Design Pattern Impact on Reliability and Usability an SMS

Brandon Uehlein and Isaac Griffith
Informatics and Computer Science
Idaho State University
Pocatello, Idaho, 83208
Email: {uehlbran@isu.edu, grifisaa@isu.edu}

***Abstract*—Blah:**

***Index Terms*—**

1. Introduction

2. Background and Related Work

2.1. Design Patterns

In software engineering, we as developers tend to encounter a lot of the same problems throughout our careers. Regardless of where we are, what we're working on or the problem we're trying to solve these issues continue to show up. They're common, reoccuring problems, and are exactly what design patterns are trying to solve. In 1977, Alexander [1], wrote a book on design patterns called "A Pattern Language". In this book, design patterns are defined as reocurring problem for which we have a common, repeatable solution that can be implemented numerous times over without ever being exactly the same. Funny enough, this book is actually about architecture in buildings and towns and not programming. This isn't the only benefit that design patterns provide however. They also give us a taxonomy in which to discuss said issues.

In 1987, following the work of Alexander, Beck et al, began working with design patterns in programming [citation]. In 1994, "Design Patterns: Elements of Reuseable Object-Oriented Software" was published by a group known as the "Gang of Four". In this book, Gamma et al. [2] describe four essential elements of a design patterns: pattern name, problem, solution, and consequnces. The pattern name is used to describe the problem, solution, and consequnces of the design problem. The problem element tells us when its appropriate to apply the pattern. The solution element describes the design, relationships, responsibilites, and collaborations used in the pattern. Finally, consequences describe the trade-offs invovled with using the pattern.

Based on the work by Gamma et al [2]., design patterns can be seperated into three distinct categories: creational, structural, and behavioral.

- **Creational design patterns -** allow for a more flexible and loosely coupled architecture by abstracting out the process of object instantiation to another part of the system. Essentially, creational patterns are a way of achieving inversion of control in a system. Examples of the creational pattern would be: abstract factory, builder, prototype, and singleton pattern.
- **Structural design patterns -** are concerned with how classes and objects are composed together to create new, larger ones while still being flexible and efficient in their designs. The idea behind these patterns is based on inheritance. Examples include: composite, decorator, flyweight, and the adapter pattern.
- **Behavioral design patterns -** like structural design patterns, behavioral patterns also use the idea of inheritance to distribute behavior between classes. Essentially, behavioral patterns modify the behavior of classes and thus change the way that classes interact with one another. Examples include: observer, and iterator design patterns.

2.2. Reliability

2.3. Usability

Usability is a quality attribute that is used to assess the level of difficulty associated with using a system [3]. Low usability is a good indicator of a system that is difficult to use and is also likely to be rejeted by users. Usability has been defined in multiple standards including ISO 9126, ISO 9241-11, ISO/TR 16982:2202, and ISO 25010:2011. In this paper, when we refer to usability we will be using the definition provided in ISO 25010:2011. ISO 25010:2-11 defines usabilty as "". Usability is not a single measure of a system, but rather, is made up of five subcharacteristics: learnability, usability, memorability, errors, and satisfaction [citation]. These subcharacteristics will be discussed in more detail in section 2.3.1.

For a long time, usability was thought to only apply to the user interface of a system and was something that could simply be added at any point in the development lifecycle. Teams would desgin these systems and then expect to hand them to a usability team where they were then expected to make the product useable [4]. It wasn't until recently that businesses began to undersand the importance

of usability and actually began to take it seriously. Usability is something that needs to be considered at every stage of the development process. Since usability is concerned with the interactions between a system and its users it makes sense that it would play an important role at every step in the process. According to a study by donahue [5], it was found that for every dollar invested in usability you can expect a return of $30.25.

2.3.1. Usability Quality Components. Nielsen defined five quality components that make up usability. According to Nielsen, usability is a quality attribute that assesses how difficult a user interface (UI) is to use. [also defined in ISO 25010:2011] - Section is still a work in progress.

