Resolving the Pedagogical Disconnect between User Stories and Use Cases in Systems Analysis and Design Textbooks

Completed Research

Gary Spurrier
University of Alabama
gspurrier@cba.ua.edu

Heikki Topi
Bentley University
htopi@bentley.edu

Abstract

Over the past 25 years, user stories and use cases have become increasingly dominant techniques for defining software project requirements. Indeed, although they originated in different camps—user stories from agile development and use cases from software engineering—numerous prominent authors in systems analysis and design (SA&D) have noted important similarities between the two techniques, such that at times one technique can be substituted for the other. However, they also may differ significantly, especially as scope is refined in preparation for software construction. As such, user stories and use cases may play different roles and relate to each other in complex ways. It is therefore problematic that most SA&D textbooks do not explicitly compare and contrast user stories and use cases, let alone explore their complex interrelationships. This paper systematically reviews the state of SA&D textbooks and makes recommendations to resolve this disconnect between user stories and use cases.

Keywords

Requirements specification, user stories, use cases, IS education

Introduction

Since the early 1990s, user stories and use cases have become increasingly important and widely used (even dominant) methods of defining software project requirements. This is evidenced by that most leading Systems Analysis and Design (SA&D) textbooks have taken to referring to other, older techniques such as data flow diagrams as “traditional” requirements approaches (Dennis et al., 2015, pp. 7, 23, 112, 127, 184, 218, 264; Satzinger et al. 2016, pp. OL-20-OL-38) or distinguishing the “traditional” structured systems development life cycle (SDLC) approaches from the more modern agile and object-oriented approaches that utilize user stories and use cases, respectively (Kendall and Kendall 2014, pp. pp. 4-13; Tilley and Rosenblatt, pp. 400-1; Valacich and George, 2017, pp. 15-23, 183-4).

As will be established below, user stories and use cases are at once highly similar, yet also clearly distinct in their nature and use in software development projects. In a nutshell, both user stories and use cases express how a user utilizes a software application to achieve a goal to derive business value. To use a now well-known formulation, the essence of both a user story and a use case can be expressed in terms of “As a <type of user actor>, I want/need to <achieve some goal via application software> in order to <obtain some business value>.” The details can, however, be very different.

When creating a backlog of items that can be constructed in the context of agile development iterations or sprints, a user story can be best viewed as an initial feature description—literally, what the system should do without respect to how the feature would be constructed. In contrast, a buildable use case “slice” (Jacobson et al., 2011) can be thought of as a transformation of one or more of those user stories (or briefly, initially-expressed use cases) that have been explicitly groomed to be constructed within a specific iteration or sprint by a team focused on the corresponding, underlying enabling technology. These transformations require consideration of several issues, including:
• **Transitions of primary actors:** Within a business process that should be addressed via different methods, data, and user interfaces.

• **Transitions of enabling technologies:** That may require different software development team members or overall teams to build.

• **Splitting the work to fit into an iteration:** Even after splitting a chunk of work by primary actor and enabling technologies, the work may be found to still be too large to be built within a single construction iteration (such as a sprint in Scrum). It therefore may be necessary to split the work into smaller units purely for sizing purposes (Leffingwell, 2011, pp. 108-9).

These issues can be well-understood from a careful reading of the practitioner literature, although, as we shall establish below, creating that understanding is complicated by differing terminology utilized across user stories and use cases, as well as within those different camps (especially for use cases) and between different authors. We would therefore hope and expect that, from an SA&D pedagogical perspective, leading SA&D textbooks would clearly explain these issues—including resolving terminology differences—to provide students with a thorough understanding of the similarities and differences between user stories and use cases. Further, we would hope and expect that transformations necessary to groom features expressed as user stories or use case briefs into buildable use case slices would be clearly explained.

We will, however, demonstrate in this paper that, in general, SA&D textbooks do not consistently and thoroughly provide this information. We suggest reasons as to why this is the case. We conclude with recommendations for improving SA&D textbooks to resolve this pedagogical disconnect.

**Terminology Challenges**

It is worthwhile noting that the exact terminology utilized for use cases varies significantly. Different authors addressing the same concepts utilize different terms or, in some cases, equivalent descriptions that do not map to a concise specific terminology. For example, the term “Main Success Scenario” as used by Cockburn (2001) is termed “Basic Flow” by Jacobson, Bittner, and Spence (2011).

