Large Scale Requirements Modeling: An Industry Analysis, a Model and a Teaching Case

Akhilesh Bajaj
School of Accounting and MIS
The University of Tulsa
Tulsa, OK 74133, USA
akhilesh-bajaj@utulsa.edu

ABSTRACT

This work provides a fresh perspective on the custom-build versus rent/buy decision that organizations have faced when dealing with large scale information systems. We first describe the historical trends in this area. Next, we present an analysis of what caused the current trend of renting/purchasing off-the-shelf systems and why recent changes may portend an increase in custom building. We make the case for why systems analysis and design will play an increasing role in the building of large custom ISs, and highlight the need for increased teaching emphasis in this area. As a first step in this direction, we present a model called ERA (Entity Relationship Activity) with an accompanying (Computer Aided Software Engineering) CASE tool and present one teaching case that has been used to teach over 250 students at the undergraduate and graduate levels.

Keywords: Systems Analysis, Systems design, requirements modeling, CASE tool

1. INTRODUCTION

The custom build versus rent/buy decision for large scale information systems (ISs) has emerged as an important decision facing organizations in the last decade. While there is a great deal of recognition of this problem in the industry literature (Higaki 1995; MindBridge 2004), it has been relatively ignored in the academic literature. Though the basic pros and cons of both approaches are mentioned in several industry articles, there has been a dominant trend amongst organizations to buy (or rent) commercial off the shelf systems (COTS) for larger ISs, rather then build customized systems (Abts 2002; Vedaris 2004). The success of these large COTS implementations, such as enterprise resource planning (ERP) systems, has been low, and has been usually accompanied by lower end-user satisfaction and unsubstantiated changes in productivity (Abts 2002; Taeschler 2002).

In this work, we consider large-scale organizational ISs that utilize a back-end database management system (DBMS) to store and retrieve information, and provide an interface to end-users to access and modify this information. As mentioned in (McManus 2003), this covers the majority of large IS systems that support business processes in an organization. The primary purpose of this work is to: a) analyze why custom building of large scale systems may become increasingly viable and highlight the need for it to be taught in MIS curricula; b) describe a model we call ERA (Entity-Relationship-Activity) with an accompanying computer aided software engineering (CASE) tool, that has been developed primarily for large scale IS analysis and design; and c) present a teaching case that has been used across multiple universities at the undergraduate and graduate IS levels to develop the skill of large scale requirements modeling.

The rest of this paper is organized as follows. In section 2, we analyze the history of IS procurement in more detail, discuss why custom building may be re-emerging and how IS curricula should adapt. In section 3, we describe the ERA CASE tool that facilitates analysis and design of large scale IS requirements. Section 4 presents an educational case that we have developed and used over the last several years, that exposes students to large scale systems analysis and design. Conclusions and future research are discussed in section 5.

2. IS PROCUREMENT AND THE POTENTIAL RE-EMERGENCE OF CUSTOM-BUILDING

Historically, organizations faced with a need for a large scale IS resorted to building **in-house** (MindBridge 2004). Since the last decade however, this trend has been almost completely reversed, so that most large scale systems today consist of COTS systems. This phenomenon began with the hope that the COTS approach would be the magic bullet (Brooks 1987) that would kill the beast of software development. This view was especially encouraged by policies at the executive level in the organization that strongly favored purchasing COTS-based systems over customized development (Abts and Boehm 1998). The perceived advantages of buying include a) cost savings due to economies of scale that allow a vendor to develop a COTS system and distribute fixed development costs over multiple customers; b) utilizing "best-of-breed" solutions that COTS vendors developed over time and working with multiple customers; and c) mitigating the risk of developing a customized large scale IS by utilizing a solution that had been known to work in other organizations.

Diverse surveys of executives indicate that they believe that ERP systems will centralize information and simplify their information technology (IT) functions by dealing with just one vendor (Taeschler 2002). A well known case of the $100 million implementation of SAP at Corning corp. resulted in data centralization as the main benefit (Ross 1999). In a Harvard Business Review article, Davenport (Davenport 1998) described the pros of COTS solutions thus:

> *"Enterprise systems present a new model of corporate computing. They allow companies to replace their existing information systems, which are often incompatible with one another, into a single, integrated system. Unlike computer systems of the past, enterprise systems are off-the-shelf systems."*

In the same article, Davenport described some of the shortcomings of legacy systems:

> *"If a company's sales and ordering systems cannot talk with its production scheduling systems then its manufacturing productivity and customer responsiveness suffer...if its sales and marketing systems are incompatible with its financial-reporting systems, then management is left to make important decisions by instinct rather than according to a detailed understanding of products and customer profitability. To put it bluntly: if a company's systems are fragmented, it's business is fragmented."* (Davenport 1998), pp. 123.
>
> The cumulative effects of customized legacy systems led to "islands of information" and fragmentation of the business processes. As mentioned in (Davenport 1998), pp. 123 *"Each of these so-called legacy systems may provide invaluable support for a particular business activity. But in combination they represent one of the heaviest drags on business productivity and performance now in existence."* In the late 1990-s, one solution was COTS ERPs, with a single database back-end. *"The database collects data from and feeds data into modular applications supporting virtually all of a company's business activities-across functions, across business units, across the world."* (Davenport 1998), pp.124.

This large scale adoption of the buy approach was further accelerated because of pressures related to the Y2K crisis. Most organizations did not have the time or resources to successfully redesign their business process systems to be Y2K compliant (Gould 1999). Increased monetary resources and reduced time led many IT departments to purchase COTS ERP packages that were Y2K compliant. By the end of 2001, enterprise level systems sales were at $47 billion and forecasted to be grow in the double digits (Johnston 2002).

On the other hand, COTS implementations are also thought to have brought significant disadvantages for the organizations that have implemented them. Early implementations of ERP systems were plagued with time and budget overruns (Abts 2002; Taeschler 2002). In some prominent cases, such as Nike, Foxmeyer and Hershey (Johnston 2002), the organizations claimed that the COTS implementation crippled the organization's core operations. As pointed out in (Eckerson 2002), COTS implementations were found to largely take about the same time as customized applications. Corning Corp. found that the amount of data input was 10 times more with the COTS implementation than with earlier customized systems (Ross 1999). COTS implementations have been found to be more "volatile" than customized applications primarily because of the frequency with which vendors release upgraded versions of their software, with an increasing number of modules in a COTS implementation leading to greater volatility because of differential upgrade releases amongst the modules (Abts 2002). More frequent vendor updates also inhibit the customizing of COTS applications, since the customizations may need to be repeated for each update.

The loss of competitive advantage has been another important issue with COTS based ISs. COTS implementations are usually not customized because of issues with costs and future compatibility with vendor upgrades (McManus 2003). This implies that the organization's data and business processes have to conform to the new model offered by the implementation. Thus, the COTS IS significantly reduces the degree of differentiation of the organization's business processes and data items compared to its competitors, who also have the same COTS implementation (Ulrich 2004).

