A Methodology Tailoring Model for Practitioner Based Information Systems Development Informed by the Principles of General Systems Theory

Timothy J. Burns
tburns1@ramapo.edu
Anisfield School of Business,
Ramapo College of New Jersey
Mahwah, New Jersey 07430, USA

Fadi P. Deek
fadi.deek@njit.edu
College of Science and Liberal Arts,
New Jersey Institute of Technology
Newark, New Jersey 07102, USA

Abstract

Information system development practitioners tailor system development methodologies to match the specific circumstances of their software projects. This is not surprising as research has shown that information systems development is a highly circumstantial process and that no one system development methodology can be optimal for every context of every project. Several formal techniques such as the contingency factors approach and situational method engineering have been introduced to facilitate the tailoring of system development methodologies to fit the needs of a project. However, there is evidence that system development practitioners have largely neglected these techniques in favor of ad hoc methodology tailoring approaches.

This paper presents a formal methodology tailoring model geared towards the practitioner. The model is based on the principles of general systems theory and is designed to provide practitioner utility, which has been shown to be a determining factor in the employment of a technological innovation.

Keywords: Information System Development Methodologies; Methodology Tailoring; Method Engineering; General Systems Theory

1. INTRODUCTION

An information system (IS) development methodology is defined as a recommended collection of phases, procedures, rules, techniques, tools, documentation, management, and training used to develop a system (Avison & Fitzgerald, 2003, Cockburn, 2006, Hoffer & Valacich, 2010). Over the years numerous IS development methodologies have emerged and many are currently taught in colleges and universities around the world (Burns & Klashner, 2005). While there has been much discussion and debate as to which of these methodologies is best, current research shows that there may not be one optimal methodology that can be universally applied to every project. This is because, while many of the methodologies are beneficial in certain situations, system development is a circumstantial process, and no one methodology will work best for every context of every project (Cockburn, 2006, Fitzgerald, Russo, & O'Kane, 2003).

Background

There have been significant advances and changes to methodologies over the last 30 years. Those changes can be characterized into specific eras that include the pre-methodology era, when no methodologies existed, and the methodology era, when a plethora of new methodologies was introduced (Avison & Fitzgerald, 2003, Fowler, 2005). Some people in the IS field feel that since 2001 we have entered a post-methodology era wherein researchers and practitioners are questioning the older methodologies (Avison & Fitzgerald, 2003, Fowler, 2005). Most of the serious criticism of the methodologies from the methodology era suggests that they are bureaucratic and labor intensive or "heavy" methodologies (Fowler, 2005)

In response to this, new methodologies introduced in the post-methodology period are considered as lightweight or agile methodologies (Fowler, 2005). These agile methodologies are considered by some people in this postmodern era to be "amethodological" (i.e., a negative construct connoting not methodological) (Truex & Avison, 2003). The biggest criticism of the agile methodologies has been the lack of empirical evidence supporting the claims of their benefits and their lack of theoretical foundation (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003). However, there is a growing body of literature both supporting and repudiating the claims of success of the agile methodologies (Abrahamsson et al., 2003, Conboy, Wang, & Fitzgerald, 2009).

Problem Description

Regardless of whether the methodology is "heavy" or "agile", current research suggests that the best methodology for a software development project may be one that has been selected, tailored, or blended (i.e. a hybrid methodology created though the blending of two or more methodologies) (McGregor, 2008) to fit the specificities of the individual system development project (Cockburn, 2006, Fitzgerald et al., 2003). In response to this discovery, several formal "methodology tailoring" (i.e. the process of selecting, tailoring, or blending methodologies) techniques have been introduced. Two examples of formal methodology tailoring techniques are the contingent factors approach and situational method engineering. The contingency factors approach suggests that specific features of the development context should be used to select an appropriate methodology from a portfolio of methodologies. This approach requires developers to be familiar with every contingent methodology or have contingency built in as part of the methodology itself.

A suggested alternative has been a technique called "Method Engineering" (ME) (Fitzgerald et al., 2003, Brinkkemper, 1996). With this technique, a methodology is constructed from a repository of "existing discrete predefined and pre-tested method fragments" (Fitzgerald et al., 2003). Using a method-engineering tool, software developers build a meta-method that is made up of fragments from popular development methodologies. The fragments are each designed to handle a particular contingency inherent to the software project. The fragments are categorized as either product or process. Product fragments are artifacts capturing the structure in deliverables such as diagrams, tables, or models, while process fragments project strategies and detailed procedures (Brinkkemper, 1996).

