service definition (WSDLs) and using an open source application server to make these available as web services. For this phase, we used Apache Tomcat as our application server, and Apache CXF to streamline our implementation of the existing Java implementations as web services.

Our initial version of the orchestration interface for this domain has been written as a plug-in to AcmeStudio (see Figure 4) [16]. We have defined an architectural style for this domain, detailing each of the services as particular component types, and defining connector types that are specific to this domain allowing the components to be chained together. The orchestration backend of the plug-in takes architectural specifications and produces BPEL definitions that can be uploaded and executed by a BPEL engine. We currently use Apache ODE as our orchestration engine for executing these orchestrations.

7. DISCUSSION & CONCLUSION

In this paper, we have discussed the requirements and design for an architecture for socio-cultural analysis. We have also described an initial implementation that provides a domain-specific approach to defining workflows that is built upon existing SOA standard technologies. In doing this, we encountered a number of additional issues and technological challenges that are imposed by this domain.

What are the appropriate connectors for long-lived service invocations? While the intent in SOAs is for service orchestrations to execute over long periods of time (e.g., many years), support for invocation of long running individual services is low. BPEL provides a way to deal with long running services through asynchronous invocation; the BPEL program then polls for results. The domain of socio-cultural analysis must support long running services because the amount of data being analyzed is typically large, and the analyses complex. However, the simplest model for analysts is to define call-return type connections between services. There needs to be a balance between conceptual ease for analysts and technical detail for developers.

Related to this is the issue of *control vs. usability*. How do we define services that have the appropriate interfaces for use in the common case, but still provide enough control for detail-oriented analysts? For many analyses in this domain, there are a set of common or default parameters that are sufficient in most cases, but we still wish to provide control for the less common cases.

Traceability and reproducibility is another challenge that we need to address. SOA platforms provide some coarse-grained traceability through provenance mechanisms. However, we require finer grained traceability so that analysts can query how analysis conclusions were made, and the reliability of the data that they were based on. Furthermore, we require the ability to rerun orchestrations with minor changes in data. Currently, there is no mechanism for providing incremental analysis or data-caching to reduce the time these analyses take.

These challenges are areas of future work. Moreover, we are planning to extend the current prototype in a number of directions, including: a) providing automated transcoding between data formats, and b) allowing the definition and reuse of workflow templates.

Another area of future work evaluating the effectiveness of the architecture for the socio-cultural domain. The architecture described in this paper matches the way that analysts think about the problem; we believe that the technical aspects balance the needs of users and developers of tools. We have some confidence that the architecture is correct through integration of existing CASOS tools. Future work will involve integrating additional components, and developing a more functional user interface for analysts to use.

ACKNOWLEDGEMENTS

This work is supported in part by the Office of Naval Research (ONR), United States Navy, N000140811223 as part of the HSCB project under OSD. Additional support for the core CASOS tools was provided by National Science Foundation (NSF) Integrative Graduate Education and Research Traineeship (IGERT) program, NSF 045 2598, the Air Force Office of Scientific Research, FA9550-05-1-0388 under a MURI on Computational Modeling of Cultural Dimensions in Adversary Organizations, the Army Research Institute W91WAW07C0063, the Army Research Lab DAAD19-01-2-0009, the office of Naval Research (ONR), N00014-06-1-0104 the Army Research Office W911NF-07-1-0060, and CASOS - the center for Computational Analysis of Social and Organizational Systems at Carnegie Mellon University (http://www.casos.cs.cmu.edu). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Office of Naval Research, the Office of the Secretary of Defense, the Army Research Lab, the Air Force Office of Sponsored Research, the Army Research Office, the National Science Foundation or the U.S. government.

REFERENCES

[1] Carley, K.M., Dynamic Network Analysis" Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers. Breiger, R., Carley, K., Pattison, P. (eds). Committee on Human Factors, National Research Council, 2003.

[2] Carley, K.M., A Dynamic Network Approach to the Assessment of Terrorist Groups and The Impact of Alternative Courses of Action. In Visualizing Network Information Meeting Proceedings RTO-MP-IST-063, Neuilly-sur-Seine, France, 2006.

[3] Carley, K.M., Columbus, D., DeReno, M., Reminga, J., and Moon, I.-C. ORA User's Guide 2007. Carnegie Mellon University School of Computer Science Institute for Software Research Technical Report CMU-ISR-07-115, 2007.

[4] Carley, K.M., Columbus, D., DeReno, Diesner, J., and Sebula, N. AutoMap User's Guide 2007. Carnegie Mellon University School of Computer Science Institute for Software Research Technical Report CMU-ISR-07-114, 2007.

[5] Carley, K.M., Diesner, J., Reminga, J., and Tsvetovat, M, Toward an Interoperable Dynamic Network Analysis Toolkit, DSS Special Issue on Cyberinfrastructure for Homeland Security: Advances in Information Sharing, Data Mining, and Collaboration Systems, 43(4), p. 1324 – 1347, 2007.

[6] Fielding, R. Architectural Styles and the Design of Network-based Software Architectures. Ph.D. Thesis, University of California, Irvine, 2000.

[7] Kott, A. and Corpac, P.S. COMPOEX Technology to Assist Leaders in Planning and Executing Campaigns in Complex Operational Environments. In 12th International Command and Control Research and Technology Symposium,, 2007.

[8] Mitra, N., and Lafon, Y. (eds) SOAP Version 1.2 Specification (Second Edition). http://www.w3.org/TR.2007/REC-soap12-part0-20070427, 2007.

[9] Naval Research Laboratory (NRL) Diplomatic, Informational, Military, Economic (DIME) Political, Military, Economic, Social, Information, Infrastructure (PMESII) Modeling Requirements Workshop. John Hopkins University, Maryland, 2007.

[10] Newcomer, E., Lomov G. *Understanding SOA with Web Services.* Addison Wesley, 2005.

[11] OASIS. OASIS Reference Model for Service Oriented Architecture 1.0. http://www.sei.cmu.edu/pub/documents/05.reports/pdf/05tn014.pdf. 2006.

[12] OASIS. Web Services Business Process Execution Language Version 2.0, April 2007. URL http://docs.oasis-open.org/wsbpel/2.0/wsbpelv2.0.html.

[13] Ockerbloom, J. Exploiting Structured Data in Wide-Area Information Systems, Ph.D. Thesis. Carnegie Mellon University Technical Report CMU-CS-95-184, August, 1995

[14] Ockerbloom, J. Accommodation: The Key to Making Widely Adopted Composable Systems. In Proc. Workshop on Compositional Software Architectures, Monterey, CA, 1998.

[15] Object Management Group. The Business Process Modeling Notation (BPMN) Version 1.2. January, 2009. URL: http://www.omg.org/docs/formal/09-01-03.pdf.

[16] Schmerl, B., and Garlan, D. AcmeStudio: Supporting Style-Centered Architecture Development (Research Demonstration) In Proc. the 26th ICSE, Edinburgh, Scotland, May 2004.

[17] Sousa, J.P., Schmerl, B., Poladian, V. and Brodsky, A. uDesign: End-User Design Applied to Monitoring and Control Applications for Smart Spaces. In Proceedings of the 2008 Working IFIP/IEEE Conference on Software Architecture, Vancouver, BC, Canada, 18-22 February 2008

[18] Yahoo!, Inc. Yahoo Pipes. http://pipes.yahoo.com/pipes. Accessed March, 2009.