2023

Navigating the Tensions between Traditional and Agile Development Approaches in Systems Analysis and Design Courses

Nik Hassan

Navigating the Tensions between Traditional and Agile Development Approaches in Systems Analysis and Design Courses

Nik Rushdi Hassan

University of Minnesota Duluth

nhassan@d.umn.edu

ABSTRACT

Recent developments in software engineering and information systems development (ISD), in particular agile development approaches, have further increased the gaps and tensions that have always existed in systems analysis and design courses. Ranging from the need to use newer models such as UML to transitioning from traditional plan-based ISD to agile-based approaches, both instructors and students are grappling with how to successfully navigate those tensions. This article surveys the landscape surrounding those tensions, offers a framework that might be used by IS undergraduate programs and proposes several strategics that could alleviate them to ensure a successful transition to more effective ISD courses.

Keywords

Systems analysis and design, information systems development (ISD), paradigms, methodologies, agile approaches, IS education

INTRODUCTION

For many information systems (IS) programs around the world, the IS development component, commonly titled "Systems Analysis and Design" (SA&D) has not changed appreciably since its inception. The IS'95 curriculum (Couger et al. 1995) begins by requiring proficiency with personal IT productivity tools, hardware and software concepts, followed by an understanding of how IT is applied within organizations, and courses in data structures, telecommunications, analysis and design, databases and a separate systems implementation course (often called the management information system or MIS capstone course). The IS'97 curriculum did not add any significant changes, but the explosion in the use of the Internet at the turn of the millennium added a new ecommerce section to the IS'2000 curriculum. The SA&D course (titled "Analysis and Logical Design") was described in the IS'97 curriculum as "an understanding of the systems development and modification process. It enables students to evaluate and choose a systems development methodology. It emphasizes the factors for effective communication with users and team members and all those associated with development and maintenance of the system. Topics included lifecycle phases: requirements determination, logical design, physical design, test planning, implementation planning, and performance evaluation; communication, interviewing, presentation skills; group dynamics; risk and feasibility analysis; group-based approaches: JAD, structured walkthroughs, and design and code reviews; prototyping" and other related topics. This description became the basis for all SA&D courses since and would not see any significant changes. The IS 2002 curriculum description for the similar Analysis and Logical Design course also did not change the previous description significantly.

It was during that period that textbook authors felt the need to revise the SA&D course to reflect the newer methodologies, techniques, models and tools being used in industry (Batra and Satzinger 2006). A special issue on SA&D (Harris et al. 2006) in the *Journal of Information Systems Education* (JISE) reinforced the status of SA&D as an essential part of IS education. Also referred as ISD, it is the IS field's oldest subarea (Hirschheim and Klein 2003) with a long and venerable tradition (Hirschheim et al. 1995). Since the inception of the IS field, ISD was conceived as the defining core of the field (Blumenthal 1969; Langefors 1973) and historically comprised as much as half of all IS research (Morrison and George 1995). Because of this historical tradition, it was viewed as the most likely to hold the potential for growing a body of knowledge unique to IS (Hassan and Mathiassen 2018; Iivari et al. 2004). However, as new models such as the Unified Modeling Language (UML) and tools arrived on the scene, the existing ISD body of knowledge have had to accommodate those new additions. Since most MIS degree programs only have one SA&D course, both instructors and textbook authors face a number of difficult questions

in trying to fit all the methodologies, techniques and models into a single course. Should that single course fit structured, iterative and agile approaches? Should students be exposed to both traditional plan-based project management or empirical-based project management? Can we teach students both data flow diagrams and UML diagrams? Of the many UML diagrams, which ones should be covered? How do we reconcile object-oriented development with relational ERDs? Should MIS programs consider splitting SA&D into the analysis phase and another for the design phase? Isn't analysis and design a completely different paradigm compared to the agile paradigm? Can the SA&D course combine both into a hybrid methodology? Should project management be taught as a separate course, or should the SA&D instructor work with the project management course and incorporate its principles into the SA&D course? These are just some of the questions that the stakeholders of the MIS program were concerned with. This paper will address them and propose some answers based on the framework for ISD practices.