Method Engineering has several shortcomings. For example, it is impossible to plan for every contingency that may arise, and therefore, critical fragments will always be missing (Rossi, Tolvanen, Ramesh, Lyytinen, & Kaipala, 2000). Also, the burden of selecting the correct fragment falls upon the analyst (Truex & Avison, 2003). Furthermore, a tool is usually required and ME tool development has been a problematic procedure (Fitzgerald et al., 2003). Thus, the evolution of software development methodologies using fragments is problematic.

Both contingency factors and ME techniques have had little success in practical industry applications (Fitzgerald et al., 2003, Rossi et al., 2000). However, ad hoc methodology tailoring (whereby practitioners use an informal process to tailor methodologies to their situation) has been an implied concept in industry (Fitzgerald, 1997). This is problematic because the lack of formality inherent to the ad hoc approach suggests that the knowledge of how to implement the approach is tacit and therefore more difficult to acquire and transfer (Howells, 1996).

As a result, simply stated, the problem is that there is currently no formal, industry accepted, widely used, system development methodology tailoring model (Fitzgerald et al., 2003, Rossi et al., 2000, Fitzgerald, 1997). While the ad hoc methodology tailoring approach may, to date, be the most widely used in industry, its tacit nature impedes the acquisition and transference of knowledge about the approach. Conversely a formalized approach permits the approach to be more easily learned and explained.

The remainder of this paper is devoted to defining a model that solves this problem. The evolution of the model is explained in terms of its utility and theoretical foundation and then a detailed definition of the model is presented. Finally, a sample application of the model is provided so that the reader may gain a complete understanding of its practicality.

2. THE MODEL

It is hypothesized that a model (i.e., an artifact used to abstract and represent phenomena) (Hevner, March, Park, & Ram, 2004, March & Smith, 1995) can be created that will provide a simple, yet formal process whereby practitioners can tailor methodologies to the context of the project. The goal of the model is to provide practitioner utility (i.e., usefulness to system developers working in industry).

It is believed that the success of this model in industry will depend on several conditions. Fitzgerald (1997) demonstrated that practitioners will bypass the use of methodologies simply because they do not see the utility in using them, therefore the model must have a perceived utility to practitioners. The second condition that the model must meet is that it must be based on sound academic theory. In order to accomplish this, a root theory must be found that can be used to explain the model and its concepts. Finally, the model must be evaluated using an accepted methodology and the results must be reported in a statistically accepted manner.

Practitioner Utility

The practitioner model described in this paper can be characterized as a technological innovation. There are several theories and models that can be used to predict the degree to which an innovation will be accepted (Riemenschneider & Hardgrave, 2001). Included in this list would be the Diffusion of Innovations Theory (Rogers, 1995), the Theory of Reasoned Action (TRA) (Fishbein & Ajzen, 1975), the Theory of Planned Behavior (TPB) (Ajzen, 1985), the Technology Acceptance Model (TAM) (Davis, 1989), and TAM2 (Venkatesh & Davis, 2000).

TAM has been proven valid in numerous studies and under a multitude of conditions (Riemenschneider & Hardgrave, 2001). TAM suggests that when users are presented with a new technology, a number of factors influence the decision about how and when they will use it. The two primary factors are perceived usefulness (i.e., the degree to which a person believes that using a particular technology would enhance his or her job performance) and perceived ease-of-use (i.e., the degree to which a person believes that using a particular technology would be free from effort). The TAM2 model extends the TAM model to include social factors (i.e., subjective norm, voluntariness, and image) and cognitive factors (i.e., job relevance, output quality, and results demonstrability) (Venkatesh & Davis, 2000).

Based on TAM2, in order for a practitioner to utilize a methodology tailoring model, they must perceive it to be useful, easy to use, and socially and cognitively acceptable. Informal, ad hoc methodology tailoring meets these requirements given its widespread use in industry (Fitzgerald, 1997). Therefore, it is hypothesized that a formal method tailoring approach that simulates the already accepted, ad hoc practitioner methodology tailoring approach would also be accepted, provided it continues to meet the conditions put forth by TAM2.

Although the literature is insufficient on the question of how practitioners informally tailor methodologies in the field, there are some things that are known. First, practitioners generally take a shorter-term view than academics and tend to emphasize the completion of tasks and the solution of problems (Lippert & Anandarajan, 2004). Second, the methodologies utilized by practitioners are influenced by the universality of the methodology, the methodology introduction process, the experience level of the developer, developer confidence in the methodology, and developer participation with the methodology (Hansen, Jacobsen, & Kautz, 2003).

