Design Thinking implemented in Software Engineering Tools
Proposing and Applying the Design Thinking Transformation Framework

Alexander Grosskopf, Mathias Weske
Hasso-Plattner-Institute, University Potsdam, Germany

Jonathan Edelman, Martin Steinert, Larry Leifer
Center for Design Research, Stanford University, CA, USA

Abstract

Design Thinking has collected theories and best-practices to foster creativity and innovation in group processes. This is in particular valuable for sketchy and complex problems. Other disciplines can learn from this body-of-behaviors and values to tackle their complex problems. In this paper, using four Design Thinking qualities, we propose a framework to identify the level of Design Thinkingness in existing analytical software engineering tools: Q1) Iterative Creation Cycles, Q2) Human Integration in Design, Q3) Suitability for Heterogeneity, and Q4) Media Accessibility.

We believe that our framework can also be used to transform tools in various engineering areas to support abductive and divergent thinking processes. We argue, based on insights gained from the successful transformation of classical business process modeling into tangible business process modeling. This was achieved by incorporating rapid prototyping, human integration, knowledge base heterogeneity and the media-models theory. The latter is given special attention as it allows us to break free from the limiting factors of the exiting analytic tools.

1. Introduction

In this paper we propose a framework to align software engineering tools with Design Thinking. We strive to apply paradigms experimentally derived from Design Thinking research with the aim to transform convergent engineering tools, designed and used for analysis, into abductive, divergent tools. In this paper we experimentally focus on the discipline of software engineering as it is fitting representative for many engineering disciplines. Its tools used are designed to analyze real problems and situations, build mathematical models based upon the same, and allow handling those models algorithmically.

It is an early observation in computer science (Brooks 1975) that effort, that is, time spent on development, is hard to predict. Software design is notoriously hard to discuss, especially with end users. This is rooted in the intangible nature of software and the very discipline specific language. The most users can see is the user interface, which literally is only the tip of the iceberg. To exchange deeper level knowledge about systems, different tools for data, process and architectural modeling have been developed. They have a mathematically defined semantics and are the sharp knife for communication amongst engineers. These analytical tools however can hardly be used to discuss software attributes with end users. Consequently, software often does not meet customer expectations (Krallmann et al. 2007).

In the last decade, the introduction of agile development as a methodology (Martin 2003) stepped up to ease this pain. Agile development demands that the evolving software product is constantly presented and discussed with end users. This helps to identify divergent expectations early on and avoids expensive misunderstandings. Agile development is a huge advancement over original software engineering methods, such as the waterfall or the V-model (Zhu 2005, p. 55-57). However, agile development treats the symptom and not the underlying problem of suboptimal early communication. Agile development has no means of communicating conceptual decisions and its impact with common end users, as end users are not "language and knowledge compatible" with the specific concepts that engineers use to capture their knowledge.

Generally tools and development behaviors employed in software engineering aim to reduce risk and to (re-)produce a product of high quality at predictable costs. The underlying logic roots in production line manufacturing (Clements & Northrop 2001). It is worth noting that management approaches for traditional product and software development are slowly converging; both rely on classical engineering philosophies. This includes well-defined procedures and analytical tools to ensure validity of results by reducing uncertainty and complexity of the problem.

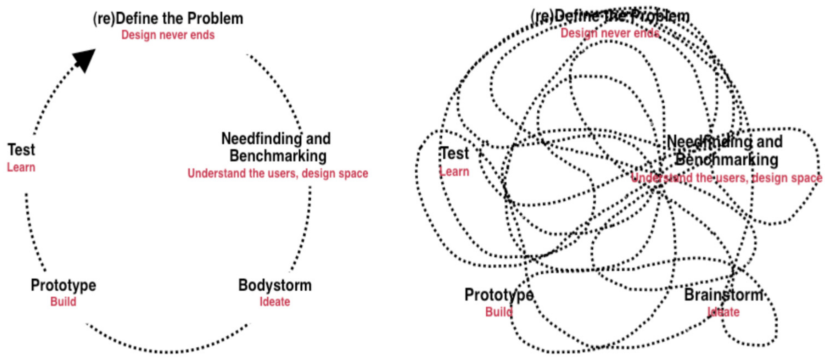

Figure 1. Design Thinking is commonly visualized as an iterative series of five major stages. To the left we see the standard form. To the right we see something closer to reality (Leifer & Meinel 2010)

Unfortunately, this formal approach to product and software development fails to support exploration and discovery of unexpected issues. Based on the research of what designers and engineers really are thinking and doing, when they successfully create products, services, and enterprises, a powerful process for innovation has emerged: Design Thinking. It integrates human, business, and technological factors in problem -forming, -solving, and –design. It creates a vibrant interactive environment that promotes learning through rapid conceptual prototyping. Figure 1 depicts design thinking as a journey through five stages as suggested by (Leifer & Meinel 2010).