THE PROBLEM STATEMENT

A survey was sent out to the AIS World listserv asking the IS community to respond to questions (See Appendix for questions) about their Systems Analysis and Design course and how they are incorporating agile approaches in that course. Most of the 29 responses received indicated that they used the Scrum methodology and it does not appear that there were any appreciable problems raised. However, several response noted:

"Theoretical courses are failing. Developing a small (3 iterations) realistic project practicing the XP-SCRUM method is better."

"It's a mess. The texts are overly complex because they cover traditional and agile. OO models still seem useful as agile has no specifications on models. Agile handling of requirements is reasonably advanced now so I cover that."

"Aspects of object-orientation and UML are quite abstract for many students. For example, just the term 'class' in 'class diagram' sounds confusing. You can prove that easily by asking students either of the following two questions:

- What does object orientation mean and why is it useful?
- What does the idea of class mean? Explain the different classes that are present in almost any real world example. Even the idea of use case can be confusing if the students do not understand the context in which the software will be used"

These responses are consistent with what is published concerning SA&D textbooks that provide only limited knowledge on how to implement agile approaches (Baham 2019). A summary of the results are given in Table 1. As Conboy (2009) explains, the concept of agility in ISD itself is plagued with many problems. The concept is not clearly communicated to its practitioners. While the agile manifesto may be portrayed as being clear and simple, in practice, it is not so. Many variants of agile approaches exists as well as their derivatives. There are no theoretical foundations that accompanied the introduction of agile approaches which led to a lack of cumulative tradition. If industry is facing these problems, they are sure to be exacerbated in academia. Some of these problems are discussed in this article.

ISD FRAMEWORK AND FUNDAMENTAL ISSUES

Before discussing specific issues, the terminology surrounding the SA&D course needs to be clarified. Using Hassan and Mathiassen's (2018) framework of ISD as a basis, we define an ISD paradigm as 'the most fundamental set of assumptions … about knowledge … and about the physical and social world' (Hirschheim et al. 1995, p. 46-47). For example, previous studies identify ISD paradigms (such as the functionalist paradigm) based on the ontological, epistemological or ethical assumptions of that practice. Paradigms are concretized into different "approaches" based on the similarity of their methodologies. We define an approach (e.g. object-oriented approach) as 'a set of goals, guiding principles, fundamental concepts and principles for the ISD practice that drive interpretations and actions in ISD' Iivari et al. (1998, p. 166). The notions of paradigms and approaches in ISD are distinctive and unique to the IS field with no equivalent concepts in software engineering. They provide important structuring of the methodology jungle (Avison and Fitzgerald 1988) by grouping similar methodologies into a single approach. A methodology is an instantiation of an approach, e.g., Rational Unified Process (RUP) is a methodology within the iterative approach while scrum is a methodology within the agile approach. Following Iivari et al. (1998), a methodology is further defined as "an organized collection of concepts, methods, beliefs, values and normative principles supported by material resources" (p. 165). This definition views methodology as more elaborate than "methods" by incorporating beliefs, values and norms. Hence, the traditional SDLC or waterfall methodology may apply the same interview methods as the object-oriented methodology, but they differ in concepts, beliefs and expectations (Sircar et al. 2001). Methods are therefore closer to technique, which is defined as 'a well-defined sequence of elementary operations, which permits the achievement of certain outcomes if executed correctly' (Iivari et al. 1998, p. 165). Most IS authors, as well as those in software engineering, use the term methods and methodology interchangeably (as in Object-Oriented Methods), which obscures the other aspects of development (e.g.

beliefs and values) contained in a methodology. We add to this ISD framework, the notion of the model (Hassan et al. 2022), which is defined as an imperfect copy of the phenomenon of interest consisting of positive and neutral analogies and in the case of ISD, commonly takes the form of graphical models such as data flow diagrams, entity-relationship and UML diagrams. Tools are software artifacts that build models. Certain Tools (e.g. the Information Engineering Workshop) are designed with specific methodologies in mind, expecting their user to comply with them, while others (e.g. generic CASE and modeling tools such as Visio) are more agnostic and admit any methodological principles.