Based on this information, in order for a formal methodology tailoring model to be utilized by practitioners, it must aid in the completion of tasks and the solution of problems. Also, it must provide universal applicability, have management support, provide utility to both experienced and in-experienced developers, and encourage developer confidence and participation.

Theoretical Foundation

The theoretical foundation for the model comes from General Systems Theory. Hungarian biologist Ludwig von Bertalanffy originally proposed general systems theory in 1928 (von Bertalanffy, 1928) as a reaction against the reductionistic and mechanistic approaches to scientific study, and in an attempt to unify the fields of science. The scientific method is based on the assumptions that an entity can be broken down into its smallest components so that each component can be analyzed independently (reductionism), and that the components can be added in a linear fashion to describe the totality of the system (mechanism). Rather than reducing an entity to the properties of its parts or elements, general systems theory focuses on the arrangement of and relations between the parts that connect them into a whole (holism).

One of the goals of general systems theory was to find common ground upon which scientific study could be conducted across all disciplines. Von Bertalanffy felt that it was futile to try and find a unitary conception of the world by reducing all levels of reality to the level of physics. He felt that the answer to a unitary conception could be found by defining the commonalities among the fields through the discovery of the isomorphy of the laws of the different fields (von Bertalanffy, 1969). Von Bertalanffy thought that the systems that are present in the various fields could identify those commonalities.

Von Bertalanffy defined a system as "complexes of elements standing in interaction". He found that conventional physics dealt only with closed systems (i.e., systems which are isolated from their environment). In particular, the laws of thermodynamics expressly stated that they were intended for closed systems. The essence of the second law of thermodynamics (law of entropy) is that entropy (i.e., the degree of disorder or uncertainty in a system) (von Bertalanffy, 1969) will increase over time in a closed system.

General systems theory realizes that many systems, by their nature, are open systems that interact with their environment. Von Bertalanffy observed that the second law of thermodynamics does not hold true in open systems. He realized that in an open system, the degree of disorder or uncertainty decreases over time or that "negative entropy" occurs (von Bertalanffy, 1969). General systems theory also realizes that open systems have a tendency to self-organize. This is a process in which the internal organization of a system increases automatically without being guided or managed by an outside source (Ashby, 1947). This happens through a process of feedback and decision-making.

An IS development methodology can be considered a "system" (von Bertalanffy, 1969), that is used to develop an information system. IS development is also a problem solving process (DeFranco-Tommarello & Deek, 2002, Highsmith, 2000). This suggests that methodologies are essentially problem solving systems with several common elements including the problems (i.e., the difference between a goal state and the current state of the system (Hevner et al., 2004), which have a hierarchical order (Ahl & Allen, 1996), problem solving processes (i.e., the tools, procedures, processes, etc. that are used to do define and understand problems, plan solutions to problems, implement solutions, and verify and present the results (Deek, Turoff, and McHugh, 1999), solutions (i.e., the answer to or disposition of a problem) (American Heritage Dictionary 2010), feedback (i.e., part of the output is monitored back, as information on the preliminary outcome of the response, into the input) (von Bertalanffy, 1969), and an environment which defines the context, contingencies, constraints, rules, laws, etc. of the organization, people, technology, etc. These systems employ incremental problem solving which involves using intermediate states as intermediate goals in solving problems (Newell & Simon, 1972).

Based on general systems theory, IS development methodologies can be characterized as collaborative, hierarchical, incremental, and problem solving systems. They are open systems that interact with their outer environment (Simon, 1996), which means that they have the propensity for negative entropy. Also, these systems all have a "system state" (Kuhn, 1974) which represents the current condition of system variables (such as the current number of open, unsolved problems in the system).

Model Definition

The practitioner based system development model is depicted in Appendix One. Based on general systems theory, the model tailors and/or combines methodologies, not by breaking the methodologies down into fragments, but by using the concepts that are isomorphic across the methodologies (von Bertalanffy, 1969). Discovering those isomorphic concepts requires abstracting methodologies to a common level. The model suggests that the commonality among all methodologies is their inherent role as problem solving systems.

The practitioner based system development model represents a problem solving system that cyclically iterates among three phases throughout the life of the project. The first phase is the "Describe" phase. It is used to understand the current state of the project. As such, it is a knowledge producing activity (March & Smith, 1995). The goal of this phase is to gain knowledge and to identify a problem or a set of problems that must be solved in order to progress to the next step of the project. It includes analyzing the current environment, identifying circumstances that have changed since the last definition phase, analyzing feedback that was obtained from the previous iteration, analyzing and parsing the list of problems still open at the conclusion of the last cycle, and adding to the list any new problems that can be identified. The knowledge gained through this phase is depicted in Appendix One by the central circle. As the project progresses, the knowledge pool expands and contributes to the actions prescribed in the other two phases.

