JOURNAL OF INFORMATION TECHNOLOGY CASE AND APPLICATION RESEARCH
2016, VOL. 18, NO. 2, 93–119
http://dx.doi.org/10.1080/15228053.2016.1198623

Routledge
Taylor & Francis Group

RESEARCH ARTICLE

Relationship analysis: Formalizing relationships within the systems analysis process

Joseph Catanio[a] and Michael Bieber[b]

[a]Shippensburg University; [b]New Jersey Institute of Technology

ABSTRACT

A significant aspect of the systems analysis process involves discovering and representing entities and their inter-relationships. This article presents relationship analysis (RA), a systematic, domain-independent systems analysis technique to determine a domain's relationship structure. The quality of design artifacts, such as class diagrams, can be improved by first representing the complete relationship structure of the problem domain. A rigorous evaluation was conducted, including a formal experiment comparing novice and experienced analysts with and without the use of RA. In addition, professional software industry engineers were interviewed and educated about RA. It was shown that the RA process provides a fuller and richer systems analysis, resulting in improved quality of class diagrams. It was also shown that RA enables analysts of varying experience levels to achieve class diagrams of similar quality.

Introduction

The literature indicates that the best way to improve the software development life-cycle is to enhance it during the early stages of the process (Booch, Jacobson, & Rumbaugh, 1998; Faulk, 2000; Sommerville, 2001; Wieringa, 1998)—in particular, during the requirements elicitation, analysis and design phases. During the system analysis phase, software components are determined through the identification process of the system's entities and relationships. Informal guidelines exist to help identify entities and objects; however prior to Yoo's (2000) dissertation on an initial version of relationship analysis (RA), no guidelines existed to analyze an application domain in terms of its relationship structure. No defined processes, templates, or diagrams existed to explicitly and systematically assist in eliciting relationships or documenting them in *class diagrams* or *entity-relationship (ER) diagrams* (Beraha & Su, 1999). However, relationships constitute a large part of an application domain's implicit structure. Completely understanding the domain means knowing how all the entities are interconnected. Relationships are only lightly addressed by class and ER diagrams. These diagrams capture a limited subset of relationships and leave much of the relationship structure out of the system model. While analyses and models are meant to be a limited representation of a system, the incomplete relationship specification is not by design, but rather a lack of good methodology to determine them explicitly (Bieber, 1998; Bieber & Yoo, 1999). As a result, many analyses miss aspects of the systems they represent. Therefore meaningful research was needed to create a process to more explicitly elicit and document the relationship structure of a problem domain, which should enhance the creation of class diagrams. We have created an improved methodology by revamping existing RA methodology with a theoretical model and a systematic approach. RA provides the requirements analyst with a straightforward method and tools that employ a systematic set of questions designed

CONTACT Joseph Catanio JTCatanio@ship.edu Department of Management Information Systems, John L. Grove College of Business, Shippensburg University, 1871 Old Main Drive, Shippensburg, PA 17257, USA.

Color versions of one or more figures in the article can be found online at http://www.tandfonline.com/utca.

to elicit the relationship structure of a problem domain from a domain expert, and document these in a set of specialized template forms and diagrams. RA thus fills a significant void in the systems analysis process (Catanio, 2004; Catanio & Bieber, 2005; Yoo, 2000).

As an example of the possibly myriad way(s) that entities are interconnected, and how one entity could lead to, influence, or otherwise impact others, consider an analyst tasked with designing or enhancing a "friend request" in a social networking application similar to Facebook. Some obvious features to include would be instructions, features to view people's profiles and search for acquaintances or people with certain characteristics, a way to accept or decline this request, and a way to block receiving such requests from certain people. However, a deeper analysis may reveal several other aspects of a friend request that could enrich the application or enable it to be utilized by other types of users, to be repurposed, or simply to discover aspects that had been overlooked initially. Each of these aspects could be modeled as a relationship between the friend request and either a new entity or an existing entity in a new way. The analyst would apply RA to elicit this enhanced relationship structure. For example, the application may assist the user in understanding the ramifications of a friend request with definitions, a way to understand the intentions of another person inviting one with a friend request, a way to define one's goals in building a particular "social" network, ways to tweak one's profile to facilitate these goals, ways to help one deal with rejection when a friend request is declined (or analyze one's online persona when friend requests are declined), a way to think about issues such as honesty and revealing personal information as part of such a request, and overall personal and societal issues surrounding friending. The preceding list is but a subset of possible relationships surrounding a friend request that the requirements analyst could elicit from a thorough analysis of the friend request entity's relationships with one or more domain experts. RA provides a systematic, theory-based methodology for conducting such a thorough analysis.

This article presents the underlying theory and the resulting well-grounded RA process, model, and tools, as well as empirical results showing RA's potential to support both experienced and inexperienced analysts in the creation of class diagrams. To support the researchers' assertion that software engineers currently do not have adequate tools to explicitly identify the complete relationship structure of a software specific problem, three practicing software engineers from three different software companies were interviewed and results of this field study are described later in text, in the section on Debriefing Session with Subjects.

Theoretical foundation

Conceptual models in systems analysis and design capture the meaning of a problem domain. The value of a model is its ability to identify and capture the relevant knowledge about a domain. In addition, models help to represent and improve user requirements to meet user needs (Bajaj & Ram, 1999). In order to categorically say that a conceptual model is complete (and thus the domain representation), it should be based on a theoretical model (Turoff, Rao, & Hiltz, 1991). In this section we provide the theoretical basis for RA. We start by introducing *structure of intellect (SI) theory* (Guilford, 1950), and describe how Turoff et al. mapped it to the computer application domain through the *hypermedia morphology model* (*HMM*; Turoff et al., 1991). We, in turn, mapped the hypermedia node and link representation to RA's entity and relationship representation for the domain of systems analysis. These mappings preserve the completeness of the representations, as we next explain.

The SI theory is a general theory of human intelligence, thus forming a basis for comparing and classifying the complete range of intellectual ability. Guilford designed SI with a focus on measuring creativity (Guilford, 1950), which is an integral aspect of systems analysis and brainstorming activities in general. The SI model classifies intellectual abilities into a three-plane system with independent cross sections, each comprised of contents, products, and operations (Guilford, 1956).

As depicted in Figure 1, SI includes five kinds of contents, six kinds of products, and five kinds of operations. Three independent planes result in theoretically 150 different components of intelligence. The three dimensions of the model specify first the operation, second the content, and third the product of a given kind of intellectual act. The convention (*Operations, Contents, Products*) is used to specify each component. For example, *Cognition, Semantic, Unit*—or CMU—represents cognition of a semantic unit. In this way the SI theory represents the major kinds of intellectual activities or processes as an interrelated three-dimensional model.

Researchers have extracted relevant subsets of the SI components for particular domains. Turoff et al. map SI to the computer application domain with their HMM (Turoff et al., 1991). Arguing that not all of the SI components are necessary for classifying computer application domains, they reduce it to two dimensions of products and operations by classifying all SI types of contents as one, namely semantic. The four SI contents—visual, auditory, symbolic, and behavioral—while useful in classifying tests of intellect, are not necessary for classifying application domains. In addition, the SI operations of evaluation and memory are also not necessary for classifying application domains (Turoff et al., 1991). As we shall describe, the RA approach maps HMM to the entity and relationship domain for systems analysis. RA repurposes HMM's resulting 2-dimensional semantic classification model (Products and Operations) specifically for relationship classification.

The concept of products (the products dimension in Guilford's model) represents the form in which information occurs (Guilford, 1967; Meeker, 1969) and will be used in RA to focus on the following aspects of entities.

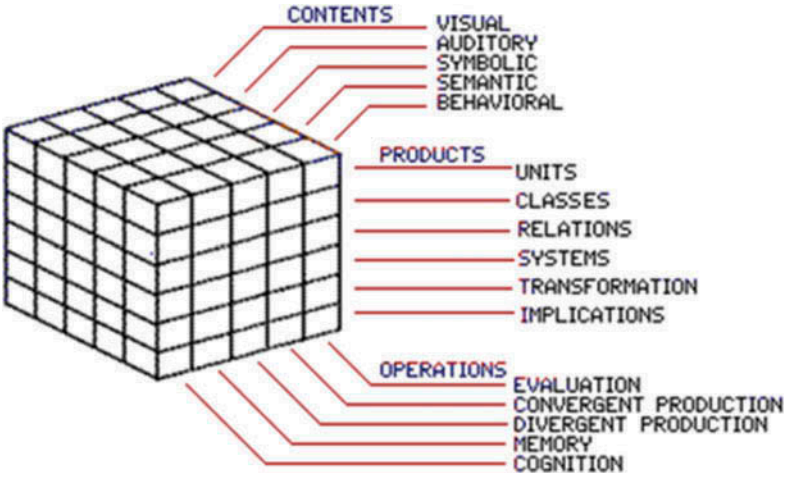

Figure 1. Guilford's structure of intellect model (Guilford, 1950).

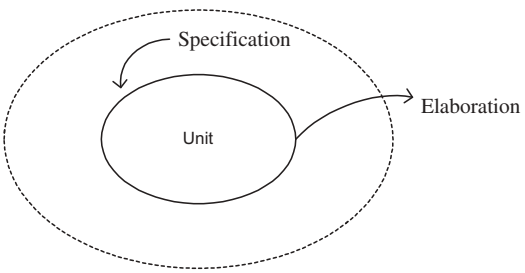

Figure 2. Unit focus.

- Units: Most basic item. Things to which nouns are normally applied. Described units of information.
- Classes: Sets of items of information grouped by virtue of their common properties.
- Relations: Connections between items of information based on variables or points of contact that apply to them.
- Systems: Organized or structured aggregates of items of information.
- Transformations: Changes, redefinition, shifts, or modifications of existing information or in its function.
- Implications: Extrapolations of information. Emphasizes expectancies, anticipations, and predictions.

The operations dimension from Guilford's model represents major kinds of intellectual activities or processes that analysts perform with information (Guilford, 1967; Meeker, 1969), and will be used in RA to elicit a complete set of relationships.