Items	Responses	
Teaching a SA&D course	Yes (21)	No (8)
Textbook adopted	Dennis et al. (3) Satzinger at al. (2) Dennis et al OO with UML (2) Kendall and Kendall (1) Whitten and Bentley (1) Degan Kettles (1) No textbook (7)	
Project management tool adopted	Microsoft Project (3) Jira (2) Trello (2) Kunagi Open Source (1) Monday.com (1) ProjectPlan 365 (1) No project management tools (8)	
Extent of agile approach adopted	Agile approach exclusively (7) Combine traditional and agile (11) No agile approach used (1)	
Agile methodology adopted	Scrum (16) eXtreme Programming (2) Kanban (2) No agile approach specified (2)	
Models adopted	Use case diagram (14) Class diagram (11) Activity diagram (9) ER Diagram (7) Sequence diagram (7) Data Flow diagram (5) Program flowcharts (2) User Stories, CRC cards, Navigation Map (1) Double diamond (1) BPMN (1)	

Table 1: Survey responses

All of these elements of ISD are included into an ISD framework (see Figure 1). The organization of the ISD framework into paradigms, approaches, methodologies, methods, models, techniques and tools will help resolve inconsistencies in the use of ISD terms and organize the issues that will be presented later. The theorizing using this ISD framework accommodates seemingly different concepts and the tenuous and often confusing methodology jungle (Conboy 2009) into a coherent structure, describing how paradigms, approaches, methodologies, models and techniques and tools relate to each other as part of ISD disciplinary knowledge. The organization of ISD into the knowledge areas of paradigm, approach and methodology supports comparisons of different methodologies, placing focus less on the "methodology of the day" and more on the theories and principles shared across methodologies. A survey of existing ISD practices suggest that the dominant paradigm in ISD still is functionalism (Hassan and Mathiassen 2018) – the notion that the subject (the developer) can rationally and objectively control and manipulate certain representations of reality in the development project. What follows from this paradigm are various management theories aligned to Anthony's (1965) and Simon's (1979) rational, empiricist, mathematical, behavioral-functional and problem-solving approach that assume existing situations ("As-Is") can be transformed into preferred ones ("To-Be") using programmatically deterministic plans and strategies. For instance, as Iivari et al. (1998) explain, the paradigm underlying both the DeMarco and Yourdon structured methodologies is to build maintainable systems that have components that are highly cohesive, but weakly coupled, and both essentially apply a dataflow concept. In contrast, object-oriented approaches underpin various methodologies such as the RUP, XP, scrum and other agile methodologies with the goal of building reusable components for rapid deployment that apply information hiding and inheritance concepts. These two groups of methodologies apply different techniques and models; the structured methodologies apply functional decomposition techniques and data flow and entity-relationship models, while the object-oriented methodologies such as RUP and Scrum apply workflow and user stories techniques and UML models. Nevertheless, both structured and object-oriented approaches are driven by the same functionalist paradigm.