Finally, COTS ISs have been found to lack flexibility because of the lack of customizability, and dependence on the vendor's ability and willingness to upgrade the software as the business environment changes (Vedaris 2004). This problem is further aggravated by the often long time lines that accompany a large scale COTS implementation. For example, the COTS implementation at Corning took approximately six years, and led to the entire organization being dependent on one vendor for its upgrade cycles, in a competitive manufacturing environment.

Based on the preceding discussion, we can conclude that, while COTS based ISs offer certain advantages over building a customized, large scale IS, they also offer significant disadvantages. In spite of these disadvantages, the current reality is that most organizations are opting to purchase/rent their large scale ISs as opposed to customized construction. However, several trends indicate that this may change. First, the disadvantages of COTS based solutions are becoming more apparent at the executive level. For example, a survey of 50 European organizations revealed a 92% dissatisfaction rate with their ERP implementations (Group 2000). Second, the phenomenon of overseas outsourcing has led to reduced development costs, making customized development more attractive (Singh and Walden 2003). Third, emerging technologies such as easy to use interface development environments (common examples include Visual Basic, Visual Café, Oracle Developer), web services that allow easier integration and reduced costs of relational DBMSs have made it conceptually easier to design and implement customized solutions, in contrast to using older technologies for networking and interface development (Baker and O'Sullivan 2001). These trends reduce the disadvantages of customized IS development, so that both the build and the buy choices are becoming viable, and the build versus buy decision in the future is likely to become increasingly important.

2.1 The Re-Emerging Importance of Systems Analysis and Design for Large Scale Modeling

It has been widely recognized that the systems analysis and design (SAND) phases of software development are critical to ensuring the success of the system, both from a technical and a business perspective (Bajaj and Ram 2002; Alter and Browne 2005; Bajaj, Batra et al. 2005; Alter 2006). However, the recent trend in COTS selections for IS development have reduced the usage of formal systems analysis when evaluating a system. Typical systems analyses during the selection of COTS involve a listing of critical activities by end-user groups, and demo-based usage of a set of COTS packages, with the intention of selecting one from the set (Verville and Halingten 2001). A critical difference between SAND methods used for customized systems versus COTS systems is the lack of data models and detailed process models when evaluating COTS systems, as opposed to considering customized systems. This reduced usage of systems analysis is primarily because COTS systems are seldom customized, for reasons discussed in the sections above. Since COTS selection occurs from a fixed, often small, set of vendors, who offer their own data and process models in their COTS, the resources required to create data and detailed process models of the actual organization are usually not expended in a COTS implementation.

We suggest that the lack of an attempt to match the data models of the organization with that of the COTS has the following implications: a) In a market with several vendors, it may be possible to omit consideration of a COTS system whose data model is a closer match, leading to poorer user acceptance and increased user-effort in switching to the new system, and b) The cost of migrating the legacy data increases for a COTS whose data model is a poor match for the data model of the organization.

A lack of a detailed data and process model of the organization also prevents a meaningful analysis of a build versus buy comparison. First, there is no basis for estimating the resources required to build a customized IS, without understanding the size of the database, and the complexity of the user interface and business logic. Second, the degree of uniqueness of the users needs cannot be determined without a detailed data and process model. Customized applications are usually considered more desirable for unique needs (McManus 2003). The lack of a data and process model may enhance the tendency to classify a large portion of the business activities of an organization as "commodities" that can be served/substituted by off-the-shelf systems. In extreme cases, this can lead to a significant loss in competitive advantage of the organization.

Based on the above discussion, we contend that the role of systems analysis will become increasingly important in the success of **large scale** ISs, as the build versus buy decision becomes more critical. While SAND is taught in most IS curricula and several modeling languages exist, we propose that there are two major shortcomings that we need to address:

a) The examples we examine in most classroom cases relate to support small scale systems. Little is known about the abilities of the models (and the software tools that support these models) to scale up to large scale requirements analysis and information system design. *E.g.,* What is the value added of a particular CASE tool if we need to collect the process and data requirements of, say, 100 plus business roles in an organization and design a custom IS for them? In the large majority of cases, IS graduates from our curricula are not adequately trained to perform SAND on a large scale.

b) As mentioned earlier, most organizational systems can be modeled as a database back-end, with front-end screens to support data –entry, modification and report generation activities. Most of the modeling languages that we teach are general purpose, and exist to support the creation of general code, rather than the creation of large scale organizational systems. *E.g.,* the Unified Modeling Language (UML) traces its roots back to facilitating the creation of object oriented code (Booch 1994). We contend that there is a need for specialized modeling languages and tools that map more directly to the constructs of large scale organizational systems: database tables and screens.

As discussed above, even though there is disillusionment with several COTS implementations the majority of organizations still purchase these implementations. What prevents organizations today from seriously considering the construction of customized ISs, as opposed to renting/purchasing COTS solutions? We list several reasons below:

a) As mentioned earlier, most IS personnel today are not adequately trained in a SAND methodology that scales to handling large scale business requirements.
b) There is a lack of real knowledge about the requirements for a software tool that can scale well to large scale business requirements (Alexander, Robertson et al. 2005).
c) COTs systems are often sold to non-IS personnel, at the executive level who are indifferent usually to whether a system was custom-developed or bought off-the-shelf. (Applegate, Austin et al. 2006), pp. 424
d) The lack of a comprehensive methodology and prior experience in custom building has led to an unfavorable risk/reward perception amongst IS personnel, when it comes to building. In other words, it is easier for an IS person to be a customer of a COTS vendor (Verville and Halingten 2001) than it is to be a provider of a custom built system to the end-users, especially when the senior executives seem to desire a COTS system and the management style in the organization is hierarchical (Khalid 1998).

The discussion above highlights an emerging need for a new breed of modeling language and supporting CASE tool: one that can support the development of a custom-built large scale IS in a business context. As a first step in this area, we next present the Entity-Relationship-Activity (ERA) model and CASE tool that has been developed specifically to support the application of SAND in a large scale business context.

3. THE ERA MODEL AND CASE TOOL

The underlying philosophy behind the ERA model is that the large majority of organizational systems that support business processes are database driven systems. Essentially, the ERA model models a large business application as a back-end database application, stored in a relational database, with a set of screens that allow the user access to view and modify the data. Business logic can be a) modeled as part of the back end design, where a good design automatically prevents incorrect data from being inserted, b) stored in the form of stored procedures or front end screen logic or c) stored as database triggers, if it is deterministic. An example of a) is providing foreign key constraints that prevent customers from making purchases if these customers do not first exist in the *customers* table. An example of b) is checking the salary information for a new employee, as it is being entered, to ensure it meets required criteria such as being over the minimum wage. An example of automated logic in c) is a situation where, if a customer purchases more than $1000 of merchandise, they automatically become a "gold" customer. We note that implementation facilities to support a), b) and c) are available on most large scale DBMSs.