- Cognition: Discovery, awareness, or recognition of information by comprehension or understanding. Guilford views the cognition process as classifying an object. Turoff et al. (1991) extend this concept to hypertext whereby cognition is represented by a node that classifies all the linked objects as related to a common concept or characteristic. Hypertext at its core, concerns nodes (elements-of-interest) and links (relationships). These links or relationships among nodes are classified under convergent and divergent production properties. RA differentiates itself from the HMM in its application of cognition. The HMM represents cognition by a node and in hypertext terms: a node is an endpoint, and relationships exist among nodes or endpoints. In contrast, the relationships of each element-of-interest in RA are represented by six cognitive focus perspectives— *units, classes, relations, systems, transformations*, and *implications*.
- Convergent production: Generation of information from the given information, where the emphasis is on achieving unique best outcomes. Guilford (1967) views convergent production as when the input information is sufficient to determine a unique answer. Turoff et al. (1991) extend this concept so that a convergent link is a relationship that follows a major train of thought. This is referred to as a *convergent relationship* in RA.
- Divergent production: Generation of information from the given information, where the emphasis is on variety and quality of output from the given information. Guilford (1967) views divergent production as fluency and flexibility of thinking. Turoff et al. (1991) extend this concept so that a divergent link is a relationship that starts a new train of thought. This is referred to as a divergent relationship in RA.

To summarize the transformation from SI, RA uses Guilford's categories from SI, condenses them in the same manner as Turoff et al. (1991), and re-labels several to reflect the goal of relationship discovery and documentation. The differences between the HMM and RA are semantically metaphoric. RA refocuses or expresses HMM anew for the domain of relationships among entities within the domain of software systems analysis. The HMM, interprets the "products" as nodes or endpoints, while RA interprets "products" to represent the six possible *cognitive foci* of an item or "element of interest" being analyzed. (Recall that Guilford views the cognition process as object classification.) The "operations" now represent relationships that either conceptually *converge* or *diverge* within this focus. Therefore, it is possible to classify relationships of an element of interest in terms of six products each of which has convergent and divergent relationships. Table 1 depicts the 18 cells of RA, which we refer to as the *relationship analysis model (RAM)*, using SI nomenclature.

The RAM contributes a new and theoretically complete model specifically classifying the relationships in a software system. It can be directly applied during the requirements analysis stage in

Table 1. Relationship analysis model using SI nomenclature.

	Operations		
Products	Cognition	Convergent production	Divergent production
Unit	Cognition, Unit	Convergent, Unit	Divergent, Unit
Class	Cognition, Class	Convergent, Class	Divergent, Class
Relation	Cognition, Relation	Convergent, Relation	Divergent, Relation
System	Cognition, System	Convergent, System	Divergent, System
Transformation	Cognition, Transformation	Convergent, Transformation	Divergent, Transformation
Implication	Cognition, Implication	Convergent, Implication	Divergent, Implication

guiding analysts to effectively elicit a domain's relationships from an expert for inclusion in the system design.

The following section applies the aforementioned concepts and describes the RAM in detail.

Relationship analysis model (RAM)

Taking SI theory from Guilford and HMM from Turoff et al., our goal is to fully describe the relationships surrounding an element of interest using the meanings of the cells outlined in Table 1. These theories give us a model comprising a set of sound relationships for describing systems. Identifying the relationships between various parts of a system represents a systems thinking approach to understanding the problem (Rubenstein-Montano et al., 2001). Our research further applies the HMM model to develop a systematic systems analysis approach. A systems thinking approach improves business processes (Chun, Sohn, Arling, & Granados, 2009). During the analysis process, documents and dialogue (discussions, knowledge elicitation and brainstorming) provide analysts with descriptions of desired system functionality. Through these interactions it is possible to identify elements of interest. Thus, the relationships surrounding an element of interest (cognitive product in SI terms) are described by six cognitive focal points. Relationships of each focal point are classified under convergent and divergent operation properties. The convergent and divergent relationships are not opposites. Rather they provide two separate emphases for exploring the relationships under each of the six cognition foci. The following sub-sections describe the six focal aspects of the corresponding RAM classification of relationships depicted in Table 2.

Elements of interest can also be seen as different objects that exist in an application's problem domain. For every object that is identified there are a number of relationships that may surround it. Each RA category contains a series of questions to be used by systems analysts and domain experts which facilitate identification of a given object's various relationships. To elicit these relationships a systems analyst would pose questions to a domain expert in relation to each of the identified objects. The following provides a description and sample questions for each of the six focus aspects.

Unit focus

Guilford views a unit as descriptions or definitions of items of information (Guilford, 1967), before they form collections or groupings. Guilford views the cognition process as the classification of an

Table 2. Relationship analysis model (RAM).

Cognition focus	Convergent relationship	Divergent relationship
Unit	Specification	Elaboration
Collection	Membership	Aggregation
Comparison	Generalization/specialization	Similar/dissimilar
System	Structure	Occurrence
Transformation	Modify	Transpose
Implication	Influence	Extrapolate

object and as such, cognition of a unit takes the form of defining the object. These descriptions or definitions can be explicit or implicit. Explicit descriptions yield specific relationship types. In contrast, implicit relationships are uncovered as descriptions are further elaborated. Descriptions provide characteristics of items of information, which are attributes, also known as metadata. Thus, metadata relationships are identified within the unit focus.

Guilford views convergence as when the input information is sufficient to determine a unique answer (Guilford, 1967). Guilford views *convergent production of a semantic unit* as explicitly *specified* in the description of the element of interest, which corresponds to the first row, second column of Table 2. Thus, the corresponding relationships connect the unit (element of interest) to other elements that provide some descriptive specification. The following list is an example of knowledge elicitation questions to find specific convergent relationships (Yoo, Catanio, Paul, & Bieber, 2004).

- Does the item have a description?
- Does the item have a definition?
- Does the item have an explanation?
- Does the item have a set of instructions?
- Does the item have an illustration?

Guilford views divergence as flexibility of thinking (Guilford, 1967). When considering *divergent production of a semantic unit* (corresponding to the first row, third column of Table 2), divergent relationships are determined as descriptions that are further *elaborated*. These types of relationships are generally found not within, but just below the surface of the description of an element of interest. The following list is an example of questions to elicit divergent relationships.

- Does the description fully describe the item?
- Does the definition fully encompass the item?
- Does the explanation make assumptions?
- Is the set of instructions complete?
- How can this item be expanded or broadened?

Both structured and object-oriented analysis utilize functional definitions to help perform the analysis (Borgida, Mylopoulos, & Wong, 1984; Brachman, 1983; Martin & Odell, 1995; Smith & Smith, 1977). Jacobson's use-case analysis technique has made the process more explicit by generating descriptions of the use-cases (Booch et al., 1998). Use-case descriptions are narratives that describe a functional aspect of the desired system. From these narratives it is possible to extract both explicit and implicit relationships. Unit (or definition) focus is depicted in Figure 2.

Collection focus

Collections are recognized sets of information grouped by virtue of their common properties (Guilford, 1967) and the collection (class[1]) focus emphasizes group or collection relationships of units of information.

Guilford views *convergent production of a semantic collection*[2] as the ability to produce meaningful collections or groups under specific conditions and restrictions (Guilford, 1967). Therefore, collection convergent relationships represent groupings or *membership* properties. Membership

[1]At the time Guilford developed SI theory, analysis—as it pertains to software systems development—did not exist. The term "class" has a different connotation in present day software engineering methodologies. In Guilford's framework, a class is a grouping of information or a collection of information. To prevent confusion with the term "class" in software engineering methodologies, the term "collection" is used in its place.

[2]This terminology continues our references to the appropriate cells of Table 1.

relationships of collections are based on aspects of the whole-part properties (Henderson-Sellers, 1997; Odell, 1994). Its intent is to represent an element of interest as a member of a collection. Membership connects a member of a collection to other members or to a whole collection or class. The following knowledge elicitation questions determine membership relationships (Yoo et al., 2004).

- Is this item a segment of a whole item?
- Is this item a member of a collection?
- What is this item a part-of?
- What components consist of this item?
- What phrases are in this whole activity?

Guilford views *divergent production of a semantic collection* as the ability to produce meaningful sub-categories of ideas appropriate to a given collection (Guilford, 1967; Meeker, 1969). Therefore, collection divergent relationships represent the components or *aggregates* of collection members. Aggregation relationships are determined for the collection members or whole-part composition (Boggs & Boggs, 2002; Booch et al., 1998; Brodie, 1981; Motschnig-Pitrik & Storey, 1995). Their intent is to represent an element's members as part-of the whole. The following list is an example of questions to help determine aggregation relationships (Yoo et al., 2004).

- Which components comprise this item?
- What materials are used to make this item?
- What is part of this item?

The collection focus is depicted in Figure 3 and represents membership relationships as looking outside the element and aggregation relationships as looking inside a collection. The premise is that membership converges to whole collection and aggregation diverges into the disparate components.

Comparison focus

The comparison[3] focus, which is equivalent to Guilford's term relation, is defined as recognized connections between items of information based upon variables or points of contact that apply to them (Guilford, 1967). Comparison focus is depicted in Figure 4.

Guilford views *convergent production of a semantic comparison* as the ability to produce an idea that conforms to specific relationship requirements (Guilford, 1967; Meeker, 1969). The ability to specify from a *general* meaning to a more specific or *specialized* meaning represents a way to represent commonality among concepts (Boggs & Boggs, 2002; Booch et al., 1998). In terms of

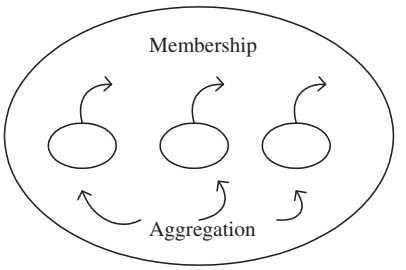

Figure 3. Collection focus.

[3]To prevent confusion with the term "relationship" in RA, the term "comparison" is used in place of Guilford's term "relation".