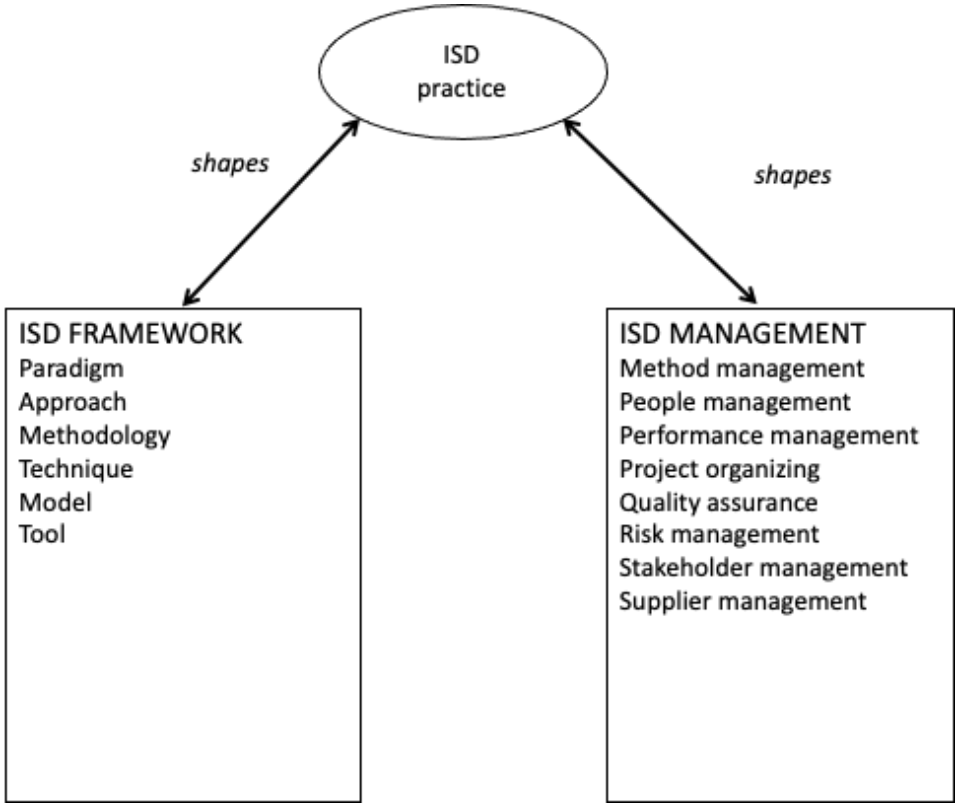

Figure 1: ISD Framework (Adapted from Hassan and Mathiassen, 2018)

On the other hand, less dominant approaches and theories such as the soft-systems methodology (Checkland and Scholes 1990) and critical theory underlying the European systems development traditions move towards achieving emancipatory goals and making sense of the environment in order to align with subjective needs. They are less dependent on the techniques and models relied on by structured and agile approaches. At the methodology level, both structured and agile approaches may use similar

techniques such as interviewing and brainstorming to elicit user requirements. The purported strength of the traditional SDLC or waterfall methodology lies in its ability to rationally organize complex systems and has been the approach of choice for large systems (Boehm 1976). The purported strength of object-oriented methodologies lies in its reusability and inherent capacity to manage changes and emergent requirements (Booch 1986). The landscape becomes more complex with the introduction of agile approaches. According to Iivari et al (2004), the agile approach still follows a functionalist paradigm. However, many authors might disagree because agile approaches following the Agile Manifesto (https://agilemanifesto.org/) value individuals and interactions over processes and tools, customer involvement over contract negotiation, and embrace change over following plans (Dingsøyr et al. 2012). Nerur and Balijepally (2007) suggest that agile approaches follow a dynamic problem-solving and emergent paradigm that embrace uncertainty and complexity instead of the traditional deterministic and mechanistic paradigm. In order for this paradigm to work accordingly, the goals, guiding principles, fundamental concepts and principles of the methodology will also need to embrace uncertainty and complexity. Notwithstanding all the claims about all these methodologies, ultimately teaching them becomes a problem just out of the sheer volume of choices and opinions surrounding each methodology.

Fitting several methodologies within a single SA&D course

The first consideration that needs to be made is whether or not project management can be taught separately from SA&D. Conceivably a separate course in project management can communicate the basics of how any project needs to be managed regardless of whether it is or not an ISD project. Programs that have the luxury of doing so or that can leverage existing project management courses should consider having two separate courses. Smaller programs that do not have such options will find solace in Hassan and Mathiassen's (2018) ISD framework that includes "ISD Management" as part of ISD that enables the coordination, organization, and practice of ISD amongst participating actors. Although ISD management contains many concepts that are generic to project management such as human resource management and risk management, ISD has enough unique needs that imply having a separate project management course might actually be counterproductive to agile approaches. This is so because agile approaches completely transforms how methods, people, tasks, risks, performances, quality and stakeholders are managed. With this background, an evaluation on whether several methodologies can be taught within a single SA&D course can be made.