The scalability problems with large scale business information systems have to deal with managing the diverse requirements of the large number of users or organizational roles the system needs to support. Thus, if a system needs to support, say, over 1000 users, then modeling and integrating the data and process needs of these users is the challenge. The overall goal of the ERA model is to provide a SAND model and tool that facilitates the capture of user requirements (both data and process) **on a large scale**, and converts them into a **code plan** to guide the development of the information system. While the underlying data and process models used in ERA are based on well known concepts, the main contribution here is the selection of concepts both at the model level and the methodology (CASE tool) level that allow the large scale collection of requirements (of the order of thousands of entity sets, relationships sets and activity sets).

To model data, the ERA model draws on the well known Entity Relationship model (ERM) (Chen 1976). While various extensions of the ERM have been proposed in the academic literature, the ERA model uses a **streamlined** version of the ER model, consisting of *entity sets, (n>=2 -ary) relationship sets, attributes* of *entity sets, attributes* of *relationship sets, primary keys* of *entity sets, superclasses* and *subclasses*. Each subclass can have only one superclass. Concepts such as *cardinality* and *participation* of relationships, the *existence* relationship, *categorization*, and aggregation were deliberately omitted from ERA in order to make it easier to use in the field, for large scale requirements gathering. For example, the cardinality concept in ERA assumes that all relationships are many-many, which is a generalized superset that covers the different cardinalities: one-one, one-many and many-many. In other words, concepts in ERA were chosen to facilitate large scale requirements gathering and systems modeling, at the expense of some of the more nuanced concepts typically found in extensions to the ERM.

ERA borrows from well known ideas of process decomposition to model processes. An activity in ERA is defined as an action that accesses or modifies data. A high level activity in ERA can be decomposed into smaller activities. Each activity can have at most one parent. Thus each high level activity forms the root of its own activity tree. Primitive activities exist as leaves of the different activity trees. A primitive activity is one that accesses information that can fit on one screen, in the judgment of the analyst. Primitive activities are either *automatable* (meaning no user input is required) or *non-automatable* (user input is required). Each activity has a *responsibility* (organizational role) linked to it.

Once the activities are depicted, the ERA model supports links between the activity and data as follows. Each data object (entity set or relationship set) is linked to **at least one primitive activity** in either a READ or a READ/WRITE mode. This ensures that each data object is "consumed" by at least one activity. Conversely, each primitive activity has to "consume" at least one data object. Again, for the purpose of scalability, we consider data objects at the entity or relationship level, not at the attribute level. Figure 1 illustrates the diagramming conventions in ERA.

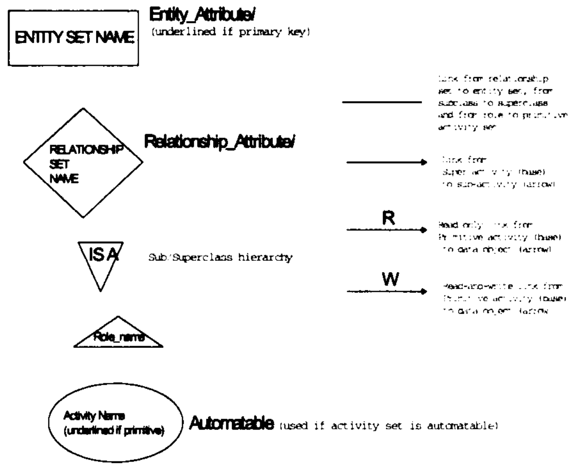

ENTITY SET NAME
Entity_Attribute/
(underlined if primary key)
RELATIONSHIP SET NAME
Relationship_Attribute/
ISA
Sub/Superclass hierarchy
Role_name
Activity Name (underlined if primitive)
Automatable (used if activity set is automatable)
Link from relationship set to entity set, from subclass to superclass and from role to primitive activity set
Link from super activity (base) to sub-activity (arrow)
R
Read only link from primitive activity (base) to data object (arrow)
W
Read-and-write link from primitive activity (base) to data object (arrow)

Figure 1. Diagram Conventions in ERA

More detailed information on ERA, including diagramming conventions and examples can be found at (Bajaj 2004).

3.1 Utilizing the ERA Methodology for Large Scale Requirements Gathering

The ERA methodology is based on a divide-and-conquer approach. The first step is to divide the entire target end-user group into sub-groups, so that the subgroups are cohesive (share a large portion of their data and have closely related business functions) and manageable. Based on our experience, we define a manageable sub-group as one whose data model will be of the order of 100 entity sets or less.

Once the subgroups have been created, the analysts meet with the different sub-groups in order to create data diagrams. These meetings are similar to JAD sessions (McConnell 1996). The output of this activity is a single data diagram for each subgroup. The different data diagrams are then examined for ambiguities, errors and overloading of terms. *E.g.,* One subgroup may use *customerId, custName, custAddress* for a customer, while another may use *customerNo, custFirstName,custLastName,custPhone.* Corrections are made to each sub-group's data diagram based on examination of the other diagrams. Finally, a complete data model is created for the application, after resolving ambiguities and error terms. This can be done by adding each subgroup to the completed set, after resolving its data diagram with the overall diagram.

Activity diagrams are created individually for each user, based on the data they will be using. The data diagrams for each sub-group act as the backbone for defining the activities that users perform, and activity trees are diagrammed for each sub-group. Finally, the primitive activities are linked to the different data objects in the data diagram.

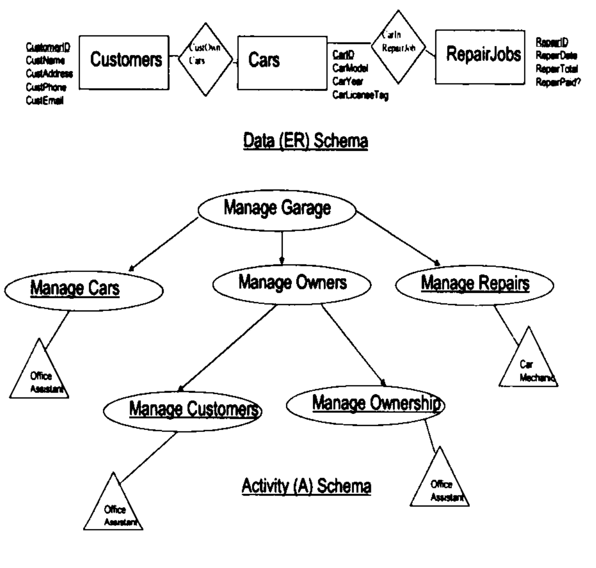


Figure 2. ERA diagram for an Automobile Garage Application: Data & Process Schemas

The ERA model provides two simple constraints: (i) Every primitive activity should have a R or W link to at least one data object (Entity or Relationship set), and (ii) Every data object must be read to or written by at least one primitive activity. These constraints serve as a check for ensuring the completeness of the ERA diagram. Figures 2 and 3 illustrate the above process with a complete ERA diagram for a simple automobile garage application.

