A Proposal of Practices, Processes and Models that Enable Innovation Potential

Dan Matheson
RWTH Aachen University
Fort Collins, CO, USA
dan.matheson.co@gmail.com

***Abstract*—Innovation is neither predictable nor guaranteed within a solution development project. Novel ideas can appear during any level of work, whether high-level design or in code algorithms. As design innovation is a creative action, this paper proposes a vision of related practices, processes and modeling ideas that enable innovation to potentially happen. The practices are mostly mental techniques for a designer to use which help to maintain an open mind to solution possibilities. The process approaches are based on and support progress within a flexible, incremental and iterative methodology that can respond to new ideas. The modeling technique ideas strive to support the clear communication of understanding and the evolution of the design through multiple alternatives. The proposals suggested are rooted in some successes from actual industry projects over the years, but can benefit from further research and empirical evaluation.**

Keywords-software engineering, modeling, innovation, development process

I. INTRODUCTION

There are many factors that contribute to establishing an environment where innovation flourishes. In [1] the following statement about engineering is made: "And though engineering is the art of rearranging the materials and forces of nature, the immutable laws of nature are forever constraining the engineer as to how those rearrangements can or cannot be made." Software engineering differs because it is, at least outside of the hardware, independent of the physical laws of nature. However, both software and physical materials engineers have the same goal of creating a good solution for their fellow humans.

Are there corresponding "laws of nature" for software engineering design that enable good design and minimize poor design? There are multiple essays and examples in [2] about bringing design to software and in [3] the idea is repeated that design is a messy creative process. Messy is a situation that the business manager of the project tries to avoid, rather predictable project execution is desired. A contradictory project situation is created from the two competing goals.

Is it possible to develop a software project approach that constrains the messiness while enabling opportunities to develop innovative and useful solutions? This paper lays out a vision for answering this question. The approach consists of three related components. The first component is a set of practices called Software Engineering First Principle (SEFP) guidelines that act as mental checks and balances on the design alternatives. A project process approach is the second component and attempts to balance the messiness needed for innovation with the reasonable progress demanded by business. Since engineers must communicate their understanding of the solution goals and needs along with designs, the third component focuses on modeling. A holistic consideration of the three components is needed for a good design environment. The suggestions that follow have been gathered and used over many years in industry projects.

The paper is organized as follows: Section II lists several SEFP practices gathered from the literature and developed in real world projects, Section III discusses characteristics of and ideas for a project process approach that enable innovation while allowing effective project management, in Section IV modeling techniques that convey designs quickly and clearly are reviewed and Section V contains a list of possible future work and some conclusions.

II. PRACTICE PROPOSALS

Practice proposals are mental or thinking aids to maintaining the thought flexibility and discipline needed to allow for innovation. A practice proposal is referred to as Software Engineering First Principle (SEFP). The analogy is to physics first principles such as the idea of inertia or the absolute speed of light. The benefits of the SEFPs are to have an evaluation framework outside of the project. Through an application of an SEFP to the design alternative, the designer can perhaps judge if the alternative can be improved or a novel alternative is possible.

The next few sub-sections list some of the SEFPs that have proven to be useful over several projects. Several of these are reiterations from other people, which are sometimes forgotten in the heat of the project, but remain valid and are strengthened when combined together.

A. Essential and Accidental Complexity

The practice of identifying the essential and accidental complexities in the problem and the design alternatives is from Frederick Brooks [4]. This SEFP forces the designer or design team to question the current understanding of the problem to be solved. Expressing the essence of a design

978-1-5386-0400-7/17 $31.00 © 2017 IEEE
DOI 10.1109/DISE.2017.8

alternative along with the accidental complexity added by that alternative can help with alternative comparisons.

Some examples of the questions to be asked to separate the essential and accidental complexities are:

- What is the problem to be solved for the business or the primary stakeholders and why?
- Does this problem need to be solved now?
- Who are the primary stakeholders?
- What value does the solution provide for the business and what is the cost?
- Will the business be disrupted by this improvement? How much?
- Can the design alternative be simplified?
- What supporting engineering or technology components are needed by a design alternative?