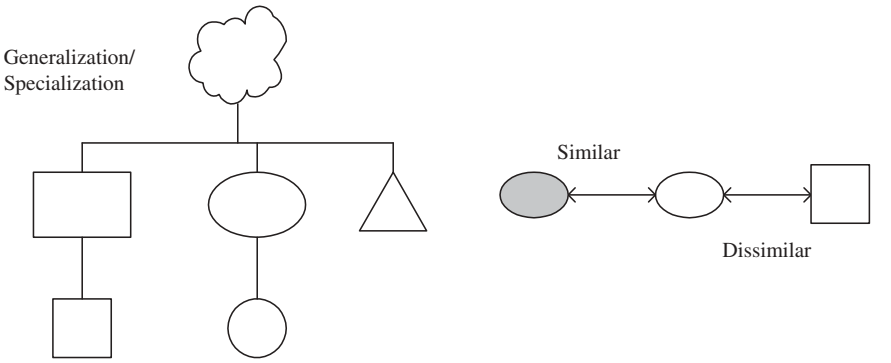

Figure 4. Comparison focus.

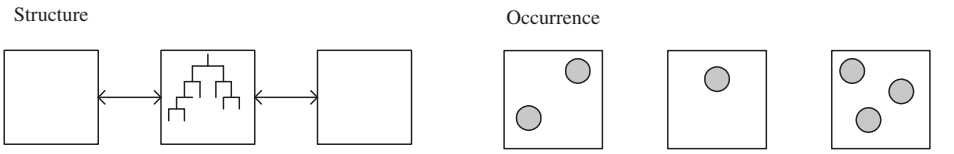

Figure 5. System focus.

analysis, generalization/specialization are the terms used to describe commonality among components and the phrases "is-a" or "a-kind-of" are used to relate objects (Booch, 1994; Rumbaugh, Blaha, Premerlani, Eddy, & Lorensen, 1991). The following knowledge elicitation questions help determine generalization/specialization relationships (Yoo et al., 2004).

- Is the item a kind of parent item?
- Does the item completely include or encompass other items?
- Is there a broader term for this item?
- Is there a narrower term for this item?

Guilford views *divergent production of a semantic comparison* as the ability to produce many relationships appropriate in meaning to a given idea (Guilford, 1967). Identifying appropriate meaning among information represents *similarity* characteristics between information components. Dissimilar characteristics also are determined as a natural result of components not being similar. Therefore, comparison divergent relationships represent both similarity and dissimilarity among elements of interest. Characteristics or attributes become criteria to determine the degree of similarity present with other elements (Belkin & Croft, 1987; Booch et al., 1998; Neelameghan & Maitra, 1978). The following questions help determine similar and dissimilar relationships (Yoo et al., 2004).

- Which other items are similar to this item?
- What serves the same purposes as this item?
- Which others items are opposite to this item?

System focus

Guilford defines a system as organized or structured items of information, a complex of interrelated parts (Guilford, 1967). Cognition of a semantic system shows comprehension of meaning derived from a system of components. System focus is depicted in Figure 5.

Guilford views *convergent production of a semantic system* as the ability to order or *structure* information into a meaningful sequence (Guilford, 1967; Meeker, 1969). Structure identifies how an item fits into the framework of a system and includes spatial perspective concepts of before, after, above, and below (Cobb & Petry, 1998; Egenhofer & Herring, 1990; Rodríguez, Egenhofer, & Rugg, 1999). The following list is an example of knowledge elicitation questions to help determine structure relationships (Yoo et al., 2004).

- What prerequisites or preconditions exist for this item?
- What follows this item for a given purpose?
- What precedes this item for a given purpose?
- Which items are close to this item?

Guilford views *divergent production of a semantic system* as the ability to organize information in various complex ideas (Guilford, 1967; Meeker, 1969). Its intent is to represent an item within the context of its appearances and uses at different places, and therefore can be viewed as *occurrence* relationships based on the temporal attributes of before, during, and after (Allen, 1983; Cobb & Petry, 1998; Egenhofer & Herring, 1990; Frank, 1998; Rodríguez et al., 1999). The following questions are examples to help determine occurrence relationships (Yoo et al., 2004).

- Where else does this item appear in the domain?
- Where else is this item used in this system and in other systems?
- What are all uses of this item?
- Where was this item used before?
- Where else is the item used now?
- Where will this item be used later?

Transformation focus

Transformations are changes or *modifications* of various kinds, of existing or known information in its attributes, meaning, role, or use (Guilford, 1967). A transformation is a matter of redefining an element. In essence, it is the ability to see potential changes of interpretations of elements and situations dependent upon a particular activity (Meeker, 1969). Therefore, it represents an element in the context of its activities.

Activities can be identified by combining SADT activity diagrams (Mylopoulos, 1998) and case relationships (Fillmore, 1968). These relationship types cover activities that involve input or output, and deal with agents and elements involved in the activities.

Convergent transformation concerns how an item can be modified focusing on the item itself and how it can change. The following knowledge elicitation questions are examples to help determine modify relationships.

- What can this item change into?
- What output results from the item's inputs?
- What resources and mechanisms are required to modify this item?
- Who can modify this item?
- Which activities result in this item being modified?

Guilford views *divergent production of a semantic transformation* as the ability to produce responses involving re-interpretations or new emphasis on some aspect of an element or situation (Guilford, 1967). Meeker extends this definition and argues that it is the ability to produce responses remote in time, remote in space, and remote in sequence (Meeker, 1969). Divergent transformations are those that *transpose* an item, reusing it in different contexts or viewing it in different ways.

Transpose relationships change an item in form or nature, or re-conceptualize it. The following list is an example of questions to help determine transpose relationships.

- How can this item be reused?
- How can this item be viewed differently?
- Can this item be used in a different context?

Transformation focus is depicted in Figure 6 with an item being modified or changed, as well as a shape being transposed into another (in this case a square being squeezed into a circle).

Implication focus

Implication emphasizes expectancies, anticipations, and predictions—the fact that one item of information leads naturally to another (Guilford, 1967). Meeker argues that cognition of semantic implication is the ability to anticipate consequences of a given situation in meaningful terms (Meeker, 1969). In essence, it is the ability to anticipate consequences of an item of interest in an organization or a social setting. Implication focus is depicted in Figure 7.

Convergent implication is dependence and control relationships both on an element and by an element, exhibiting some type of *influence* on other elements. It is how an element of information influences, controls, impacts, or (if conscious) thinks about other people or things in the social environment. The following knowledge elicitation questions are examples to help determine influence relationships (Yoo et al., 2004).

- What items or people cause this item to be created, changed, or deleted?
- What items or people have control over this item?
- What is this item dependent on?
- What is dependent on this item?

Divergent production of a semantic implication is the ability to produce many antecedents, concurrents, or consequents of given information (Guilford, 1967; Meeker, 1969). In contrast, to influence relationships, there is more freedom to produce information in divergent production of semantic implications. In context of a social setting, relationships are *extrapolated* from the given information. Divergent implication is impacts, consequences, extrapolations, rationale, deductions,

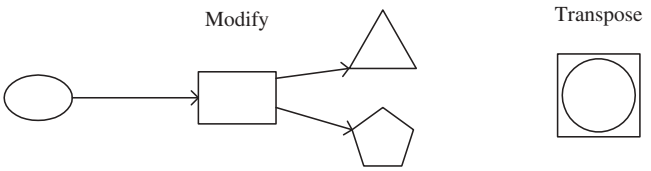

Figure 6. Transformation focus.

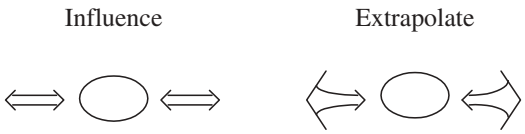

Figure 7. Implication focus.

and opinions both on an element and by an element. The following questions are examples to help determine extrapolate relationships (Yoo et al., 2004).

- Which goals, issues, and arguments involve this item?
- What are the positions and statements on the item?
- What are the comments on this item?
- What are the opinions on this item?
- What is the rationale for this decision?

Relationship analysis process (RAP)

The *relationship analysis process (RAP)* is a rigorous and systematic technique to identify the relationship structure of an application domain. A systematic process is an essential element of process improvement (Becker-Kornstaedt, 2001). A systematic approach to knowledge elicitation makes requirements gathering and problem understanding less dependent on the experience level of the process engineer (Bandinelli, 1995). A systematic approach to requirements elicitation helps to improve accuracy and provide a greater level of detail. In addition, a systematic technique helps to ensure analysis completeness and has been identified as an important criterion to develop efficient and effective information systems (Bajaj & Ram, 1999).

The RAP uses three steps in its elicitation process. The first step utilizes use-case analysis as a way to acquire system familiarity. In the second step, the process extracts detailed knowledge from information obtained earlier through use-cases by explicitly identifying the relationships between system components using a *relationship analysis template (RAT)*. The resulting relationship information is then depicted in a *relationship analysis diagram (RAD)*. The process consists of the following three process steps:

- Perform a use-case analysis to identify items of interest
- Identify the relationship structure utilizing the RATs, for each item of interest
- Graphically depict the relationships utilizing the RAD, for each item of interest

Use-cases are the feeder into the RAP. The RAP explicitly identifies the relationship structure of an application domain and provides more information than use-case analysis alone and helps in the creation of class diagrams.

Relationship analysis template (RAT)

Each item of interest can be described in terms of relationships based on the RAM. Each relationship focus has its own RAT for each item of interest, outlined in Table 3 and illustrated in Table 4, that can be used to document the relationships discovered during the elicitation process. Each RAT is used by the analyst to record the analysis results and help to track analysis decisions (Booch et al., 1998).

It is important to note that the template provides a way for analysts to communicate and document the process of discovering relationships. To this end, the template provides cells that contain brainstorming questions to help elicit and identify specific relationships. In particular, the template contains a cell that captures domain independent generic questions. These generic questions can be used to help create more domain dependent specific questions, which are also captured in the template. The results recorded in the template systematically document both the process and the relationships, which as we shall show in the experiment section, afford lesser-experienced ("low experience") analysts the ability to create class diagrams of similar quality to those of high experienced analysts.

Table 3. Generic relationship analysis template.