So far, the deployment of Design Thinking specific insights and procedures for software engineering has been rather constrained (Lindberg & Meinel 2010). Mainly the intangible nature of software, its inherent complexity and specific language hampers user integration. Our aim is to attempt to bridge this gap by proposing a Design Thinking Transformation framework. With it, we aim to firstly identify and secondly alter Design Thinking compatible software engineering tools.

Our framework is grounded on insights gained from having extensively studied the impact of media choice on the design process. Section two we introduce the media-models framework as a heuristic for understanding the convergent or divergence nature of tools. In section three, we show how we have applied the knowledge about behavioral change induced by media to a specific software engineering tool: Business Process Modeling. In section four, we propose a general framework based on the results from a series of studies, experiments, and observations. This framework is comprised of four qualities to identify transformable engineering tools and to highlight possible leverage points. In the final section of the paper we conclude with a discussion.

2. Media-Models: Steering discussions in Team-based design

The past few years have yielded powerful insights into how to gear analytic tools towards design thinking: Through the substitution of analytic media tools with generative media tools designers can trigger increased abductive activities and potential in development scenarios.

Media filters and characterizes information. Every class of instantiation of media is different in respect to the information which is embodied in it. Hence, media with different characteristics enables us to do different kinds of thinking; different media afford different kinds of thought. A prototype made of plasticine will provoke different feedback than an computer rendered image. The choice of media characterizes the information transmission in a similar way. Media provides affordances for thinking and action, because it conditions how information can be communicated and what can be done with it. Thus, we see media as an "actant", and not just a passive container.

When people externalize knowledge they use media, whether in the form of sound waves in the air (spoken words), e-mails or paper. Contemporary studies in cognitive psychology have emphasized the effect of media on what people can think about and how they think about it (Tversky 1999; Maglio et al. 1999; Clark 2008). If media directly influences the direction, breadth, and depth of communication, the question is how to maximize the effectiveness of media in a given phase of product (or software) development. The Media-Models framework (Jonathan Edelman 2009) considers the dimensions of media and their effect on the conversations that designers have during the development process.

Media-models can be seen as intermediary objects used during the development and negotiation of designs, processes, products, and services in team based design. The framework is based on field studies and experimental evidence examining three dimensions of intermediary objects in use by design teams: abstraction, resolution, and ease of change (Jonathan Edelman 2009).

All models enlist abstraction. **Abstraction** is defined as the highlighting and isolation of specific qualities and properties of an object, such as color, size or functions. Fewer represented properties indicate a greater abstraction from the actual object. Because representations with higher levels of abstraction have fewer properties with which to contend, they are easier for designers to work with than models with many properties. This ease comes at a price: abstract models are not complete, but offer only a slice of or a perspective on a product or service.

Resolution refers to the fidelity with which an object is defined with respect to its final form. For example if a final product is a car, a Lego model of a car would be considered low resolution. However, if the deliverable is made of Lego and the Lego model shares the same dimensions, than the Lego model must be considered fully resolved.

Resolution and abstraction are orthogonal properties inherent in media-models used for communication. In figure 2 we show some sample media in a coordinate system by abstraction and resolution.

The third dimension that media-models share is **ease of change**. It refers to the amount of effort required to change an idea embedded in a specific media-model. The resistance to change is also referred to as viscosity by (Blackwell et al. 2001), though his focus is on dimensions of notations, whereas the ease of change that we refer to here is a dimension of media-models.

All three media dimensions are at play when people express their ideas in a model. Less abstract models require designers to consider more properties. Higher resolution models afford high precision when making parametric changes to a model. On the other hand, abstract, low resolution models afford global or paradigm changes.