- **Learnability -** Deals with the level of difficulty a user will experience when first interacting with a system to accomplish basic tasks.
- **Efficiency -** Looks at how long it takes a user to complete a certain task once they've gained familiarity with the system.
- **Memorability -** Focuses on a users ability to return to a system after a period of time and regain efficiency using it.
- **Errors -** Looks at the frequency, severity, and recoverability of errors users make while interacting with the system.
- **Satisfaction -** Deals with the amount of fulfillment a user gets from using the system.

2.3.2. Usability Inspection. According to Nielsen et al, usablity inspection is a set of methods where evaluators inspect a user interface (UI) to identify usability issues, severeity of issues, and the overall usability of a design. There are four basic types of usability inspection methods: automatic, empiracally, formal, and informal. Automatic usability inspection is accomplished by running a UIs specifications through a computer program. Empiracal usability inspection invovles using actual users to test the UI. Formal usability inspection is done using models and formulas to calculate a usability score. Finally, informal usability inspection is based on general guidelines and principals as well as the skill and experience of the evaluators testing the UI [6]. Next, will take a look at the actual methods involved in usability inspection.

- **Heuristic evaluation -** is an informal usability inspection method where evaluators, usability experts, independently judge a UI based on its compliance with a set of usability principles known as heuristics. This method relies solely on the skills and experience of the evaluators to find potential issues in the design.
- **Cognitive walkthroughs -** is an formal usability inspection method where evaluators, usually the developers and designers of the software, take turns role playing as users of the software to perform specific tasks while the other members evaluate and document the process for later refinement.
- **Formal usability inspections -** is a formal usability inspection method and is a combination of heuristic evaluation and the cognitive walkthough methods. It is a six step process that involves: assembling a team, assigning roles to each member, distributing appropriate documentation (design, technical, etc), giving out instructions on what is to be done, independent evaluation of the software, and finally, a discussion of issues and concerns that were identified during the process.
- **Pluralistic walkthroughs -** is an informal usability evaluation where a group of individuals, users, developers and designers, and usability experts walk through a scenario together and discuss any issues or concerns they might have.
- **Feature inspections -** is an formal usability inspection method where a set of features used to accomplish a specific task are evaluated based on the given user story.
- **Consistency inspections -** is an informal usability inpsection method where designers from multiple projects inspect an interface based on how similiar it is to their own projects.
- **Standards inspections -** is an informal usability inspection method involving usability experts with knowledge of interface standards use their skills and experience to evaluate a UI based on its compliance with said standards.

2.3.3. Usability Testing Methods. Usability testing is used to determine the level of difficulty associated with using a system. Numerous usability testing methods exist, however, we will only be going over the most commonly used methods in this paper. However, very little research into usability testing exists this section merely aims to provide an overview of the field.

- **Hallway testing -** is a usability testing method where randomly selected individuals, people passing by in the hallway, are brought in to test the usability of a system.
- **Remote usability testing -** is broken down into two categories: synchronous, and asynchronous usability testing. Synchrounous usability testing employs software such as skype, discord, or any other software that allows a user to be observed while using a system from anywhere in the world. Asynchrounous usability testing is where a users actions are automatically tracked and logged by the system. Things like how long a user was on the page, what and where they clicked on the screen, how often they visited a certain page, the previous page that was used to get to the current one, and so-on.
- **Expert review -** involves the use of usability experts who are brought in to judge a system based on their skills and knowledge of the field.
- **Automated expert review -** similiar to expert reviews, however, the testing is done via a program

that has been encoded with an experts knowledge in the form of a set of rules that are used to evaluate the usability of a system.
- **A/B Testing -** also known as split testing, is where we have two identical versions of a system, usually a web site, where one version contains a slight modification with the intent of studying whether this variation will impact a users behavior.
- **Think aloud Protocol -** is a process where a user or group of users are brought in to use a system and during that time are instructed to share their thought process out loud while doing so.

3. Systematic Mapping Approach

A systematic mapping study (SMS) is used to provide a broad overview of a particular research area by identifying previous primary studies whose results are then categorized to create a visual summary, the map. In this paper, the systematic approach used is defined by Kitchenham et al. [7] and later refined by Peterson et al [8]. There are five steps to this process: definition of research questions, searching for relevant papers, screening of papers, and keywording of abstracts. We chose to use this approach because it is well-defined, allows for more general conclusions, and in doing so helps to reduce bias. We decided to go with an SMS instead of a systematic literature review (SLR) for two reasons. The first reason is because we wanted a broader overview of the field and secondly, we were more interested in general trends rather than the in-depth analysis that an SLR provides. Our goal with this paper is to identify what research has already been done with regards to design patterns and their affect on usability and reliability to determine where new or better research can take place.