Item of interest	Each item of interest should have a unique name suggesting its purpose.
Description	Description and purpose of the item of interest.
Focus	Unit, Collection, Comparison, System, Transformation, or Implication. *{Each template takes one focus.}*
Convergent relationship	Specification, Membership, Generalization/Specialization, Structure, Modify, or Influence. *{Corresponds to the template focus.}*
Generic question(s) (optional)	The generic questions provided in the RAM section above for the corresponding convergent relationships could be contained here. This helps generate the specific question(s) to ask for a particular application domain.
Specific question(s)	This cell contains the exact question(s) to ask to determine appropriate convergent relationships and is/are tailored to the particular application domain.
Results	The results generated from the specific question(s) are recorded in this cell.
Divergent relationship	Elaboration, Aggregation, Similar/Dissimilar, Occurrence, Transpose, or Extrapolate. *{Corresponds to the template focus.}*
Generic question(s) (optional)	The generic questions provided in the RAM section above for the corresponding divergent relationships, could be contained in this cell.
Specific question(s)	This cell contains the exact question(s) to ask to determine divergent relationship and is/are tailored to the particular application domain.
Results	The results generated from the specific question(s) are recorded in this cell.

Table 4. Order relationship analysis template.

Item of interest	Order
Description	An order is a list of one or more products that a customer or vendor desires to purchase.
Focus	Unit
Convergent relationship	Specification
Specific question(s)	Does an "order" have an explanation?
Results	Customers place orders. Vendors place orders. Sales Representatives take orders. Orders consist of products. Orders have an order number. Orders are packaged by fulfillment clerks. Orders are shipped by shipping clerks. Orders are filled by fulfillment clerks. Orders are processed. Orders written on an order form.
Divergent relationship	Elaboration
Specific question(s)	How does a customer select a product to purchase? How does a sales representative fulfill an order? How is an order shipped?
Results	A customer selects items from a drop down list box and a purchase order request form is generated for an order. A sales representative views an order request form, fetches and packages the order items from the warehouse and updates the product database. The shipping department receives the packaged items of an order and delivers the package.

Relationship analysis diagram (RAD)

In addition to capturing the relationship structure in the aforementioned templates, it is also possible to consolidate and present the information from a set of RATs in a graphical representation—see Figure 8.

Both the RAT and RAD represent a new way to document the relationship structure of a domain, which will greatly assist in developing class diagrams. The relationships of each item of interest are documented in six templates and one diagram. The collection of all RAT and RAD components comprise the relationship structure of the problem domain. The RAD provides an information rich graphic, while more details can be accessed via the information recorded in the templates. Utilizing the collection of RADs of a problem domain, which depict the discovered relationships, should enhance the generation of class diagrams.

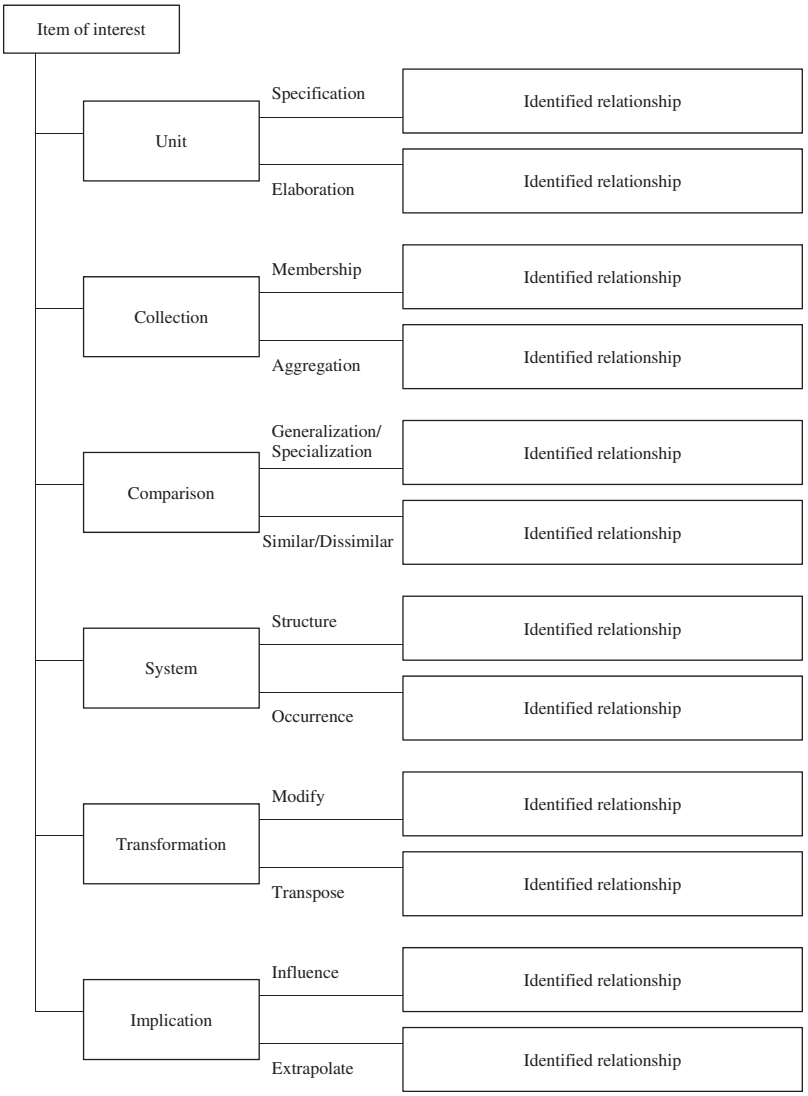

Figure 8. Relationship analysis diagram (RAD).

Concrete example to explain the concepts

Consider a basic inventory control system in which a store is set up to fill customer and vendor orders. To fill these orders, products are stored and maintained in a warehouse. These products are not given directly to the customer; they are packaged first and then shipped to the customer. Once a customer order is placed, the order fulfillment employee locates the product in the warehouse; places requisite number of the product in a proper package; updates the inventory list to reflect the fact that a particular product item was taken from inventory. Once all the product items of the order are packaged, the order-processing department is notified that the order has been filled. Some of the use cases associated with this example are, place order, fill order, and ship order. Use case analysis can be utilized to outline the flow of events of each of these use cases.

Beyond use cases, this example focuses on demonstrating the RAP regarding relationship discovery and documentation.

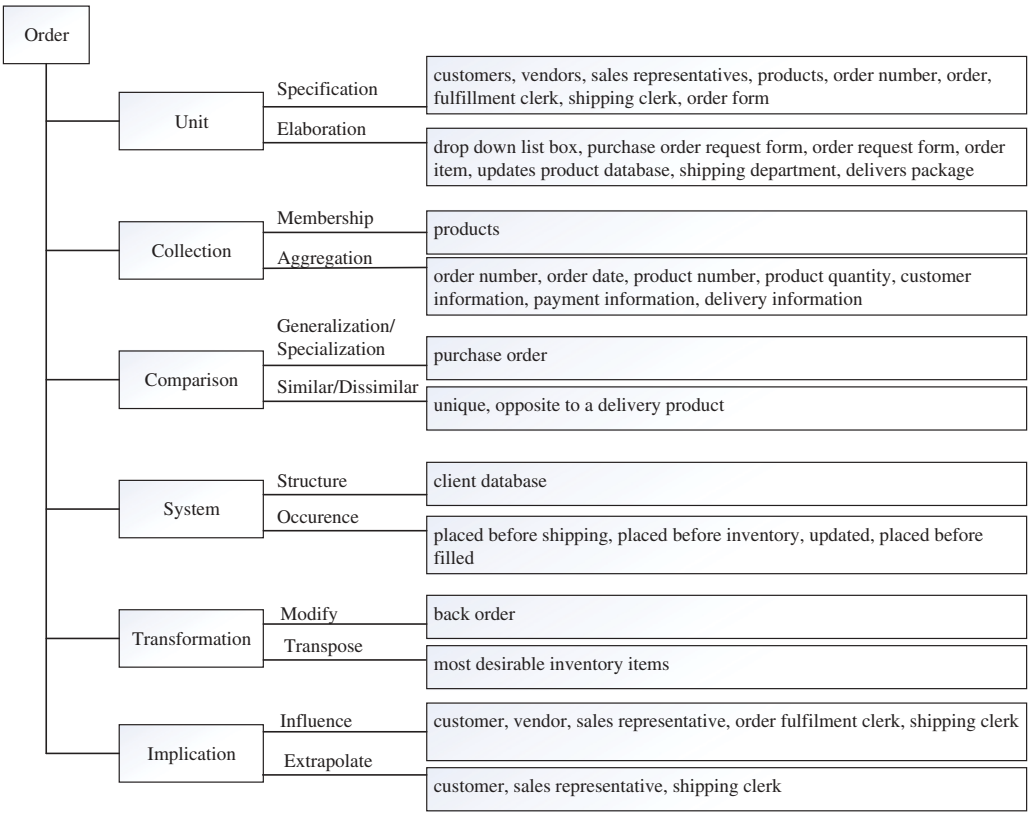

Figure 9. Order relationship analysis diagram.

Normally a RAT would be created for each of the other focus components namely, *class, relation, system, transformation*, and *implication*, however for brevity only the unit focus component is provided. The next step would be to transcribe the elicited information from the six RATs for the item of interest, Order, into the RAD depicted in Figure 9. Note that the RAD contains the results of all of the RATs.

The set of RADs constitute the complete relationship structure of all the items of interest for the problem domain. Our proposed RATs and the RADs will be useful in the creation of class diagrams.

Experiment

While much IT project and systems analysis research involves case studies, a number of solid studies based on theory-based models have been conducted using laboratory experiments (Keil, Tan, Wei, & Saarinen, 2000; Lee, Keil, Smith, & Sarkar, 2016; Mangalaraj, Nerur, Mahapatra, & Price, 2014; Marakas & Elam, 1998; Shaft & Vessey, 2006). An experiment was designed to assess whether the RAP is an effective technique to explicitly identify the relationship structure of a problem. While not an integral aspect of this study, the experiment was conducted using groups of analysts since software product development is generally realized as a team effort. Teams have been shown to generate better solutions than individual solutions (Baroudi, Olson, & Ives, 1986; Connolly, Jessup, & Valacich, 1990; DeSanctis & Gallupe, 1987; Gallupe, DeSanctis, & Dickson, 1988; Goguen & Linde, 1993; Kontonya & Sommerville, 1996; Rumbaugh, 1994; Sommerville, 2001).