The second phase is the "Problem Solve" phase. During this phase, solutions are found for the problem(s) identified in the "Describe" phase. If the problem is something simple, for instance a task that needs to be completed, then it can immediately pass to the next phase where an action is prescribed. However, if the problem is complex, then a problem–solving technique must be applied in order to find a solution to the problem. The final phase is the "Prescribe" phase. This is a knowledge using activity (March & Smith, 1995). Using the knowledge gained during the previous two phases the next course of action is prescribed. The next course of action could take virtually any form. It depends on what was identified as the highest priority problem in the "Describe" phase and the solutions discovered in the "Problem Solve" phase. The prescribed action may be a methodology fragment. For instance, it may be determined that the best action at this point in time for the project would be to build a prototype or to create a UML diagram.

It must be pointed out that the principle of equifinality (von Bertalanffy, 1969) holds true in the model. Equifinality is a condition in which different initial conditions lead to similar effects or in which different courses of action lead to similar results. Application of this principle suggests that there are multiple methodologies and instantiations that would fit the model and still produce the desired result.

A Sample Walkthrough of the Model

A sample walkthrough of the practitioner model is illustrated in Appendix Two. This walkthrough is designed to show how system developers can use the model to tailor system development methodologies to a project. The process begins with the "Describe" phase of the model. During this phase, the developers identify the highest priority problem to be the selection of a base system development methodology that will be used to implement the project. For instance, should the developers use a traditional approach such as the waterfall or spiral method or perhaps should the developers use the object-oriented approach or one of the agile methodologies?

The problem then passes to the "Problem Solve" phase where problem solving tools and techniques are used to select a base methodology with core competencies, (i.e., the set of the most strategically significant and value-creating skills in any organized system or person), that most closely match the context of the project and organization. Several key factors contribute to this selection process. For instance, the knowledge and background of the developers, the risk of change inherent to the project, and the visibility of the project development process required by the organization's management will all have to be considered when selecting a development methodology.

The project then progresses to the "Prescribe" phase where the recommended action is to select the base methodology. For this walkthrough, given the factors mentioned previously, the developers decide to implement a traditional SDLC such as the waterfall methodology. Given that this selected methodology provides a framework and not a mandate, only base fragments will be selected to be implemented. So, for instance, only the *phases* of the waterfall approach will be selected but the activities typically inherent to those phases may be supplanted with other "actions" or activities. As an example, typically during the requirements specification phase interviews with system users are conducted. However, using the model, the developers determine that JAD sessions would be a better requirements gathering method for this project.

Once a base methodology has been selected, the model suggests that we should cycle back to the "Describe" phase. For this walkthrough, the developers identify the next problem to be the identification and extraction of the fragments from the base methodology that will serve as a skeleton methodology for the project. The "Problem Solve" and "Prescribe" phases are used to identify these fragments and determine their arrangement in a temporal fashion, with intentional gaps left in the prescribed process. This is represented by the base fragments in figure two.

We continue to cycle through the phases of the model. As we do, we describe problems and then use problem solving mechanisms to identify and prescribe activities that will extend, contribute to, and replace parts of the base methodology. "Extends" and "contributes" alters the base methodology by adding additional activities, while replaces removes a fragment of the methodology and replaces it with an activity (McGregor, 2008). The end goal is to enhance the base methodology and provide a methodology that is more of a custom fit to the project.

The walkthrough continues to follow this cycle throughout the course of the project. The base methodology fragments that were initially extracted as the skeleton methodology serve as anchor points which keep the project grounded. The prescribed actions must be collated within the fragments of the base methodology that were initially prescribed. The methodology can continue to be employed throughout the lifecycle of the project, even after the project as progressed into the maintenance phase.

3. DISCUSSION

The goal of the model is to provide practitioner utility (i.e., usefulness to system developers). The model attempts to reach that goal by presenting a simple process that is intuitive to the system developer and simulates the developer's typical procedure. The hope is that the model will be perceived by developers to be easy to use, and useful, and thus in accordance with the primary conditions set forth by the technology acceptance model. Furthermore, the model is based on a sound academic theory as it draws its basis from general systems theory.

