8-6-2011

# Teaching Enterprise Integration and Architecture – Tools, Patterns, and Model Problems

M. T. Gamble
*University of Tulsa*, mtg@utulsa.edu

# Teaching Enterprise Integration and Architecture – Tools, Patterns, and Model Problems

**M. T. Gamble**
University of Tulsa
mtg@utulsa.edu

**ABSTRACT**

This paper describes the design and delivery of a new course introducing the joint topics of enterprise integration and enterprise architecture. Planned for an Information Technology curriculum, the course sits at the crossroads of Computer Science and Information Systems and is intended for upper level undergraduate and graduate students pursuing degrees in any of these three disciplines. The course builds from an introduction to business operating models into a study of enterprise architecture and finishes with a design project in enterprise integration. The ArchiMate enterprise architecture modeling standard is applied using an open source tool. Model problems are developed in the tool to motivate class lectures and student design assignments. Design patterns form the basis of learning recurring solutions of integration problems. A final project challenges students to respond to a Request for Proposal to provide integration consulting services to a recently merged enterprise business.

**Keywords**

Enterprise architecture, enterprise integration, model problems, modeling tools, education, ArchiMate

## INTRODUCTION

There is a growing focus to reach a middle ground, both in pedagogy and in curriculum, between Computer Science (CS) and Information Systems (IS) programs. That middle ground is one of Information Technology (IT). IT practitioners are an advocate of the user – in most cases an enterprise business or one of its organizational units – attempting to find or construct software-based solutions for business problems and opportunities. In order to understand the context of complex business applications of IT, students require a foundation to address many situations. A study of enterprise architecture (the recurring structural patterns of business) and enterprise integration (the reality that most "new" solutions involve working with prior systems – both technological and human) provides such a foundation. Evidence of the ongoing debate around defining the distinction between IS and IT is given in this excerpt from the IS 2010 curriculum guide jointly prepared by the Association for Computing Machinery (ACM) and the Association for Information Systems (AIS),

> "… we can clearly see that the disciplines share areas of interest, such as Data and Information Management, IT Infrastructure, and Human Computer Interaction, but that there are also specific areas of distinction. Particularly important is the IS emphasis on Systems Analysis and Design (including Business Analysis and Business Process Design and Management), IT Strategy, Management, and Acquisition, and Enterprise Architecture. It is very likely that the discussion regarding the identities of the IT and IS disciplines will continue actively." (Topi et al., 2010)

This paper covers the development of a course in Enterprise Integration and Architecture as part of recent efforts to introduce a new IT degree program. The overarching goal is to balance coverage of business operating models with learning a solid set of design knowledge and tools usable within the scope of a one semester course. The course is aimed at upper level undergraduates and early graduate students preparing for professional careers.

The demand for skills in integration and architecture is acknowledged by the recent curriculum recommendations focusing on an IT curriculum with a cross cutting concern of integrative programming (Lunt et al., 2008) and an IS curriculum (Topi, et al., 2010) with enterprise architecture as one of its core seven courses. Integrative programming assumes that all information system development tasks will involve some degree of integration and provides the low level skills and tools to work in those situations. From an enterprise integration standpoint, the integrative programming theme is realized more by modeling tools which facilitate enterprise system analysis, identifying large scale design patterns for integration, and using off-the-shelf integration systems (e.g., middleware) to achieve integration goals. The course also offers considerable coverage of business concerns that may be very new material to crossover students from Computer Science and other more technical disciplines. Cases and reading assignments are carefully selected to provide a means for the student unfamiliar with this type of material

to reduce the learning curve. As with any course using case study materials, substantial class participation by the student is expected and forms a portion of their overall grade result.

An ongoing challenge with teaching coursework is crafting a curriculum that directly relates and prepares students for the complexity and demands of professional practice within a business enterprise. Case based teaching certainly helps, but there is no substitute for actual problem solving where failure is an option. While enterprise businesses have access to modern tools and ongoing processes around them, such tooling is most certainly beyond the budgets of students and, most often, also beyond that of departments teaching enterprise architecture. Our approach is to take advantage of recently developed open source tools and freely published enterprise architecture methods and frameworks. Admittedly, these are new opportunities unavailable to prior course developers but the time is right to couple these tools with the enterprise architecture body of knowledge for instruction.

