INTRODUCING AN EMPIRICAL MODEL OF DESIGN

Paul Ralph, Lancaster University, paul@paulralph.name

Abstract

The dominant view of design in information systems and software engineering, the Rational Model, views design and engineering as a methodical, plan-centered, approximately rational process of optimizing a design candidate for known constraints and objectives. It persists despite extensive empirical evidence that it does not reflect design practice and no evidence that attempts to adopt rationalistic processes improve outcomes. One reason for its resilience against empirical critique may be the lack of a comprehensive alternative. This paper addresses this gap by enumerating the Rational Model's components and proposing a comprehensive, better-supported alternative, an "Empirical Model of Design". The Empirical Model is intended to replace the Rational Model as a foundation for design methods and practices, design education and design science research.

Keywords: Design Science, Rationalism, Empiricism, SCI Theory, Software Development.

1 INTRODUCTION

Brooks (2010) attacked the assumptions underlying "The Rational Model of Design", the dominant view of design in information systems (IS) and software engineering (SE), which views design and engineering as a methodical, plan-centered, approximately rational process of optimizing a design candidate for known constraints and objectives. His most compelling criticism was that empirical research does not support this view (e.g., Cross et al. 1992, Truex et al. 2000, Ralph 2010a, Checkland 1999, Bansler and Bødker 1993, Zheng et al. 2011). Despite this evidence, the Rational Model continues to exert substantial influence in scientific and popular discourse surrounding design in IS and SE, as displayed in the design science approach (Hevner et al. 2004), textbooks (e.g., Laudon et al. 2009, Kroenke et al. 2010), standards (e.g., IEEE 1998), official body of knowledge compilations (e.g., Bourque and Dupuis 2004), Wikipedia articles[1], modern design methods (Jacobson et al. 1999) and even the popular conception of the scientific process (e.g., Yin 2003, Campbell and Stanley 1963, Trochim 2001).

The Rational Model's continued dominance is unsurprising. It came first. It has intuitively appealing characteristics including modeling design as search, satisficing instead of optimizing, and viewing professionals as rational actors executing reliable plans. Furthermore, a host of social and cognitive factors including the validity effect (Renner 2004) and status quo bias (Samuelson and Zeckhauser 1988) inhibit shifts to alternative models. Moreover, no comprehensive alternative to the Rational Model is available, which motivates this paper.

> ***Purpose:*** *The purpose of this paper is to explicate the meaning of The Rational Model of Design and to propose a comprehensive alternative.*

Software design science includes a minimum of two research streams (Hevner and Chatterjee 2010) – 1) a research method where knowledge is gained by building innovative artifacts (Hevner et al. 2004) and 2) "the philosophical, theoretical and empirical study of software creation and modification including its phenomenology, methodology and causality" (Ralph 2010b, p. ii). The paper contributes to the latter by organizing alternatives to elements of the Rational Model into an "Empirical Model of Design" (§2). Section 3 summarizes empirical evidence concerning both models, followed by a discussion of the effects of the Rational Model's dominance (§4). The paper concludes with a summary of its contributions and remaining questions (§5).

2 THE RATIONAL MEMEPLEX AND ITS ALTERNATIVE

The dominant view of design has been called "Technical Problem-Solving" (Schön 1983), the "Reason-Centric Perspective" (Ralph 2010b) and the "Rational Model" (Brooks 2010). This paper adopts "Rational Model" as it conveys not only its rationalist epistemology (§2.2) but also its rational designer and process assumptions.

As the Rational Model is both multifaceted and socially constructed, various thinkers may disagree on its nature, hindering unequivocal definition. Therefore, this paper analyzes the Rational Model using mimetics, the study of memes. Dawkins (1989) coined the term "meme" as a "noun that conveys the idea of a unit of cultural transmission, or a unit of imitation" (p. 192). In this view, memes include beliefs, theories, methods, models, philosophies and best practices. Memes may interact, e.g., a rationalist philosophy justifies analytical evaluation methods. Several mutually-reinforcing memes form a complex of memes or "memeplex". By conceptualizing the Rational Model as a memeplex it may be defined by its component memes and their interactions.

2.1 Rational Memes and Alternatives.

Rationalism vs. Empiricism. Brooks (2010) argues that the Rational Model rests on the philosophy of rationalism. Broadly speaking, rationalism is the belief that knowledge derives from reason and

[1] At the time of writing, Wikipedia's ubiquitous "Software Development Process Template" sidebar listed eight "Activities and Steps" including "Requirements", "Design", "Implementation" and "Testing".

intuition, in contrast to empiricism, the view that knowledge is primarily derived from sensory experience (Markie 2008). Empiricism is the basis of popular epistemologies including critical realism, social constructivism, falsificationism and logical positivism.