Furthermore, the experiment was designed to assess whether a rigorous and systematic process helps low experienced groups achieve a similar level of quality as high experienced groups (Amento, Terveen, & Hill, 2000; Bandinelli, 1995; Becker-Kornstaedt, 2001; Carter, Cushing, Sabers, Stein, & Berliner, 1988; Hillerbrand & Claiborn, 1990; Saleem, 1996; Schenk, Vitalari, & Davis, 1998; Spence & Brucks, 1997).

Method

The method of the experiment is a 2 × 2 factorial design. The two independent variables are *analyst experience* and *analysis tool*, depicted in Table 5.

Therefore, the four conditions in this experiment are:

- Use-case only, low experience
- Use-case & RA, low experience
- Use-case only, high experience
- Use-case & RA, high experience

The use-case analysis tool represents the control group category and the treatment group represents the use-case & RA category. Thus, it is possible to measure the effects of the RA technique.

Hypotheses: Analysis quality

(1) The class diagram generated by performing a RA will be more accurate and complete than by groups using use-case analysis alone.
(2) The groups with high experience will generate more accurate and complete class diagrams than low experience groups using use-case analysis or the RAP.
(3) The low experience groups utilizing Use case & RA will generate more accurate and complete class diagrams than high experience groups using use-case analysis alone.

Analysis quality is measured by the quality of the class diagram generated by the group subjects that performed the same task utilizing different analysis tools. Schenk's research findings indicate that novices exhibit less detail in problem-solving tasks than do experts, resulting in lower quality (Schenk et al., 1998). In addition, a study of novice and expert programmers found that novices tend to employ easy methods for their tasks (Vessey, 1985). The results from these previous studies indicate that novices were unable to formulate an overall structure to the task. However, these experiments did not include a systematic process to follow. We speculate that low experience groups utilizing the systematic well-structured process (RAP, for example) will produce documents of equal quality as high experience groups. This speculation is supported by Spence and Brucks, who provide convincing empirical evidence that the benefits of expertise are less pronounced when analyzing and solving a problem with a well-defined technique (Spence & Brucks, 1997). In addition, another study concluded that experts, compared to novices make qualitatively different inferences in their reasoning, focus on different problem features, and thereby reason to different conclusions (Hillerbrand & Claiborn, 1990). We speculate that the very nature of a well-defined process, namely RA, will permit novices to reach the same conclusions as experts.

Table 5. Factorial design experiment.

		Analyst experience	
2 × 2 factorial design		Low	High
Analysis tool	Use-case only		
	Use-case & RA		

Experience has been used extensively in experiments to determine its effect on the learning process (Amento et al., 2000; Carter et al., 1988; Hillerbrand & Claiborn, 1990; Saleem, 1996; Schenk et al., 1998; Spence & Brucks, 1997). To determine experience level, subjects completed a pre-experiment questionnaire (Appendix B) that identified academic background, software background, and professional work experience relating to software system analysis and design. The authors divided the subjects into low and high experience based upon the scores from the pre-experiment questionnaire. The criteria assigned a range of scores from 26 to 203 points. Those below 100 points were considered low experience subjects and were then randomly selected and placed in a team consisting of three low experience individuals. Similarly, high experience subjects were those that scored above 100 points and were then randomly selected and placed in a team consisting of three high experience individuals.

Table 6 provides the means and standard deviation calculations of the subjects' experience level score for each of the conditions. In addition, the sample size is also provided.

Subjects

The subjects in the experiment consisted of both undergraduate and graduate students enrolled in the College of Computing Sciences at the New Jersey Institute of Technology.

Procedures

The main experiment took place during the semester and lasted one week, whereby the first day included a training session. One week prior to the experiment, the subjects completed a pre-experiment questionnaire (Appendix B) to determine analysis experience level. Each group, consisting of three subjects, was placed in one of the four conditions. Each group, from the four different group types, performed the same task (Appendix A). This permits the pure effect of the treatments to be isolated because the difference in tasks is controlled. This will increase the internal validity of the research (Straub, 1989).

The experiment was conducted at the end of the semester so all subjects had some level of modeling experience. All subjects were taught how to develop use-case analysis diagrams and generate class diagrams prior to the experiment. The treatment groups were trained in RA. To eliminate any training effect, the control groups were provided an equivalent enrichment topic, namely entity relationship (E/R) analysis. After the training, all groups were provided the same task, (Appendix A), to solve with their team members. All groups had one hour to create the use-case analysis diagram. This afforded all groups adequate time to become familiar with the problem domain. At the conclusion of the session, all groups were provided with an expert-generated use-case analysis diagram to the problem statement. All groups used this as a basis to complete the remaining experimental steps. The control groups generated class diagrams after use-case analysis. The treatment groups performed RA and then generated class diagrams. This allows the effect of RA to be measured. All groups had one week to complete the task and submit all analysis documents and class diagrams.

Table 6. Subject experience score mean, standard deviation calculations, sample size.

		Analyst experience	
2 × 2 factorial design		Low	High
Analysis tool	Use-case only	Mean, 62.13 SD ,14.17 Sample, 15	Mean, 127.36 SD, 12.87 Sample, 13
	Use-case & RA	Mean, 58.98 SD, 15.14 Sample, 16	Mean, 122.33 SD, 11.59 Sample, 13

Measures

Two professional software engineers were deemed as experts. Each rated the quality of each group's generated class diagram. Expert judges have been used in many studies to evaluate quality of system design and decision-making (Ocker, Fjermestad, Hiltz, & Johnson, 1998; Shaft & Vessey, 1998).

The expert judges had their own training session to ensure that their evaluations were compatible. The expert judges used a 10-point scale whereby a 10 represents a perfect score. In order to eliminate potential bias of individual experts, each expert judge evaluated each group's class diagrams independently and the average evaluation was computed and used as the final score. Whenever the difference in their individual evaluations exceeded 1 point, (an acceptable 10% threshold), the two experts met to resolve the issues and cooperatively assigned a final score.

Data analysis

This section describes the evaluation of the data collected with respect to the analysis quality of the class diagram created by each group in each of the four conditions. The results were compared against the hypotheses described earlier in this section.

To determine if the data is normally distributed, SAS was used to run the Kolmogorov-Smirnov normality test and the results ($p< 0.01$) indicate that the data is not normally distributed. Therefore, to determine the interaction effect among the two experience level groups, we divide the data into two groups, namely low and high, and apply the following non-parametric methods (available in SAS) to both groups. The FREQ procedure compares the two groups and tests for independence between two variables. The asymptotic Wilcoxon rank sum test is obtained by using SCORES = RANK in the TABLES. This method alters the original 10 point scale range by creating mean ranks and compares the difference between UC and UC&RA in low and high experience groups separately. If the "differences" are significantly different, then there is an interaction effect. For example, suppose the difference between UC and UC&RA in the low experience groups is 4 and the difference between UC and UC&RA in high experience groups is 1. Then, the effect of UC/UC&RA is bigger in low experience groups than high experience groups, which means that the two variables (Analysis Tool: UC/UC&RA and Experience: Low/High) interact with each other. Table 7 provides the analysis quality score mean, standard deviation calculations, and sample size for each of the conditions.

Main effect 1

The analysis tool independent variable main effect shows a significant effect at alpha = 0.05 level (p = 0.0001) and mean ranks of 38.36 and 19.30 (Table 8) for UC&RA and UC respectively.

The results shown in Table 8 support H1 (*The class diagram generated by performing a RAP will be more accurate and complete than by groups using use-case analysis alone.*) and indicate that this variable is statistically significant at alpha = 0.05 level. The mean ranks of those using RA (38.36) is much better than not using RA (19.30) and indicate that RA significantly improves analysis quality.

Table 7. Quality score mean, standard deviation calculations, sample size.

		Analyst experience	
2 × 2 factorial design		Low	High
Analysis tool	Use-case only	Mean, 5.87	Mean, 6.81
		SD, 2.53	SD, 0.97
		Sample, 15	Sample, 13
	Use-case & RA	Mean, 7.78	Mean, 7.96
		SD, 0.55	SD, 0.78
		Sample, 16	Sample, 13

Table 8. Quality score main effect 1 (analysis tool).

Independent variable	*N*	Mean ranks	Significance
Use-case only	28	19.303	0.0001
Use-case and relationship analysis	29	38.362	

Main effect 2

The analyst experience level independent variable main effect does not show a significant effect at alpha = 0.05 level (p=0.5892) and mean ranks of 30.27 and 27.94 (Table 9) for high and low experience respectively.

The results shown in Table 9 do not support H2 (*The groups with high experience will generate more accurate and complete class diagrams than low experience groups using use-case analysis or the RAP.*) and indicate that this variable is not statistically significant at alpha=0.05 level.

Interaction effect

An interaction effect is present when the main effect of independent variable analysis tool changes at different levels to the main effect of independent variable analyst experience. The data (Table 10) indicates at alpha = 0.05 level (p=0.0023) that the mean differences percentage increase for low experience analysts groups using analysis tools is 32.54% and high experience analysts groups using analysis tools is 16.89%.

The results shown in Table 7 support H3 (*The low experience groups utilizing the RAP will generate more accurate and complete class diagrams than high experience groups using use-case analysis alone.*) and indicate that this variable is statistically significant at the alpha=0.05 level. Table 7 indicates the mean score of low experience groups using RA (7.78) is higher than the mean score of high experience groups not using RA (6.81). Figure 10 confirms that there is an interaction effect between Analyst Experience and Analysis Tool.

Debriefing session with subjects

At the conclusion of the experiment, a debriefing session was performed and the following comments (paraphrased) from the experimental subjects deserve mention.

- I am not able to draw class diagrams by looking at use-case diagrams. I would have preferred to be in a group that did the RA.
- It is difficult to create class diagrams and I agree that something is missing to help do this. The technique was helpful, but very long.
- After our group did one, the process was pretty straightforward.
- I liked the way the final class diagram was developed by the previous pieces.