As an example of media-model choice, a car manufacturer might build an actual roadworthy prototype of the next generation car product. This type of media model is highly resolved, absolutely not abstract and hard to change. That prototype is well suited for detailed examination just before mass production. It is not suited to question the fundamental design. In contrast, a miniature car made of plasticine is highly abstract, less resolved and easy to change. This type of representation might be suited for general design discussions but does not reveal details. These two extreme model choices showcase the contrast between "analytic" media-models and "generative" media-models.

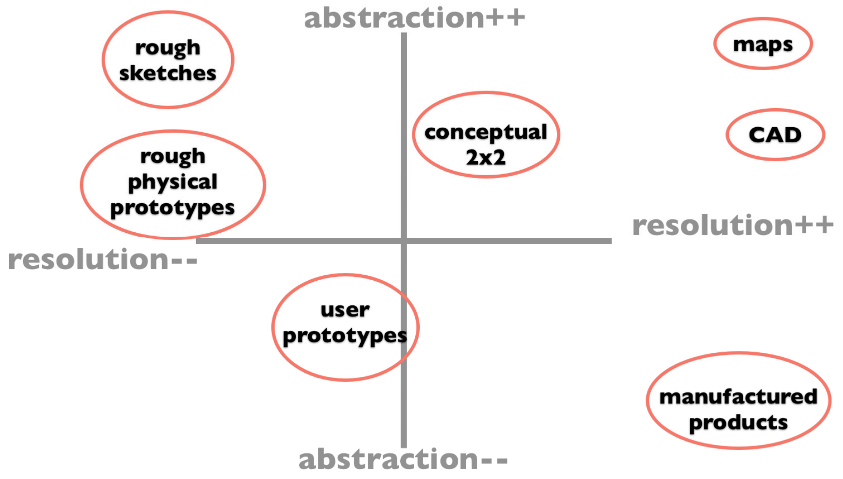

Figure 2. Media used in product development characterized by the media-models framework

We call media which affords parametric change **analytic media**, and media, which affords a multiplicity of potential global solutions **generative media**. In lab experiments with designers we have observed that analytic media leads people to discuss adjustments of parameters within the design, while generative media affords discussions of the general concept of the design. It is simply meaningless to discuss parametric adjustments with a low resolution model. In other words, media choice conditions communication in product design.

This effect is also implicitly known in software engineering. Best practices for User Interface (UI) designers suggest to use sketched paper prototypes to discuss UIs with end users rather than polished screenshots or even the actual UI (Buxton 2007). The sketched representation abstracts e.g. from color and does not resolve e.g. the actual size of buttons. Thus it allows the user and UI designer to concentrate on the underlying concepts of the human-computer-interaction. However, in software engineering, UI design practices are an exception. Classical software engineering is mathematically driven and committed to analytical techniques and media. We applied our insights from the media-models theory to one particular software engineering tool, Business Process Modeling.

3. A Software Engineering tool transformed

Business process modeling is the act of mapping knowledge about working procedures in organizations to a graphical representation, the business process model. This is popular in the context of Business Process Management, an approach to structure work in organizations (Burlton 2001). That includes modeling, analyzing and improving the working procedures. Automating processes in software systems offers great potential to save time, enhance reliability and deliver standardized output (Davenport 1993; Hammer & Champy 2003). In the last decade, business process management, and therefore also business process modeling, has become an IT-driven topic (van der Aalst et al. 2003). IT-support for business processes requires significant software engineering effort. As typical for software projects, misunderstandings in early stages lead to expensive change requests at later stages of the project (Boehm 1981).

In current practice, requirements are gathered in interviews and workshops. Post-its and software tools dominate the employed media. In explorative studies we observed that Post-its allow end users to easily map their knowledge. However, Post-its do not embody concepts. Thus, the resulting Post-it stream does not express the knowledge in the frame of a business process. It typically requires a process analysts to collect the information and create process models from end user input. The model is then discussed with end users and is refined until fully accepted. Process analysts can choose from a wide variety of modeling software that supports language specific iconographies, syntax verification and process automation qualities.

For novices, these are expert tools. Thus, for efficient use, modeling software typically remains in the hands of the experts. Changes to the model have to be channeled through them.