Technical Problem-Solving vs. Reflection-in-Action. The foundations of the Rational Model were independently developed in computer science by Herbert Simon, Allen Newell and their collaborators (cf., Newell and Simon 1972, Simon 1996) and engineering by Gerhard Pahl, Wolfgang Beitz and their collaborators (cf., Pahl and Beitz 1996). Design professionals are modeled as rational agents attempting to optimize a design candidate for known constraints and objectives. Where the problem space is so large that finding an optimal solution is beyond the designer's limited processing power, the designer will "satisfice" or "find decisions that are good 'enough'" using heuristic search (Simon 1996, p. 27). Simon coins the term "procedural rationality" for this "finding a way of calculating, very approximately, where a good course of action lies" (p. 27), thereby differentiating a (boundedly) rational action from rational outcomes. Schön (1983) called this paradigm "Technical Problem-Solving" (TP-S). TP-S exhibits rationalism in assuming that designers are capable of evaluating the effects of decisions by inspection and that good processes produce good systems.

Building on empirical studies of professional practice, Schön (1983) devised an alternative to TP-S – Reflection-in-Action (RiA). RiA models design as a reflective conversation between the designer and the situation. The designer alternates between framing (conceptualizing the problem), making moves (where a move is a real or simulated action intended to improve the situation) and evaluating those moves. Multiple agents may collectively reflect in action using boundary objects (Levina 2005). (Boundary objects, including diagrams and prototypes, are simultaneously robust enough to maintain their identities and flexible enough to serve multiple parties). Reflection-in-Action differs from Technical Problem-Solving in many ways, e.g., professionals respond to a problematic situation (rather than a given problem) with many possible interpretations and form and explore hypothesis about potentially beneficial actions (rather than optimizing or satisficing design candidates). RiA is consistent with empiricism in assuming that the problem and solution are coconstructed by reflecting on observations and simulations.

Planning vs. Ethnomethodoligcal views of Human Action. The "planning model of cognitive science ... posits that action is a form of problem solving," and that "the plan is prerequisite to the action" (Suchman 1987, p. 28-29). This 'cognitivist view' is consistent with Rationalism in two senses – 1) designing an artifact requires a substantial planning phase as seen in plan-driven methods and 2) designing is itself planning – a design is a plan for building an artifact. In contrast, the "ethnomethodological view" (EV) of action posits that "the organization of situated action is an emergent property of moment-by-moment interactions between actors, and between actors and the environments of their action" (Suchman 1987, p. 179) while "plans are representations, or abstractions over action" (p. 186). This is consistent with Empiricism in that design is seen as an interaction between an actor and its environment; a design project is seen as a system of inquiry where the designer searches for the right solution by building artifacts in the world rather than planning them in the mind.

SDLC vs. SCI. Another key feature of the Rational Model is The Systems Development Lifecycle (SDLC). Although SDLC (Figure 1) was initially presented as a method (Royce 1970), many treat it as a lifecycle process theory – an explanation of how and why an entity changes and develops according to a prefigured, unitary sequence of phases (Van de Ven and Poole 1995). For example, Fitzgerald (2006) states that "in conventional software development, the development lifecycle in its most generic form comprises four broad phases: planning, analysis, design, and implementation" (p. 3). Similarly, Ewusi-Mensah (2003) says that "regardless of the particular process model … every software project will feature: (1) the requirements-definition and functional-specification phase; (2) the design phase; … (3) the implementation; … and (4) the installation, operation, and maintenance phase" (p. 51). Additionally, Laudon et al. (2009) state that "systems development … consist[s] of systems analysis, systems design, programming, testing, conversion and production and maintenance … which usually take place in sequential order". Moreover, traditional SDLC phases are explicitly adopted by the IEEE Guide to the Software Engineering Body of Knowledge (Bourque and Dupuis 2004).

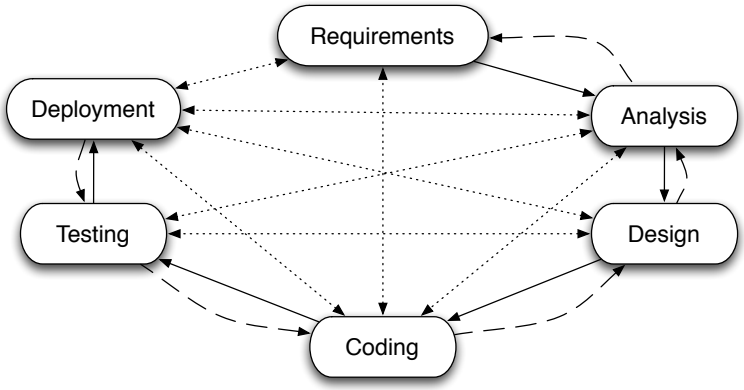

Fig. 1. *SDLC (adapted from Royce 1970). Solid lines indicate the original, no-backtracking version. Solid lines plus dashed lines show the original version with backtracking. Solid, dashed and dotted lines indicate the clique version (see below).*

SDLC has been criticized for many years (cf., Royce 1970, McCracken and Jackson 1982, Gladden 1982, Boehm 1988, Brooks 2010, Beck 2005). However, most criticism concerns the order of activities rather than their conceptual separation; e.g., Royce (1970) argued that leaving testing to the end of the lifecycle would increase risk (§2.5). Many of these criticisms therefore may be avoided by redefining SDLC as a clique (Figure 1), i.e., adding all possible phase transitions. This re-conceptualization reveals the fundamental premise of SDLC – analysis, design, coding and testing are temporally and conceptually separable. This is consistent with Rationalism as separating design from implementation assumes that the designer is capable of determining which type of system to build and approximately how it will work by *reasoning.*

An alternative to SDLC is Sensemaking-Coevolution-Implementation Theory (SCI). SCI (Ralph 2010b) is a teleological process theory, an explanation of how and why an entity changes where change is manifested by a goal-seeking agent engaging in activities in a self-determined sequence (Ralph 2010b, Van de Ven and Poole 1995). The core claim of SCI is that complex software systems are produced primarily through its three titular activities. SCI is consistent with empiricism in that the design agent explicitly gains knowledge through sense experience (sensemaking).

