The Design Space of Software Development Methodologies

Kadie Clancy, CS2310 Term Project

I. INTRODUCTION

The success of a software development project depends on the underlying framework used to plan and structure the process of development tasks. Presently, there are a multitude of software development methodologies in which to choose. The selection of an appropriate methodology for a specific project may not always be a straightforward process and, further, the methodology that provides the most benefit to the project may be overlooked. In an effort to organize the multitude of software development methodologies in a coherent and visual manner, I created the design space of software development methodologies in which I seek to comprehend different methodologies plotted as points. Constructing such a design space allows for a fair comparison of the various methodologies based on the underlying features and characteristics, rather than lists of advantages and disadvantages for each method, which depend more on the application to a specific project.

II. RELATED WORK

Most engineering disciplines organize designs or concepts as some sort of abstraction, like taxonomies or (less popularly) design spaces. Previous work has been done to apply the principles of design space analysis to a variety of projects, demonstrating the usefulness of this type of abstraction. A design space for bug fixes and how developers navigate it was constructed in [1] and [4]. The authors of [2] generated a morphological design space of computer input devices. Zwicky [3] used a similar method to generate the design space of jet engines. The design spaces introduced in these works resulted in properties of the design space as well as insight into spaces where novel designs could be produced.

III. DESCRIPTION OF THE DESIGN SPACE

I employed a generate-and-test paradigm to decide upon the dimensions of the design space. These dimensions correspond to features or characteristics that the methodologies share and may be differentiated based upon. To decided upon these dimensions, I conduced a review of popularly used development methodologies. The design space is illustrated in Figure 1.

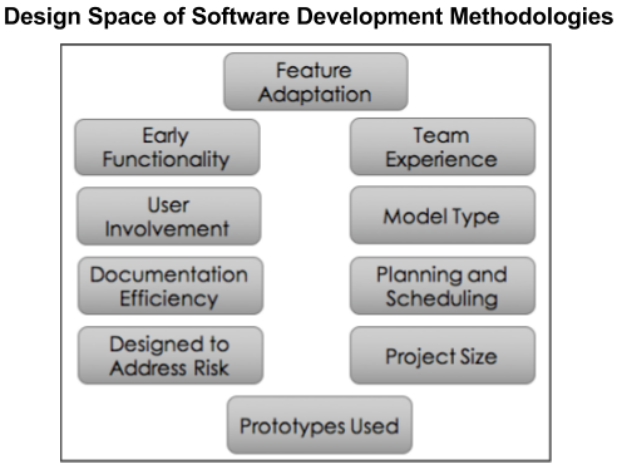


Fig. 1. Characterizing the design space of software development methodologies

Each of these dimensions can take on certain values in either a continuous or binary manner. The possible values of each dimension are elaborated on below and a summary of possible dimension values is illustrated in Figure 2.

A. Feature Adaptation

This binary dimension describes if the methodology allows for new feature integration at intermediary stages of development. For example, the waterfall methodology is not flexible to the addition of new features after the initial requirements phase. As this process is sequential, developers cannot return to a previous phase without scratching the entire project and starting from the beginning. Conversely, the incremental methodology would be plotted on the opposite end of the dimension as this process allows developers to iteratively introduce features to the system based on customer feedback.

B. Early Functionality

Early Functionality is also a binary dimension in which a methodology can either produce a usable