Batra and Satzinger (2006) recommend having two SA&D courses to fit all the different methodologies. The two courses can be organized in five different ways depending on the goals and needs of the program. The first option is to split between analysis and design related tasks. This option becomes problematic if the program wishes to focus on agile approaches since agile approaches do not separate between analysis and design. The second option is to split between traditional and agile approaches. This option is probably available only to larger IS programs. The third option is to split between project management focus and SA&D focus. This option is already discussed previously and could be counterproductive for ISD. The fourth approach is to repeat each content in both courses, but in an iterative fashion with the first course focusing more on theory and introductory tasks and the second course completing the implementation-type tasks. This option is most effective in larger programs that have two courses that are required to be taken in tandem, otherwise students will not retain what they have learned in the first course. The fifth approach splits the content based on the architecture of the system being developed. This option is probably suited for larger programs and more advanced students who are capable of working on larger systems. One option that many undergraduate IS programs have adopted is to maintain the single SA&D course and require a capstone course where students code and implement a working system. The capstone course is not strictly a SA&D course so there is no splitting of SA&D content.

One novel perspective surrounding ISD is not to view traditional and agile methodologies as two necessarily opposing paradigms but to view them according to the lens of paradox theory. Similar to Hegelian thinking, "To be effective, an organization must possess attributes that are simultaneously contradictory, even mutually exclusive" (Cameron 1986, p. 545). What this paradigm implies is that students who are familiar with both traditional and agile approaches will perform better than those who are just familiar with any one approach. When implementing both approaches in a single SA&D course, it could be designed with students briefly introduced to the traditional approach but with more work performed using the agile approach. The other term that refers to the same paradigm is ambidexterity. Students that are ambidextrous in both traditional and agile approaches are more likely to be able to exploit and explore which of these contradictory approaches will be beneficial for the ISD project (Iivari 2021). Instructors who have taught both approaches in the same SA&D course provide evidence for the effectiveness of this approach (Baham 2019).

Choosing the right techniques and models for the course

With so many techniques and models available, a decision needs to be made about with concepts, techniques and models that can be taught in either the single or two SA&D courses. Batra and Satzinger (2006) strongly suggest eliminating the data flow

diagram from the content since it is not suitable for object-oriented analysis and design that has become the industry standard. Program flowcharting is approach-agnostic so can be included if need be. The next consideration has to do with the system architecture. With the data flow diagram absent, a hierarchy chart of the system architecture is the simplest model that represents the highest level architecture. UML models are categorized into structural, behavioral, grouping and annotational models. The latter two models are either part of or made up of the first two categories of models. Because structural models represent the static aspects of the system, system architectures can be represented by structural models like the component, deployment or grouping models like the package diagram, depending on what needs to be described. The other structural models are class and object diagrams. The class diagram represents the core model for object-oriented systems, so naturally should not be excluded from any SA&D course. Behavioral models represent the dynamic aspects of the system and can be represented by activity (dynamics of objects and their flow of control), use case (actions performed by the system to provide value to the actor), sequence (time ordering of messages sent between objects), collaboration (structural organization of objects and their messaging) and state chart (flow of control of objects from state to state) diagrams.

The syntax of each of the behavioral diagrams can be overwhelming for students, so only up to two to three diagrams can be fitted effectively into one SA&D course. The use case is deceptively simple but useful because it can be used as the model for communicating with the client, since it's the easiest to understand without prior training. The use case diagram corresponds directly with the functional user requirements of the proposed system. The four most popular models used in industry are the use case, activity, class and sequence diagrams (Dobing and Parsons 2006). Because UML was developed independently from agile approaches, several UML models are not consistent with agile techniques. For example, agile methodologies apply user stories which are not the same as UML's use case diagrams and narratives. The developer of use cases, Ivar Jacobson, has since linked use cases to agile approaches (Jacobson et al. 2016) although some problems still linger (Spurrier and Topi 2018). Another example of a disconnect lies between UML and the well-known entity-relationship diagram (ERD) model that was developed in the 1970s. Efforts have also been made to synchronize the object-oriented foundations of UML with ERD (Rumbaugh 2006).

