

# Software Architecture

## When to use Alternative Formats for Communicating Ideas from Stakeholders to Software Developers

Johnny King and John L Williams Jr<sup>\*</sup>  
UCCS  
Colorado Springs, CO, USA  
jwilli11@uccs.edu  
jking4@uccs.edu

### ABSTRACT

Documenting software in a way that avoids verbosity, ambiguity, and confusion is a goal that all software architects should aspire to. The authors will identify and compare several formats that may be used to document software. The authors will list some of the pros and cons of each format, helping the readers to make an informed choice of the format to use for their own software projects.

### Keywords

Software, Architecture, Documentation

### 1. INTRODUCTION

### 2. BACKGROUND

Software architects are responsible for communicating ideas from stakeholders to software coders with the goal that the software coders accurately and completely implement those ideas into a working software product.

With an estimated software project failure rate of over 40%, we need to investigate possible reasons for such a high percentage of failures. One such reason may be a failure to communicate effectively. Communication is required between stakeholders and requirements engineers, between requirements engineers and software architects, and finally between software architects and software coders and testers [4]. The authors will focus their investigation on communication failures between software architects and software coders and testers. This approach will assume that documents created prior to this communication phase are both complete and correct.

Software architects have several formats to choose from when it comes to creating documentation for software. We will assume that most software projects are too complex to support an all oral format using natural language to commu-

nicate the ideas, so we will assume that at least a written format using natural language is required. Other formats that are more formal and structured in nature include UML diagrams and mathematical models.

### 3. RELATED WORK

Miles and Hamilton describe why and how to use UML (Universal Modeling Language) for documenting software. They list the following cases for using UML [2].

1. Requirements using Use Case Diagrams
2. System Workflows using Activity Diagrams
3. A Systems Logical Structure using Class Diagrams
4. Ordered Interactions using Sequence Diagrams
5. Interaction Links using Communication Diagrams
6. Interaction Timing using Timing Diagrams
7. Interaction Picture using Interaction Overview Diagrams
8. Internal Class Structure using Composite Structures
9. System Parts using Component Diagrams
10. Organize Your Model using Packages
11. Object State using State Machine Diagrams
12. The Deployed System using Deployment Diagrams

Wieggers and Beatty discuss the importance of using both textual and visual formats to capture the full intentions of the intended system. Their book describes the following visual requirements models [5].

1. Data flow diagrams (DFDs)
2. Process flow diagrams
3. State-transition diagrams and state tables
4. Dialog maps
5. Decision tables and decision trees

---

<sup>\*</sup>King and Williams are graduate students at UCCS

6. Event-response tables
7. Feature trees (Ch 5)
8. Use case diagrams (Ch 8)
9. Activity diagrams (Ch 8)
10. Entity-relationship diagrams (Ch 13)

Wiegiers and Beatty also briefly discuss the use of visual models in agile driven projects [5].

The American Institute of Architects discuss the need for standard content and graphics to be used by Construction Architects to document their projects (Appendix E). [3]. Although software architecture and construction architecture are technically different fields, they share many common goals of turning stakeholder ideas into a real products.

Ingeno discusses methods for documenting and reviewing software architectures [1] In particular, he discusses

1. Uses of software architecture documentation
2. Creating architecture descriptions (ADs), including architecture views
3. Using UML to document software architecture
4. Reviewing software architecture documents

#### 4. PROPOSED APPROACH

The authors will review some of the literature written by leading experts on the topic of software requirements documentation content and format. A table will be generated listing the different formats proposed and the pros and cons of each format.

#### 5. EXPERIMENTAL RESULTS

#### 6. DISCUSSION

#### 7. THREATS TO VALIDITY

#### 8. CONCLUSIONS AND FUTURE WORK

Our conclusion is ...

#### 9. ACKNOWLEDGMENTS

We would like to thank Dr. Kristen Walcott, Dr. Armin Moin and the Software Engineering Department at UCCS for helping us gain a deeper understanding of the challenges and opportunities in software architecture.

#### 10. NOTES AND BRAINSTORMING - DELETE BEFORE SUBMITTING FINAL PAPER

1. Understanding Research Problems
  - (a) Topic: We are evaluating the usefulness of presenting ideas in different formats
  - (b) Question: because we want to find out if certain formats are more useful than others in communicating ideas from stakeholders to software developers

- (c) Significance: so that we can help others choose a document format that will result in successful communication of ideas from stakeholders to software developers.

#### 2. Levels of Idea Formats from Abstract to Concrete

- (a) Mental
- (b) Natural Language
  - i. Oral
  - ii. Written
- (c) Formal Languages
  - i. Mathematical
  - ii. Models (UML)
- (d) Source Code

#### 11. REFERENCES

- [1] J. Ingeno. *Software Architect's Handbook*. Packt Publishing, 2018.
- [2] R. Miles and K. Hamilton. *Learning UML 2.0*. O'Reilly, 2006.
- [3] A. I. of Architects. *Architectural Graphic Standards, 11th Edition*. John Wiley and Sons, Inc., 2007.
- [4] A. van Lamsweerde. *Requirements Engineering*. Wiley, 2010.
- [5] K. Wiegiers and J. Beatty. *Software Requirements*. Microsoft Press, 2013.