The limited access to the model for non-experts motivated us to change the media for business process modeling. We aimed to empower end users to express their knowledge as processes and directly apply changes to the model. The early development of generative media for Business Process Modeling included Lego, crafting accessories, and Post-its. After some iterations we found that, acrylic tiles with process modeling iconography sharpened the discussions of process modeling experts and domain experts (J. Edelman et al. 2009; Grosskopf et al. 2009). The specific iconography embodies process modeling concepts and thereby enforces basic framing. At the same time, the rough media is more flexible than a digital model and lifts resolution constraints. In other words, software and logic does not restrict the use of elements. Thus, logical constructs can be less accurate or can be ignored entirely during modeling. It is even possible to (ad hoc) break the process modeling frame and incorporate different concepts into the model for discussion.

During process modeling workshops a process analyst is enlisted to explain process modeling and to guide the group through the workshop. Instead of *filtering* and *translating* input from different stakeholders, the analyst becomes a *facilitator* of the group's internal consensus finding. This enables an integration of human, social interaction into process design. Now multiple people work together at the same model which is laid out at a table top. They can immediately point at, touch and change the model to demonstrate ideas. The shared common knowledge is represented at the table in tangible media. We therefore call this approach t.BPM, tangible business process modeling.

Figure 3 depicts t.BPM in the framework of media models. Traditional Business Process Modeling software allows for different levels of granularity. A range of simple to technically sophisticated representations can be contained in one model. The resolution of each piece of information however is typically high. By contrast, t.BPM can only embody a limited set of information at a time, but through more or less accurate use of the process modeling concepts, the resolution might vary. In comparison to figure 1, please note that these are relative measures.

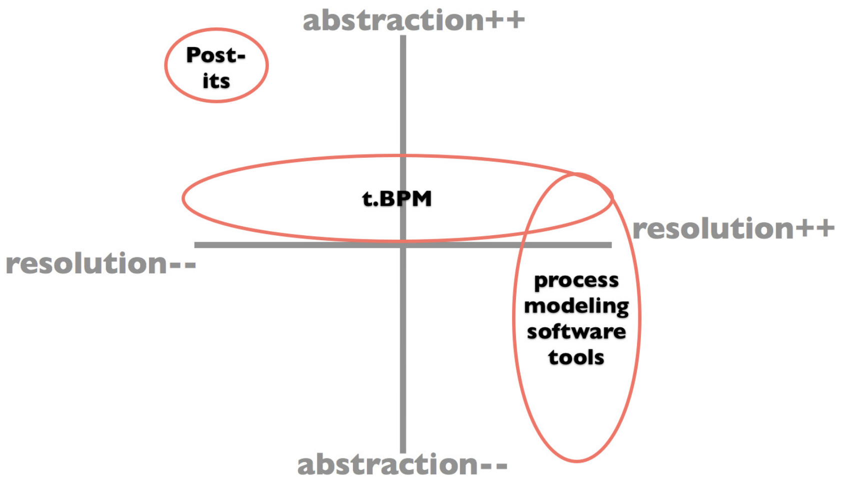

Figure 3. How t.BPM changed process modeling in the light of the media models framework

We conducted first user studies with t.BPM in 2009 (see figure 4, middle). As predicted by the media-models theory, we observed people to question the overall design of the process more often. Global changes were acceptable with easily changeable media. We also found that the absence of computers for process modeling narrowed the gap between modeling experts and novices. Only a few concepts have to be explained before the modeling can start. Application specific knowledge is not needed as the tool is intuitive to use.

In a Case Study conducted in a hospital environment (see figure 4, right) we also observed a flexible abstraction level for t.BPM as a tool. A five day workshop with hospital doctors on clinical pathway modeling (processes in hospitals) started with rough mapping and became a detailed and sophisticated model for discussion over time. The workshop was facilitated by an experienced BPM consultant, who used t.BPM with only a few concepts as a minimal ground for information sharing.