3.1. Research Questions

- **RQ1:** What design pattern types do research study when considering usability and reliabiity?
- **RQ2:** What recomendations have been made regarding application of design patterns and in the context of usability and reliability?
- **RQ3:** How is the impact on usability and reliability from design patterns currently evaluated?
- **RQ4:** What type of projects or domains were studied?
- **RQ5:** Where are papers concerning design patterns and reliability and usability published?
- **RQ6:** What types of studies are conducted regarding design pattern impact on usability and reliability?
- **RQ7:** In what phase of development do proposed results apply?
- **RQ8:** What tools are utilized for the research and to which languages do they apply?

3.2. Data Sources

The data sources used in this paper included: IEEEXplore, ACM Digital Library, SpringerLink, Web of Science, and the Science Direct database. The search results were as followed: IEEEXplore returned 1410 results, ACM Digital Library returned 886 results, SpringLink returned 4619 results, Web of Science returned 2314 results, and Science direct returned 8219 results. All search results were also limited to the years between 2009-2019.

We choose to use the ACM Digital Library along with IEEEXplore for their large academic databases that largely focuses on computer science as well as the fact that together they cover articles, books, conference papers, technical standards, and journal papers. We also decided to go with SpringerLink, Web of Science, and Science Direct because of there wide coverage of software engineering journals as well as the fact that using multiple databases helps to eliminate bias in our results.

3.3. Search Queries

The search string used in this paper was created from keywords identfied based on our reseach questions and included: usability, reliability, design pattern, and pattern as well as common synonyms and subcharacteristics of usability and reliability. The specific search string used is: (usability OR useable OR learnability OR learnable OR efficiency OR satisfaction OR memorability OR memorization) AND (reliability OR reliable OR integrity OR maturity OR compliance OR “fault tolerance”) AND (“design pattern” OR pattern). We aslo made use of boolean operators such as OR and AND as well as operator precedence through the use of ().

3.4. Paper Selection

The first step in our selection process was to manually search through the electronic databases listed in sections 3.2. After this initial search we had gathered x papers for possible inclusion or exclusion. The next step in our process was to manually read through each papers title, abstract, and keywords in order to determine a papers elgigibility. If more information was still needed after this to determine a papers eligibility the full paper was read. After determining a papers eligiblity we used the snowballing process listed in section 3.8 to discover additional papers that might have been missed earlier in our process. After identifying any missed papers, we then repeated the above process to determine their eligibility.

Of the x papers that were initially included, only y were actually included after running them through the inclusion and exclusion criteria. Of the y papers that were eligible for inclusion we identified z additonal papers after performing our snowballing approach. Of the additional z papers that were found only w papers were further included.

3.5. Inclusion Criteria

In our Systematic Mapping Study only papers published between 2009 and 2019 were available for inclusion.

- Only papers whose main focus is on design patterns and their effect on usability or reliability were available for inclusion.
- For papers where multiple iterations of the same paper exist only the latest version was eligible for inclusion.
- Papers that included either reliability or usability and design pattern or pattern as keywords to identify the paper topics.

3.6. Exclusion Criteria

- Papers not written in the English language.
- Papers whose title or abstracts did not contain the selected keyword phrases.
- Papers which discussed the impacts of design patterns on quality attributes exclusing reliability or usability or their sub-characteristics.

3.7. Quality Criteria

3.8. Snowballing Approach

The snowballing process used in this paper was described by Wohlin [9] and defines two types of snowballing procedures: forward and backward snowballing. The first step in the snowballing process is to obtain the initial set of papers, known as the starting set, that will be used to begin the snowballing process.

Backwards snowballing is an iterative process where the references for each paper in the set are used to identify new papers that might have been missed during the initial gathering. Each paper is then ran through the same exclusion and inclusion criteria as the initial starting set to determine its eligibility. This process is then repeated for each reference until all references have been exhausted for each paper.

