Letter from Dan Power, President of MWAIS

The Midwest chapter of the Association for Information Systems (MWAIS) was founded in Fall of 2005. With the help of many people, especially a core leadership team, the organization seems firmly established. The upcoming MWAIS 2007 conference will mark our first leadership transition. At the business meeting, Deepak Khazanchi will assume the duties of President of MWAIS for 2007-2008 and I will take on the role of Past President. My good friend Ilze Zigurs has served as founding Past President and she will continue to serve as an At-Large Board member.

Every year we will need new colleagues to step forward to serve our shared interests and promote the importance of studying information systems and information technology. From my vantage point we have an enthusiastic, talented pool of people who are ready to serve and I'm sure that group will grow as MWAIS expands its membership.

So what are my hopes and dreams for the future of MWAIS? The broad purpose of MWAIS is to help meet the professional needs of Midwest U.S. members of AIS. "The goal of the Chapter is to promote the exchange of ideas, experiences, and knowledge among scholars and professionals in the Midwest U.S. engaged in the development, management, and use of information and communications systems and technology."

I hope that MWAIS will help members stay current with the rapidly evolving software technologies. Perhaps MWAIS can help members cope with this challenging task. Most of us have limited time and resources to learn new practical IS/IT knowledge and skills. Additionally, I hope that MWAIS will help develop meaningful professional relationships with colleagues. We are generally facing the same challenges and issues and MWAIS can provide opportunities to discuss the issues and search for shared approaches and joint action. MWAIS can provide the forum where we can share research and get helpful feedback.

I envision MWAIS being more than an annual conference and business meeting. With new tools for "mass" collaboration and video conferencing, I hope MWAIS will become an active virtual community. Through this, MWAIS members will cooperate to insure the quality of instruction and research in Information Systems. With the advent of web-based courses, we need to actively evaluate how faculty jobs are changing and how we can insure the quality of what we teach and certify what our students have learned.

Finally, I believe MWAIS can help institutionalize the Information Systems discipline and promote the importance of understanding information technologies and information systems.

Dan Power
Editor, DSSResources.COM
Professor
University of Northern Iowa
Adopting Service Oriented Architectures for Designing Information Systems

Matt Germonprez
University of Wisconsin – Eau Claire

Introduction

Systems design is subject to changing environments including new approaches, techniques, and technologies. How system designers are supposed to utilize these amorphous environments is vague and we can not provide a ‘how to’ guide that designers can follow pragmatically in the successful design of a new system. What we can provide is a post hoc view of the choices that designers made in the selection of certain design environments over others. In this paper we explore the emerging design environment of service oriented architecture (SOA), an architecture that explicitly represents the use of components across the HTTP protocol. Specifically, we ask why designers are driven to design in this environment in spite of technical inefficiencies that come with using SOA.

The SOA design environment supports designer selection and modification of functions to be assembled into a larger system. It is an environment that enables designers to select and integrate pre-built technology functions in the ongoing creation and recreation of unique information systems. In addition, flexibility that comes with the SOA environment is realized at the customer, business logic, and data layers associated with system design. SOA is adopted around such technologies as Enterprise Resource Planning systems, operating system desktops, and word processing software where users pick and choose parts they want without having to create the parts themselves. Specific, reusable functions can be integrated by designers in the formation of numerous unique technologies (Baldwin and Clark 2003). The SOA environment is billed at being beneficial on many fronts yet carries with it a certain amount of technical overhead that other design environments do not. This leads us to our single research question: Why do designers adopt the SOA design environment in light of its technical shortcomings?

In the remaining sections we will discuss SOA and the technical inefficiencies associated with it. We then present our findings from three interviews and site-visits in response to our research question. At the end of the paper, we present our concluding remarks.