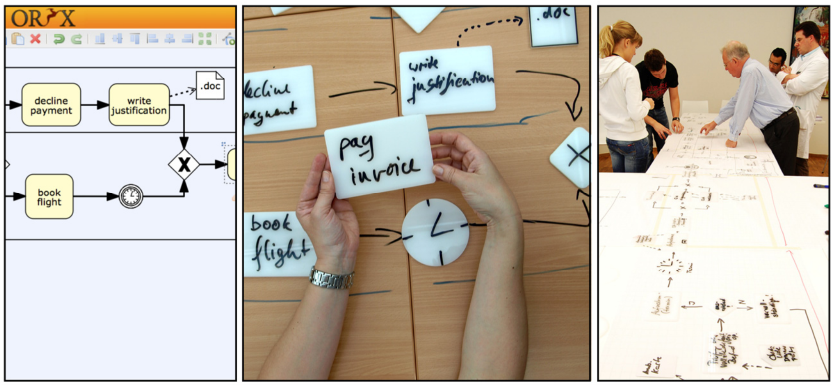

Figure 4. Software Modeling tools (left), t.BPM first user studies (middle), t.BPM field study with hospital processes (right)

Over time the BPM facilitator introduced more concepts when needed, and when discussions focused on details of the models. On day three, t.BPM was complemented with other media such as software modeling tools and print-outs. Our analysis of the workshop showed that software-based process modeling media is more suited for navigating through large sets of process models than t.BPM. However, when creating new models, discussions genuinely turned back to the table with t.BPM.

4. Design Thinking Transformation Framework

The insights and experience we gained during iterating and testing t.BPM led us to identify some core qualities that we believe are key for transforming analytical convergence tools into abductive divergence tools. Therefore, based on our roughly twenty prototypes we would like to suggest a Design Thinking Transformation Framework consisting of the following four qualities:

Q1 Iterative Creation Cycles

As depicted in figure 1, we identify five major development stages: problem definition, need finding & specifications, ideation, development, deployment/testing. The classical analytical process goes through those five stages once. Iterative approaches in analytical disciplines, e.g. in software engineering, propagate iterations by slicing the problem into small pieces and solving one piece within one iteration. This is what we call "iterative refinement cycles". Parts or details are determined iteratively. Fundamental, underlying ideas are not questioned.

In contrast, Design Thinking suggests to iteratively create new solution ideas. Inexpensive prototypes are key enablers to explore the solution space by trying out many different ideas. This is what we call "iterative creation cycles". In t.BPM, easily movable elements enable fast prototyping of ideas.

Q2 Human Integration in Design

In both product and software development ambiguous, informal human needs are transformed into formal requirements to be used in analytical reasoning. Therefore, analytical disciplines tend to limit the user interaction in order to reduce ambiguity and uncertainty in the analytical process. Users present needs to developers, who in turn present well crafted solutions to users.

Design Thinking calls for repeated, physical integration of stakeholders in the design and ideation phase. This is not only helpful to get instant feedback on ideas. Integrating users into the design process makes them engaged and advocates for the solution they helped design. Often, members of the user community can help promote new solutions, get them accepted and avoid resistance to adoption.

Q3 Suitability for Heterogeneity

A high degree of specialization leads to a strong fragmentation of knowledge. The more sophisticated the discipline, the stronger the fragmentation, even within disciplines. Sophisticated models that are used in those disciplines for detailed discussions and deep reasoning are not well suited for the incorporation of heterogeneous knowledge bases.

Design thinking works with interdisciplinary teams by using simple visualization and prototyping techniques to transport ideas and integrate knowledge from stakeholders with heterogeneous background. As a rule of thumb, the further apart the disciplines, the simpler the models that they can use to share and integrate information. By 'simple' we mean the amount of concepts required to understand the model. As an example, in t.BPM we condense the BPMN standard (OMG 2009) to the basic concepts of control flow, data and resource allocation. This is embodied in four shapes and markings drawn on the table.

Q4 Media Accessibility

Typically, in analytical disciplines, solution designs are digital throughout. Advantages such as versioning, computer-assisted analysis and easy distribution have outweighed advantages of physical representations. Really?

Design Thinking research suggests that each instantiation of media affords particular types of interactions and changes to a designed solution. This happens because the media-model dimensions (abstraction, resolution, ease of change) define the interaction space in which people can define their solution. We learned with t.BPM, that tangible media removes barriers for participation. No expert knowledge is required to handle the media. This enables people to change the models, and therefore the solution themselves.

Framework Applied

We now propose an early stage framework based on the four qualities that can be used to identify the level of Design Thinking factored into existing tools. We test it by using common conceptual modeling approaches from software engineering.