The practice of enterprise integration is often more art than science. In consideration of this, the course uses design patterns as a basis of developing an understanding of proven integration methods. Design patterns for representing enterprise architectures and as solutions for enterprise integration problems span the gamut of complexity from enterprise applications (Fowler, 2002), message-oriented integration (Hohpe & Woolf, 2003), general software and component-based architecture (Shaw & Garlan, 1996), and finally to the modern use of service-oriented architectures (Erl, 2009). Web 2.0 (Governor, Nickull, & Hinchcliffe, 2009), and cloud computing. Design patterns from each of these areas provide applicable and reusable knowledge for enterprise integration. They also serve to provide a common "design language" usable by all the students, regardless of their background during case discussion and debate.

Where design patterns provide the design language of integration, model problems provide the common ground to test that knowledge in a more uniform and controlled way. Many disciplines, such as mathematics and the natural sciences, have their own set of model problems that almost any student in the discipline will encounter during their education. Computer Science does this as well with the known problems of sorting, searching, and graph traversal across a myriad of data structures. However, enterprise integration and architecture reside in the area of systems science where problems are substantially more complex than sorting a list in a Java program. Replicating that complexity in the classroom can be daunting, but it can be done by focusing on a well-known domain with either a limited scope or carefully selected portion of the problem. The reward is a much more meaningful series of course assignments and a learning experience that translates well into "real life" for the student.

## RELATED WORK

A few recent publications have touched on similar course development. In (Davis & Comeau, 2004) the focus is on assessing, acquiring, and integrating a packaged software system supporting the Enterprise Resource Planning (ERP) function in a business. The context is less about system-to-system integration and more about the integration of an off-the-shelf system into an organization, dealing with all the issues of adapting existing processes and practices to the reality imposed by the new software system. Another effort (Recker & Rosemann, 2010) describes a new course on business process modeling (BPM). Similar to our approach, the BPM course developers are faced with a substantial body of knowledge spread across a wide range of published sources. Their representation of knowledge in a concept map for learning clearly shows a substantial learning challenge. The authors acknowledge the issues of identifying good assignment problems while avoiding areas where students lack domain knowledge. Instead of model problems, they define scenarios of process design derived from modern consumer online purchasing and collaboration (e.g., making a purchase on Amazon's used book marketplace) which are already familiar to students.

## TOOLS

The course uses the following categories of tools for instruction and exploratory learning:

- **Texts and published references**: The course uses two required texts. The first text (Ross, Weill, & Robertson, 2006) gives a high level view of the structural implications of business operating models for enterprise architecture. The second text (Schmidt & Lyle, 2010) focuses on the implementation of an integration competency in an enterprise setting. Additional case studies and online materials (especially for design pattern catalogs) supplement the texts.
- **Enterprise architecture modeling language**. ArchiMate (The Open Group, 2009) is a recently standardized (February 2009) enterprise architecture (EA) modeling language. Better suited to EA than the more implementation focused Unified Modeling Language (UML) (Rumbaugh, Booch, & Jacobson, 1999), ArchiMate has a limited number of concepts and notations that are easily learned by students within course timeframes. The complete standard is freely available online.

- **Visual modeling tool**. An open source modeling tool called Archi fully implements the ArchiMate standard and is also freely available. It requires limited resources and can be run on student's laptops on Windows, Mac OS, or Linux without issue. Use of these tools allows ongoing incremental development of models so student assignments can build from one to the next or be crafted to extend designs provided to them.
- **Integration experimentation tools**. Beyond the design part of the course, implementation of actual integrations provides a good foundation for understanding the underlying problems. The cloud-based Yahoo! Pipes application is used to implement mashups.