This, not surprisingly, increases the challenge of achieving a common understanding of the concepts and their implications for SA&D practice. It follows that part of the challenge of SA&D textbooks should be to provide key terminology from key practitioner authors and their equivalencies. Such equivalent terminology used by other authors is shown in Table 1. In Table 1, “N/S” means “Not Specified” by that author.

**The Relationship Between User Stories and Use Cases**

As per the discussion above, both user stories and use cases are modern approaches to expressing software requirements features. More specifically, they are both ways of articulating how a user (or “actor”) interacts with a system to achieve some business goal or benefit. The differences between the two concepts arise from the level of detail that may ultimately be included in each approach—relatively less in user stories and relatively more in use cases.

A key reason that user stories and use cases must be considered as distinct concepts is that they originated independently from different software development schools of thought. Specifically, user stories originated from Kent Beck’s XP agile development methodology (Leffingwell, 2007, p. 29) and then spread to other agile approaches, including Scrum (Leffingwell, 2007, pp. 219-220; Rubin, 2013, p. 80). Use cases, in contrast, originated from the software engineering approach of Ivar Jacobson (Tiwari and Gupta, 2015) and have come to be closely associated with the Unified Process (Leffingwell, 2007, pp. 221-2).

Thus, we see at the outset that user stories and use cases are at once similar and also distinct—a recipe for confusion in understanding their interrelationship.

**Similarities Between User Stories and Use Cases**

Importantly, however, practitioner-oriented literature (unlike SA&D textbooks) note that it is also true that user stories and use cases are strikingly similar in certain key respects, especially when comparing
### Table 1. User Story and Use Case Terminology

<table>
<thead>
<tr>
<th>Core Concept</th>
<th>Agile Terminology</th>
<th>Use Case Terminology</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single feature or activity</strong></td>
<td>User Story (pp. 56-7)</td>
<td>User-goal Level Use Case (p. 3)</td>
</tr>
<tr>
<td><strong>Compound story describing multiple features or activities</strong></td>
<td>Epic (pp. 85-90)</td>
<td>Summary Level Use Case (p. 3)</td>
</tr>
<tr>
<td><strong>Briefly stated format</strong></td>
<td>User Story (pp. 102-4)</td>
<td>Casual Use Case (p. 120-1)</td>
</tr>
<tr>
<td><strong>Highly elaborated, fully-detailed format</strong></td>
<td>User Story Detail Attachment or Token (p. 104)</td>
<td>Fully Dressed Use Case (pp. 119-20)</td>
</tr>
<tr>
<td><strong>Acceptance criteria</strong></td>
<td>Acceptance Criteria (pp.104-5)</td>
<td>Business Need Work List (pp. 173-4)</td>
</tr>
<tr>
<td><strong>Basic flow of steps without exceptions, errors, or variations</strong></td>
<td>No direct equivalent</td>
<td>Main Success Scenario (pp. 87-8) or Brief Use Case (pp. 17, 37-8)</td>
</tr>
<tr>
<td><strong>Steps supporting exceptions, errors, or variations</strong></td>
<td>No direct equivalent</td>
<td>Extensions (pp. 90ff) or Alternate Scenarios (p. 138)</td>
</tr>
<tr>
<td><strong>Use Case split into specific subsets of the basic flow and other steps, often created to provide chunks of functionality buildable in a single sprint</strong></td>
<td>User Story Vertical Slice (p. 111)</td>
<td>Recognizes Use Case Scenarios as a basis for some design approaches; generally does not recommend use cases as basis for planning design (Ch. 17)</td>
</tr>
</tbody>
</table>

Table 1. User Story and Use Case Terminology

User stories to shorter versions of use cases, e.g., “use case briefs” or “features” to use Alistair Cockburn’s influential use case terminology (Cockburn, 2001, p. 187). For example, one source points out that “Use cases...are essentially stories” (Bittner and Spence, 2003, p. 4), in that both use cases and stories explain “…how an actor uses the system to achieve some end result” (Bittner and Spence, 2003, p. 207). Similarly, another source asserts that “…the text on a [user] story card 'plus acceptance tests are basically the same thing as a use case’” (Cohn, 2004, pp. 137-8). Indeed, cutting across agile and software engineering camps, another author baldly states that, when expressed briefly as “bite-sized chunks” of functionality, use cases and user stories are fundamentally equivalent and interchangeable (Leffingwell, 2007, pp. 72, 102, 118-120).
Further, it can be seen that user stories and use cases are similar in that they both can refer to either a single task or activity vs. multiple tasks or activities. In the context of user stories, the former is typically (and somewhat confusingly) denoted as a “user story,” while the latter is generally called an “epic” (Leffingwell, 2011, p. 111). In the context of use cases, the former is in the most widely used approach called a “user-goal level use case” (Cockburn, 2001, p. 3), while the latter is called a “summary level use case” (Cockburn, 2001, p. 3) or, alternately, simply a use case that covers “multiple stories” (Jacobson, 2011, p. 25). Note that multiple tasks or activities are frequently part of complex end-to-end business processes, as can be portrayed in a swim lane UML Activity Diagram.