SCI (Figure 2, Table 1) describes the process where an agent designs a complex system. The agent may be an individual or team. The arrows in Fig. 2 indicate relationships between concepts and activities, not the sequence of activities. Since implementation depends on the Mental Picture of the Design Object, which is initially generated by the coevolution process, some coevolution must precede implementation. By equivalent logic, some sensemaking must precede coevolution. However, once the two mental pictures are initially formed, the agent may transition between activities in any order. One possible sequence would be as follows. The design agent begins the project by perceiving its environment and the environment where the design object is intended to operate. Organizing its perceptions, the agent forms a mental picture of the context, including some tentative goals and constraints. Based on this, the agent formulates some beliefs about one or more possible design objects. It then iterates between these two sets of beliefs, refinements of one triggering reframing of the other and the inverse (the inner iterative loop). The agent may externalize its cognition in boundary objects including conceptual models, design models, mockups and prototypes. Eventually the agent begins to build the design object (or perhaps to convert an existing prototype into the design object). The design object's existence then changes the context, which triggers further sensemaking, and so forth (the outer iterative loop). Testing would therefore mostly involve the outer loop.

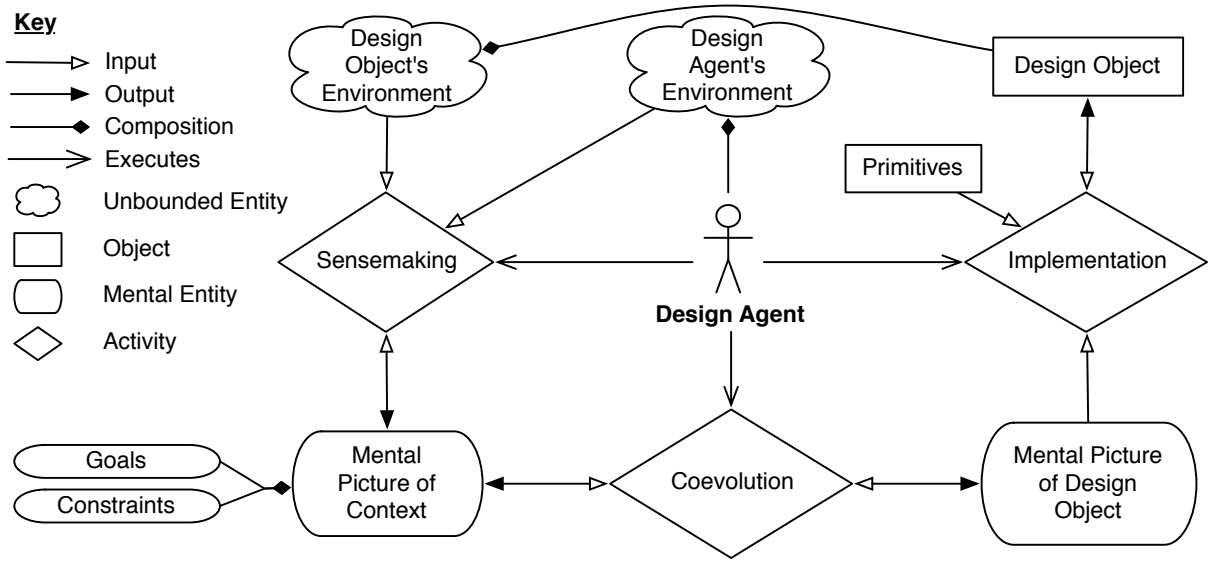

Fig. 2. Sensemaking-Coevolution-Implementation Theory (from Ralph 2010a)

Concept / Activity	Meaning
Constraints	the set of restrictions on the design object's properties
Design Agent	an entity or group of entities capable of forming intentions and goals and taking actions to achieve those goals and that specifies the structural properties of the design object
Design Object's Environment	the totality of the surroundings where the design object exists or is intended to exist
Design Agent's Environment	the totality of the surroundings of the design agent
Design Object	the thing being designed
Goals	optative statements about the effects the design object should have on its environment
Mental Picture of Context	the collection of all of the design agent's beliefs about its and the design object's environments
Mental Picture of Design Object	the collection of all of the design agent's beliefs about the design object
Primitives	the set of entities from which the design object may be composed
Sensemaking	the process where the design agent perceives its and the design object's environments and organizes these perceptions to create or to refine the mental picture of context
Coevolution	the process where the design agent simultaneously refines its mental picture of the design object, based on its mental picture of context, and the inverse
Implementation	the process where the design agent generates or updates the design object using its mental picture of the design object

Table 1. SCI Theory Concepts and Relationships (adapted from Ralph 2010a)

System Development Methods (SDMs). Many modern SDMs still retain SDLC's basic structure. For example, Extreme Programming (Beck 2005) involves running through SDLC phases in linear, time-boxed iterations (Figure 3), while the Unified Software Process (USP) (Jacobson et al. 1999) adopts them as parallel "disciplines" within an alternative phase model (Figure 4). Even Boehm's (1988) spiral model is just SDLC reorganized into a spiral with added risk analysis. However, modern SDMs are often characterized according to their degree of planning and analysis.