B. Conceptual Integrity

The SEFP of *Conceptual Integrity* from Frederick Brooks [4] is closely related to the understanding of the essence of a solution. As Brooks puts it:

"The hardest single part of building a software system is deciding precisely what to build."

"I will contend that conceptual integrity is the most important consideration in system design."

The concept of a solution to a problem is the distillation of the essence of what should be built. A properly expressed concept will clearly communicate the essence of the project, problem and solution, to all concerned people. More importantly, it provides a test metric for requirements and design alternatives through the question: "Does the requirement or design align with and directly support the concept?" In practice the concept alignment question can help purge false requirements, for example the "wouldn't it be cool to …" ones from brainstorming. For design alternatives the concept alignment question can raise questions about technology choices, whether the choice makes engineering sense or whether it is current fashion.

C. Optimal Performance

The SEFP for optimal performance is *the fewest operations on the fewest pieces of data*. This approach to thinking applies from the gathering of requirements to the design of code methods. It helps in discovering the essence.

In practice some of the optimal performance SEFP questions are:

- Is this feature or this data essential to achieving the solution concept or not?
- Is the design of the feature or data as efficient as possible?
- Does the collection of features work in an optimal manner or are translations involved?
- Has the design been translated into good code?
- Are the best coding practices being used in relation to the life of the solution?

D. Change Language or Active Thesaurus

A mantra that evolved from practice that helps to discover the essence or express the concept is *Change your words to change your perception to open innovation opportunities*. Almost every domain has its own set of vocabulary. In some cases the domain vocabulary must be used to ensure clear communication. However, when a designer uses alternative expressions and synonyms, opportunities are opened to think in new directions. This is the mental equivalent of picking an object up and turning it around in your hands to view all sides.

A co-approach to new words is the communication richness provided by the graphical expression of the design idea or concept [5]. It is possible to articulate the solution concept graphically [6]. User Requirements Notation (URN) [7] and *i** [8] can be used to express the essence of the business value. Many aspects of a design alternative can be conveyed graphically in Unified Modeling Language (UML) [9] or a modified sub-set of UML.

E. Modules

David Parnas [10] makes the point that a decomposition of the solution based on information or decision hiding often provides the better level of development management, product flexibility and comprehensibility. This is one of the earliest statements of abstraction and the power of abstraction. Models can be an abstraction of the solution requirements and the design of the solution.

One of the goals he values is that modules should be capable of being assembled, reassembled and replaced within the completed system. His technique for this is expressed in the following quote, "...module is considered to be a responsibility assignment."

From a requirements perspective this translates into specifying "what" should happen or "what" is needed from a business view, while delaying the specification of the "how" it is accomplished until the design phase.

1) Coupling and Cohesion

When attempting to satisfy the goals of assembling modules, then the design of how things are connected and the consequences of the connections are important [11]. The term cohesion expresses the singularity of purpose of the module. A module with the best cohesion stands on its own and does one thing. Coupling is used to indicate the closeness of interaction between two modules with no interaction the best.

F. Patterns

Patterns can be used to assist in achieving the necessary completeness while deferring details and as such they can be used to maintain the appropriate level of abstraction. Patterns work well with module design. The lesson for innovation from [12] is that there are levels of patterns that can be used in designing a solution. For software there are collections of design patterns available [13] [14]. In addition there are the patterns conveyed by the acronyms, such as CRUD (Create, Read, Update, Delete) and ACID (Atomicity, Consistency, Isolation, Durability) which can be used to evaluate designs.

While there are lists of well-known patterns, often an aspect of innovation is the discovery of new or domain

specific patterns. Design innovation can be stimulated by the following questions:

- Can an existing pattern simplify or improve the design?
- Is there a new pattern emerging from the design?

III. Process Methodology Proposals

The process used to control and manage the project can either enable or suppress innovation. What are the characteristics that could enable innovation during the project?

A. Incremental and Iterative

An incremental and iterative or agile approach has several features that can enable innovation to occur. The primary feature is the short cycles of work, which allows for change of direction. Not just during an iteration, but especially at the end there is an ideal opportunity to review the work using the SEFPs. Whether the iteration is used for requirements gathering, designing or implementation, review offers an opportunity for innovation. In [3] the cycle of evaluation of design alternatives and back to earlier decisions is clearly explained.