Service Oriented Architecture

As information systems designers perform new tasks, form new groups, or develop new processes, architectures should support their ability to make changes in the functionality of their applications, rather than simply support a set of pre-specified, anticipated designer actions. This flexibility relies on a component model and the evolution of component relationships during the ongoing use of a technology (Domingos and Martins 2000; Wang and Haake 2000). Service oriented architectures are, in part, based on the principles of information systems architecture that enable designers to be able to select from a set of services during use (Morch and Mehandjiev 2000; Hummes and Merialdo 2000). Service oriented architectures support designer discovery of services distributed across nodes within a network, and at each node, designers can integrate specific, reusable services into their applications and thus form unique, customized applications. In a service-oriented architecture, designers are envisioned as continually integrating smaller, independent components into increasingly complex, integrated systems, in order to manage increasing levels of innovation and growth.

As an example of SOA in practice, we illustrate web-based SOA. Web-based SOA is specified through the standards of the Simple Object Application Protocol (SOAP), the Web Services Description Language (WSDL), and the Universal Discovery and Descriptive Interface (UDDI). All standards rely on the Extensible Markup Language (XML) and are highly interrelated. From a design perspective, the only critical component is the SOAP message as it is through this message that data is exchanged between system components. Yet many systems do rely on all three parts. If the UDDI and the WSDL are bypassed interaction with a service can only occur through undocumented knowledge of where the service is located and how the service works.

The SOAP message itself is a markup document that carries data to be used by another application. The SOAP message is comprised of three core elements: The envelope, the body, and the data payload. In this example, a Google web service is called passing the information that the query is ‘University,’ two results are requested, and a license key for accessing the service.

```xml
<soap:Envelope xmlns:typens="urn:GoogleSearch">
<soap:Body>
<doGoogleSearch>
<key>QM6kFR1+UJlrEQ4smLr</key>
<query>University</query>
<maxResults>2</maxResults>
</doGoogleSearch>
</soap:Body>
</soap:Envelope>
```
If this service was passed to Google, the site would respond with a similarly packaged SOAP message containing the results. This provides access to a part of a larger system and designers can access those parts without concern of how a service is programmed or what language it is programmed in. In this example, a designer and user of this part of Google’s larger computing infrastructure, you become capable of consuming the programmatic pieces needed to build the system by simply knowing where they are located and how to interact with them.

SOA has a long history in computing stemming back to early systems such as CORBA and DCOM where users were able to build services that could be consumed by others within an organization. Both technologies worked well in homogeneous computing environments but scaled poorly to the heterogeneous environment of the web and highly distributed organizations. The XML based packaging system associated with SOA has proven to be highly beneficial in abstracting the programmatic details of a service down to a universally readable interface. Changes to a service at the programmatic level do not affect its interface, thereby enabling system designers to choose services based on abstract interfaces and not specific languages.

**Service Oriented Architecture Inefficiencies**

An important part of SOA is the aforementioned use of XML as a data packaging system. The effect that this has on performing programmatic tasks is significant due to the fact that every SOA message requires, at minimum, an exchange of data between two applications and as part of that exchange, an application layer protocol check to determine the contents of an SOA message.

To demonstrate this effect, we wrote two applications representing System 1 and System 2. The first application relied on the Perl programming language to perform three different tasks. The tasks were (1) to write one piece of data to a single table database, (2) create a hash of a string of data, and (3) create a hash of a string and write it to a system file. In the case of SOA, the exact same code was wrapped in the proper SOA wrapper and called.

The code is similar in both cases, the SOA approach requires some programmatic overhead required to call the service. In this example no interface is read prior to accessing the service. That is, additional overhead would be expected if the SOA code had to read a programmatic interface (WSDL) prior to interacting with the service. When we ran the code (n=400), we see that SOA increased technical deficiencies. Table 1 illustrates the differences between the two applications.

<table>
<thead>
<tr>
<th>Applications</th>
<th>Time (Seconds)</th>
<th>User CPU Utilization</th>
<th>Server CPU Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-SOA Database</td>
<td>.04</td>
<td>0.086%</td>
<td>0.012%</td>
</tr>
<tr>
<td>Non-SOA Hashing</td>
<td>0</td>
<td>0.002%</td>
<td>0%</td>
</tr>
<tr>
<td>Non-SOA Hashing/Write to File</td>
<td>.008</td>
<td>0.006%</td>
<td>0.003%</td>
</tr>
<tr>
<td>SOA Database</td>
<td>3.2</td>
<td>1.674%</td>
<td>0.066%</td>
</tr>
<tr>
<td>SOA Hashing</td>
<td>2.4</td>
<td>1.562%</td>
<td>0.056%</td>
</tr>
<tr>
<td>SOA Hashing/Write to File</td>
<td>2.4</td>
<td>1.56%</td>
<td>0.058%</td>
</tr>
</tbody>
</table>

On average, SOA provided an additional 2-3 second overhead on running small applications as well as an increase in CPU utilization at both the client and the server (again, all findings were significant at p=.001).