Plan-driven methods including Waterfall and USP and are often contrasted with Agile methods (Beck et al. 2001) including Extreme Programming (Beck 2005), Lean (Poppendieck and Poppendieck 2003) and Boomerang (Stacey and Nandhakumar 2008). However, a starker contrast is available in the form

of *amethodical development*. "Amethodical systems building implies management and orchestration of systems development without a predefined sequence, control, rationality, or claims to universality. An amethodical development activity is so unique and unpredictable ... that even the criteria of contingent development methods are irrelevant" (Truex et al. 2000, p. 54).

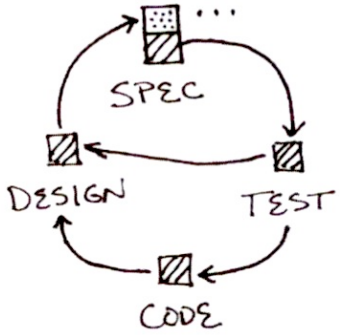

Fig. 3. XP's "Pull Model of Development" (Beck 2005)

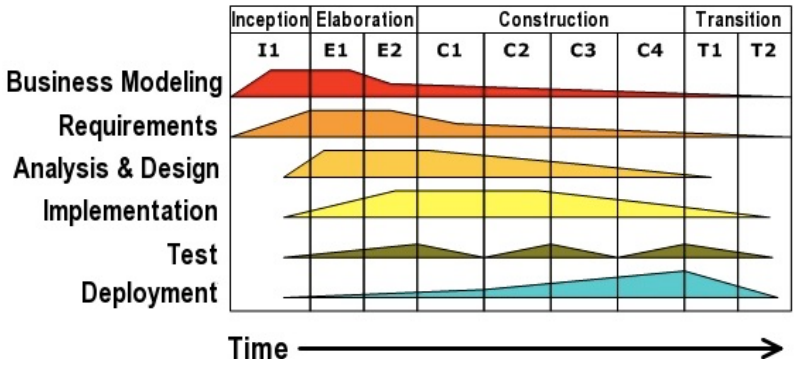

Fig. 4. Unified Software Process Phase Model (adapted from Wikimedia Commons)

Like SDLC, plan-driven methods are consistent with Rationalism in their separation of design from coding and focus on ensuring outcome quality through good plans and processes. Analogously, amethodical development, with its unpredictability and focus on improvised action, is clearly more consistent with Empiricism. In contrast, Agile methods may combine rationalist and empiricist elements. For example, Extreme Programming prescribes both separating planning, analysis, design and coding (rationalist) and keeping a user near the development team for questioning (empiricist).

Project Management Frameworks (PMFs). While a software design method conveys advice for designing and building software, a software project management framework conveys advice for organizing the software creation enterprise (e.g., distributing and scheduling tasks, types of meetings). USP (above) is both an SDM and a PMF as it contains advice for both designing (e.g., 'organize code into components') and organizing the project (e.g., 'create an executable architecture during the elaboration phase') (Jacobson et al. 1999). As a PMF, USP is consistent with rationalism in its organization of the design project as separate disciplines named after SDLC phases.

Scrum (Schwaber and Beedle 2001) is an alternative PMF commonly combined with Agile SDMs. Scrum comprises a set of recommended roles, artifacts meetings and practices for developing software in time-boxed iterations called sprints but no phases or disciplines; rather, work is organized into time-boxed 'sprints', each producing a shippable product increment. Scrum's consistency with empiricism is evident in its lightweight planning and explicit focus on continual interaction with external stakeholders.

2.2 Defining the Rational and Empirical Models of Design

The previous section identified various elements (memes) related to software design and consistent with rationalism. Many of these are interconnected. TP-S rests on the assumption that "a physical symbol system … has the necessary and sufficient means for general intelligent action" (Simon 1996, p. 23). More specifically, intelligent agents model their environments and possible actions using symbol structures; "hence the programs that govern the behavior of a symbol system can be stored, along with other symbol structures, in the system's own memory, and executed when activated" (Simon 1996, p. 22). This is equivalent to the planning model of cognitive science. The centrality of plans also manifests in SDLC as both the project plan and the design specification (as a construction plan). Like SDLC, TP-S clearly separates phases with design beginning with known constraints and objectives and producing a plan for an artifact rather than the artifact itself. Plan-driven methods likewise seek to ensure outcomes through planning and process, and separate design from other phases. SDLC, TP-S and plan-driven methods are all intensely methodical.