The complexities of design and implementation

Another reason why a second SA&D course may be necessary is the time and effort taken to cover complex design and implementation issues. Systems design is much more complex today than it was decades ago, making it more difficult to decide what to emphasize in the analysis and design course. First, there are Web applications that range from informational sites to dynamic data-driven sites with hypermedia functionality that includes enhanced navigation, highly visual interfaces, and multimedia content. Second, mobile applications that require responsive interfaces and screen adapted features add to the application development work. Third, cloud services require applications to adhere to distributed architectures and international legal rules and policies.

Most of this learning is undertaken concurrently as students complete the core capstone course as they are forced to deal with these tasks in that course. Typically, the amount of work expected for the capstone course is kept to a minimum to allow it to be completed within a semester. One common option for doing so is to complete a web-based application or form that is connected to a database. In order to accomplish this level of performance, many IS programs require that students complete prerequisite database, object-oriented programming and web development courses at the lower sophomore or junior levels before taking the SA&D or capstone course (Baham 2019).

DISCUSSION, RECOMMENDATIONS AND CONCLUSION

Based on the developments in industry, it is unlikely that ISD will revert back to its traditional separate analysis and design phases or that it can be separated from code generation. With the advancement in artificial intelligent code generators such as ChatGPT and GPT-4, the programming tasks in ISD will likely not be relegated to later stages. Therefore it makes sense that the IS programs need to reevaluate how they should be structured. Here are several recommendation stemming from this article's analysis.

Rebranding of System Analysis and Design Courses

As part of the efforts to clarify and better communicate the latest developments in ISD, it is time to do away with the title "Systems Analysis and Design" and perhaps use more current and relevant labels such as ISD. Several major IS programs such as University of Texas Austin ("Business Systems Development") and Carnegie Mellon ("Application Design and Development") have already moved away from SA&D. Ironically, even though research in design science in IS has begun to accumulate some tradition (Gregor and Hevner 2013; Hevner et al. 2004), it is not clear how that tradition is connected to the

long history of teaching in SA&D. Such a disconnect needs to be resolved and elements of ISD practice and design science need to be more cohesive and clearly communicated.

Closer connection to programming and implementation tasks

With the dominance of agile approaches in industry, ISD courses need to work closer in tandem with programming courses. In the past, logical analysis and design was taught separate and even disconnected from programming courses. As mentioned above, many IS programs either require programming as prerequisites to the ISD course or require their students to take introductory computer science courses outside of the school of business in order to be able to deliver code in short iterations or sprints. Ideally, the kind of prerequisite programming and coding taught should reflect the business background of the student and therefore should be offered within the school of business. The way programming is taught in computer science lacks the business context in the examples being coded. Additionally, computer science programs often require specific math prerequisites that are not taken by students in the school of business. All of these issues encourage business schools to offer their own introductory programming courses.

Structure all ISD courses to exclusively teach or apply agile approaches with a background of the traditional approach

Given the limited time students can be exposed to the ISD course and the unlikely possibility that the school can offer two ISD courses, all ISD courses should teach or apply agile approaches exclusively. At the same time, some background of traditional methodologies can serve as the introduction so that students will appreciate the benefits of the agile approach. This recommendation implies that ISD textbooks be revamped to deliver not just the agile techniques and models, but also the paradigms and methodologies that undergird those agile techniques and models. The most difficult task in teaching agile models is internalizing the philosophy, values and attitudes that support the practice. For example, agile approaches imply a bottom-up, self-organizing coordination of tasks instead of the top-down project manager's control of the project. The philosophy of embracing change and variety instead of preventing them requires a transformation in the students' mindset.