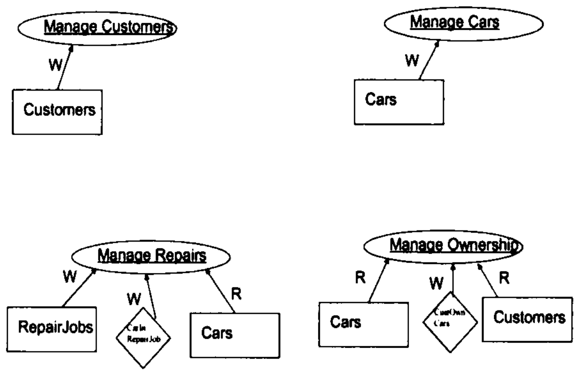
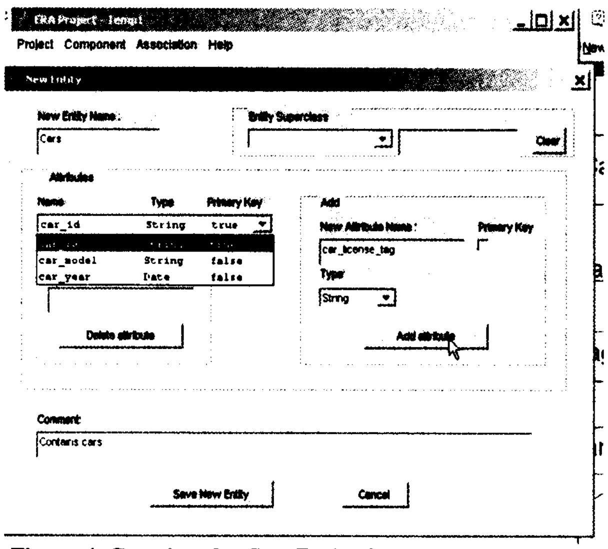


Figure 3. ERA diagram for an Automobile Garage Application: Linking primitive activities to data objects

Once the data and process diagrams are created, the ERA CASE tool (Bajaj 2004) is used to store the information in the diagrams. This tool is written completely using Java^{TM} to allow it to be run on diverse platforms. In the interests of scalability, the ERA CASE tool *does not* offer the ability to create diagrams in the software. This prevents analysts from expending resources constructing complicated diagrams when faced with large scale applications. Instead, the CASE tool acts as a repository for all the information elements in the diagrams. Hence, when using ERA in the field, analysts create multiple diagrams on paper or some other medium, resolve different diagrams *outside of the tool* and then finally input the corrected information about each subgroup's data model into the CASE tool.



New Entity Name: Cars

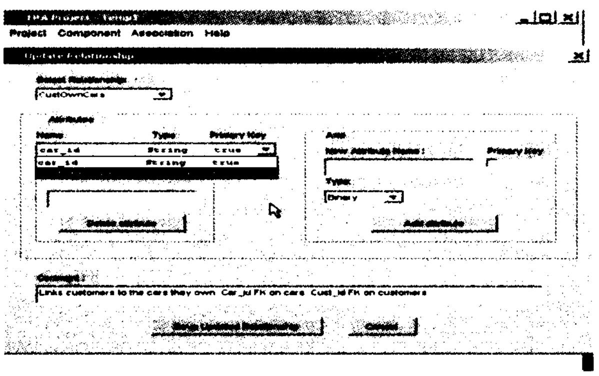
Name	Type	Primary Key
car_id	String	true
[illegible]	[illegible]	[illegible]
car_model	String	false
car_year	Date	false

New Attribute Name: car_license_tag

Type: String

Comment: Contains cars

Figure 4. Creating the *Cars* Entity Set in the ERA CASE Tool

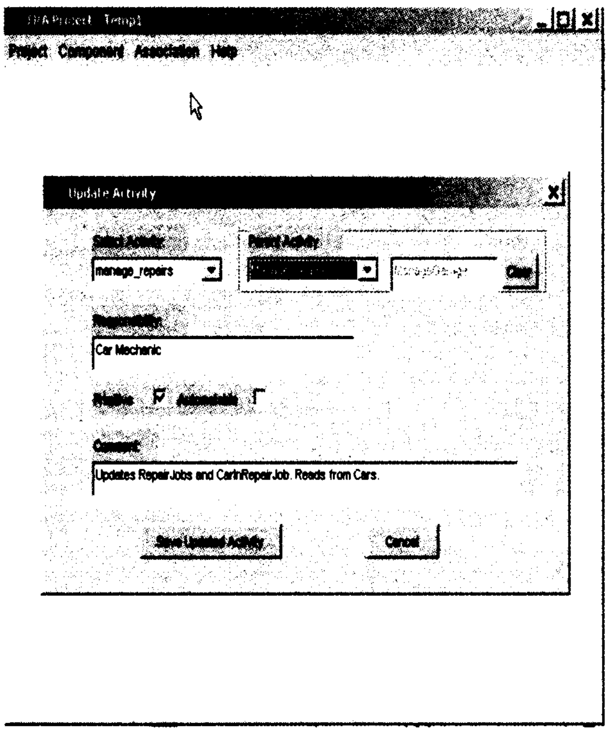


CustOwnCars

```
car_id          String    true
car_id          String    true
```

Links customers to the cars they own. Car_id FK on cars. Cust_id FK on customers

Figure 5. Creating the *CustOwnCars* Relationship set in the ERA CASE Tool



manage_repairs

Car Mechanic

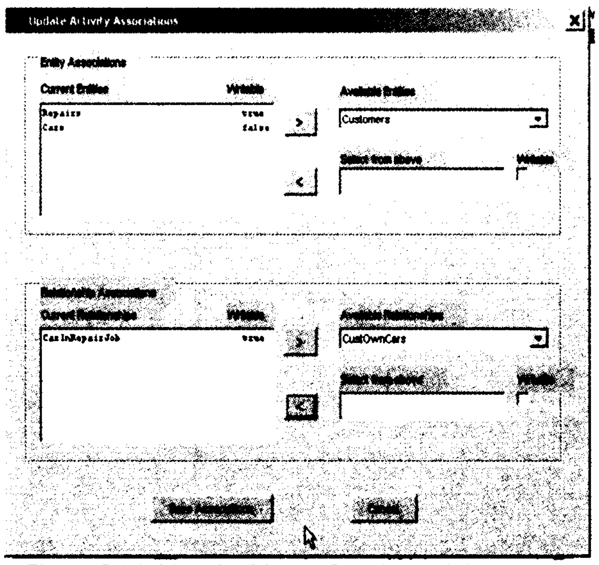
Updates RepairJobs and CarInRepairJob. Reads from Cars.

Figure 6. Creating the *ManageRepairs* Activity set in the ERA CASE Tool.

The ERA methodology facilitates the creation of several sub-diagrams for data and activities for different sub-groups in parallel. Figures 4, 5 and 6 show screenshots of the ERA CASE tool's interface screens to create entity sets, relationship sets and activities as they are added to the ERA repository.

Figure 7 illustrates how the Manage Repairs activity set is linked to the relevant data objects in the ERA tool.