**Differences Between User Stories and Use Cases**

However, the relevant practitioner literature also makes it abundantly clear that, in at least two important respects, user stories and use cases can be quite different.

**Fully Dressed Use Cases vs. User Stories**

First, use cases can routinely be elaborated to be far more detailed than user stories or use case briefs. Cockburn calls these highly detailed versions “fully dressed use cases,” in contrast to more succinctly expressed use cases that are more similar in detail to user stories; Cockburn denotes these as “use case briefs” or “features” (Cockburn, 2001, p. 187). In contrast to a user story, which at a maximum will contain the single feature statement “As a <type of user role>, I want/need to <achieve some goal via application software> in order to <obtain some business value>” plus a list of acceptance criteria, a fully dressed use case will contain that information plus significant additional details, including (Cockburn, 2001, p. 119-20):

- **Stakeholders and Interests**: Including those in addition to the primary actor.
- **Preconditions**: To the execution of the use case execution.
- **Minimal Guarantees**: How stakeholder interests are protected in all outcomes.
- **Success Guarantees**: The state of the world in the event of a successful outcome.
- **Trigger**: What starts the use case execution.
- **Main Success Scenario**: Step by step description of the execution of use case in the absence of errors or exceptions.
- **Extensions**: To the Main Success Scenario detailing what can happen differently, including errors and exceptions.

**Decomposition of Use Cases to Buildable Chunks of Functionality**

A key implication of the prior section is that summary feature statements—whether initially expressed as use case briefs or user stories—can be elaborated to a level of detail via use cases that enables them to be readily decomposed into smaller chunks of work. These smaller chunks can be used to form a product backlog consisting of items that can be readily mapped to development teams for construction within individual sprints (Jacobson, et al., 2011, p. 33; Leffingwell, 2011, p. 111). The practitioner literature focused on this decomposition—variously termed “splitting” or “slicing”—makes it clear that there are many ways that this decomposition can be done (Cockburn, 2001, pp. 67, 143-5, 182-3; Jacobson, 2011, pp. 24-5; Larman and Vodde, 2010, pp. 248-65; Leffingwell, 2011, pp. 111-6).

Importantly, while it is true that user stories can frequently be mapped one-to-one with (perhaps more detailed) use cases—either from the epic user story to a summary use case level or from an (individual) user story to an elaborated description in a user-goal level use case—the more general case is that user stories and use cases may have a many-to-many relationship (Leffingwell, 2011, p. 379). These complex, M:N relationships can arise as user stories are mapped via decomposition, optional elaboration (of user stories to use cases), and merging of similar features.

In particular, decomposition approaches can be quite complex, including but not limited to splitting features, expressed either as (often epic) user stories or (often summary level) use cases by:
• **Configurations**: Such as different operating systems.
• **I/O channels**: Such as command line vs. GUI or data entry methods.
• **Data parts**: Such as a subset of data entities or elements.
• **Knowledge of requirements**: Such that the best understood requirements are split out to be built first.
• **Importance of requirements**: Such that the essential or most valuable requirements are split out to be built first.
• **Non-functional requirements**: Such as supporting high throughput or high availability.
• **Operations**: Such as splitting by HTTP GET operations or by CRUD operations.
• **Integration**: With an existing component.
• **Test cases**: Chunked into meaningful sub-groups.
• **Scenarios/scenario steps**: Representing different combinations of the basic flow and extensions.

An inspection of these points indicates that, while this list might be effectively parsed and utilized by experienced systems analysts, there are likely too many options for SA&D students to usefully consider. In other words, students are likely to be overwhelmed by facing too many choices that have no clear order of implementation.

Given this, for pedagogical purposes, we propose an approach to splitting use cases into “use case slices” (to use the term used by Jacobson et al., 2011) with a process that focuses on a series of three general steps that are both represented in the practitioner literature and which are likely to be understandable, manageable, and, therefore, useful to SA&D students:

1. **Split by primary actor**: This allows an epic or summary level use case to be decomposed to a level expressible via a series of “As a <type of user actor>, I want/need to <achieve some goal via application software> in order to <obtain some business value>” statements.