<table>
<thead>
<tr>
<th>SOA Overhead for Respective Programs</th>
<th>Time (Greater for SOA)</th>
<th>Total CPU (Greater for SOA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>3.1 seconds</td>
<td>1.642%</td>
</tr>
<tr>
<td>Hashing</td>
<td>2.4 seconds</td>
<td>1.618%</td>
</tr>
<tr>
<td>Hashing/Write</td>
<td>2.4 seconds</td>
<td>1.609%</td>
</tr>
</tbody>
</table>

It is expected that the findings would hold scaling up from the fairly simple application presented here as what is presented here represents actual services used on the web. For sample services see xmmethods.com, strikeiron.com, or bindingpoint.com.

**Question and Methodology**

These findings represent a significant difference between the two forms of development, showing that SOA built applications are significantly slower and require significantly greater CPU utilization than the same applications built using a non-SOA approach. On a small scale, these findings prove insignificant but in an organization with 10,000 transactions on one service a day, the results become more dramatic. With these findings, we return to the original research question:

*Why do designers adopt the SOA design environment in light of its technical shortcomings?*

To answer this question, we visited three companies and interviewed their chief designers involved in SOA design projects. The organizational units were comparable across organizations, representing the Information Technology divisions of their respective companies. The companies included one Fortune 500 insurance organization involved in redesigning traditional systems into an SOA.
environment; one consulting company who is using SOA for software distribution for a multi-national company, and one international trucking company who is using SOA for internal systems design.

In some situations, more than one designer was present at the site visit and interview. In all cases, the interview was focused on understanding why SOA was a chosen design environment for the aforementioned projects, in light of these technical deficiencies. While there was agreement with our earlier technical findings, the issues as to why SOA was chosen varied. In the next section we present our findings and illustrate consistency across all organizations.

Findings

In this section, we present the findings from our three site visits and interviews. The central ideas are listed in Table 3, showing the company and their primary purpose for using the SOA environment.

<table>
<thead>
<tr>
<th>Company</th>
<th>Primary Purpose for Using SOA and Definition of that Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>International Trucking</td>
<td><strong>Loose Coupling:</strong> Remove tight binding between programmatic modules in an information system.</td>
</tr>
<tr>
<td></td>
<td><strong>Centralized Business Logic:</strong> Manage the business logic layer more effectively through a centralized repository of services.</td>
</tr>
<tr>
<td></td>
<td><strong>Application Updates:</strong> Have software updates sent to users. Updates include changes to data types and service locations.</td>
</tr>
<tr>
<td></td>
<td><strong>Data Management:</strong> Managing how data logic layer connects to various databases.</td>
</tr>
<tr>
<td></td>
<td><strong>Locking Management:</strong> Regulating who has access to data layer.</td>
</tr>
<tr>
<td>Multi-National Consulting</td>
<td><strong>Loose Coupling:</strong> Remove tight binding between programmatic modules in an information system.</td>
</tr>
<tr>
<td></td>
<td><strong>Application Distribution:</strong> Provide access to services across organizational boundaries such that applications in one organizational division can use services from another.</td>
</tr>
<tr>
<td></td>
<td><strong>Proxy Classing:</strong> Create new business logic that can analyze an incoming request for a service and act as a proxy in regulating access to or from a service.</td>
</tr>
<tr>
<td></td>
<td><strong>Application Updates:</strong> Have software updates sent to users. Updates include changes to data types and service locations.</td>
</tr>
<tr>
<td>Fortune 500 Insurance</td>
<td><strong>Loose Coupling:</strong> Remove tight binding between programmatic modules in an information system.</td>
</tr>
<tr>
<td></td>
<td><strong>Centralized Business Logic:</strong> Ability to control the business logic layer in a heterogeneous environment.</td>
</tr>
</tbody>
</table>