### Modeling tools

We used an open source tool, Archi[1], which provides a full implementation of the ArchiMate 1.0 standard. Easy to acquire, install and use, the students quickly adopted the tool. Instruction on the tool and corresponding modeling techniques were interwoven with other lecture topics. Cases presenting specific business models and enterprise architecture implementations were related to ArchiMate modeling features and demonstrated with model "snippets" from the standard and model problems. ArchiMate provides a clear separation of modeling concepts allowing both the structure (i.e., topology) as well as the behavior of enterprise architectures to be modeled. This separation allows ArchiMate to model modern service-oriented architectures at the business, application (as illustrated in Figure 1), and infrastructure levels.

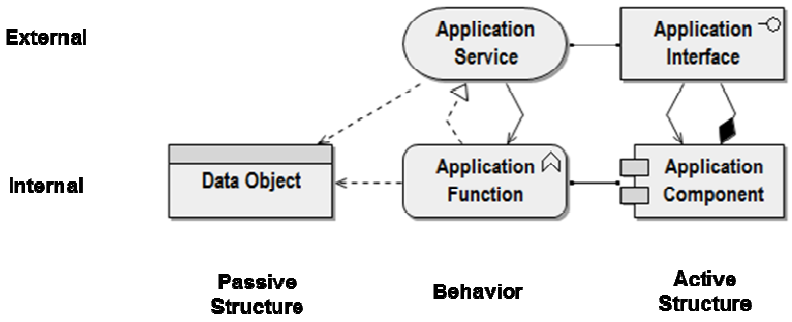


**Figure 1. Structure of ArchiMate Modeling Levels.**

Targeted towards use in higher education, the tool is in early development phases. Its inception was to provide a means for students to have a tool at a reasonable cost. Most enterprise modeling tools either are ad hoc (e.g. Microsoft Visio or some other drawing program) or prohibitively expensive costing thousands of dollars per seat. Continuing releases in 2010 (during the course) saw rapid expansion of the tool's capabilities and stability. Recent updates to the open source tool offer additional features making it especially useful in easing the learning curve for a new modeling method. Updates include the "magic connector", a user interface component that allows students to create relationships between modeling concepts without knowing which relationships are correct in the model. The magic connector presents the user with a set of valid choices for relations once the concepts are selected, thus enabling a trial-and-error approach to creating portions of the model.

### Prepare for opportunistic design

Given the competitive and economic conditions of the modern business environment, many companies are forced to "work with what they have got" when determining which IT solutions are to be used. This situation may involve making an existing system fit with a newly acquired piece of software or determining the "best" resulting IT architecture from the merger of two companies. In most scenarios, the result is much more integration than construction. Students need training in the general practice and mindset of integration along with exposure to tools that make integration possible when it is needed. The reality in modern business is that most enterprise systems are formed by combining already existing software system assets that either (1) exist as "legacy" systems or (2) come into the environment as pre-built software from commercial or open source suppliers. The IT 2008 model curricula (Lunt, et al., 2008) call this competency "integrative programming". A core learning objective of the IS 2010 Enterprise Architecture course is to be able to "evaluate and plan for the integration of emerging technologies"(Topi, et al., 2010).

[1] Archi is an open source, cross-platform tool to create ArchiMate models available from http://archi.cetis.ac.uk/index.html.

Almost every act of software-based system development is now an act of integration. Which parts get used, how they are combined, and which tools are used is often an opportunistic decision (Gamble & Gamble, 2008). Yahoo! Pipes is a mashup tool designed to collect and combine data from Internet-based resources. As part of an assignment (see Figure 2) students are asked to find a way to combine sales receipts across multiple, merged grocery stores into a single central sales record organized by per product sales across the entire enterprise. The tool is visual, requiring no coding. A series of "pipes" and "filters" is constructed. Filters perform computation, while pipes interconnect filters.

Using this approach has the following benefits:

- Students unfamiliar with programming can quickly focus on the *skills of design* at the level of architecture without the need to write code.
- The tool is readily/freely available and easily combined with other familiar tools (e.g. Microsoft Excel was used to create the input data used by the Pipes SalesPerProduct application).
- An obvious solution to the problem would have been to use a database application. Use of the pipe-and-filter architecture forces students to consider alternative approaches and prepares them for situations when tools are not ideally suited to the problem at hand.
- A visual representation of the problem solution is very useful in class discussion.