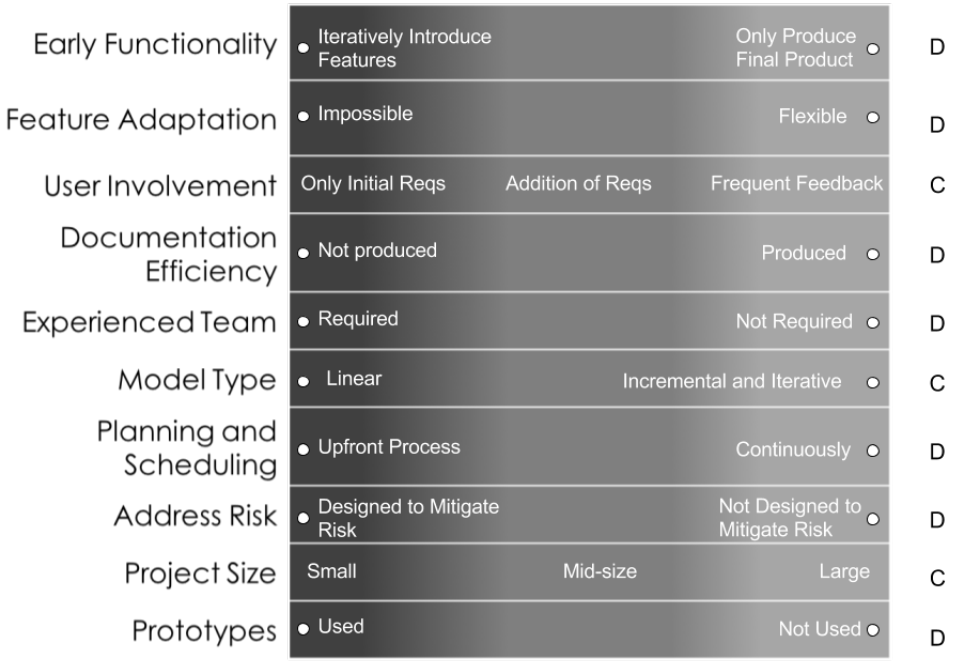


Fig. 2. Possible dimension values; C represents a continuously valued dimension and D represents a discretely valued dimension

system early in the development process or produce only a final functioning system at the completion of the project.

C. User Involvement

This dimension describes the degree to which client feedback is required. The values composing this dimension are continuous in nature. At one end of the dimension, client involvement is limited to producing the initial list of requirements. At the other end of the dimension, the client provides feedback at each step during the development process. At the middle point of this dimension, the client occasionally produces more requirements to be added to the overall system functionality.

D. Documentation Efficiency

This binary dimension describes if the methodology produces efficient documentation or not. For example, the Agile approach does not produce efficient business documentation. Rational Unified Process (RUP) methodology, however, places emphasis on precise documentation.

E. Designed to Address Risk

Some development methodologies were specifically designed to address and mitigate risks to the project. One end of this binary dimension expresses that risks are addressed in some way by the methodology, the other side of the dimension expresses that risks are not addressed.