Conclusions

This research has explored this and has proven interesting for three reasons. First, there are technical inefficiencies to SOA that are not otherwise present in non-SOA systems. Second is that companies are willing to take on take on that overhead in order to provide design flexibility across all business layers. The flexibility provided by SOA gives designers the ability to distribute and update applications, centralize business logic, and manage data in ways they were unable to with non-SOA development. Finally, these issues are particularly relevant in a heterogeneous computing environment where there is less control over how services are built and consumed. In all this research provides a clear base for why designers are using SOA in light of the technical inefficiencies. It also provides future designers reasons to consider the SOA environment when redesigning or building new systems.

Future research for this study entails understand the affect that accepting an SOA model will have on organizational design. It is expected that moving away from traditional application development towards a service based system will affect the way that business analysts and systems designers define their job roles and interact with other parts of the business. While the affects of SOA at a local development level may not be felt beyond technical inefficiencies, it is expected that the impact across an organization entails technical and social reconstruction.

REFERENCES


Domingos, H. and Marins, J. (1997). Coordination and tailorability issues in the design of a generic large scale groupware platform, Proceedings of Group ’97, Phoenix, AZ.


To better understand this competitive tension, consider as an example a project team that identifies low operating cost and high availability as both critical. The common objective for operating cost is to keep it as low as possible. The tension here exists because highly available systems tend to drive up operating costs through the need for redundant hardware and additional software to handle clustering and failover situations. It is not possible to achieve both criteria equally well. Left to be discovered later in the project, this type of tension easily becomes a show-stopper: a major issue that disrupts the team and the plan and that may even jeopardize the perceived success of the effort. But when examined and answered at the outset of the project the team can calmly and more cost effectively resolve the conflict.

The second way in which the design stance introduces clarity to the project is in providing a formal mechanism to identify and then to qualify or quantify non-functional requirements. A great deal of attention will be given to the core business functionality of the new system: workshops may be held, use case models may be drawn up, user interface designs will be sketched and prototyped. These functional requirements may imply specific nonfunctional requirements which, if not taken into consideration, may end up missing in the final system. You can’t have functionality that the system wasn’t designed to support, but at the same time you can’t provide the necessary support unless you know what support is required. The two go hand-in-hand. Too often, the existence and details of non-functional requirements are left to the experience of the architect, to the casual hallway conversation between technical staff, or to internal IT meetings that do not involve the business community.

Returning to our previous example, having identified the importance of high availability in the success of the system, we would as a natural course of developing the design stance take the opportunity to quantify exactly what the group means by “high availability” and more importantly, to document why this stance is important to the business goals of the system. In so doing, we make it possible to trace architectural decisions back to concrete, measurable business objectives and at the same time make it possible to better evaluate the competitive tensions that may exist with other criteria. We can consciously and logically determine the point at which the need to contain operating costs outweighs the benefit of high availability.

Common Design Stance Elements
If you examine the design stances across multiple projects you will notice that there are a number of characteristics that are common to all of them. These characteristics tend to be easily measurable and make sense as indicators of success. I find it helpful to organize them into three groups: run-time properties, design-time properties and usability properties.
The following run-time properties describe a system in operation and can be considered to affect the system’s performance agility – its ability to scale to ever-larger volumes of work while maintaining acceptable levels of response. Run-time properties are typically quantifiable through some discreet, objective unit of measurement.

**Response-time Performance:** Commonly measured through minimum, maximum and average response times that users are willing to wait for a response. When ranked high and especially when more than one rank higher than flexibility, this suggests an architecture with relatively less code and less abstraction.

**Throughput / Scalability:** The volume of messages or transactions that can be handled by the system, the rate at which the number of users and work may increase over time, the degree to which the system can be scaled through addition of processors, memory and storage, and the anticipated marginal costs of scaling up. For the application, this stance may imply clustered and/or stateless components that can more easily be distributed across a number of machines or perhaps smarter client side software than can pin communications to a stateful component.