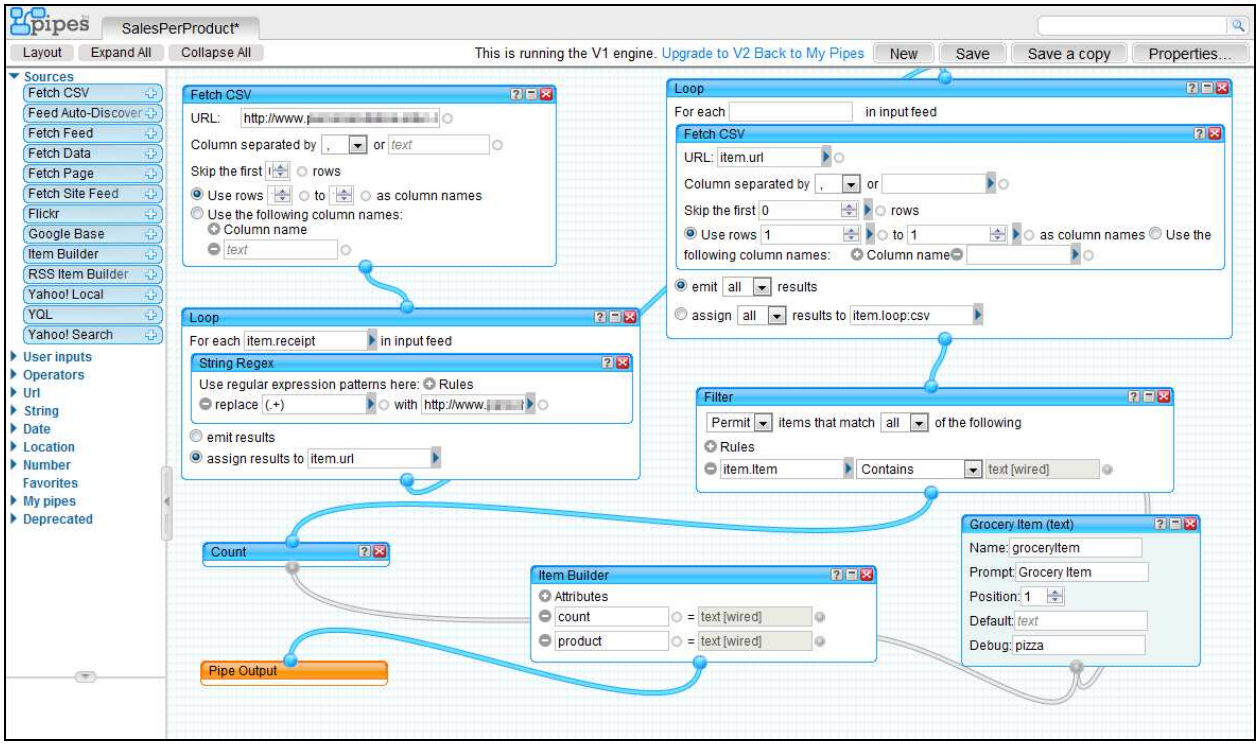


**Figure 2. Yahoo! Pipes ArchiMart Collection of Sales Receipts per Product.**

## PATTERNS

A design pattern is a general reusable solution to a commonly occurring design problem. Adapted from the building architecture and urban design work of Christopher Alexander (Alexander, Ishikawa, & Silverstein, 1977), design patterns have been used extensively in software engineering to the point that they have become embedded in the design language of practitioners. As future designers of integration solutions, students need a means to quickly understand and communicate that understanding of system and organizational structures in order to determine options for integration. Knowledge of design patterns gives them a mental toolkit for analysis as well as future learning in situations where the structure of the design problem is similar but the technologies and specific business scenarios change.