Meanwhile, many of the alternatives identified above are also interconnected. RiA is motivated by Schön's (1983) finding that a designer "does not keep means and ends separate, but defines them

interactively as he frames a problematic situation [and] does not separate thinking from doing" (p. 69). This rejection of planning as the foundation of action is central to EV and reframes design as an improvised interaction between an actor and its environment. SCI is explicitly informed by RiA (Ralph 2010b) and enshrines both the actor/environment interaction and the actor's ability to choose its own path. Agile methods, like SCI, focus on producing code rather than plans, however, SCI and RiA are consistent with entirely amethodical development with neither assuming a structured process. Moreover, as Scrum's planning is mostly just choosing the current sprint's goal and distributing work, the plan is an abstraction over action, as in EV. Scrum's explicit focus on interaction with external stakeholders is broadly consistent with SCI and Agile methods. Additionally, Scrum explicitly encourage reflection in action through its retrospective meetings.

Organizing design-related memes around their epistemological underpinnings facilitates definitions for The Rational Model of Design and an alternative as follows. I call this alternative "The Empirical Model of Design" not only as it posits that designers approach projects as empirical inquiries but also to remind us of its grounding in empirical observation (§3). Table 2 summarizes the models.

> *The Rational Model of Design*: a collection of interconnected design-related memes with a rationalist epistemology including Technical Problem-Solving, the Systems Development Lifecycle, plan-driven development methods, and their assumptions.
>
> *The Empirical Model of Design*: a collection of interconnected design-related memes with an empiricist epistemology including Reflection-in-Action, Sensemaking-Coevolution-Implementation Theory, amethodical development, and their assumptions.

Dimension	Rational Model	Empirical Model	References
Philosophy	Rationalism	Empiricism	(Markie 2008)
Design Paradigm	Technical Problem-Solving	Reflection-in-Action	(Schön 1983, Simon 1996, Checkland 1999)
Theory of Action	Plan-Centred	Ethnomethodological	(Suchman 1987)
Process Theories	FBS, Basic Design Cycle, SDLC, USP	SCI Theory, Selfconscious Process	(Gero 1990, Roozenburg and Eekels 1995, Royce 1970, Ralph 2010b, Alexander 1964, Jacobson et al. 1999)
Methodicalness	Methodical	Amethodical	(Truex et al. 2000)
SDMs, PMFs, etc.	FBS, SDLC, USP, Spiral Model	Scrum, Extreme Programming, Lean	(Royce 1970, Jacobson et al. 1999, Boehm 1988, Beck 2005, Schwaber 2004, Poppendieck and Poppendieck 2003, Gero 1990)

Table 2. Components of the Rational and Empirical Models

3 THE JURY IS IN

The above elucidation of the Rational and Empirical models raises many questions – can the model elements be empirically tested? are some elements superior to others? does one model better reflect software development practice overall? The answer to all three questions is "yes" and substantial empirical study has already been done.

Concerning TP-S and RiA, Schön (1983) and Cross et al. (1992) explicitly found that real designers do not operate according to TP-S. The core TP-S assumption that problems are known, unambiguous, and agreed is incommensurate with empirical evidence of goal conflict in design projects (cf., Checkland 1999). Instead, Schön (1983) found that a designer "does not keep means and ends separate, but defines them interactively as he frames a problematic situation. He does not separate thinking from doing" (p. 69). RiA is explicitly based on this empirical foundation provided by Schön's (1983) case studies. This was later extended for team design (see Levina 2005 for discussion of Collective RiA).

Concerning theories of human action, Suchman (1987)'s comprehensive comparison of EV and the plan-centered view concludes that action is inherently improvised, consistent with EV. In software

development specifically, Zheng et al. (2011) show how "collective agility" in a large project was enacted through improvisational behavior under minimal strategic planning.

Concerning process theories, a survey of more than 1400 software developers (Ralph 2010a) found that SCI more accurately represented software development than SDLC or the Function-Behavior-Structure Framework (FBS) (Gero 1990), another rationalist design process theory.

Concerning the methodicalness of development, Nandhakumar and Avison (1999) found that "traditional [information systems] development methodologies are treated primarily as a necessary fiction to present an image of control or to provide a symbolic status, and are too mechanistic to be of much use in the detailed, day-to-day organization of systems developers' activities" (p. 176). Baskerville et al. (1992, 2004) similarly found evidence of amethodical systems development in several case studies of software developers. Truex et al. (2000) summarized the argument by suggesting that "methods [are] merely unattainable ideals and hypothetical 'straw men' that provide normative guidance to utopian development situations" (p. 53).

Concerning SDMs, research by The Standish Group (2006) found that the shift from Waterfall to Agile methods has driven the reduction in project failure since the original Chaos Report. Concerning PMFs, a meta-analysis of empirical studies of Scrum teams concluded that introducing Scrum is associated with higher productivity and customer satisfaction (Cardozo et al. 2010).

The above summary may appear biased toward the Empirical Model. However, an extensive search using Academic Search Premier and Google Scholar using Query 1 produced only one study supporting a Rational-Model element. Palvia and Nosek (1990) evaluated SDLC against a prototyping methodology using a survey; however, it explicitly assumed that SDLC describes all software development, creating a circular argument.

> ***Query 1:*** *("Waterfall Model" OR SDLC) AND (experiment OR "laboratory study" OR "case study" OR "field study" OR survey OR "variance model" or "econometric analysis")*

While it may appear incredulous that the dominant view of design lacks empirical support, it is worth remembering that Simon (1996) provided no empirical support for TP-S; Royce (1970) specifically said the Waterfall Model "has never worked" (p. 335); Gero (1990) provided no empirical support for FBS; and Rationalism as a basis for science fell out of fashion with Kant's *Critique of Pure Reason*. Meanwhile RiA, SCI and amethodical development were all developed based on empirical observations. In conclusion, notwithstanding that existing evidence is incomplete and imperfect, the balance of evidence supports the Empirical Model.