Forward snowballing, like backward snowballing, is an iterative process where new papers are identified by searching for papers that reference the ones already being examined. These papers are then ran through the same inclusion and exclusion criteria to determine their eligibility. Usually, forward and backward snowballing involve reading only the title or abstract to determine a papers eligibility, however, is both of these should prove inadequate the entire paper is then read.

Throughout the process there may false positives (papers were included that shouldn't have been) that require all papers identified from this one to be removed and the snowballing process to start over. It's important to note that only the papers identified from the current one being examined are to be removed.

Once all of the papers in the start set have been examined the process is repeated for each new paper that identified from the previous round of snowballing. This process will continue until no new papers are identified. It is at this point that Wohlin recommends contacting the authors for each paper that was included to ensure that no other papers exist. If still no new papers are identified at this point the snowballing process is complete.

4. Threats to Validity

In the software engineering field there are four commonly looked at types of validity that need to be considered: Internal Validity, External Validity, Conclusion Validity, and Construct Validity. Each of these will be further discussed in their corresponding subsections: 4.1, 4.2, 4.3, and 4.4.

4.1. Internal Validity

Internal validity deals with the degree to which a study is able to rule out alternative explanations in order to show that the evidence presented throughout the paper is adequate in proving that a causal relationship exists between two variables. Therefore, since this study makes no such argument, there are no internal threats to validity.

4.2. External Validity

External validity is concerned with the degree to which the results of a study can be further generalized to other populations. Therefore, since this paper merely aims to provide an overview of the field by looking at previous research there are no threats to external validity.

4.3. Conclusion Validity

Conclusion validity is concerned with whether the operations of a study are able to be replicated with the same results. Therefore, since an SMS aims to provide a broad overview of a specific field through the collection of previous primary studies we do have threats to conclusion validity.

- **Search Terms -** the search string used in the paper may not contain all appropriate keywords, synonyms, or alternate keywords that would otherwise ensure all appropriate papers could be located. However, in an attempt to minimize this issue we included all synonyms and alternate keywords that are in common use when discussing usability and reliability as defined in ISO 25010:2011 which is the standard that deals with these ideas.
- **Search Methods -** the search methods used in this paper to locate relevant papers could possibly lead to missing relevant papers due to the fact that only online databases were queried.
- **Limited Access -** since we were unable to query every relevant database or seach engine in finding appropriate studies it's possible that not every relevant paper was identified.

4.4. Construct Validity

Construct validity deals with whether a test actually mesaures what it claims to measure. Therefore, since this paper is merely an overview of a field and doesn't contain any tests there are no threats to construct validity.

4.5. Reliability

5. Results

6. Conclusion

7. Recommendations

References

[1] C. Alexander, "A pattern language : towns, buildings, construction / Chistopher Alexander... [et al.]." 1977.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns: Abstraction and Reuse of Object-Oriented Design," in *Software Pioneers: Contributions to Software Engineering*, M. Broy and E. Denert, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 701–717.

[3] N. Juristo, A. M. Moreno, and M.-I. Sanchez-Segura, "Analysing the impact of usability on software design," *Journal of Systems and Software*, vol. 80, no. 9, pp. 1506–1516, Sep. 2007.

[4] X. Ferre, N. Juristo, H. Windl, and L. Constantine, "Usability basics for software developers," *IEEE Software*, vol. 18, no. 1, pp. 22–29, Jan. 2001.

[5] G. M. Donahue, "Usability and the bottom line," *IEEE Software*, vol. 18, no. 1, pp. 31–37, Jan. 2001.

[6] R. Mack and F. Montaniz, "Usability Inspection Methods," J. Nielsen and R. L. Mack, Eds. New York, NY, USA: John Wiley & Sons, Inc., 1994, pp. 295–339.

[7] B. A. Kitchenham, D. Budgen, and O. Pearl Brereton, "Using mapping studies as the basis for further research – A participant-observer case study," *Information and Software Technology*, vol. 53, no. 6, pp. 638–651, Jun. 2011.

[8] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic Mapping Studies in Software Engineering," in *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, Italy, 2008, pp. 68–77.

[9] C. Wohlin, "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, London, England, United Kingdom, 2014, pp. 38:1–38:10.