| **Software Architecture Patterns (Integration Styles)** | **Enterprise Integration (Messaging) Patterns** | **Enterprise System Archetypes** | **Service-oriented Architecture** |
|---|---|---|---|
| Pipe-and-filter<br>Hub-and-spoke<br>Event-based<br>Layered<br>Repositories<br>Remote Procedures | Message Router<br>Message Translator<br>Message Endpoint<br>Content-based Router<br>Message Filter<br>Content Enricher | Enterprise Resource Planning (ERP)<br>Customer Relationship Management (CRM)<br>Supply Chain Management (SCM)<br>Master Data Management (MDM)<br>Product Lifecycle Management (PLM) | Enteprise Service Bus<br>Service Façade<br>UI Mediator<br>Trusted Subsystem<br>Orchestration<br>Composite Application<br>Federation |

**Figure 3. Exemplary Integration Patterns at Different Levels of Detail.**

This portion of the course draws on materials in published design pattern catalogs in Enterprise Applications (Fowler, 2002), Enterprise Integration (Hohpe & Woolf, 2003), Software Architecture (Shaw & Garlan, 1996), and Service-oriented Architecture (Erl, 2009). Enterprise systems are not often described as patterns, but they do have archetypes and are similar in both design as well as intended use. Figure 3 details various design patterns for each category. While vendors of Enterprise System software might disagree, one Customer Relationship Management (CRM) system is much like another. When students learn this design language, it provides a common baseline for understanding and discourse. They were also encouraged to use this design language to communicate results in course assignments.

## MODEL PROBLEMS AND CASES

Few things convey a complex idea more effectively than a good example. Using model problems for teaching is well established in such disciplines as mathematics and the natural sciences. The same is true of computer science (e.g. basic manipulation of data structures for searching and sorting). Case studies present good opportunities for conversation as well as set the stage for students to build up their understanding of recurring design patterns by seeing instances of implementation. They are written to capture the most interesting and focused details. However, cases alone are not usable for student design work because they deal with limited views of real systems outside the educational setting. In some instances a case example of enterprise architecture might be "cloned" and then adapted by the student. But this technique is superficial since details of the architecture are proprietary and only specific information is revealed in a publicly available case study. To truly invest themselves in the design, students need to "own it" and that means they need a completely realized architecture to manipulate.

Model problems in enterprise integration and architecture are needed because:

- Reference architectures are too high level and the best examples are often vendor influenced.
- Software architecture patterns and model problems are too low level or only at the componentry level

The structure of model problems relies both on industry norms, usually in the forms of reference architectures from vertical industries (e.g., groceries, libraries, etc.) and the more general structural patterns identified by operating models . Design patterns are applied at different levels to address recurring problems in growth and integration across companies with such operating models. Students learn to identify patterns and both apply and adapt known solutions.

| **Model Problem and Domain** | **Description** | **Course Use** |
|---|---|---|
| ArchiSurance<br>*Insurance services* | Extended case study of a hypothetical merger of three insurance provider companies | Examples appear throughout the ArchiMate standard and prior publications on which the standard is based |
| ArchiMart<br>*Supermarket grocery chain* | A supermarket grocery using the replication operating model | Classroom discussion and illustration of integration design issues with examples |
| ArchiLib<br>*Public library system* | An open ended problem of combining two library information systems a city-county library system is merged | Presented as a series of assignments where students develop an enterprise architecture and then solve integration problems during a merger |

**Table 1. Model Problems in Enterprise Integration.**

The course relies on three model problems: ArchiSurance is a pre-existing model of an insurance company that is used within the ArchiMate standard and by its authors in prior publications to illustrate concepts. While illustrative, it quickly becomes uninteresting to students given that it is used so extensively within the standard.

Two additional model problems (see Table 1) are developed specifically for the course. The reasoning behind having two problems is that one model problem is used for demonstrations and to motivate open class discussion while the other is reserved for use in assignments. Selection and development of the model problems followed these informal criteria:

1. A domain that is already known or easily understood by a student
2. Sufficient detail to require a variety of modeling concepts and techniques to represent
3. Sufficiently high level to allow for variation during use within a particular class session
4. Representative of "real world" issues both in modeling and, later, integration issues

ArchiLib targets the representation of a community public library system that is later integrated with another library. ArchiLib is woven through a series of assignments. Both developed model problems are business domains with which most students are familiar and therefore provide a simple foundation for early discussion. Exploration of the business models and representative information systems within these contexts allow incremental complexity to be added during the course.