Table 9. Quality score main effect 2 (analyst experience level).

Independent variable	*N*	Mean ranks	Significance
High experience	26	30.269	0.5892
Low experience	31	27.935	

Table 10. Interaction effect.

Interaction effect	*N*	Mean differences	% increases	Significance
High experience analysts using analysis tools	26	1.15	16.89%	0.0023
Low experience analysts using analysis tools	31	1.19	32.54%	

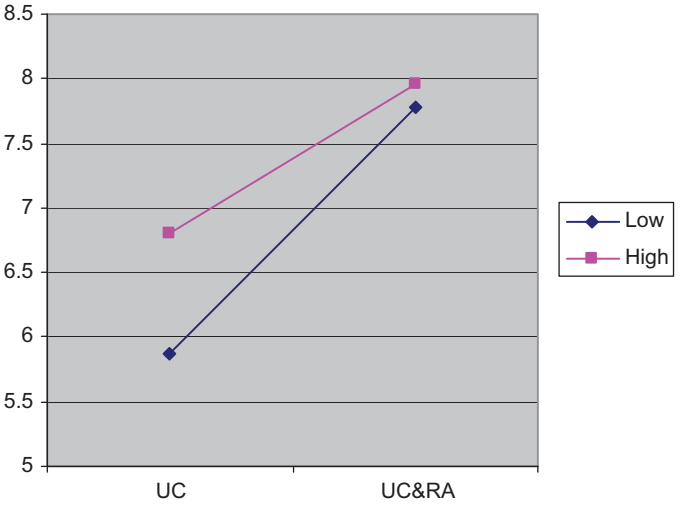

Figure 10. Quality score for groups.

- The technique was helpful in creating the class diagram, but I saw the solution about half-way through.
- It would have been nice to have a computer tool to help generate the templates and the diagrams instead of doing it on paper.

From the discussions during the debriefing sessions two conclusions can be drawn. The first is that the RA process does aid in class diagram generation. In addition, the subjects believed that the templates were easy to use to elicit and document the relationships. Second, however, most subjects expressed that the time needed to perform the analysis was too long and required too much paper.

Interviews with practitioners

To help ascertain whether RA is a viable technique in the software industry field, three practicing, professional software engineers from three different software companies were interviewed during January–February 2016. Subject 1 has 20+ years of practical experience developing both embedded systems firmware and software application software. Subject 2 has 5+ years of experience working as a business analyst designing, documenting, and implementing application software components that are incorporated into a larger information system. Subject 3 has two years of experience creating client server applications using Microsoft SQL server and Microsoft Access as the back-end database layer and Microsoft Visual Basic as the front-end application layer.

To make best use of subjects' time and expertise, an open-ended interview was conducted using five open-ended questions. The results are paraphrased and summarized below.

- Question 1: Explain the concept of relationship analysis and its application to the development of class diagrams.

The researcher presented the concept of RA by providing an overview of the topic and suggested that the technique can be used to develop class diagrams. These discussions involved the explanation of the RAT and the RAD.

Subject 1: The development of software applications always involves a backend database, and as such this subject creates a database diagram first prior to writing any software code. The subject

further noted that relationships with which he or she is familiar are those displayed on the database diagram. The relationships depicted in the RAT seem to be very detailed and at first glance somewhat complicated. In contrast, the RAD provides a good overview diagram and appears easy to read and understand.

Subject 2: Documentation is a very important element to the process that Subject2 follows. There are many different individuals working on the same project and the better it is documented, the fewer questions occur during implementation. Both the RAT and RAD may be ideal ways of having the implementation team develop code as outlined by the business analyst in the specification.

Subject 3: The subject simply mentioned that it looks good and immediately wanted to see an example. The researcher noted that Subject 3 worked in the implementation capacity and did not seem to have a role in design.

- Question 2: Ask how the subject currently develops and documents software prior to implementation.

Subject 1: A separate team decides what projects the company will pursue and creates a limited system request that is distributed to the software department for evaluation. The software department assigns a software project lead who becomes responsible to more concretely elicit and document the functional requirements of the proposed system. In further conversation it is important to note that the subject believes that the system request provided generally lacks important information and much is left to the software department to determine. The subject uses an in-house created template, similar to a use case analysis template. This template identifies system functionality down into a collection of data to be processed from a user's perspective. A data model is then created that groups data elements together. Subsequent steps when developing code in C++ simply involve the creation of classes based on each of the entities depicted on the database diagram. Class methods are written to access the data variables located in each class. The researcher believes that this process is primarily driven by the highly experienced engineers of the software department.

Subject 2: Uses a combination of agile and rational unified processes. Daily scrum meetings set the team's objectives for the day. Use case templates are employed to determine functional requirements with special attention paid to the flow of data. Subject 2 reviews identified data components and develops methods detailing data inputs and outputs. The subject then groups common methods and data together in a class. This information is thoroughly documented using a series of in-house process steps. Once the documentation is completed it is passed to the implementation team. Prior to implementation, the affected individuals involved in the project meet and discuss the project to hammer out unclear details.

Subject 3: The documentation has already been completed and Subject 3 implements the classes, methods, and data variables that have already been documented. In further discussions, the researcher believes that the process followed by Subject 3 is similar to that of Subject 1. Namely, an experienced software engineer architects the software system and junior less experienced software engineers help to implement the software system in a modular fashion.

- Question 3: Discuss results of laboratory experiment.

All three subjects were equally impressed that both low and high experienced groups achieved a similar level of quality of design artifacts using the RA technique.

Subject 1: Mentioned that this might be a useful tool to train entry-level software engineers.

Subject 2: Elaborated that it is the responsibility of the specification writer to design the class diagrams and that RA might help make that easier.

Subject 3: Mentioned that perhaps this tool will allow him or her to become a software architect instead of just a "code monkey."

- Question 4: Demonstrate a brief example.

Both Subject 1 and Subject 2 are familiar with use case analysis so they provided several use cases from a previous project during this portion of the interview. Both subjects identified an item of interest that they wanted to know more about and together with one of the researchers performed RA. As described in the RAM section of this article, the relationships of the item of interest were elicited and documented using the RAT and elicitation questions were already preprinted on the RAT. During the demonstration, subjects recorded the answers to the elicitation question in the RAT. In addition, the answers were transcribed onto the RAD. Both subjects mentioned that the process was very easy but was very time-consuming and requires lots of paper.

Subject 1: Asked if RA must be performed for every item of interest identified in the collection of use cases. The subject suggested a modified form of RA to speed up the process. The proposal was to only apply RA to items of interest that need more elaboration. Since the goal is to create a class diagram, then let us only use the tool if we need more information on a "thing" so as to properly classify the item as a class, method, data element, or relationship.

Subject 2: Mentioned that a benefit of RA is that it can be deployed as a validation technique for the actual class diagram to be created. The subject went on to describe that documents created undergo a thorough scrutiny process and perhaps RA could be used to justify the correctness of the documented solution.

Subject 3: The researcher demonstrated RA to Subject 3 using the laboratory experiment example. Subject 3 believes he or she would be able to use this tool and create a system software architecture broken down into classes.

- Question 5: Ask if the technique would add-value to their process of software development.

Subject 1: This subject did not think that the tool was needed to design software but thought that maybe it would be a useful tool for less experienced designers. Potentially more junior level software engineers could become involved in the software architecture creation and not just experienced software engineers.

Subject 2: Yes! Subject 2 was a proponent of RA and does plan on incorporating it into the process of software design. The subject mentioned again personally believing it to be a very useful validation technique that can be used concurrently during the software design process.

Subject 3: This subject was not sure how his or her company actually designs software but thinks that with explicit tools like RA, he or she would be able to become a software architect.

Conclusions from the subjects' comments

Both experimental subjects and professional software engineers mentioned that the RA process is very lengthy and requires a lot of paper. Ovaska and Stapleton also investigated the requirements understanding process of information systems and similar results indicate that working professionals believe that too much time is spent during the requirements capturing phase (Ovaska & Stapleton, 2010). Therefore, a way must be found to reduce the time needed to perform the analysis without reducing the effectiveness of the technique and are common goals of process improvement. Shankararaman et al. also support process improvement through the reduction of time without sacrificing effectiveness (Shankararaman, Gottipati, & Duran, 2012). The researchers also believe that implementing the technique using paper needs to be improved. To these ends, the researchers plan to develop a computerized version to create the RATs and RADs. This computerized version will provide an easy, and hopefully streamlined automated process, thereby addressing both concerns, namely too much paper and too much time.

In addition, one subject of the field study suggested not performing a complete RA on every item of interest in the use-case diagrams. Instead, perform RA on a reduced set of items. Since the

fundamental goal is to create a complete class diagram, a minimal RA may only be needed. The researchers like this idea and will consider positioning RA to allow for both a complete and reduced analysis once computerized RAT and RAD versions are created.

Conclusions

RA provides the software community a usable technique that improves an analyst's effectiveness in relationship discovery and documentation. Based on the experimental results, RA does provide a fuller and richer systems analysis, resulting in improved quality of class diagrams, and that RA enables analysts of varying experience levels to achieve a similar level of quality of class diagrams. RA significantly enhances the systems analyst's effectiveness, especially in the area of relationship discovery and documentation, resulting in improved analysis and design artifacts.

In a post-experiment debriefing session many subjects mentioned that current software object-oriented analysis techniques provide little assistance in identifying classes and how they interrelate. Much of the system analysis process and class diagram creation is delegated to highly experienced system analysts. RA attempts to level the playing field among analysts of varying experience by providing a process to explicitly identify and document classes and their relationships. RA is poised to serve as the missing link, which can supplement the everyday techniques for documenting and creating class diagrams.

Although a methodology independent technique, RA can be positioned seamlessly between the use-case analysis and class diagram generation steps of the widely used object-oriented paradigm. In future work, the researchers plan to conduct additional field studies using a computerized version of RA. Showing that RA improves the development process, is compatible with current approaches, and that practitioners are satisfied with and accept it will be the first stage towards RA's inclusion into the object-oriented paradigm and toolkits used by software engineers.