F. Team Experience

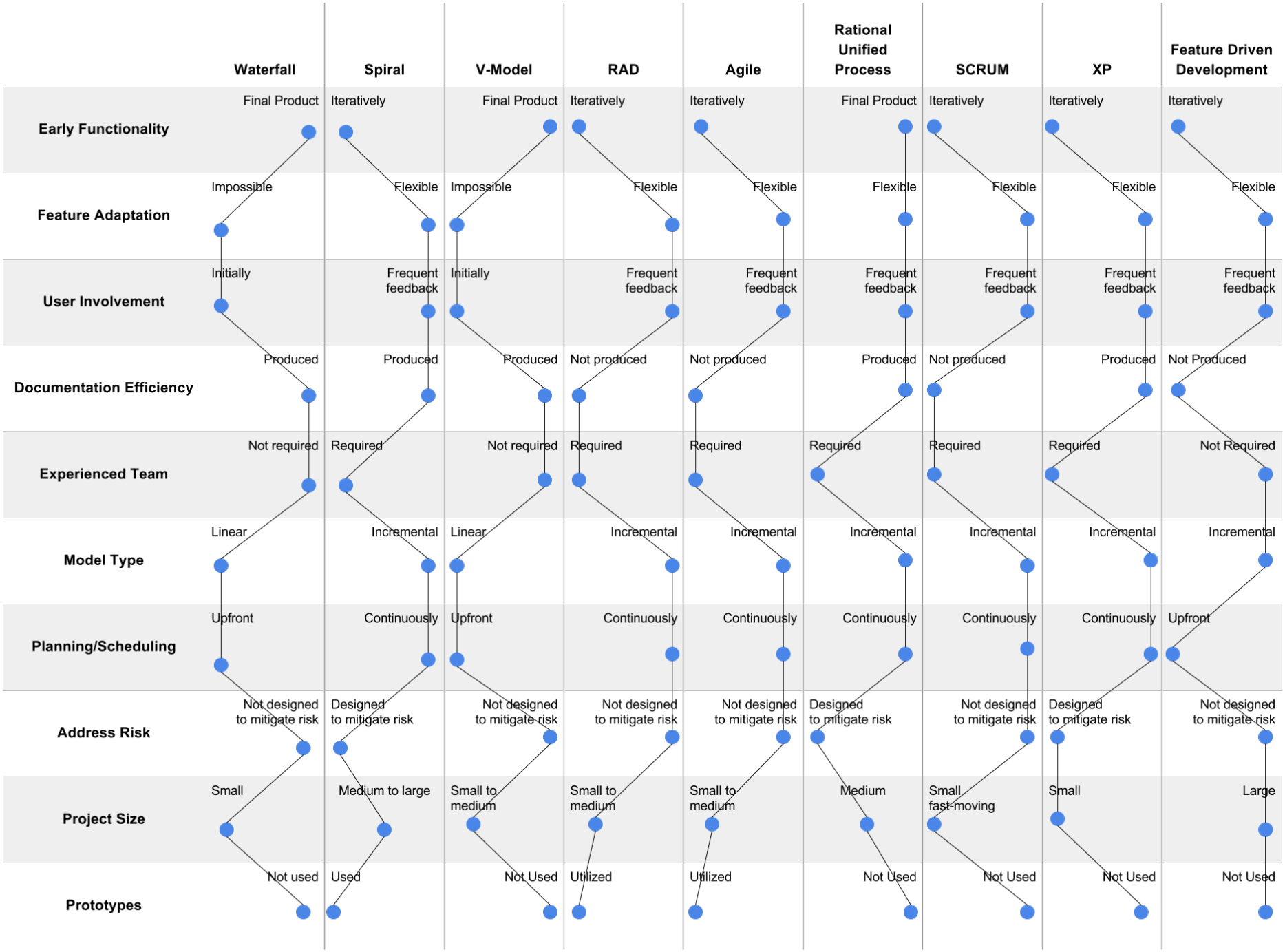
Team Experience is a binary dimension in which a methodology can either require some form of specialized team member experience, or may be suitable for novice teams. Type of experience may be diverse. For example, the risk analysis component of the Spiral methodology requires highly specific expertise. Scrum requires a different type of experience in the form of an experienced designated Scrum master.

G. Model Type

The values of this dimension consist of discrete values of linear or incremental and iterative. For example, the waterfall methodology is a linear process in which each phase follows linearly from the previous and the revisiting of past phases is not permitted. On the other hand, Feature Driven Development methodology follows an iterative approach because it is based on separate development of features.

H. Planning and Scheduling

This binary dimension describes whether the planning and scheduling of development tasks and phases is done once at the beginning of the project, or if these


Waterfall
Spiral
V-Model
RAD
Agile
Rational Unified Process
SCRUM
XP
Feature Driven Development
Early Functionality
Final Product
Iteratively
Final Product
Iteratively
Iteratively
Final Product
Iteratively
Iteratively
Iteratively
Feature Adaptation
Impossible
Flexible
Impossible
Flexible
Flexible
Flexible
Flexible
Flexible
Flexible
User Involvement
Initially
Frequent feedback
Initially
Frequent feedback
Frequent feedback
Frequent feedback
Frequent feedback
Frequent feedback
Frequent feedback
Documentation Efficiency
Produced
Produced
Produced
Not produced
Not produced
Produced
Not produced
Produced
Not Produced
Experienced Team
Not required
Required
Not required
Required
Required
Required
Required
Required
Not Required
Model Type
Linear
Incremental
Linear
Incremental
Incremental
Incremental
Incremental
Incremental
Incremental
Planning/Scheduling
Upfront
Continuously
Upfront
Continuously
Continuously
Continuously
Continuously
Continuously
Upfront
Address Risk
Not designed to mitigate risk
Designed to mitigate risk
Not designed to mitigate risk
Not designed to mitigate risk
Not designed to mitigate risk
Designed to mitigate risk
Not designed to mitigate risk
Designed to mitigate risk
Not designed to mitigate risk
Project Size
Small
Medium to large
Small to medium
Small to medium
Small to medium
Medium
Small fast-moving
Small
Large
Prototypes
Not used
Used
Not Used
Utilized
Utilized
Not Used
Not Used
Not Used
Not Used

tasks are completed sporadically throughout the entire process.