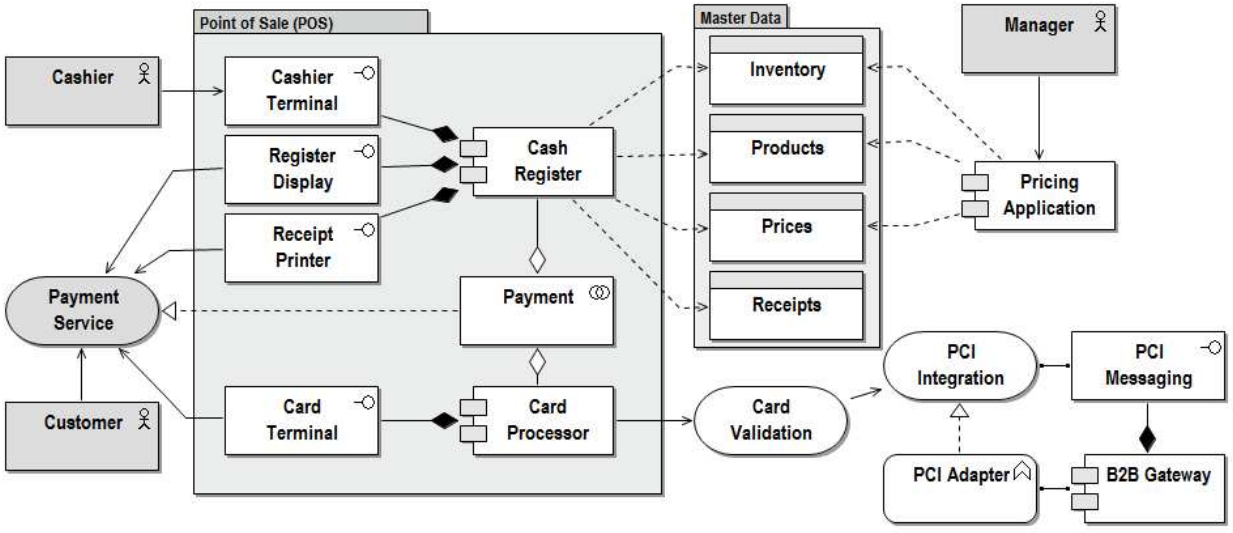

**Figure 4. The Supermarket Model Problem - ArchiMart.**

The ArchiMart model problem provides opportunities to convey numerous concepts in a compact and easily understood format. Use of the modeling representations early in the course schedule prepares students to use those representations in their own mental models and also offers substantial situations for open-ended design discussions. From a single diagram (see Figure 4) we can launch discussions across the following key areas:

- Handling of master data and the data architectures required for the Replication operation model used by ArchiMart.
- Roles of business actors and their relationship to the model of a single market as well as multiple market location within a supermarket chain
- Representation of business services as concepts delivered through real interfaces (see the Payment Service in Figure 4 which is a representation of the Customer's interaction with the devices at the point-of-sale)
- Service integration with business-to-business partners or vendors (see the B2B gateway for payment card validation)
- Incorporation of multiple built or purchased software systems (e.g., point-of-sale and a general B2B gateway)
- Application interactions which deliver services (see Payment as the association between the Cash Register and Card Processor applications)
- Differences between administrative and transactional uses of enterprise applications and data.

## COURSE STRUCTURE

The course is delivered in a traditional classroom setting, meeting one night per week. Assignments, reading materials, and grading results are communicated online to students. The first half of the course focuses on more traditional assignments while the latter half introduces the final project problem. Early assignments build knowledge in the areas of business foundations, modeling, and general enterprise architecture issues. Introducing design patterns is a semester transition point where students become more autonomous and focused on design thinking to produce results.

### Traditional assignments (preceding the final project)

Throughout the first half of the course short one to two-week assignments build the student's understanding of core principles (through case studies), key integration patterns (as applied to practical integration problems), and use of the tools (particularly the fundamentals of modeling enterprise architectures).