The figures above illustrate the ease of use inserting, updating and deleting objects and links between objects in the repository of the ERA CASE tool.



```
Repairs          true
Cars             false
```

Customers

```
CarInRepairJob   true
```

CustOwnCars

Figure 7. Linking the *ManageRepairs* Activity to Data Objects in the Data Schema

After the information is input in the ERA CASE tool, a code plan can be printed out. The code plan lists all the entity sets and relationship sets as tables. The *comments* fields that apply to every object in ERA can be used to capture special requirements such as foreign keys, table sizes and the meanings of certain attributes. The non-automatable primitive activities are shown as screens. Again, the *comments* fields can be used to capture information such as activity order and a logical description of each activity. Finally, automatable primitive activities are shown as DBMS triggers in the code plan. For each activity, the data objects it consumes are also shown. This information, combined with the description of the activity in the *comments* section provides insight into the complexity of constructing that particular screen or trigger.

It is important to note that ERA **is not** a code generation tool; rather, it is used at the conceptual modeling level, where it can be used to capture detailed user requirements for large scale ISs. The output is a code-plan that is **independent** of what DBMS will be used, or what technology will be used to generate the screens and host them. Next, we illustrate the usage of ERA via a teaching case in the classroom. .

4. TEACHING CASE ILLUSTRATING ERA USAGE: THE CASE OF ACME CORP.

Appendix 1 contains an extended case that has been used in two courses at the undergraduate and graduate MIS level, across two different universities for teaching large scale information systems modeling using the ERA model and CASE tool. The case describes two departments of ACME Corp (Operations and Sales & Marketing) in considerable detail. The key contribution here is that the level of **detail** and **scope** of requirements is similar to what would be found in a real world implementation. Students form teams and first have to utilize the ERA methodology to create separate paper diagrams for each department. Next, these diagrams are input into the ERA CASE tool, and a code plan is output. The typical size of the systems analysis and design output consists of over 50 entity sets and relationships sets. Activities have to be created for each of these data objects. The final code plan generated by the CASE tool typically consists of between 80-120 tables, and approximately the same range of screens and triggers. Students are encouraged to write comments for each entity, relationship and activity set in the ERA case tool, since this leads to a more detailed code plan. Common comments include the types of constraints that will be required for tables in the construction phase, as well as the basic functionality of each activity (screen or trigger).

In the second phase of the project, the team uses the output of phase 1 to perform a build versus buy comparison for ACME by analyzing real COTS products available out there. The deliverable in this phase is a detailed report that describes the time and cost to build a system for ACME versus buying a COTS system. The team has to make a presentation of their final recommendation to the "ACME board" which consists of the Professor and the rest of the class. The key **learning outcomes of this project** are: a) the skill-set to perform large scale SAND, b) the experience of performing a detailed build analysis from a time and cost perspective, and c) the ability to compare build time/cost to buying a COTS system. As highlighted earlier in the paper, these are the skills that are lacking in our IS curricula today even though there is likely to be strong demand from industry for these skill-sets.

While the feedback from students who have gone on to industry has been extremely positive, one limitation of this project is that access to COTS systems information is somewhat limited, since the students are not actually considering a purchase. In an industrial setting, the main difference would be more access to functionality of the different COTS systems that could allow a formal "fit-gap" analysis between the data & process needs of the enterprise and the COTS system.

Next, we summarize approaches, based on our teaching experience that can help educators teach large scale modeling using ERA more effectively. First, as with most SAND methodologies, it is advisable to build up slowly, using examples solved slowly by the Professor followed by in-class exercises of small requirements. We use mini cases that consist of one or 2 paragraphs each. Second, we recommend students should already have been exposed to data modeling in a prior database course. The major new concept that students need to grasp is the linking of activities with data (entity and relationship sets). The pedagogical strategy that has worked well for the smaller examples is to develop the data model first and then think about the activities. As the examples get bigger, students may be encouraged to think about iterating between activity and data diagrams and using one to add to the other.

Third, as students get exposed to activity diagrams in ERA, it seems beneficial to compare these with other process models they may have been exposed to, such as data flow diagrams or use-case in UML. This allows students to build on existing knowledge. The chief differences between ERA activities and processes in data flow diagrams are that processes in data flow diagrams are linked only informally to data using data flows, whereas activities in ERA are linked to formal data elements. There is no notion of information flow between processes and data stores in ERA. The use case model from UML is analogous to the higher level activities in ERA, and the main difference here is that ERA allows for activity decomposition as well as explicit links between activities and data.

Fourth, as students attempt to model primitive activities, it is easier to depict the standard create-update-delete (CRUD) data maintenance activities first. The next step is to model activities that represent business processes, either by combining different CRUD activities under a higher level activity that represents the business process or by depicting individual business process activities at the primitive level. After this step, students can also model reports required from the system, as activities with read-only access to data elements.

Fifth, as students enter data from the diagrams into the CASE tool, they should be encouraged to use the *comments* field to capture information that will be helpful to developers of the system, downstream. This includes foreign key dependencies and CHECK clause constraints for the data elements, and overall logic and expected behavior for the activities. The degree of information in the code-plan should be one quality determinant, with the correctness being the other.

Finally, once the code plan is generated, students can be encouraged to apply it further in at least two ways: a) to generate a prototype consisting of screens that conform to the code-plan, and/or b) to do a cost and time analysis for building the system and comparing it to off the shelf systems of comparable scope.

5. CONCLUSION

The decision to build or buy a large scale information system is becoming increasingly important. While historically, organizations built customized large scale ISs, the trend over the last decade, has reversed overwhelmingly to purchasing/renting of COTS based systems. However, we see the choice of building custom ISs becoming increasingly attractive, for the following reasons: a) organizations have become increasingly more aware of the disadvantages of COTS based ISs, b) technologies to build custom ISs have become easier to understand and manage, c) the resources required to outsource the construction of customized ISs have become less, as programming costs have declined, and d) organizations have become increasingly aware of testing and matching an IS system before approving any large capital outlay.

Given this choice of build versus buy, systems analysis and design will play an important role in documenting end-user data and process needs. A detailed documentation will provide: a) a basis for estimating the resources required to build a customized system, b) an ability to compare the functional "fit-gap" between the end-user requirements and the features offered by different COTS systems and c) the ability to better manage the construction phase if custom development is selected. In this work, we described how one possible model: the ERA model and its supporting CASE tool can be used to document data and process requirements on a large scale. The field study example provides a flavor for how easy the ERA model is to use with large scale requirements when building actual systems. The teaching case illustrates how large scale SAND modeling can be developed as a skill-set in MIS curricula. ERA has been successfully used in many organizations, and has also been taught in the classroom, in various forms, to over 250 graduate and undergraduate students. The ERA CASE tool is available as a free download at http://nfp.cba.utulsa.edu/bajaja/ERA/. We hope that this tool is considered for adoption in IS curricula as they move towards increased emphasis on SAND for large scale IS applications.