When the increments of work are kept small, then the design momentum is kept small. The main component to the momentum is the personal investment by the team members in their work. Design momentum interferes with innovation, because people get invested in a direction and resist admission that the earlier choices were poor [15]. Project process must not be confused with design momentum.

B. Abstraction Guidance

The project process should move the project forward at a reasonable pace. However, "reasonable pace" is a subjective measure that varies between the project team members and project stakeholders. Reasonable pace also varies based on the project maturity, whether at the beginning with many unknowns or just before solution deployment. One aspect of reasonable pace is the delivering of artifacts that show progress from broad general questions to specific design details. A good project process will provide guidance from an abstract solution design through a concrete implementation.

The Open Distributed Processing - Reference Model, ISO-10746, (RM-ODP) [16] is one example of a framework that provides abstraction guidance. The desired feature is an approach that helps to order the questions to be addressed by importance. The answers are either a requirement to be achieved or a design decision to be made. The RM-ODP standard uses viewpoints to order the abstraction level and therefore the question importance. The path through the viewpoints, from the Enterprise Viewpoint (highest abstraction level) to the Technology Viewpoint (lowest abstraction level), is not a straight line, but can be done in an iterative and incremental manner. The work in an iteration can cause a higher-level design decision to be revisited as new constraints are discovered [3].

C. Process Artifacts

During the project process many decisions will be made. Reports of progress will be created and communicated. In a medium to large project the people needing this information will be distributed geographically and with realistically accounting for people turnover there is temporal distribution. As the Agile community [15] points out creating documentation is accidental complexity [4]. While the effort in creating the documentation of decisions and reports should be minimized, it does have some value for innovation.

The act of documenting design decisions, constraints and technology choices does provide a review of those things both for the documenter and outside experts, especially if unsuccessful decisions are included. Innovation does not occur just in the initial solution creation work, but also in maintenance and extension work, which is a temporal distribution. The design process documentation is required in many regulated domains, such as life science.

IV. Modeling Proposals enabling Innovation

The modeling of requirements and designs is standard engineering practice [1] [6] [17] [18]. While there are many types of modeling available such as natural language and mathematics, this section will concentrate on the advantages of graphical or visual models for innovation.

A. Visual Modeling with Sketching

Mechanical engineers, civil engineers and building architects use graphical models or drawings as the primary documentation artifact for their solutions. The main reason for this is the communication density and clarity over text [5] [19]. The stories of sketching the new idea on a napkin during a lunch with colleagues abound. The lunch time sketch is a visual model used to document and communicate the idea.

Creating graphical models of the requirements and the design alternatives to those requirements is an application of change language SEFP. The graphical expression allows for different perspectives to be generated. The graphical models are often faster to generate, especially in a sketching mode.

Fast exploration of new ideas aids innovation. It is rare that an innovative idea is fully formed on its first expression. Rather, the innovation happens in multiple refinement steps as the idea is evaluated, discussed with colleagues and compared to alternatives. The faster these steps can be accomplished, the sooner the idea is fully formed and sometimes the process is known as "fail fast". Equally important is keeping a record of the failures or weaknesses so that the effort is not duplicated.

B. Innovation in Modeling

There are times when a visual modeling technique does not exist. This sometimes happens when a new domain or problem is attached. For the new domain the creation of a new visual modeling capability can be used to express the acquired insights into the essence of the problem. Expressing a fundamental insight into the essence of a problem is a type of innovation, which can have great value.

Innovation in modeling is needed as well as innovation in the solution.

The various design alternatives will have multiple innovation models attached. This raises the issue of managing the alternatives, models and relationships. In [20] a proposal for the management issue is defined.

C. Rigorous Visual Models

As the project proceeds, the need for more rigorous modeling appears. The amount of rigor needed varies according to the cost and the benefit. A benefit of a more rigorous model is the ability to test or analyze for weaknesses, which is an opportunity for innovation. Rigorous modeling at more detailed design level offers the opportunity for some implementation (code) generation. When the human developer is relieved of implementation effort, then more time can be spent on innovative design.

As an example, the UML and the URN graphical modeling capabilities can be used to express new ideas within their domains with rigor at different levels of abstraction.