We assume as working hypothesis that, for any software development tool, this Design Thinking Transformation framework allows us to judge its closeness to divergent Design Thinking vs. its closeness to convergent analytical thinking. To test this hypothesis we have selected four standard analytical modeling techniques from software development. We choose:

- Data Modeling, often done with Entity-Relationship Diagrams (Chen 1976), is used to represent information objects (e.g. dog), their attributes (e.g. age) and relation to other information artifacts (e.g. is owned by). It is used to specify database schemas.
- Use Case Modeling, part of UML (Fowler & Scott 2000), is used to depict roles (dog owner), their applications (e.g. go for walk) and dependencies between applications (e.g. go for walk requires find leash). It is used to visualize complex application scenarios.
- Object Oriented Modeling, also part of UML (Fowler & Scott 2000), describes classes of objects (e.g. dogs) with attributes (eg. age), behavior (e.g. bite, sit) and interrelation to other classes of objects (e.g. has owner of type human). It is used to model the structure of object oriented programs.

- Classical process modeling (Scheer et al. 2005; OMG 2009)depicts steps (go for walk) their interdependencies (e.g. find leash before going for a walk), responsibilities (owner has to find leash) and data used in the process (e.g. newspaper to read in the park). Process modeling is used for analysis, simulation and automation of working procedures, business processes.

All these modeling approaches were developed in the IT community and are software supported. In t.BPM we use the idea of classical process modeling. We transform a tool's media, from software to tangible, we involve the stakeholders in an iterative creation process and we simplify the notational system. Thus, we can integrate more people with heterogeneous knowledge bases into the creation of the solution.

In the following section, we rate the software development tools for their Design Thinkingness, and therefore the effort required for transformation. We rate according to industry's best practices which implies that individual applications of the tools might differ from our assumptions. We rate with a (-) if current practice is contrary to proposed Design Thinking practice. We rate (o) if it is not in line but aspects point into the right direction. And we use (+) if it is in line with the Design-Thinking-Way that this aspect is practiced.

	Data Modeling	**Use Case Modeling**	**Object Oriented Modeling**	**Classical process modeling**	**t.BPM process modeling**
Q1 Iterative Creation Cycles	-	-	-	-	+
Q2 Human Integration in Design	-	o	-	o	+
Q3 Suitability for Heterogeneity	-	+	-	o	+
Q4 Media Accessibility	o	o	o	o	+

Table 1. Design Thinking Transformation Framework applied to Software Modeling Techniques

We rate classical modeling tools with (-) for Q1 as they rely on software support to visualize their concepts. Software tools in this area afford iteration for refinement but not iteration for creation. For Q2 we rate (o) if users in practice typically provide feedback to intermediate solutions, otherwise (-). A (+) is given only if customers are actively participating in the design process.

For Q3 we rate (+) if concepts can easily be understood and adopted with little introduction time. We rate (o) if the representation is simple enough to be read and mainly understood without expert knowledge. We rate (-) if it requires expertise and experience to read the model and understand the implications. For Q4, we rate (o) if only experts can create models and others can only make comments, e.g. on printouts. t.BPM here scores (+) as it allows non-experts to apply changes.

We note that no classical tool is purely analytical. Nonetheless, no classic tool or method is truly an abductive, divergent tool in the sense of Design Thinking tools.

At the present time, we have not developed proper scales for the qualities proposed. However, the framework has enabled us to identify and to separate the very classical convergent analytical software tools from other tools that have incorporated some Design Thinking rules.

Indeed, as we have shown with the creation of t.BPM from classical process modeling, it is possible to transform an analytical tool into an abduction divergence tool. The framework at hand provides us with four fundamental questions as a starting point for the transformation:

1. How to incorporate rapid prototyping and iterate for the creation of new ideas rather than refinements?
2. How to ensure the continuous integration of the user and his participation in the design process?
3. Which concepts are required for communication in order to establish a model for shared understanding amongst participants with heterogeneous background?
4. How to choose media to support questions 1,2,3 and realize media accessibility?

Although we currently only have one, though very successful example, we believe that the same approach may be used for other software engineering tools, to judge their level of Design Thinking and to identify starting point for a possible transformation. We would like to call our framework Design Thinking Transformation Framework (DTTF) and, with this paper, put it forward for discussion.