The work presented here can be extended in several ways. First, we plan to test the scalability of ERA more formally with laboratory based experiments. Second, while the selection of the concepts in ERA was based on our intuitive understanding of concepts that promote scalability, it would be interesting to evaluate the richness versus complexity trade-off by adding concepts such as weak entity sets to ERA. Finally, we plan to evaluate the impact of the ERA methodology on software engineering practices of an organization, i.e., measure the extent to which utilizing the ERA methodology promotes conformity with software engineering aspirations such as achieving higher capability maturity levels.

6. REFERENCES

Abts, C. (2002). COTS-based Systems and Make vs. Buy Decisions: The Emerging Picture. International Workshop on Reuse Economics, Austin, TX.

Abts, C. and B. Boehm (1998). COTS Software Integration Cost Modeling Study. Los Angeles, CA, University of Southern California.

Alexander, I., S. Robertson, et al. (2005). What Influences the Requirements Process in Industry? A Report on Industrial Practice 13th IEEE International Conference on Requirements Engineering (RE'05), IEEE.

Alter, S. (2006). The Work System Method: Connecting people, Processes and IT for Business Results. Larkspur, CA, Work System Press.

Alter, S. and G. Browne (2005). "A Broad View of Systems Analysis and Design." Communications of the AIS **16**(50): 981-999.

Applegate, L. M., R. D. Austin, et al. (2006). Organizing and Leading the IT Function. Corporate Information Strategy and Management
McGraw-Hill Irwin: 424-425.

Bajaj, A. (2004). "The ERA CASE Tool." Retrieved Feb 1, 2006, 2004, from http://nfp.cba.utulsa.edu/bajaja/ERA/index.html.

Bajaj, A., D. Batra, et al. (2005). "Systems Analysis and Design: Should We Be Researching What We Teach?" Communications of the AIS **15**(April): 478-493.

Bajaj, A. and S. Ram (2002). "SEAM: A State-Entity-Activity Model for a Well Defined Workflow Methodology." IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE) **14**(2): 415-431.

Baker, S. and D. O'Sullivan (2001). Positioning CORBA, J2EE, web services and other middlewares. Third International Symposium on Distributed Objects and Applications, IEEE.

Booch, G. (1994). Object Oriented Analysis and Design with Applications. Redwood City, Benjamin/Cummings.

Brooks, F. P. (1987). "No Silver Bullet: Essence and Accidents of Software Engineering." IEEE Computer **20**(4): 10-19.

Chen, P. P. (1976). "The Entity-Relationship Model: Towards a Unified Model of Data." ACM Transactions on Database Systems **1**(1): 9-36.

Davenport, T. H. (1998). "Putting the Enterprise Into the Enterprise System." Harvard Business Review **76**(4): 121-131.

Eckerson, W. E. (2002). "The Rise of Analytic Applications: Build or Buy?" Retrieved July 15, 2004, from www.dw-institute.com.

Gould, L. S. (1999). "Reinventing ERP After Y2K." Retrieved July 30, 2004, from http://www.autofieldguide.com/articles/119907.html.

Group, P. C. (2000). "Unlocking the Value in ERP." 2000, from http://pa-consulting.com/publications/index.html.

Higaki, W. H. (1995). "Applying an Improved Economic Model to Software Buy Versus Build Decisions." Hewlett-Packard Journal **46**(4): 61-66.

Johnston, S. J. (2002). ERP: Payoffs and Pitfalls. Harvard Business School: Working Knowledge.

Khalid, S. (1998). "Maintaining Professional Competence in Innovation Organizations." Human Systems Management **17**(1): 69-87.

McConnell, S. (1996). Rapid Development: Taming Wild Software Schedules, Microsoft Press.

McManus, D. J. (2003). "A Model of Organizational Innovations: Build versus Buy in the Decision Stage." The International Journal of Applied Management and Technology **1**(1).

MindBridge. (2004). "Build versus Buy: Which Software Implelementation Strategy Should You Choose for Your Organization?" Retrieved July 25, 2004, from http://www.mindbridge.com/white_papers/ExaminingROIForIntranet.html.

Ross, J. (1999). "Dow Corning Corporation C:
Transforming the Organization." Retrieved July 30, 2004, from http://web.mit.edu/cisr/www/html/jwr_cases_6.html.

Singh, P. and E. A. Walden (2003). Flexibility and Cost in Information Technology Outsourcing: Balancing Opposing Goals. Ninth Americas Conference on Information Systems, Tampa, FL.

Taeschler, D. (2002). Metrical Approaches to Customer Equity Through CRM. Montgomery Research CRM Project. 3.

Ulrich, W. M. (2004). Stop Treating the Symptoms and Treat the Disease. Business Intelligence Journal Online.

Vedaris. (2004). "The Buy Versus Build Decision." Retrieved July 30, 2004, from www.vedaris.com/pdf/buybuild.pdf.

Verville, J. and A. Halingten (2001). Acquiring Enterprise Softwar: Beating the Vendors at their Own Game, Prentice Hall.

AUTHOR BIOGRAPHY

Akhilesh Bajaj is Chapman Associate Professor of MIS, at the University of Tulsa. He received a B. Tech. in Chemical Engineering from the Indian Institute of Technology, Bombay in 1989, an MBA from Cornell University in 1991, and a Ph.D. in MIS (minor in Computer Science) from the University of Arizona in 1997. Dr. Bajaj's research deals with the construction and testing of tools and methodologies that facilitate the construction of large organizational systems, as well as studying the decision models of the actual consumers of these information systems. He has published articles in several academic journals such as *Management Science, IEEE Transactions on Knowledge and Data Engineering, Information Systems* and the *Journal of the Association of Information Systems.* He is on the editorial board of several journals in the MIS area. His research has been funded by the department of defense (DOD). He teaches graduate courses on basic and advanced database systems, management of information systems, and enterprise wide systems.

Appendix 1: Requirements for the ACME Corporation

The ACME corporation makes different types of products for sale directly to the general public. Examples of product types include the ACME bicycle, the ACME scooter and the ACME skateboard. ACME consists of a chief executive officer (CEO) who runs the company overall and oversees two departments. These departments are the sales & marketing department and the operations department. Next, the departments are described in detail.

ACME Sales & Marketing

There are 20 people in ACME's Sales and Marketing Department. The entry level position is sales associate or marketing associate (different positions). A group of sales and marketing associates is managed by a sales and marketing manager. All the sales and marketing managers form one group managed by the Sales and Marketing VP (Vice President). The VP reports directly to the CEO. In addition, there is one front desk for the department, managed by the sales and marketing receptionist, who supports everybody, but reports to the sales and marketing VP. The performance of the receptionist is measured by conducting a quarterly survey that is filled in by all the other employees of sales and marketing. Each quarterly survey has a *survey_id*, a *date* and a *rating* (on a 1-10 scale) from each sales and marketing employee.

Each employee in the sales & marketing department has an *employee_id*, a *name*, an *office address*, a *home address*, an *office phone*, a *home phone*, a *cellular phone*, an *e-mail*, and a *title* (which is either sales associate, marketing associate, sales and marketing manager, sales and marketing VP or sales and marketing receptionist).