2. **Split by enabling technology**: In many cases, a high level use case may consist of multiple enabling technologies. For example, a process to take in external data, summarize it for historical trends, model the summary data to project future trends, and then generate an elaborate report might utilize, respectively, an ETL tool, a general programming language, a business intelligence tool, and a high-quality report generator. It is likely that at least some of these would need to be built by different developers or teams with a command of each specific technology. Even if this is not the case, the change in technology can provide a convenient basis for splitting the high level use case into more detailed,
implementable use case slices.

3. **Split by size using scenarios**: As a final step, if the use case slice is still too large to build in a single sprint, then the use case slice can be further sliced by focusing on first building the smallest collection of basic flow and extensions that offer value to a user, and then defining additional use case slices that add in extensions or alternate scenarios.

Figure 1 includes a graphical representation of a process that systematically addresses all of these transformations.

We suggest that many of the more specific splitting methods listed above can be mapped to these three generic steps. To this end, Table 2 presents our proposed three-step splitting approach with mappings from the more specific, highly granular splitting techniques offered by practitioner authors who endorsed these techniques. A few of these splitting techniques do not appear to map cleanly to our three steps; these are listed in the final row of Table 2 as “Other/Not Mapped Splitting Techniques.”

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1: Split Epics/Summary Use Cases to Single Actors</strong></td>
<td>• Find Actors (p. 24)</td>
<td>• Identify Summary Level Use Cases (pp. 64-5)</td>
<td>• Role or persona</td>
<td>• Data entry Methods</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Type (e.g., of trades)</td>
<td>• Variations in data</td>
</tr>
<tr>
<td><strong>Step 2: Split by Enabling Technology</strong></td>
<td>• By Development Team (p. 144); Identify Technology and Data Variations (Chapter 9)</td>
<td>• Configuration (e.g., operating system)</td>
<td>• I/O channel (e.g., GUI or command line)</td>
<td>• Defer system qualities (e.g., non-functional requirements)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Data part or format (e.g., XML or JSON)</td>
<td>• Non-functional requirements</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Integration (e.g., of an existing technology)</td>
<td>• Data entry Methods</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3: Split by Basic and Alternate Scenarios</strong></td>
<td>• Find Use Cases (p. 25) and “Stories” (NOTE: “Stories” in this context is different from the standard “Use Story” definition)</td>
<td>• Write the Main Success Scenario (Chapter 7) and Write Extensions (Chapter 8)</td>
<td>• Use cases</td>
<td>• Use case scenarios</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Scenario or steps</td>
<td>• Workflow steps</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• CRUD (create/retrieve/update/delete) or other operation</td>
<td>• Business rule variations</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Sub-group of acceptance tests</td>
<td>• CRUD operations</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Major effort (to build first slice)</td>
</tr>
<tr>
<td><strong>Other, Not Mapped Splitting Techniques</strong></td>
<td></td>
<td></td>
<td>• Knowledge of requirements (e.g., best understood requirements)</td>
<td>• Break out a spike (of complexity)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Stub (e.g., fake implementation)</td>
<td>• Simple/complex</td>
</tr>
</tbody>
</table>

Table 2. Proposed 3-Step Splitting Process, with Mappings from Practitioner Book Authors
Evaluation of Systems Analysis and Design Textbooks

Evaluation Criteria

Based on the discussion above, we evaluated the extent to which current, leading SA&D textbooks would provide three key types of information regarding user stories and use cases:

- **Comparison and contrast of user stories and use cases**: The extent to which the textbook highlights the fundamental similarities and differences between the two concepts.

- **Explanation of terminology differences and equivalencies**: Students are likely to encounter these in professional contexts, as portrayed in Table 1. For example, explain that a “main success scenario” and “basic flow” are synonyms.

- **Method for use case splitting/slicing**: Explaining the need to transform initial user stories or use cases to create a buildable product backlog, including a straightforward, comprehensible method of conducting the splitting/slicing, using an approach at a similar level of complexity as the one we proposed in Figure 1.

Textbooks Evaluated

We selected SA&D textbooks for evaluation utilizing the following criteria:

- **Systems Analysis and Design textbook**: The title of book had to include the phrase “Systems Analysis and Design.” Further, the book had to be represented as being a college-level textbook on Amazon.com.

- **Latest Edition**: We selected the most recent edition of each SA&D textbook to review.