4 DISCUSSION

In addition to the components discussed above, the Rational Model makes several key assumptions – 1) design projects begin with known, unambiguous, agreed goals (cf., Schön 1983); 2) "design" occurs after analysis and before implementation (cf., Schön 1983); 3) the solution space is bounded and conceptually navigable (cf., Brooks 2010); 4) the system being designed is simple, i.e., its properties are predictable from its components (cf. Gero 1990 on predicted behavior). Neither Simon (1996) nor Pahl and Beitz (1996) clearly state these assumptions.

Brooks (2010) attacked the Rational Model on multifarious grounds including – 1) "We don't really know the goal when we start" (p. 22); 2) The desiderata and constraints keep changing; and 3) the designer cannot explore the tree of design decisions as it is never fully understood and the impacts of individual decisions cannot be determined without following them through to complete designs. He thereby joins a chorus of conceptual excoriation of the Rational Model and related concepts (e.g., McCracken and Jackson 1982, Gladden 1982, Truex et al. 2000, Nandhakumar and Avison 1999).

Following this, the Empirical Model makes opposing assumptions – 1) design projects involve multiple stakeholders having poorly-understood, ambiguous, conflicting goals (Checkland 1999, Schön 1983, Brown et al. 2008); 2) "design encompasses all the activities involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems" (Freeman and

Hart 2004, p. 20); 3) the solution space is unbounded; and 4) the system being designed is complex, i.e., it exhibits properties not predictable from its constituent parts (cf., Waldrop 1992).

As mentioned in the introduction, support for the Rational Model is not surprising. This discussion is not intended to insult those who accepted the Rational Model in the past (as the author did) but to reveal the harm of continuing to do so. The Rational model is not *wrong*; it is simply misapplied where its assumptions do not hold and this misapplication creates specific, observable problems.

In real-world design projects, for example, developers are unable to make reliable time and cost estimates as they neither conceptualize their work through detailed plans nor have sufficient information about the problem to accurately estimate its solution's difficulty (Ralph and Wand 2009). This creates tension with managers attempting to drive projects through cost estimates (Beck 2005) and is used to justify contracting development with fixed price / fixed schedule contracts, thereby increasing overall project risks (Brooks 2010). Furthermore, Rational-Model thinking encourages focusing on an oversimplified "given" problem (e.g., USP's focus on use cases) instead of multiple stakeholders having poorly-understood, ambiguous, conflicting goals (cf., Checkland 1999).

Similarly, unfounded adherence to the Rational Model affects design research in at least two ways. First, the design science research approach, as formulated by Hevner et al. (2004), was heavily influenced by TP-S (Hevner et al. cite Simon 13 times). It conceptualizes "the design science paradigm" as "fundamentally a problem-solving paradigm" (p. 76), marginalizing the crucial interplay between problem framing and problem solving at the heart of RiA. Furthermore, it recommends that researchers "Design as a search process" (p. 83), following Simon's model of design as heuristic search. However, empirical evidence reveals that designers do not design as a search process (Schön 1983, Cross et al. 1992) as designers rarely have sufficient information to navigate the tree of design decisions (Brooks 2010).

Second, many design scientists are engaged in developing new and improved SDMs and PMFs (i.e., method engineering). The Rational Model's continued dominance has, at least in part, created a research culture that accepts methods making Rational-Model assumptions despite empirical evidence that these assumptions do not hold in practice. For example, the extensive research on goal-oriented requirements engineering methods assumes that requirements can be "derived" from stakeholder goals (cf., van Lamsweerde 2001). This is based on the Rational-Model assumption of a limited number of knowable solutions. Given the opposite, Empirical-Model assumption of an unknown, possibly infinite number of solutions, requirements cannot generally be derived from goals. For example, one cannot infer specific requirements from the goal of a virtual learning environment, "to enhance student learning", as there is an unknown number of strategies to achieve this goal, some of which may not need any particular "requirement". The fact that proponents of new methods need not produce evidence of effectiveness to be taken seriously exacerbates this problem. Not requiring empirical evidence supporting SDMs and PMFs is arguably also a manifestation of Rationalism.

Moreover, software engineering courses embracing the Rational Model have become misaligned with practice, as expressed by Graham (2003) – "I was taught in college that one ought to figure out a program completely on paper before even going near a computer. I found that I did not program this way … I tended to just spew out code that was hopelessly broken, and gradually beat it into shape." The Rational Model's influence is evident in the ACM model curricula for software engineering and information systems, which cover neither coevolution nor generating design alternatives (Ralph 2011).

Finally, two certain misconception bears preempting. First, many design-related concepts do not fit cleanly in either model. For example, peer programming has no innate epistemology. Second, the Rational and Empirical Models are not methods. Software development teams do not adopt one or the other of these models, and more than Galapagos finches adopt evolution, natural selection and positivism and scientific realism. This is not a debate between agile and plan-driven methods – adopting a plan-driven method is not akin to adopting the Rational Model. The Rational and Empirical models are memeplexes, constructed by the scientific community to conceptualize design work.