The department maintains a listing of customers. Customers are people or organizations who have bought from ACME in the past. Each customer has a *customer_id*, a *name*, a *company name* (the company that they work for), *address*, *phone*, *cell phone*, *e-mail*, and a *status* (which is either gold, silver or bronze). The sales and marketing employees keep records of all conversations with customers. Each conversation has a *customer_conversation_id*, and a *summary of conversation*. Each customer conversation takes place between exactly one customer and one or more employees in the sales and marketing department.

The department also has a list of leads. Leads are people or organizations who have not yet bought anything from ACME, but may buy in the future. Each lead has a *lead_id*, a *name*, a *company name*, *address*, *phone*, *cell phone*, *e-mail* and a *status* (cold, warm or hot). The sales and marketing employees keep records of all conversations with leads. Each conversation has a *lead_conversation_id*, and a *summary of conversation*. Each lead conversation takes place between exactly one lead and one or more employees in the sales and marketing department

The performance of sales associates is measured by the dollar value of the products that they sell every month. Some of the times, customer conversations or lead conversations result in orders. An order has an *order_id*. Each order is for one or more product items, with each product item being of one product type. Each product type that ACME sells has a *unit price*, which is the price for one item of the product type. In addition, each product type has a *product_type_id*, a *name* and a *quantity_on_hand* (the number of product items of that product type, available for sale). The *dollar value* of each order is *added up*, and the credit for each order goes to at least one sales and marketing associate, though possibly more. Each order is given by only one customer. If a lead conversation results in an order, then that lead is converted into a customer. As soon as a lead becomes a customer, then their previous lead conversations are deleted.

The department is responsible for advertising campaigns. Each advertising campaign is the responsibility of one marketing associate. Advertising campaign are conducted in *media outlets* (like newspapers, trade journals, television and radio). Each advertising campaign has an *ad_campaign_id*, a *name*, a *budget_total*, a *costs_to_date*, *date_began* and *date_ended*. Each media outlet has a *media_outlet_id*, *name* (*e.g.*, post-gazette is a name), a *unit_cost_of_ad*, an *ad_sales_person_name*, *phone*, *cell phone* and *e-mail*. Each ad campaign is linked to several media outlets. *E.g.*, an ad campaign can be one that is linked to ads running in the post-gazette and WFUN FM (both media outlets) from the dates Jan 20, 2000 to June 20, 2000.

In order to keep track of advertising effectiveness, ACME also asks each customer and lead, in each conversation with them, if they heard about ACME since the time of the last conversation. The date and the media outlet which the customer or lead says is recorded and linked back to the ad campaign that it belongs to. Of course, in each conversation, a customer or lead can say that they heard about ACME through a number of different media outlets, on different dates. Each of these is recorded and linked back to the relevant advertising campaign.

The performance of the sales and marketing managers is measured by the total dollar of sales generated by the sales associates who work for that manager, as well as by the advertising effectiveness of the advertising campaigns that were run by that marketing associates who work for the manager.

The performance of the VP sales & marketing is measured by the total dollar value of the sales generated by the sales & marketing department, as well by the advertising effectiveness of the advertising campaigns run by the department. Performances of all sales and marketing employees are assessed every quarter. Of course, advertising campaigns may span quarters, so that, for example, one campaign may run for five months, while another may run for two years.

In order to perform its activities, the department gets a budget that is approved by the CEO. They also keep track of past budgets. Each budget has a *budget_id, date_began*, a *date_ended*, and a *total* field. Each budget has a list of budget items, with each item in the budget having an *item_id*, *activity_name*, and *amount*. Budgets are for a quarter. Quarters are from Jan - March, April - June, July – September and Oct. - December.

Some processes that the sales and marketing department performs include:
a) making a sales call to a customer or a lead,
b) creating a new or conducting an ongoing advertising campaign,
c) creating new products, or updating information about existing products, and
d) in general creating new information or updating information on existing data that is necessary for the day to day running of the department
e) converting a lead to a customer.
f) changing the status of a customer or a lead

However, other processes are also performed, so that all information items are modified in-house.
All of the information needs of the ACME sales & marketing department are contained in the description above.

ACME Operations

ACME currently has 35 people in the operations department. The department is responsible for actually manufacturing the different products that ACME sells. It is broken up into different shops, like the fabrication shop, the assembly shop and the packaging shop. Each shop has several different machines (like lathes, milling machines, drills and packaging machines). Each machine has a *machine_id*, *machine_name*, *location_description*, *function_description*, *date_machine_was_purchased* and *date_last_maintenance*. Each shop has machinists, with at least one and possibly more machinists per machine. The machinists work in shifts, with each shift being 8 hours. Each shift takes place in one shop, and has a *shift_id*, a *time_began*, and a *time_ended*. As per union laws, each machinist can work in a maximum of 6 shifts a week, and a minimum of 4 shifts a week. Each shift is managed by one foreman. The foremen report to the shop managers (each shop is managed by a shop manager). The shop managers report to the Operations Vice President (VP).

Each employee in the operations department has an *employee_id*, a *name*, an *office_address*, a *home_address*, an *office_phone*, a *home_phone*, a *cellular_phone*, an *e-mail*, and a *title* (which is machinist, foreman, shop manager, receiving clerk, stocking clerk, shipping clerk or operations VP).

The operations department follows a defined process for manufacturing each product type. A product type is a *type of product* (like the ACME skateboard), as opposed to the *product item*, which is the actual item itself (like an actual, physical ACME skateboard item). The operations department makes product items. Each product type has a *product_type_id*, a *name*, a *date_production_began* (the date it was first produced), and a *date_production_ended* (the date it was not offered anymore and ACME stopped manufacturing it). Each product also has associated with it a *work plan*, which includes a set of *drawings*, and a *process flow*. A drawing can only belong to one product. A process flow can only belong to one product. Each product item is typically made on a number of machines, and may go through different shops. For each product type, ACME has the amount of time required on each machine to make a product item for that product type. For each product, ACME also records the different types of raw material needed for the product type, as well as the required quantity of each raw material type.

The operations department also has a list of raw material types (like steel rods, steel bars, cast iron bars, aluminum rods, and so on). Each raw material type has a *raw_material_type_id*, and a *name*. Raw material items are supplied by suppliers. Each supplier has a *supp_id*, *name*, *address*, *phone*, *contact_name*, *e-mail* and *web_page*. A supplier can supply many raw material types. Each raw material item is supplied by one supplier. A raw material item is an actual physical item of a raw material type (*e.g.*, 5 steel bars that were supplied by supplier Bob in one shipment).

When ordering raw material items, the operations department uses purchase orders. *E.g.*, When a supplier supplies 5 steel bars that are requested on one purchase order, then that is considered to be one raw material item on that purchase order.