- **Top sellers**: We selected the five best selling SA&D textbooks, as per rankings using the Amazon rankings: “Books > Computers & Technology > Computer Science > Systems Analysis & Design.” Other SA&D textbooks that appear in the Amazon search are older versions of the latest editions.

Based on these selection criteria, we reviewed five current SA&D textbooks (listed alphabetically by first author):

- **System Analysis & Design: An Object-Oriented Approach with UML, 5th Ed.**, by Alan Dennis, Barbara Haley Wixom, and David Tegarden

- **Systems Analysis and Design, 9th Ed.**, by Kenneth E. Kendall and Julie E. Kendall


- **Systems Analysis and Design, 11th Ed.**, by Scott Tilley and Harry Rosenblatt

- **Modern Systems Analysis and Design, 8th Ed.**, by Joseph S. Valacich and Joey F. George

As a practical matter, these five textbooks are the leading textbooks on the market today. They therefore are the leading sources of information to SA&D students to understand user stories, use cases, and their interrelationship.

Evaluations

For each textbook, each author of this paper independently evaluated the content and presentation of each of the three categories of information listed above, utilizing a five point scale:

- **0-None**: The textbook provides no mention of the information category.

- **1-Minimal**: The textbook mentions the information category, but provides no significant discussion that would inform SA&D students per the discussion above.
<table>
<thead>
<tr>
<th>Textbook</th>
<th>Comparison/Contrast of User Stories and Use Cases</th>
<th>Explanation of Terminology Differences and Equivalencies</th>
<th>Method for use case splitting/slicing</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Analysis &amp; Design: An Object-Oriented Approach with UML, 5th Ed., by Dennis, Wixom, and Tegarden</td>
<td>1-MINIMAL: Focused on object-orientation via use cases, the book only briefly discusses user stories (pp. 14, 110-2) and does not compare/contrast them with use cases. P. 14 does state “user stories... are invaluable tools from which object-oriented systems development could benefit.”</td>
<td>1-MINIMAL: User story terms are largely lacking. Use case terms presented are idiosyncratic, albeit meaningful (e.g., “Normal flow of events” vs. “Subflows,” “Overview use case” vs. “Detail use case”). Specific terminology from Cockburn and Jacobson et al. is missing.</td>
<td>2-PARTIAL: Concepts of product backlogs of stories in agile development using timeboxing are presented (pp. 12-5, 67-9), but without discussing specific techniques for splitting. Discussion of use cases does mention splitting based on complexity of steps and flows (pp. 141-8).</td>
</tr>
<tr>
<td>Systems Analysis and Design, 9th Ed., by Kendall and Kendall</td>
<td>2-PARTIAL: Both are introduced in Ch. 1, but without comparisons. User stories are explored in Ch. 6, while Use Cases are explored both Chs. 2 and 10. There are no comparisons, although Ch. 2 p. 38 does suggest using agile stories as a starting point for creating use cases.</td>
<td>1-MINIMAL: Some of Cockburn’s use case terminology is included (e.g., Summary Level, User Goal Level, Main Success Scenario, Extensions or Alternate Scenarios). Terms from Jacobson et al. are not included, nor is there any comparison to user story terms.</td>
<td>1-MINIMAL: Although different use case levels are noted, there is no discussion of splitting use cases. Regarding user stories, the text merely notes the need to timebox development into short releases without discussing how to create those backlogs.</td>
</tr>
<tr>
<td>Systems Analysis and Design, 7th Ed., by Satzinger, Jackson, and Burd</td>
<td>3-SUBSTANTIAL: Ch. 3 discusses key similarities and differences between user stories and use cases (p. 71). This relationship between user stories and use cases is noted in numerous other places in the text (e.g., pp. 16, 94, 96, 313, 350).</td>
<td>1-MINIMAL: The text in Chapters 3 and 5 uses idiosyncratic, albeit meaningful, terms for most concepts, e.g., “Use Case Description,” “Flow of Activities,” “Exception Conditions.” There is no discussion of use case levels.</td>
<td>2-PARTIAL: Chs. 1 and 11 note user stories and use cases must be mapped to an “iteration schedule.” However, there is no splitting technique, other than noting use cases can be decomposed from business processes.</td>
</tr>
<tr>
<td>Systems Analysis and Design, 11th Ed., by Tilley and Rosenblatt</td>
<td>0-NONE: User stories are presented in Chapter 11 and use cases are presented in Chapter 6, with no comparison.</td>
<td>1-MINIMAL: User stories include the concept of epics (p. 376), but no acceptance criteria. Use cases are only briefly introduced (pp. 186-8), use idiosyncratic terms, and say nothing about levels.</td>
<td>1-MINIMAL: “Iteration planning” is mentioned as needing to “break down user stories into tasks...” (p. 376), but without meaningful guidance.</td>
</tr>
<tr>
<td>Modern Systems Analysis and Design, 8th Ed., by Valacich and George</td>
<td>0-NONE: User stories are presented in Chapter 6 and use cases are presented in Appendix 7A, with no comparison.</td>
<td>2-PARTIAL: Cockburn’s use case terminology is largely included (e.g., Summary Use Case, User Goal Use Case, Main Success Scenario, Extensions), but not that of Jacobson et al. nor any comparison to user story terminology.</td>
<td>2-PARTIAL: The need to create “vertical slices” that can be built “during this iteration” is identified in an appendix to Chapter 3. However, there is only a barebones process described to create these slices (p. 172).</td>
</tr>
</tbody>
</table>