The paper has two primary limitations. First, it is not a comprehensive review of every study on ever Rational- and Empirical-Model concept. It claims only that a good-faith search for empirical evidence supporting the Rational Model found only a dearth of empirical study, while empirical studies

supporting corresponding Empirical-Model concepts were plentiful. Second, the analysis applies only to the software development domain. The Rational Model may comprise excellent concepts for understanding other design domains (e.g., electrical engineering).

5 CONCLUSION

This paper makes two contributions. First, it extends Brooks' (2010) identification of The Rational Model of Design by clarifying its meaning, providing a sample of its components and their relationships and enumerating its negative effects on research, practice and education. Second, it organizes Rational Model alternatives into The Empirical Model of Software Design and summarizes the evidence supporting these alternatives. This results in the challenging conclusion that the new Empirical Model is superior to the status quo Rational Model.

These contributions have implications for research, practice and teaching. Researchers studying design may stop basing surveys, coding schemes and experimental protocols on unsupported Rational-Model phases – opting instead for SCI or RiA. Furthermore, the idea of designing through a search processes in the design science approach may be revised. Practitioners may use the Empirical Model and its supporting evidence to argue against fixed-price, fixed-schedule contracts and SDMs and PMFs based on unsupported Rational-Model assumptions. Finally, instructors may replace SDLC, USP and related Rational-Model concepts, with SCI Theory, Scrum and other Empirical-Model alternatives in software development and project management courses. Moreover, it is crucial that software engineering education shift away from toy problems and idealized processes toward ambiguous situations, and complex artifacts that emphasize the problem-framing / problem-solving interplay.

The purpose of this paper is not to disparage proponents of the Rational Model but to point out that the status quo should be subject to the same standard of empirical testing as its alternatives. Following this, several avenues of future research are evident. First, direct empirical comparisons of SDMs, a methodologically challenging task, are needed. Second, some descriptive research on the extent to which Rational-Model assumptions are met in diverse projects would be beneficial. Finally, the components of The Empirical Model may be improved, e.g., SCI Theory may be generalized to account for multiple agents pursuing conflicting agendas.

In conclusion, while many have criticized the dominant view of design, such criticisms are of limited use when no comprehensive alternative is available. This paper presents a comprehensive alternative, clearing the way for a paradigm shift in design research, practice and education.

References

Alexander, C. W. (1964). *Notes on the synthesis of form.* Harvard University Press.

Bansler, J. & Bødker, K. (1993). 'A reappraisal of structured analysis: Design in an organizational context.' *ACM Transactions on Information Systems,* 11 (2), 165-93.

Baskerville, R. & Pries-Heje, J. (2004). 'Short cycle time systems development.' *Information Systems Journal,* 14 (3), 237-64.

Baskerville, R., Travis, J. & Truex, D. P. (1992). 'Systems without method: The impact of new technologies on information systems development projects.' Paper presented at the IFIP WG8.2 Working Conference on The Impact of Computer Supported Technologies in Information Systems Development.

Beck, K. (2005). *Extreme programming explained: Embrace change.* Boston, MA, USA: Addison Wesley.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001). 'Manifesto for agile software development.' [online at http://www.agilemanifesto.org/].

Boehm, B. (1988). 'A spiral model of software development and enhancement.' *IEEE Computer,* 21 (5), 61-72.