A purchase order has a *po_id*, a *date_sent* and a *list of raw material types that were ordered*, including the *unit price* and *quantity* of each raw material type ordered. Each purchase order is made out by one receiving clerk. Receiving clerks report directly to the Operations VP. The receiving clerks also follow up periodically on the purchase orders they have made. Each purchase order followup is identified by the *po_id* and the *date on which it was made*. In addition, *a summary of the conversation* that took place during the followup between the *supplier representative* (who may be different from the supplier *contact_name*) and the receiving clerk is also recorded. While all the raw materials on a purchase order are ordered on the same date (the date of the purchase order), the supplier may deliver the items on *different dates*. So, for each raw material type on a purchase order, the *date_requested* and *date_actually_received* are captured.

The performance of the receiving clerks is measured by the number of total purchase orders they process in each quarter and the average delay in receiving raw materials (computed by looking at the average difference between *date_requested* and *date_actually_received* for all items processed by the receiving clerk in a quarter).

Once the raw material items are received, they are handed over to a stocking clerk. All stocking clerks also report directly to the Operations VP. The stocking clerk puts the items in raw material inventory. Each raw material inventory shelf has a *raw_shelf_id* and *a description of the shelf location*. Each item in the inventory has a *raw_material_type_id*, a *raw_material_item_id*, a *name*, a *quantity*, a *date_put_in_inventory*, a *raw_shelf_id* and the *supplier_id* of the supplier who supplied it and the *employee_id* of the stocking clerk who shelved it.

The performance of stocking clerks is based on the number of raw material items they shelve every quarter, as well as the average delay in shelving materials (computed by looking at the average difference between *date_actually_received* for a raw material item and the *date_put_in_inventory* for that item, for all items shelved by that stocking clerk in that quarter).

When a product item is to be manufactured, the operations department creates a *new product item* in its system before it starts manufacturing the item. Each product item has a *prod_item_id* and is of one product type. The relevant raw materials are removed (from different raw material items in the raw materials inventory) and the manufacturing of the product item is started. Typically, the raw materials make their way through the machines that are required to make the product item and are converted by each machine. We can think of each machine as receiving some unfinished items in each shift, and converting part or all of it to finished items (for that machine). The finished items are then fed to the next machine, and become the unfinished items for that machine. Each unfinished item for a machine has a *raw_material_item_id* (this links it to the raw material item it came from), a *time_reached_machine* (the time it reached that machine), and a *time_manufactured* (the time it came out of that machine as a finished item).

The productivity of each machinist is measured by dividing the amount of time that *should have* been taken to make all the finished items that were made on that machine in that shift divided by the time duration of the shift. Of course, in the same shift, the machine may be used to help in the manufacture of product items for different product types. So, if the duration of a shift is 8 hours and the machine produced finished items on it that should have taken 10 hours (according to the different product specifications), then the machinist's performance is 10/8. If another machine produced only 6 hours worth of work in an 8 hour shift, then that machinist's performance is 6/8. A performance of 1 or higher is excellent, 0.8 to less than 1 is good and below 0.8 is poor.

The performance of each foreman is measured by the combining the performance of all the machinists in the shift. The performance of each shop manager is measured by combining the performances of all the foremen in that shop.

Once a product item is completed (this means that a complete product item has been made), it is ready to be move to the *finished goods inventory*. Each completed product item is taken by one shipping clerk and moved to the finished goods inventory. Each finished inventory shelf has a *finished_shelf_id*, and a *description of where the shelf is*. Each completed product item is of one product type, and has a *prod_item_id*, a *finished_shelf_id*, a *date put_in_finished_inventory* and the *employee_id* of the shipping clerk who put it in the finished goods inventory.

The performance of the operations VP is measured by combining the performance of all the shop managers, the receiving clerks, the stocking clerks and the shipping clerks.

In order to perform its activities, the operations department gets a budget that is approved by the CEO. They also keep track of past budgets. Each budget has a *budget_id*, *date_began*, a *date_ended*, and a *total* field. Each budget has a list of *budget_items*, with each item in the budget having an *item_id*, *activity_name*, and *amount*. Budgets are for a quarter. Quarters are from Jan - March, April - June, July – September and Oct. - December.

<u>Some processes that the operations department performs include:</u>

a) creating and sending a purchase order (by the receiving clerk),
b) manufacturing a product item,
c) moving a finished item to shipping inventory (shipping clerk)
d) performing maintenance on a machine
e) getting a new machine,
f) moving raw materials into raw material inventory (stocking clerk)
g) receiving or updating information on new or existing product types

However, other processes are also performed, so that all information items are modified in-house. All of the information needs of the ACME operations department are contained in the description above.

Project Requirements:

ACME corporation is considering using an EWS system that will initially combine the marketing and operations departments, but which can hopefully later be extended to include human resources, accounting, finance, customer service and other departments as well. A senior manager at ACME has approached your consulting firm to develop a recommendation report for them. ACME currently has 4 people in their IT department, including one DBA, one systems administrator and two network administrators. They are open to expanding this, as they either build or buy an EWS.

Our consulting report should consist of the following:
Phase 1: The Enterprise model of ACME, using the ERA CASE tool. Deliverables here will consist of neat sketches of the ER diagram, activity trees (automatable & non-automatable), and the output of the code plan. Note that we'll have multiple roles (responsibilities) in ACME.
Phase 2: A **build versus buy** decision for ACME. In order to do this, we need to actually do a market analysis of vendors and consultants, using whatever sources we can lay our hands on (including making phone calls to vendors, web browsing, trying to obtain market surveys if vendors, reading up trade publications, and looking at other companies that have acquired EWS systems (our case studies). Based on our search, we should come up with a **cost analysis** of how much it will cost to buy an OTS EWS. We should also come up with a **time-line** of how long it will take to implement the OTS EWS. We should note that a cost analysis includes items like hardware, software, consulting time, and the time spent by ACME employees who will be involved in the acquisition and implementation team.

To analyze the **build** option, we should consider different technologies that can be used to build in-house, large database systems. These include JSP or ASP and Oracle or Microsoft SQL server. We can select one technological solution, and, based on the enterprise model created in phase 1, provide a detailed costing of how much money it would cost (including hiring developers or technical consultants, hiring new IT people, new hardware, software, etc) to build the system. We should also provide a detailed time line for building the system.

Finally, we should provide a one page memo that recommends whether ACME should build the system in house, or buy the system, and if so, from which vendor. We should note that vendors are not just the large ERP vendors, but there are many smaller vendors also, that may be more suitable for ACME.
A detailed description of the report will be available in about 10 days.
As part of the project, we will do a presentation to the ACME board (an in class presentation).

Grading criteria:
Phase 1: -The correctness of the ERA model.
-The completeness and detail captured in the ERA model.
Phase 2: -The completeness of the analysis and the attention to detail
-The quality of the presentation (professional, convincing).

Due Dates:
Phase 1: Due in 2 weeks
Phase 2: Due in 5 weeks

Team Size: Min 1 – max 4. .

Information Systems & Computing
Academic Professionals

Serving Information Systems Educators

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096