**Security:** Capabilities having to do with user authentication, authorization, network access and intrusion detection.

**Survivability:** The ability of the production system to survive physical disasters, often through the use of multiple, geographically disperse operating facilities. In addition to the raw cost of these facilities, there may be additional operating costs and procedures to keep the sites synchronized.

**Reliability / Availability:** A measurement of how much down time can be tolerated, either accidental or scheduled, such as for maintenance or upgrades.

Design-time properties are relevant while the system is being developed or maintained. These affect the business and technical agility of the system; that is, its ability to adapt to changing business needs and technical environments. Although some of these are objectively quantifiable, many are more subjective and must be qualified differently. Examples include the following.

**Flexibility / Extensibility:** A measurement of the effort required to add new or to modify existing features. These are controlled through use of architectural elements such as abstraction, interface design and component semantics.

**Openness:** Measures the degree of difficulty in replacing a component of the system with one that is different but functionally equivalent and, where appropriate, the degree to which a component’s interface adheres to a known standard.

**Portability:** The degree to which changes in the platform technology – moving from Windows to Solaris, for example – affect the components of the system. The measurement scale might include No Change, Recompile, Code Change or Does Not Port.

The third category contains properties having to do with usability.

**Ease of Use:** Refers specifically to the measurable difficulty of using a system. These metrics must be reasonably objective in order to provide points of comparison. It is more useful to say that it takes 30 seconds on average to enter user information than to provide anecdotal comments about the same task. This stance refers not only to the needs of occasional or new users, but also to those who are experienced with it. It is possible that a user interface design may fail in production because ease-of-use features become cumbersome to the experienced user.

**Ease of Learning:** The ease with which the functions of the system can be learned. For example, a library’s on-line card catalog interface must be easily learnable since most casual users won’t remember how to use it from session to session.

Many of the design-time properties and both of the usability properties described are more subjective than objective in nature. It is difficult, for example, to quantify the degree to which a system is expected to be flexible. Instead, we can qualify it through the use of change cases.

A change case is similar to a use case. Rather than describing a current requirement, a change case defines an anticipated future requirement or requirement change based on known business strategies, goals and operational history. The importance of this thought process cannot be underestimated. Time is the enemy of every business systems development project because the business does not usually stand still while new systems are being built.

Consider for example financial services organizations that create, buy and sell financial instruments. This is a very dynamic industry that is constantly striving to innovate new products. Flexibility is clearly an important property in their design stance, but to truly understand why and what it means that flexibility is important, the participants in an IT project need to consider not only what types of products the business currently handles, but how to build a system that will be helpful in handling products not yet invented.

If the team follows through in the thought process, it may be decided that the best solution is one that enables the business users to define and organize elements of financial products interactively rather than requiring programmer intervention as the product line evolves. The resulting decrease in time to market for new products may end up being worth the price. If they can’t or don’t consider change cases, the resulting system will most likely have a shorter or more costly life and because of the resulting change in how its return on investment is perceived, may ultimately be considered a failure.
Midwest Association for Information Systems

INFORMATION TECHNOLOGY: TRANSFORMING THE HEARTLAND

Purpose
To share ideas and research addressing behavioral, organizational, and technical aspects of information systems.

Keynote Speakers
Professor Blake Ives
C.T. Bauer Chair in Business Leadership
C.T. Bauer School of Business
University of Houston

Ronald S. Swift
Vice President of Cross-Industry Solutions of Teradata
A Division of NCR

Awards
Top 3 papers will receive cash awards and an expedited review process for publication in the Mid-American Journal of Business