I. Project Size

This continuous dimension describes the ability of a methodology to scale to larger project sizes. For example, the V-Model works best for small-to-medium sized projects, but doesn't scale well to larger projects.

J. Prototypes Used

Prototypes Used is another a binary dimension in which a methodology can either produce prototypes in the design and development of systems or does not explicitly make use of prototyping. For example, the Spiral methodology makes use of prototypes after the risk analysis phase of each iteration.

IV. Points in the Design Space

Using the dimensions I selected to characterize the design space, I plotted a number of different methodologies to test the descriptiveness of the space. This served as a singular visualization of the array of methodologies and also as a test to make sure that different methodologies can be well distinguished from one another. In other words, this serves as a check to ensure that the constructed space is mutually exclusive. This visualization is included on page 3.

A. Explanation of Waterfall

I will use the waterfall model as an example to explain the plotting of a methodology in the design space. The waterfall model is considered the classic style of software development. This model proceeds in a linear sequential flow, meaning that any phase in the development process begins only after the earlier phase is completed. Therefore, this methodology has a value of "linear" in the model type dimension. This development approach does not define the process to go back to the previous phase to handle changes in requirements. Therefore, the method has a value of "impossible" on the feature adaptation dimension. The waterfall method places emphasis on documentation efficiency and is plotted as such in that dimension.

The Waterfall model is simple and easy to understand and is beneficial for the beginner or novice developer and is such plotted as "not needed" in the experienced team dimension. There is no possibility to produce any working software until it reaches the last stage of the cycle, and therefore has a value of "only final product produced" on the early functionality dimension. This model is good for a small project but not ideally suitable for long and ongoing projects, and is plotted as such in the project size dimension.

V. Application to Patterns

To put this project in the context of patterns as discussed in class, I defined the selection of a software methodology for a specific problem in terms of a problem, a context and a solution. The problem is the problem the software is being designed to address, i.e. the purpose of creating a new system. The context is external factors like budget, client, project type, deadlines, team, etc. The solution is the methodology chosen to be most appropriate out of all others plotted in the space. The most appropriate method is decided upon based on how the components of the context constrain certain dimensions of the design space. For example, if client time is limited (something that will be defined by the context), the method chosen must be restrained to "only initial reqs" on the user involvement dimension. Further restraints may be defined by other facets of the context, leading to the most suitable methodology.

VI. Conclusions

In this paper, I have created a new design space to organize and visualize software development methodologies. I then plotted methodologies as points on this space to test that it is mutually exclusive, even when plotting similar methodologies. This space, while not meant to be completely exhaustive, can act as a starting point to effectively distinguish between the vast number of available methodologies. It can also be used to visualize similar groups, and to note areas for the creation of new methodologies.

A visualization of this type has a number of possible applications. Notably, project managers may use a space of this type as a tool in the selection of an appropriate methodology for specific projects as all methods may be visualized in one place.

References

[1] E. Murphy-Hill, T. Zimmermann, C. Bird, and N. Nagappan, The design of bug fixes, in Proc. Int. Conf. Softw. Eng., 2013, pp. 332341.

[2] S. Card, J. MacKinlay, and G. Robertson. A morphological analysis of the design space of input devices. ACM Transactions on Information Systems, 9:99122, 1991.

[3] Zwicky, F. The morphological approach to discovery, invention, research, and construction. In New Methods of Thought and Procedure, F. Zwicky and A. G. Wilson, Eds. Springer-Verlag, New York, 1967, 273-297.

[4] Murphy-Hill, E., Zimmermann, T., Bird, C., and Nagappan, N., 2015. The Design Space of Bug Fixes and How Developers Navigate It. IEEE Transactions on Software Engineering 41, 1, 65-81.

[5] J. Wang, Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style Submitted for publication), IEEE J. Quantum Electron., submitted for publication.