1. Case study in enterprise architecture – operating models and "core" diagrams for company architectures
2. Modeling business level architectures – representation of high level organizations and business functions
3. Information integration – implementation of data sources and creation of online mashup solutions
4. Business model integration – teams of two integrate the business architectures developed in an earlier assignment
5. Modeling application layer architectures – detailed process modeling with application layer services and interfaces; solution delivered in a "professional consulting" format
6. Application integration – develop and present a solution architecture merging the student's design from an earlier assignment with a "generic" architecture provided by the instructor

Each of these assignments is designed to build up the student's proficiency in order prepare for work on the final project. In additional to the case study analysis and technical skills of modeling, student grades also depend on the quality of their written and oral presentation results throughout the course.

### Final project

The final project assignment in the course is designed to fulfill multiple learning objectives including not only applying the modeling and design skills acquired earlier in the course, but also expressing results in a professional manner consistent with a consulting engagement. The assignment is given in the form of a Request for Proposal (RFP), which consists of three components:

- **Bidders' conference**. This exercise implements a pre-bid meeting where bidders (students) ask questions of the client (i.e., the instructor acting as client). It is designed to mimic a real world scenario of participating in a corporate procurement of consulting services. Questions are submitted in advance and the meeting follows a strict protocol which deliberately alters the normal dynamic of a classroom setting. Students must accept the limitations of dealing in a contractual setting where the transparency of communications along with the time and opportunity to ask questions regarding the RFP are limited.
- **Written proposal**. Students complete a professional RFP response document, prepared according to a template and consistent with requirements in the RFP. Students are required to restate earlier work as results that prove their ability as a consultant to respond to the RFP. Written justifications for design approaches demonstrate their ability to restate the problem for the "client" as well as apply new skills. Results include a *transition architecture* showing the intermediate state of the enterprise during the merger. Depending on the operating models of the student's original design and that of their target (merged) company, these transition architectures may vary significantly in design tradeoffs.
- **Oral presentation**. For some students this task is the most daunting. The situation is setup to mimic the actual procedure of a consulting vendor presenting to a client during competitive bidding. The instructor asks questions as the client and the presentation is graded independent of the written assignment. Given that other students are also "competitors" in this mock procedure, questions are filtered through the instructor during the presentations.

*Evaluation criteria*

The RFP instructions provide the evaluation criteria in **Table 2**.

| | |
|---|---|
| 1. | Understanding and ability to explain the current context/situation of the client |
| 2. | Understanding of the services required by the client |
| 3. | Relevant experience and knowledge of lead consultants in developing an integration architecture (based on reporting and representation of prior results) |
| 4. | Ability to refine the Statement of Work (SOW) to demonstrate expertise in integration dependencies/requirements |

**Table 2. RFP Response Evaluation Criteria.**

Where prior assignments focus on the ability to do requirements extraction, design, and modeling, the final assignment turns to providing reasoned argument for doing integration. It defines and proposes work based on prior learning but does not require complete implementation of that work within a single semester course (which would be unreasonable).

*Statement of Work*

By the end of the course students have developed sufficient learning and skills to know how to approach integration problems. The final project (an integration of two public library systems) tests their ability to analyze and, more importantly, describe what would be involved in approaching the solution to an enterprise level integration problem. By refining a Statement of Work (SOW), these skills are put into practice and shift the focus from classroom assignments to a full-scale enterprise integration problem. Students are asked to expand and refine the following SOW points:

1. **Develop a target operating model for the combined libraries that meets the concerns of the library, staff, and patrons**. Students must now describe as a consultant how this problem is to be approached and what the expected outcomes are for the client.
2. **Develop an integration architecture that aligns with the target operating model**. In expanding this point, students are encouraged to justify their effort and identify what key points are needed to
3. **Conduct an analysis of the current systems in order to provide a complete model to be used for integration design**. Here the objective is to actually define and limit the scope of investigation and outputs to that which is necessary for a result. The careful art of managing expectations as a consultant comes into play.