Bourque, P. & Dupuis, R. (2004). 'Guide to the software engineering body of knowledge (SWEBOK).' IEEE Computer Society Press.
Brooks, F. P. (2010). *The design of design: Essays from a computer scientist.* Addison-Wesley Professional.
Brown, A. D., Stacey, P. & Nandhakumar, J. (2008). 'Making sense of sensemaking narratives.' *Human Relations,* 61, 1035-62.
Campbell, D. T. & Stanley, J. (1963). *Experimental and quasi-experimental designs for research.* Boston: Houghton Mifflin.
Cardozo, E., Neto, J., Barza, A., França, A. & da Silva, F. (2010). 'Scrum and productivity in software projects: A systematic literature review.' *14th International Conference on Evaluation and Assessment in Software Engineering (EASE)*. Keele University, UK,.
Checkland, P. (1999). *Systems thinking, systems practice.* Chichester: Wiley.
Cross, N., Dorst, K. & Roozenburg, N. (1992). *Research in design thinking.* Delft: Delft University Press.
Dawkins, R. (1989). *The selfish gene.* Oxford University Press.
Ewusi-Mensah, K. (2003). *Software development failures.* MIT Press.
Fitzgerald, B. (2006). 'The transformation of open source software.' *MIS Quarterly,* 30 (3), 587-598.
Freeman, P. & Hart, D. (2004). 'A science of design for software-intensive systems.' *Communications of the ACM,* 47 (8), 19-21.
Gero, J. S. (1990). 'Design prototypes: A knowledge representation schema for design.' *AI Magazine,* 11 (4), 26-36.
Gladden, G. R. (1982). 'Stop the life-cycle, I want to get off.' *SIGSOFT Software Engineering Notes,* 7 (2), 35-39.
Hevner, A. & Chatterjee, S. (2010). *Design research in information systems: Theory and practice.* Springer.
Hevner, A. R., March, S. T., Park, J. & Ram, S. (2004). 'Design science in information systems research.' *MIS Quarterly,* 28 (1), 75-105.
IEEE (1998). 'IEEE standard 830-1998: Recommended practice for software requirements specifications.' [online at http://standards.ieee.org/reading/ieee/std_public/description/se/830-1998_desc.html].
Jacobson, I., Booch, G. & Rumbaugh, J. (1999). *The unified software development process.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.
Kroenke, D., Gemino, A. & Tingling, P. (2010). *Experiencing MIS.* Toronto: Pearson, Prentice Hall.
Laudon, K., Laudon, J. & Brabston, M. (2009). *Management information systems: Managing the digital firm.* Toronto: Pearson, Prentice Hall.
Levina, N. (2005). 'Collaborating on multiparty information systems development projects: A collective reflection-in-action view.' *Information Systems Research,* 16 (2), 109-30.
Markie, P. (2008). 'Rationalism vs. Empiricism', *Stanford Encyclopedia of Philosophy* [online at http://plato.stanford.edu/entries/rationalism-empiricism/].
McCracken, D. D. & Jackson, M. A. (1982). 'Life cycle concept considered harmful.' *SIGSOFT Software Engineering Notes,* 7 (2), 29-32.
Nandhakumar, J. & Avison, D. (1999). 'The fiction of methodological development: A field study of information systems development.' *Information Technology & People,* 12 (2), 176-91.
Newell, A. & Simon, H. (1972). *Human problem solving.* Prentice-Hall, Inc.
Pahl, G. & Beitz, W. (1996). *Engineering design: A systematic approach.* London: Springer-Verlag.
Palvia, P. & Nosek, J. (1990). 'An empirical evaluation of system development methodologies.' *Information Resource Management Journal,* 3 (3), 23-32.
Poppendieck, M. & Poppendieck, T. (2003). *Lean software development: An agile toolkit.* Addison-Wesley Professional.
Ralph, P. (2010a). 'Comparing two software design process theories.' In Proceedings of the Fifth International Design Science Research in Information Systems and Technology Conference. Jun 4-5.
Ralph, P. (2010b). 'Fundamentals of software design science.' *Sauder School of Business:* 177. Vancouver, Canada: University of British Columbia.
Ralph, P. (2011). 'The coevolution gap in software developer education.' Working Paper, Lancaster University, 12 pages.

Ralph, P. & Wand, Y. (2009). 'A proposal for a formal definition of the design concept.' In K. Lyytinen, P. Loucopoulos, J. Mylopoulos & W. Robinson (Eds.) *Design requirements engineering: A ten-year perspective:* 103-36. Springer-Verlag.
Renner, C. H. (2004). 'Validity effect.' In R. F. Pohl (Ed.) *Cognitive illusions: A handbook on fallacies and biases in thinking, judgement and memory:* 201-13. Hove, UK: Psychology Press.
Roozenburg, N. & Eekels, J. (1995). *Product design: Fundamentals and methods.* Chichester, UK: Wiley.
Royce, W. W. (1970). 'Managing the development of large software systems: Concepts and techniques.' In proceedings of Wescon.
Samuelson, W. & Zeckhauser, R. J. (1988). 'Status quo bias in decision making.' *Journal of Risk and Uncertainty,* 1, 7-59.
Schön, D. A. (1983). *The reflective practitioner: How professionals think in action.* USA: Basic Books.
Schwaber, K. (2004). *Agile project management with scrum.* Microsoft Press.
Schwaber, K. & Beedle, M. (2001). *Agile software development with scrum.* Prentice Hall.
Simon, H. A. (1996). *The sciences of the artificial.* Cambridge, MA, USA: MIT Press.
Stacey, P. & Nandhakumar, J. (2008). 'Opening up to agile games development.' *Communications of the ACM,* 51 (12), 143-46.
Standish Group (2006). 'Chaos database: Chaos surveys conducted from 1994 to fall 2004'.
Suchman, L. (1987). *Plans and situated actions: The problem of human-machine communication.* Cambridge University Press.
Trochim, W. (2001). *Reserch methods knowledge base.* Cincinnatti, OH, USA: Atomic Dog Publishing.
Truex, D., Baskerville, R. & Travis, J. (2000). 'Amethodical systems development: The deferred meaning of systems development methods.' *Accounting, Management and Information Technologies,* 10 (1), 53-79.
Van de Ven, A. H. & Poole, M. S. (1995). 'Explaining development and change in organizations.' *The Academy of Management Review,* 20 (3), 510-40.
van Lamsweerde, A. (2001). 'Goal-oriented requirements engineering: A guided tour.' Proceedings of the Fifth IEEE International Symposium on Requirements Engineering. Aug.
Waldrop, M. M. (1992). *Complexity: The emerging science at the edge of order and chaos.* New York, USA: Simon and Schuster.
Yin, R. (2003). *Case study research: Design and methods.* California, USA: Sage Publications.
Zheng, Y., Venters, W. & Cornford, T. (2011). 'Collective agility, paradox and organizational improvisation: The development of a particle physics grid.' *Information Systems Journal*, 21(4), 303-33.