Program Co-Chairs
Arkalgud Ramaprasad
prasad@uic.edu

Matthew Nelson
mlnelso@listu.edu

Conference Co-Chairs
Rassule Hadidi
University of Illinois at Springfield

In Lee
Western Illinois University

Conference Location Chair
Michele Gribbins
University of Illinois at Springfield

For Complete Guidelines:
http://mis.uis.edu/mwaismwais2007

MW AIS
2007

CONFERENCE

2nd Annual Conference
May 18-19, 2007
University of Illinois-Springfield

Topics will include:
Accounting Information Systems
Computer Aided Software Engineering
Computing History
Data Warehousing and Mining
Database Management Technologies
Decision Support Technologies
Distance Learning Technologies
e-Collaboration in Organizations
Electronic Commerce Technologies Mgmt
End User Computing
Enterprise Resource Planning
Global IT Management
Human Computer Interaction
The Human Side of IT
Information Ethics Issues
Information Security Management
IT and Small Business
IT Business Value
Information Technology Education
Information Technology Standards
Intelligent Information Systems
IT Evaluation Methods and Management
IT Management in Healthcare
IT Teaching Cases
Knowledge Management
Medical Information Technology
Mobile Computing & Commerce
Multimedia Information Management
Object Oriented Technology
Open Source Technologies and Systems
Project Management & Information Technologies
Software Engineering
Software Process Improvement
Strategic IT Management
Telecommunication and Networking Technologies
Unified Modeling Language and Unified Process
Vertical Standards Consortia
Virtual Organizations and Society
The 3rd Annual MWAIS Conference

May 23-24, 2008

University of Wisconsin-Eau Claire

MWAIS Sponsored Journal Special Issues

Mid-American Journal of Business Special Issue based on 1st Annual MWAIS Conference (2006)

From the papers accepted for presentation at the 1st Annual MWAIS conference, 4 were invited to submit expanded versions for a fast-tracking opportunity with the Mid-American Journal of Business.

e-Service Journal Special Issue based on the 2nd Annual MWAIS Conference (2007)

Editor-in-Chief, Ilze Zigurs, has agreed to work with MWAIS in fast-tracking selected best papers from the 2nd Annual MWAIS Conference in 2007. Authors with topics of relevance to e-Service Journal will be invited to be part of the special issue. The journal covers a broad range of topics related to electronic services. For details, see the home page at http://www.e-sj.org/ or contact Ilze Zigurs at izigurs@mail.unomaha.edu.

MWAIS Sponsors

MWAIS would like to express its sincere appreciation to the following sponsors. Current and future MWAIS activities would not be possible without their generous support.

http://vig.prenhall.com/

http://www.bsu.edu/web/majb/
Acknowledgements

Thanks to Simha Magal and Paul Leidig for their excellent work at the 1st Annual MWAIS Conference.
Thanks to Rassule Hadidi for his work on the upcoming 2nd Annual MWAIS Conference
Thanks to Dan Power for serving as the first President of MWAIS

MWAIS Officers

**President:** Dan Power, University of Northern Iowa, power@dssresources.com

**President-Elect:** Deepak Khazanchi, University of Nebraska-Omaha, khazanchi@mail.unomaha.edu

**Secretary:** Vance Wilson, University of Wisconsin-Milwaukee, wilsonv@uwm.edu

**Treasurer:** Chelley Vician, Michigan Technological University, cvician@mtu.edu

**At-Large Director:** Ilze Zigurs, University of Nebraska-Omaha, izigurs@unomaha.edu

**Membership and Publicity Director:** Matt Germonprez, University of Wisconsin-Eau Claire, germonr@uwec.edu

**Newsletter Editor:** Matt Germonprez, University of Wisconsin-Eau Claire, germonr@uwec.edu

**Newsletter Assistant:** Alanah Davis, University of Nebraska at Omaha, alanahdavis@mail.unomaha.edu

**Conference Planning Chair for MWAIS 2007:** Rassule Hadidi, the University of Illinois at Springfield

**Conference Planning Chairs for MWAIS 2008:** Matt Germonprez, University of Wisconsin – Eau Claire and Jacob Iverson, University of Wisconsin - Oshkosh

URLs

**AIS website:** http://aisnet.org

**MWAIS website:** http://www.sba.uwm.edu/MWAIS/

**MWAIS Newsletter:** http://www.sba.uwm.edu/MWAIS/

**MWAIS Photo Gallery:** http://www.sba.uwm.edu/MWAIS/

**AISWORLD website:** http://www.isworld.org/