4. **Identify functional gaps and overlaps between the two libraries**. Based on prior class assignments the students have a good understanding of both their own library model and that of the acquired library. Results here tended to suggest both previously identified gaps and overlaps as well as argue for those areas of library's reference architecture which likely hold such functionality. The choice of operating models also provides significant design guidance to the student. For example, the design patterns involved in consolidating to a single, unified operating model often result in the same types of overlaps from merged companies.
5. **Involve all necessary stakeholders when gathering information and reporting results**.

## FUTURE DIRECTIONS

Further offerings of the course will provide a larger student population for evaluation metrics. A serious concern was also choices (and costs) for textbooks. Given the substantial amount of online resources, it may be possible to eliminate a required text. One of the challenges involved is finding sufficient time to develop baseline models which could then be used for integration analysis. Further elaboration of model problems should provide this baseline at varying levels of detail. For example, while the ArchiSurance model is given in the ArchiMate standard, it is not available in the modeling tool. The model problem could be reworked and provided as a template earlier in the course. Streamlining the modeling effort required would leave more time for implementation work and the addition of an open source enterprise service bus (ESB) product to supplement the mashup tools. Finally, team assignments proved to be impractical for a number of reasons. Most importantly, the "normal" integration problem usually involves some fixed design and some variable design that is under the control of one designer. In the case where two or more students are integrating their models, this much variability proved impractical. The genesis for detailing more of the model problems came out of this result. The group project component of the course will be removed in the future unless we develop it into a two course sequence where there is sufficient time to deal with that complexity.

## REFERENCES

1. Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A pattern language : towns, buildings, construction*. New York: Oxford University Press.
2. Davis, C. H., & Comeau, J. (2004). Enterprise integration in business education: Design and outcomes of a capstone ERP-based undergraduate e-business management course. *Journal of Information Systems Education, 15*(3), 287-300.
3. Erl, T. (2009). *SOA design patterns*. Upper Saddle River, NJ: Prentice Hall.
4. Fowler, M. (2002). *Patterns of enterprise application architecture*: Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.
5. Gamble, M. T., & Gamble, R. (2008). Monoliths to mashups: Increasing opportunistic assets. *IEEE Software, 25*(6), 71-79.
6. Governor, J., Nickull, D., & Hinchcliffe, D. (2009). *Web 2.0 Architectures*. Sebastopol, CA: O'Reilly Media, Inc.
7. Hohpe, G., & Woolf, B. (2003). *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Boston: Addison-Wesley.
8. Lunt, B., Ekstrom, J., Gorka, S., Hislop, G., Kamali, R., Lawson, E., & Reichgelt, H. (2008). Information Technology 2008 Curriculum Guidelines for Undergraduate Degree Programs in Information Technology: Association for Computing Machinery (ACM).
9. Recker, J. C., & Rosemann, M. (2010). Teaching business process modelling: experiences and recommendations. *Communications of the Association for Information Systems, 25*(32), 379-394.
10. Ross, J. W., Weill, P., & Robertson, D. (2006). *Enterprise architecture as strategy: Creating a foundation for business execution*: Harvard Business Press.
11. Rumbaugh, J., Booch, G., & Jacobson, I. (1999). *The unified modeling language reference manual*. Reading, Mass.; Harlow, England: Addison-Wesley.
12. Schmidt, J. G., & Lyle, D. (2010). *Lean integration : an integration factory approach to business agility*. Upper Saddle River, NJ: Addison-Wesley.
13. Shaw, M., & Garlan, D. (1996). *Software architecture : perspectives on an emerging discipline*. Upper Saddle River, N.J.: Prentice Hall.
14. The Open Group. (2009). *ArchiMate 1.0 specification*. Zaltbommel: Van Haren Publishing.
15. Topi, H., Valacich, J. S., Wright, R. T., Kaiser, K., Nunamaker Jr, J. F., Sipior, J. C., & de Vreede, G. J. (2010). IS 2010: Curriculum Guidelines for Undergraduate Degree Programs in Information Systems. *Communications of the Association for Information Systems, 26*(1), 18.