Future research: Extending the RAT with risk and other broad metadata

A possible extension would be to capture metadata about a relationship that would be useful for determining whether to include the relationship in the final system design. Aspects such as the cost of gathering and importing/inputting the relationship data; the risk that this data may be biased, incorrect or incomplete; conditionality (e.g., that a relationship may be temporary, or only valid under certain circumstances or locations); that it may be open to cultural interpretation (organizational or regional; Keil et al., 2000) all may influence whether or in what way the relationship is implemented. For such a full understanding of relationships important to a system, it may be useful for conducting the RAP with experts from different levels and regional locations of an organization, and engaging a cross-cultural team to resolve any important differences of opinion among these experts regarding the true system relationship structure.

Acknowledgments

We thank the JITCAR reviewers and the Editor-in-Chief Dr. Shailendra Palvia for their detailed comments and suggestions to improve this article during the review process. James Bell provided input on the introductory friend-request example.

In loving memory of Shadow.

Funding

We gratefully appreciate funding support for this research by the United Parcel Service, the New Jersey Institute of Technology, the National Science Foundation under grants IIS-0135531, DUE-0226075, and DUE-0434581, and the Institute for Museum and Library Services under grant LG-02-04-0002-04.

Notes on contributors

Joseph T. Catanio is currently an Associate Professor of Management Information Systems at the John L. Grove College of Business, Shippensburg University of Pennsylvania. He earned his Ph.D. in Information Systems from the New Jersey Institute of Technology (NJIT) in Newark, New Jersey. His research interests include business systems analysis and design, project management, and management information systems. His research has been published in journals including, *International Journal of Information Technology Project Management, Journal of Advances in Information Technology, Journal of Human Resources Education, Requirements Engineering Journal, Journal of Technical Writing & Communications, Journal of Digital Information*. In addition, he has worked in the private sector as a software engineer for 16+ years.

Michael Bieber is Professor in the Information Systems Department at NJIT's Ying Wu College of Computing Sciences. He is conducting research in several related areas: learning sciences, cyber-learning, lightweight systems integration, relationship analysis (as part of the software engineering process), automatic link and meta-information generation, and hypermedia. He has received several NSF grants.

References

Allen, J. (1983). Maintaining knowledge about temporal intervals. *Communication of the ACM*, *26*(11), 832–843. doi:10.1145/182.358434

Amento, B., Terveen, L., & Hill, W. (2000). Does "authority" mean quality? Predicting expert quality ratings of web documents. In Association for Computing Machinery (Ed.), *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '00)* (pp. 296–303). New York, NY: Editor. doi:10.1145/345508.345603

Bajaj, A., & Ram, S. (1999). Evaluating completeness of Conceptual Business Process Models (CBPMs): A metric based on case studies. *Journal of Information Technology Case and Application Research*, *1*(4), 5–30. doi:10.1080/15228053.1999.10855945

Bandinelli, S. (1995). Modeling and improving an industrial software process. *IEEE Transactions on Software Engineering*, *21*(5), 440–454. doi:10.1109/32.387473

Baroudi, J., Olson, M., & Ives, B. (1986). An empirical study of the impact of user involvement on system usage and information satisfaction. *Communications of the ACM*, *29*(3), 232–238. doi:10.1145/5666.5669

Becker-Kornstaedt, U. (2001). *Towards systematic knowledge elicitation for descriptive software process modeling*. Proceedings of PROFES, pp. 118.

Belkin, N., & Croft, W. (1987). Retrieval techniques. *Annual Review of Information Science and Technology (ARIST)*, *22*(4), 109–131.

Beraha, M., & Su, J. (1999). Support for modeling relationships in object-oriented databases. *Data and Knowledge Engineering*, *29*(3), 227–278. doi:10.1016/S0169-023X(99)80001-9

Bieber, M. (1998). *Hypertext and web engineering*. Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia, ACM Press, pp. 277–278.

Bieber, M., & Yoo, J. (1999). Hypermedia: A design philosophy. *ACM Computing Surveys*, *31*(4es). doi:10.1145/345966.346028

Boggs, W., & Boggs, M. (2002). *UML with rational rose*. Alameda, CA: Sybex.

Booch, G. (1994). *Object-oriented analysis and design* (2nd ed.). Redwood City, CA: Benjamin/Cummings.

Booch, G., Jacobson, I., & Rumbaugh, J. (1998). *The unified modeling language users guide*. Reading, MA: Addison Wesley.

Borgida, A., Mylopoulos, J., & Wong, H. (1984). Generalization/specialization as a basis for software specification. In M. L. Brodie, J. Mylopoulos, & J. W. Schmidt (Eds.), *On conceptual modeling: Perspectives from artificial intelligence, databases, and programming languages* (pp. 87–117). New York, NY: Springer-Verlag.

Brachman, R. (1983). What IS-A is and isn't: An analysis of taxonomic links in semantic networks. *IEEE Computer*, *16*(10), 30–36. doi:10.1109/MC.1983.1654194

Brodie, M. (1981). Association: A database abstraction for semantic modeling. In P. P. Chen (Ed.), *Entity-relationship approach to information modeling and analysis* (pp. 583–608). Amsterdam, the Netherlands: Elsevier.

Carter, K., Cushing, K., Sabers, D., Stein, P., & Berliner, D. (1988). Expert-novice differences in perceiving and processing visual classroom information. *Journal of Teacher Education*, *39*(3), 25–31. doi:10.1177/002248718803900306

Catanio, J. (2004). Relationship Analysis: Improving The Systems Analysis Process (Ph.D. Dissertation). New Jersey Institute of Technology.

Catanio, J., & Bieber, M. (2005). Relationship analysis: A technique to enhance systems analysis for web development. In W. Suh (Ed.), *Web engineering: Principles and techniques* (pp. 97–113). Hershey, PA: Idea Group.

Chun, M., Sohn, K., Arling, P., & Granados, N. (2009). Applying systems thinking to management systems: The case of Pratt-Whitney Rocketdyne. *Journal of Information Technology Case and Application Research*, *11*(3), 43–67. doi:10.1080/15228053.2009.10856164

Cobb, M., & Petry, F. (1998). Modeling spatial relationships within a fuzzy framework. *Journal of the American Society for Information Science*, *49*(3), 253–266. doi:10.1002/(ISSN)1097-4571

Connolly, T., Jessup, L., & Valacich, J. (1990). Effects of anonymity and evaluation tone on idea generation in computer-mediated groups. *Management Science*, *36*(6), 305–319. doi:10.1287/mnsc.36.6.689

DeSanctis, G., & Gallupe, B. (1987). A foundation for the study of group decision support systems. *Management Science*, *33*(5), 589–609. doi:10.1287/mnsc.33.5.589

Egenhofer, M., & Herring, J. (1990). Categorizing binary topological relations between regions, lines, and points in geographic databases. *Technical Report*, Department of Surveying Engineering, University of Maine.

Faulk, S. (2000). Software requirements: A tutorial. In *Software requirements engineering* (2nd ed., pp. 158–179). Los Alamitos, California: IEEE.

Fillmore, C. J. (1968). The case for case. In E. Bach, & R. T. Harms (Eds.), *Universals in linguistic theory*. New York, NY: Holt, Rinehart & Winston.

Frank, A. (1998). Different types of times in GIS. In M. Egenhofer, & R. Golledge (Eds.), *Spatial and temporal reasoning in geographic information systems* (pp. 41–62). New York, NY: Oxford University Press.

Gallupe, B., DeSanctis, G., & Dickson, G. (1988). Computer-based support for group problem-finding: An experimental investigation. *MIS Quarterly*, *12*(2), 277–296. doi:10.2307/248853

Goguen, J., & Linde, C. (1993). *Techniques for requirements elicitation*. Proceedings from the International Symposium on Requirements Engineering, pp. 152–164.

Guilford, J. P. (1950). Creativity. *American Psychologist*, *5*, 444–454. doi:10.1037/h0063487

Guilford, J. P. (1956). The structure of intellect. *Psychological Bulletin*, *53*(4), 267–293. doi:10.1037/h0040755

Guilford, J. P. (1967). *The nature of human intelligence*. New York, NY: McGraw-Hill.

Henderson-Sellers, B. (1997). OPEN relationships—Compositions and containments. *Journal of Object-Oriented Programming*, *10*(7), 51–72.

Hillerbrand, E., & Claiborn, C. (1990). Examining reasoning skill differences between expert and novice counselors. *Journal of Counseling & Development*, *68*(6), 684–691. doi:10.1002/j.1556-6676.1990.tb01437.x

Keil, M., Tan, B., Wei, K. K., & Saarinen, T. (2000). A cross-cultural study on escalation of commitment behavior in software projects. *MIS Quarterly*, *24*(2), 299–325. doi:10.2307/3250940

Kotonya, G., & Sommerville, I. (1996). Requirements engineering with viewpoints. *Software Engineering Journal*, *11*(1), 5–18.

Lee, H. K., Keil, M., Smith, H. J., & Sarkar, S. (2016). The roles of mood and conscientiousness in reporting of self-committed errors on IT projects. *Information Systems Journal*. Advance online publication. doi:10.1111/isj.12111

Mangalaraj, G., Nerur, S., Mahapatra, R., & Price, K. (2014). Distributed cognition in software design: An experimental investigation of the role of design patterns and collaboration. *MIS Quarterly*, *38*(1), 249–274.

Marakas, G., & Elam, J. (1998). Semantic structuring in analyst acquisition and representation of facts in requirements analysis. *Information Systems Research*, *9*(1), 1–37. doi:10.1287/isre.9.1.37

Martin, J., & Odell, J. (1995). *Object-oriented methods: A foundation*. Englewood Cliffs, New Jersey: Prentice Hall.

Meeker, M. (1969). *The structure of intellect: Its interpretations and uses*. Columbus, Ohio: Merrill Publishing.

Motschnig-Pitrik, R., & Storey, V. (1995). Modelling of set membership: The notion and the issues. *Data & Knowledge Engineering*, *16*(2), 147–185. doi:10.1016/0169-023X(95)00014-J

Mylopoulos, J. (1998). Information modeling in the time of the revolution. *Information Systems*, *23*(3–4), 127–155. doi:10.1016/S0306-4379(98)00005-2

Neelameghan, A., & Maitra, R. (1978). Non-hierarchical associative relationships among concepts: Identification and typology (Part A of FID/CR report No. 18). Bangalore, India: Documentation Research and Training Centre, Indian Statistical Institute.