D. Overview Models and Drawings

Other engineering disciplines use overview drawings to communicate the purpose or makeup of the solution. In mechanical engineering a drawing will show multiple views of the part, such as top, side, front, cross section and often a perspective view. Architects produce multiple drawings of the house like an exterior in a landscape setting, floor plan layouts and construction detail blueprints needed for the building permit. For both mechanical engineers and architects the various drawings can be generated from 3-D Computer Aided Design (CAD) tools. The 3-D CAD model is the authoritative definition with the generated drawings used only for communication, which is a state software engineering has yet to reach.

Software engineering would benefit from a similar modeling approach. Currently the majority of UML modeling tools handle the different diagram creation activities as independent actions. Just as other engineering disciplines have a comprehensive overview drawing, a Solution Overview Drawing (SOD) for a high-level and more comprehensive specification of a software solution would be useful. In [6] a suggestion for the SOD is made based upon some successes in industry projects.

V. Conclusions and Future Work

The desire for innovation in a development project is often at odds with desire to control the project execution process from a business perspective. This paper suggests that through considering the components of SEFP practices in the context of project process and using effective visual models to communicate, a balance can be achieved between the opposing desires. The three components interact and influence each other. While there is some understanding of the extent of the components interaction, much is not completely understood and might vary from domain to domain.

Future work in part would consist of activities such as the theoretical exploration of the components and the empirical evaluation of the effectiveness. Are there other important SEFPs? What modeling approaches and model communication are most effective and at what point in the project? How do the SEFPs and model artifacts interact during the project process? Do the proposals scale up or down to match different project sizes or domains? Are the proposals effective only for new projects or do they work for the evolutionary releases of long-lived solutions?

References

[1] H. Petroski, Invention by Design-How Engineers Get From Thought to Thing, Harvard University Press, 1996.

[2] T. Winograd, Bringing Design to Software, ACM Press, 1996.

[3] F. P. Brooks, Jr., The Design of Design: Essays from a Computer Scientist, Addison-Wesley, 2010.

[4] F. P. Brooks, Jr., The Mythical Man-Month Essays on Software Engineering 20th Anniversary Edition, Addison-Wesley, 1995.

[5] J. H. Larkin, H. A. Simon, "Why a Diagram is (Sometimes) Worth Ten Thousand Words", Cognitive Science, volume 11, pages 65-99, 1987.

[6] D. Matheson, "Modeling Requirements: The Customer Communication", Requirements Prioritization and Communication Workshop, Karlskrona, Sweden, 2014.

[7] ITU=T Z.151 User Requirements Notation – Language definition, 10/2012, 2012.

[8] E. S. Yu, "Social Modeling and *i**", in Borgida, A. T., V. Chaudhri, P. Giorgini, E. S. Yu (eds.),Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos, LNCS volume 5600, Springer, 2009.

[9] Object Management Group Unified Modeling Language (UML), version 2.5, 2015, http://www.omg.org/spec/UML/2.5/.

[10] D. L. Parnas, "On the Criteria to be Used in Decomposing Systems into Modules", Communications of the ACM, volume 15, number 12, December 1972, pages 1053 - 1058.

[11] G.J. Myers, Reliable Software through Composite Design, Petrocelli/Charter, 1975.

[12] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, S. Angel, A Pattern Language, Oxford University Press, 1977.

[13] M. Fowler, Patterns of Enterprise Application Architecture, Addison-Wesley, 2003.

[14] E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Prentice Hall, 1995.

[15] M. Cohn, Succeeding with Agile: Software Development Using Scrum, Addison-Wesley, 2010.

[16] ISO, Open Distributed Processing - Reference Model (RM-ODP) ISO 10746-1, 1998.

[17] W. G. Vincenti, What Engineers Know and How They Know It, The John Hopkins University Press, 1990.

[18] J. Beatty, A. Chen, Visual Models for Software Requirements, Microsoft Press, 2012.

[19] D. L. Moody, "The 'Physics' of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering", IEEE Transactions on Software Engineering, volume 35, issue 6, pages 756-779, 2009.

[20] D. Matheson, Innovation Information Management Model, U.S. Patent 6,994,514 & 7,050,872, issued 2005 & 2006.