Table 3. Evaluations of SA&D Textbooks Against Key Criteria
Pedagogical Disconnect between User Stories and Use Cases

- **2-Partial**: The textbook discusses the information category, including a subset of the relevant concepts and principles discussed above, in a manner that would provide some meaningful information and guidance to SA&D students.

- **3-Substantial**: The textbook thoroughly discusses the information category, including substantially all relevant concepts and principles, with only minor omissions.

- **4-Thorough**: The textbook discusses the information category in essentially complete detail.

Our evaluations of these SA&D textbooks are discussed below. Further, to promote comparisons between the books, key findings of individual textbooks are summarized in Table 3.

**Findings and Recommendations**

We would hope that, at a minimum, all SA&D textbooks provide “3-Substantial” or better support for each of the three categories of information detailed above. However, as Table 3 indicates, only a single textbook provides that level of information in even a single information category. This clearly indicates the need for substantial revisions and updates to all of these textbooks to address the needs of SA&D pedagogy.

A question that naturally arises considering these findings is why these leading SA&D textbooks suffer from these deficiencies. Although the following is speculation, we believe that several reasons may be in play:

- **Differing origins of user stories vs. use cases**: As noted above, user stories originated from the concepts of agile development as articulated by Kent Beck. In contrast, use cases came from the software engineering environment as articulated by Ivar Jacobson. As such, a significant amount of the original practitioner literature pertaining to each of these respective camps did not explicitly reference the other camp. Thus, SA&D textbook authors may have similarly introduced these sets of concepts separately.

- **Lack of standards for use case narratives**: While use case diagrams are a UML standard, use case narratives are not (Cockburn, 2001, p. xxi). As such, while influential authors such as Alistair Cockburn have exerted significant influence on the field (Cockburn, 2001; Leffingwell, 2011, p. 368), it is still the case that individual practitioners—and, it follows, individual SA&D textbook authors—have been free to utilize their own use case narrative forms and terminology. That the originator of use cases, Ivar Jacobson, chose in some of his most recent writings to utilize terminology different from that of Cockburn (Jacobson et al., 2011) has likely contributed to this variation in use case terminology.

- **Lack of easily ascertained explanations regarding many-to-many relationships between user stories and use cases**: A careful review of key practitioner authorities (Jacobson et al., 2011; Leffingwell, 2011) reveals unambiguously that this complex relationship does exist between these concepts. In general this is, however, the least easily ascertained aspect of user stories vs. user cases, with many sources remaining silent or at least implicit about this relationship. It is, therefore, not surprising that, coupled with the first point above, many SA&D textbook authors either did not recognize this relationship or, perhaps, simply chose not to emphasize it in their works.

To conclude, we strongly recommend that SA&D authors take the time and effort to improve their textbooks per the discussion in this paper. However, this is clearly not a set of updates and revisions that can be expected to occur in the immediate future, given the amount of time required to revise and issue new editions of these works. Indeed, for most of these textbooks, adequately knitting together the concepts of user stories and use cases would likely require a major organizational revision to bring the agile and structured/object-oriented sets of concepts together.

Given this likely delay, we urge professors and instructors of SA&D to take matters into their own hands in order to bridge the gap, up to and including referencing practitioner texts cited herein and utilizing this paper as source material for their classes.
REFERENCES

https://www.ibm.com/developerworks/rational/library/content/RationalEdge/jun01/GeneratingTestCasesFromUseCasesJune01.pdf