In summary, a serious reevaluation of the existing SA&D courses is necessary in order for IS programs to keep abreast of the recent transformations in ISD. Research has shown that not only will this transformation result in better quality code and more manageable systems, it engenders a more engaged and motivated IS student body.

REFERENCES

Anthony, R.N. 1965. *Planning and Control Systems: A Framework for Analysis.*, Cambridge, MA: Harvard University Press.

Avison, D.E., and Fitzgerald, G. 1988. *Information Systems Development: Methodologies, Techniques and Tools*, Oxford: Blackwell Scientific Publications.

Baham, C. 2019. "Implementing Scrum Wholesale in the Classroom," *Journal of Information Systems Education* (30:3) pp. 141-159.

Batra, D., and Satzinger, J.W. 2006. "Contemporary Approaches and Techniques for the Systems Analyst," *Journal of Information Systems Education* (17:3) pp. 257-265.

Blumenthal, S.C. 1969. *Management Information Systems: A Framework for Planning and Development*, Englewood Cliffs, N. J.: Prentice-Hall, Inc.

Boehm, B.W. 1976. "Software Engineering," *IEEE Transactions on Computers* (25:12) pp. 1226-1241.

Booch, G. 1986. "Object-Oriented Development," *IEEE Transactions on Software Engineering* (SE-12:2) pp. 211-221.

Cameron, K.S. 1986. "Effectiveness as Paradox: Consensus and Conflict in Conceptions of Organization Effectiveness," *Management Science* (32:5) pp. 539-553.

Checkland, P., and Scholes, J. 1990. *Soft Systems Methodology in Action*, Chichester: John Wiley & Sons.

Conboy, K. 2009. "Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development," *Information Systems Research* (20:3) pp. 329–354.

Couger, J.D., Davis, G.B., Dologite, D.G., Feinstein, D.L., Gorgone, J.T., Jenkins, A.M., Kasper, G.M., Little, J.C., Herbert E. Longenecker, J., and Valacich, J.S. 1995. "IS '95: Guidelines for Undergraduate IS Curriculum," *MIS Quarterly* (19:3) pp. 341 - 359.

Dingsøyr, T., Nerur, S., Balijepally, V., and Moe, N.B. 2012. "A Decade of Agile Methodologies: Towards Explaining Agile Software Development," *Journal of Systems and Software* (85:6) pp. 1213-1221.

Dobing, B., and Parsons, J. 2006. "How Uml Is Used," *Communications of the ACM* (49:5) pp. 109–113.

Gregor, S., and Hevner, A.R. 2013. "Positioning and Presenting Design Science Research for Maximum Impact," *MIS Quarterly* (37:2) pp. 337-355.