Ocker, R., Fjermestad, J., Hiltz, S. R., & Johnson, K. (1998). Effects of four modes of group communication on the outcomes of software requirements determination. *Journal of Management Information Systems*, *15*(1), 99–118. doi:10.1080/07421222.1998.11518198

Odell, J. (1994). Six different kinds of composition. *Journal of Object-Oriented Programming*, *5*(8), 10–15.

Ovaska, P., & Stapleton, L. (2010). Requirements engineering during complex ISD: Case study of an international ICT company. *Journal of Information Technology Case and Application Research*, *12*(2), 36–59. doi:10.1080/15228053.2010.10856182

Rodríguez, A., Egenhofer, M., & Rugg, R. (1999). Assessing semantic similarity among geospatial feature class definitions. In A. Vckovski, K. Brassel, & H.-J. Schek (Eds.), *Interoperating Geographic Information Systems—Second International Conference, INTEROP'99* (Vol. 1580, pp. 1–16). Berlin, Germany: Springer-Verlag.

Rubenstein-Montano, B., Liebowitz, J., Buchwalter, J., McCaw, D., Newman, B., & Rebeck, K. (2001). A systems thinking framework for knowledge management. *Decision Support Systems*, *31*(1), 5–16. doi:10.1016/S0167-9236(00)00116-0

Rumbaugh, J. (1994). Getting started: Using use cases to capture requirements. *Object-Oriented Programming*, *7*(5), 8–23.
Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorensen, W. E. (1991). *Object-oriented modeling and design*. Englewood Cliffs, NJ: Prentice Hall.
Saleem, N. (1996). An empirical test of the contingency approach to user participation in information systems development. *Journal of Management Information Systems*, *13*(1), 145–166. doi:10.1080/07421222.1996.11518116
Schenk, K. D., Vitalari, N., & Davis, K. (1998). Differences between novice and expert systems analysts: What do we know and what do we do? *Journal of Management Information Systems*, *15*(1), 9–50. doi:10.1080/07421222.1998.11518195
Shaft, T. M., & Vessey, I. (1998). The relevance of application domain knowledge: Characterizing the computer program comprehension process. *Journal of Management Information Systems*, *15*(1), 51–78. doi:10.1080/07421222.1998.11518196
Shaft, T., & Vessey, I. (2006). The role of cognitive fit in the relationship between software comprehension and modification. *MIS Quarterly*, *30*(1), 29–55.
Shankararaman, V., Gottipati, S., & Duran, R. (2012). A retail bank's BPM experience: Research note. *Journal of Information Technology Case and Application Research*, *14*(3), 46–51. doi:10.1080/15228053.2012.10845706
Smith, J., & Smith, D. (1977). Database abstractions: Aggregation and generalization. *ACM Transactions on Database Systems*, *2*(2), 105–133. doi:10.1145/320544.320546
Sommerville, I. (2001). *Software engineering* (6th ed.). Reading, MA: Addison-Wesley.
Spence, M. T., & Brucks, M. (1997). The moderating effects of problem characteristics on experts' and novices' judgments. *Journal of Marketing Research*, *34*, 233–247. doi:10.2307/3151861
Straub, D. W. (1989). Validating instruments in MIS research. *MIS Quarterly*, *13*(2), 147–169. doi:10.2307/248922
Turoff, M., Rao, U., & Hiltz, S. R. (1991). *Collaborative hypertext in computer mediated communications*. Proceedings of the 24th Annual Hawaii International Conference on System Sciences, Vol. IV.
Vessey, I. (1985). Expertise in debugging computer programs: A process analysis. *International Journal of Man-Machine Studies*, *23*(5), 459–494. doi:10.1016/S0020-7373(85)80054-7
Wieringa, R. (1998). A survey of structured and object-oriented software specification methods and techniques. *ACM Computing Surveys*, *30*(4), 459–527. doi:10.1145/299917.299919
Yoo, J. (2000). Relationship Analysis (Ph.D. Dissertation). Rutgers University.
Yoo, J., Catanio, J., Paul, R., & Bieber, M. (2004). Relationship analysis in requirements engineering. *Requirements Engineering Journal*, Springer Verlag London Ltd., *9*(4), 238–247. doi:10.1007/s00766-004-0205-5

Appendix A: Task

Topic: University On-Line Registration System

- At the beginning of each semester, the registrar's office will provide a list of courses to students through a new on-line registration system. Information about each course, such as professor, department, and prerequisites will be included to help students make informed decisions.
- The new system will allow students to review available courses and select four of them for the coming semester. In addition, each student will indicate two alternative choices in case a course becomes filled or canceled. No course will have more than ten students. No course will have fewer than three students. A course with fewer than three students will be canceled. If there is enough interest in a course, then a second session will be established.
- Professors must be able to access the on-line system to indicate which courses they will be teaching. They will also need to see which students have signed up for their courses. Professors are expected to maintain a list of their research interests and projects as well as a list of publications. All students have access to each professor's research interests and projects lists. Each professor has access rights to their own publications list and can assign individual students permission rights to view these publications.
- The registration process will last for three days. The first day will be freshman orientation and registration. All other students will arrive on the second day of the semester to register. The third day will be used to resolve any outstanding course assignment conflicts.
- Once the course registration process is completed for a student, the registration system sends information to the billing system, so the student can be billed for the semester.
- As a semester progresses, students must be able to access the on-line system to add or drop courses.

Appendix B: Experience questionnaire

Name:	_____
Last 4 digits of your SS#:	_____
Course #:	_____
Section #:	_____

Academic Coursework:

Please place a check mark (✔) in the box next to any of the following courses that you have already completed or are currently enrolled. Also, indicate the grade you received in the already completed courses.

Course number and name	✔	Grade
CIS 101 Computer Programming and Problem Solving		
CIS 103 Computer Science Business Problem Solving		
CIS 113 Introduction to Computer Science I		
CIS 114 Introduction to Computer Science II		
CIS 280 Programming Language Concepts		
CIS 375 Application Development on the WWW		
CIS 381 Object-oriented Software Systems		
CIS 390 Analysis & System Design		
CIS 431 Introduction Database Systems		
CIS 434 Advanced Database Systems		
CIS 464 Advanced Information Systems		
CIS 490 Design in Software Engineering		
CIS 491 Computer Science Project		
CIS 601 Object-oriented Programming in C++		
CIS 602 Java Programming		
CIS 631 Data Management Systems Design		
CIS 632 Advanced Data Management Systems Design		
CIS 663 Advanced Systems Analysis & Design		
CIS 673 Software Design & Production Methodology		
CIS 676 Requirements Engineering		
CIS 683 Object-oriented Software Development		

General Software Background:

1. Circle the number that indicates your familiarity with the following programming languages.
a. C# Very Familiar With ← 7–6—5–4—3–2—1 → Have Not Used
b. C++ Very Familiar With ← 7–6—5–4—3–2—1 → Have Not Used
c. C Very Familiar With ← 1–2—3–4—5–6—7 → Have Not Used
d. Java Very Familiar With ← 1–2—3–4—5–6—7 → Have Not Used
e. Basic Very Familiar With ← 1–2—3–4—5–6—7 → Have Not Used
f. Pascal Very Familiar With ← 7–6—5–4—3–2—1 → Have Not Used
g. Cobol Very Familiar With ← 1–2—3–4—5–6—7 → Have Not Used
h. Assembly Very Familiar With ← 7–6—5–4—3–2—1 → Have Not Used
i. Fortran Very Familiar With ← 1–2—3–4—5–6—7 → Have Not Used

2. Circle the number that indicates your familiarity with the following scripting languages.
a. XML Very Familiar With ← 7–6—5–4—3–2—1 → Have Not Used
b. HTML Very Familiar With ← 7–6—5–4—3–2—1 → Have Not Used
c. VBScript Very Familiar With ← 1–2—3–4—5–6—7 → Have Not Used
d. JavaScript Very Familiar With ← 1–2—3–4—5–6—7 → Have Not Used

3. Circle the number that indicates your familiarity with the following Database Management System (DBMS) packages.
a. Oracle Very Familiar With ← 7–6—5–4—3–2—1 → Have Not Used
b. SQL Very Familiar With ← 1–2—3–4—5–6—7 → Have Not Used
c. Access Very Familiar With ← 1–2—3–4—5–6—7 → Have Not Used

Software Development Background: (Please indicate your skill level in the following areas)

1. Software development life-cycle models (waterfall, spiral, iterative, RUP, etc.)
High Skill ← 7–6—5–4—3–2—1 → Low Skill

2. Software Economics (cost-benefit analysis, software cost estimation, feasibility studies, etc.)
High Skill ← 7–6—5–4—3–2—1 → Low Skill

3. Generating software system analysis documents.
High Skill ← 1–2—3–4—5–6—7 → Low Skill

4. Generating software system design documents.
High Skill ← 1–2—3–4—5–6—7 → Low Skill

5. Generating software system class diagrams.
High Skill ← 7–6—5–4—3–2—1 → Low Skill

6. Developing software code.
High Skill ← 7–6—5–4—3–2—1 → Low Skill

7. Using Modeling Languages and Techniques (UML, etc.)
High Skill ← 1–2—3–4—5–6—7 → Low Skill

Work Related Experience: (Please indicate your skill level in the following areas)

1. Working as a software engineer.
High Skill ← 7–6—5–4—3–2—1 → Low Skill

2. Working as a software developer.
High Skill ← 1–2—3–4—5–6—7 → Low Skill

3. Working as a system analyst.
High Skill ← 7–6—5–4—3–2—1 → Low Skill

End of Experience Questionnaire

Decision Criteria

The score possible ranges from 26 to 203 points, lowest to highest respectfully.
The cutoff point was 100, those below were classified as low experience and those above were classified as high experience.

Criterion	Possible points	Earned points
Academic performance	0 to 21	
General software background	16 to 112	
Software development background	7 to 49	
Software engineering professional background	3 to 21	
	Total	