5. Discussion and Future Research

We have proposed the Design Thinking Transformation Framework. DTTF consists of four qualities: Iterative Creation Cycles, Human Integration in Design, Suitability for Heterogeneity and Media Accessibility.

The DTTF is grounded in years of Design Thinking Research on media and the insights gained from transforming classical business process modeling into t.BPM. We have used the DTTF to assess different software modeling tools. However, we believe that this DTTF is not limited to contrasting Design Thinking with Computer Science.

We invite other researchers to use this Design Thinking Transformation Framework for other disciplines and show applicability and shortcomings.

For our future research, we aim to develop mature scales to measure the Design-Thinkingness of tools with respect to the qualities described in this paper.

References

van der Aalst, W.M., Hofstede, A.H. & Weske, M. (2003) *Business process management: A survey*. Lecture Notes in Computer Science Vol.2678, 1–12.

Blackwell, A.F., Britton, C., Cox,A., Green,T.R.G., Gurr, C., Kododa,G. Kutar, M.S., Loomes, M., Nehaniv, C.L., Petre, M. & others (2001) *Cognitive dimensions of notations: Design tools for cognitive technology*. Lecture Notes in Computer Science Vol.2117, 325–341.

Boehm, B.W. (1981*) Software engineering economics*, Prentice Hall.

Brooks, F.P. (1975) *The Mythical Man Month: Essays on Software Engineering*, 2/e, Pearson Education.

Burlton, R.T. (2001) *Business process management*, Sams.

Buxton, W. (2007) *Sketching user experiences: getting the design right and the right design*, Morgan Kaufmann.

Chen, P.P. (1976*) The entity-relationship model—toward a unified view of data*. ACM Transactions on Database Systems (TODS), Vol. 1, p 36.

Clark, A. (2008) *Supersizing the mind: Embodiment, action, and cognitive extension*, Oxford University Press, USA.

Clements, P. & Northrop, L. (2001). *Software product lines*, Addison-Wesley Reading MA.

Davenport, T.H. (1993) *Process innovation: reengineering work through information technology*, Harvard Business School Press.

Edelman, J., Grosskopf, A. & Weske, M. (2009) *Tangible Business Process Modeling: A New Approach*. In Proceedings of the 17th International Conference on Engineering Design, ICED'09.

Edelman, J. (2009) *Hidden in Plain Sight: Affordances of Shared Models in Team Based Design*. In Proceedings of the 17th International Conference on Engineering Design, ICED'09.

Fowler, M. & Scott, K. (2000) *UML distilled: a brief guide to the standard object modeling language*, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Grosskopf, A., Edelman, J. & Weske, M. (2009). *Tangible Business Process Modeling - Methodology and Experiment Design*. 1st International Workshop on Empirical Research in Business Process Management (ER-BPM'09). Ulm, Germany: Springer Verlag, p. 53-64.

Hammer, M. & Champy, J. (2003) *Reengineering the corporation: A manifesto for business revolution*, Collins Business.

Krallmann, H., Schönherr, M. & Trier, M. (2007) *Systemanalyse im Unternehmen*, Oldenbourg Verlag.

Leifer, L.J. & Meinel, C. (2010) *The Philosophy behind Design*, Springer

Lindberg, T. & Meinel, C. (2010) *Design Thinking in IT Development?* Available at: http://ecdtr.hpi-web.de/report/2010/001/.

Maglio, P.P., Matlock, T., Raphaely, D. Chernicky ,B. & Kirsch, D. (1999) *Interactive skill in Scrabble*. In Proceedings of the Twenty-First Annual Conference of the Cognitive Science Society, 1999, Simon Fraser University, Vancouver, British Columbia. p 326.

Martin, R.C. (2003) *Agile software development: principles, patterns, and practices*, Prentice Hall PTR Upper Saddle River, NJ, USA.

OMG (2009) *Business Process Modeling Notation (BPMN) 1.2*, OMG.

Scheer, A.W., Thomas, O. & Adam, O. (2005) *Process modeling using event-driven process chains*. Process-aware information systems: bridging people and software through process technology, 119–145.

Tversky, B. (1999) *What does drawing reveal about thinking. In Visual and spatial reasoning in design*. S. 93–101.

Zhu, H. (2005) *Software design methodology*, Butterworth-Heinemann.