Harris, A.L., Lang, M., Oates, B., and Siau, K. 2006. "Systems Analysis & Design: An Essential Part of IS Education," *Journal of Information Systems Education* (17:3) pp. 241-248.
Hassan, N.R., Lowry, P.B., and Mathiassen, L. 2022. "Editorial-Useful Products in Information Systems Theorizing: A Discursive Formation Perspective," *Journal of the Association for Information Systems* (23:2) pp. 418-446.
Hassan, N.R., and Mathiassen, L. 2018. "Distilling a Body of Knowledge for Information Systems Development," *Information Systems Journal* (28:1) pp. 175-226.
Hevner, A.R., March, S.T., Park, J., and Ram, S. 2004. "Design Science in Information Systems Research," *MIS Quarterly* (28:1) pp. 75-105.
Hirschheim, R., Klein, H.K., and Lyytinen, K. 1995. *Information Systems Development and Data Modeling Conceptual and Philosophical Foundations*, Cambridge, UK: Cambridge University Press.
Hirschheim, R.A., and Klein, H.K. 2003. "Crisis in the IS Field? A Critical Reflection on the State of the Discipline," *Journal of the Association for Information Systems* (4:5) pp. 237-293.
Iivari, J. 2021. "A Paradox Lens to Systems Development Projects: The Case of the Agile Software Development," *Communications of the Association for Information Systems* (49:1) p. Art 4.
Iivari, J., Hirschheim, R., and Klein, H.K. 1998. "A Paradigmatic Analysis Contrasting Information Systems Development Approaches and Methodologies," *Information Systems Research* (9:2) pp. 164-193.
Iivari, J., Hirschheim, R.A., and Klein, H.K. 2004. "Towards a Distinctive Body of Knowledge for Information Systems Experts: Coding ISD Process Knowledge in Two IS Journals," *Information Systems Journal* (14:4) pp. 313-342.
Jacobson, I., Spence, I., and Kerr, B. 2016. "Use-Case 2.0," *Communications of the ACM* (59:5) pp. 61-69.
Langefors, B. 1973. *Theoretical Analysis of Information Systems*, Philadelphia, PA: AUERBACH Publishers Inc.
Morrison, J., and George, J.F. 1995. "Exploring the Software Engineering Component of MIS," *Communications of the ACM* (38:7) pp. 80-91.
Nerur, S., and Balijepally, V. 2007. "Theoretical Reflections on Agile Development Methodologies," *Communications of the ACM* (50:3) pp. 79–83.
Rumbaugh, J.E. 2006. "Er Is Uml," *Journal of Information Systems Education* (17:1) pp. 21-25.
Simon, H.A. 1979. "Rational Decision Making in Business Organizations," *The American Economic Review* (69:4) pp. 493-513.
Sircar, S., Nerur, S.P., and Mahapatra, R. 2001. "Revolution or Evolution? A Comparison of Object-Oriented and Structured Systems Development Methods," *MIS Quarterly* (25:4) pp. 457-471.
Spurrier, G., and Topi, H. 2018. "Resolving the Pedagogical Disconnect between User Stories and Use Cases in Systems Analysis and Design Textbooks," *Americas Conference on Information Systems, August 16-18*, New Orleans.

APPENDIX: SURVEY QUESTIONS

This brief survey collects information about how IS professors are teaching Agile approaches in their Systems Analysis and Design or Software Development courses. Thank you for contributing to the practice of teaching IS. This survey should not take more than 2 minutes.

Q1 Are you teaching a Systems Analysis and Design or Systems/Software Development course?

- Yes (1)
- No, but I'm interested in this topic (2)

Q2 Which textbook have you adopted for your Systems Analysis and Design or Systems/Software Development course?

- Satzinger et al's Systems analysis and design in a changing world (1)
- Valacich et al's Modern systems analysis and design (2)
- Dennis et al's Systems analysis and design (3)
- Dennis et al's Systems analysis and design: An object-oriented approach with UML (4)
- Kendall et al's Systems analysis and design (5)
- Other (please specify) (6) _____

Q3 What project management tool do you use for your Systems Analysis and Design or Systems/Software Development course, if any?

- No, I don't incorporate project management tools in my course (1)

- Microsoft Project (2)
- Jira (3)
- Monday.com (4)
- Wrike (5)
- Trello (6)
- Other (please specify) (7) _____

Q4 Are you using Agile approaches to teach your Systems Analysis and Design or Systems/Software Development course?

- Yes, I use Agile approaches exclusively (1)
- No, I don't use Agile approaches at all (2)
- I combine both traditional (SDLC/Waterfall) and Agile approaches (3)
- Other (please specify) (4) _____

Q5 Which Agile methodology do you use? (Select all that apply)

- Scrum (1)
- Extreme Programming (2)
- Kanban (3)
- Lean (4)
- DevOps (5)
- Other (please specify) (6) _____

Q5 Which analysis and design models do you use in your Systems Analysis and Design or Systems/Software Development course? (select all that apply)

- ER diagram (1)
- Data Flow Diagram (DFD) (2)
- Activity diagram (3)
- Use case diagram (4)
- Class diagram (5)
- Sequence diagram (6)
- Program flowcharts (7)
- Other models (list all separated by commas) (8)