Comparing the model to other known methodology tailoring techniques illustrates its advantages. The inadequacies of the contingency factors approach are apparent (Fitzgerald et al., 2003). It is just not feasible or possible for all the developers in an organization to be familiar with all of the possible methodologies that would work best for a given situation (Fitzgerald et al., 2003). Plus as the contingent factors of the project change over time, so will the optimum methodology.

If method engineering is analyzed through the lens of general systems theory, it becomes apparent that it is both a reductionistic and mechanistic solution to the problem. It is reductionistic in the sense that it attempts to solve the problem by reducing the phenomenon (the methodology) to its smallest component (method fragments) and analyzing the components. It is mechanistic because it attempts to build a whole meta-methodology from the sum of its parts, with no regard for the interrelationships of those parts.

The practitioner model, as specified in general systems theory, presents an anti-reductionistic and anti-mechanistic approach. It seeks to integrate by identifying the isomorphic characteristics of the IS development methodologies. In particular, the model capitalizes on the common inherent problem solving nature of the various methodologies.

4. CONCLUSION

The separation of the IS development methodology community around heavy, proprietary tool oriented approaches versus "amethodological", light, open source approaches distracts us from more basic issues. None of the IS development methodologies that have been developed to date work well in the majority of situations. They all have to be refined and tailored extensively to the actual needs of the development context (Cockburn, 2006, Fitzgerald et al., 2003). The existing accepted approaches to method tailoring (i.e., contingency and ME) have shortcomings as noted earlier.

The model presented in this research directly addresses the problems inherent with other development methodology adaptation approaches. This general systems approach facilitates an IS community effort to normalize system development methodologies. The adherence to design science guidelines lends itself to the legitimacy of the model. Practitioners who use this method will not have to learn methodologies that are not normalized. Thus, they will have a shorter learning curve to implement this technique versus the other method tailoring techniques. Our research community can work collaboratively to reduce ambiguity in methodologies by using the theoretical foundation presented here.

Future research is needed in several areas. First, lab experiments are needed to validate the model. Second, field experiments are needed that will test the model in a realistic setting and against other popular methodologies and approaches. Finally, specific methodologies and instantiations of the model need to be developed and evaluated accordingly.

5. REFERENCES

Abrahamsson, P., Warsta, J., Siponen, M., Ronkainen, J., "New Directions on Agile Methods: A Comparative Analysis", IEEE. 2003.

Ahl, V., Allen, T. F. H. "Hierarchy theory, a vision, vocabulary and epistemology", Columbia University Press. 1996.

The American Heritage Dictionary of the English Language, Fourth Edition, Houghton Mifflin Company, 2010.

Ashby, W.R., "Principles of the Self-Organizing Dynamic System", Journal of General Psychology, 1947, volume 37, pages 125—128.

Avison, D., Fitzgerald, G., "Where Now for Development Methodologies?", Communications of the ACM, 46,1 2003, 79-82.

Ajzen,I., "From intention to actions: a theory of planned behavior, in: J. Kuhl, J. Beckmann (Eds.), Action Control: From Cognition to Behavior", Springer-Verlag, New York, NY, 1985, pp. 11-39.

Brinkkemper, S., "Method Engineering: engineering of information systems development methods and tools" Elsevier Science B.V. 1996.

Burns, T., Klashner, R. "A Cross-Collegiate Analysis of Software Development Course Content", Proceedings of the 6th Conference on Information Technology Education, Newark, NJ, USA, pp. 333-337. 2005.

Cockburn, A., "Agile Software Development" 2nd edition, Addison-Wesley, 2006.

Conboy K, Wang X, Fitzgerald B. Creativity in Agile Systems Development: A Literature Review. In: Information Systems – Creativity and Innovation in Small and Medium-Sized Enterprises. Vol 301/2009. Springer; 2009. p. 122-34. (IFIP Advances in Information and Communication Technology ; vol 301/2009).

Davis, F.D., "Perceived usefulness, perceived ease of use, and user acceptance of information technology", MIS Quarterly 13, 1989, pp. 318-339

Deek, F.P., Turoff, M., McHugh, J., "A Common Model for Problem Solving and Program Development", Journal of the IEEE Transactions on Education, Volume 42, Number 4., pp. 331-336, November 1999.

DeFranco-Tommarello, J., Deek, F., "Collaborative Software Development: A discussion of Problem Solving Models and Groupware Technologies" Proceedings of the 35th Annual Hawaii International Conference on System Sciences. IEEE. 2002.

Fishbein, M., Ajzen, I., "Belief, Attitude, Intention, and Behavior: An Introduction to Theory and Research", Addison-Wesley, Reading, MA, 1975.

Fitzgerald, B., "The use of systems development methodologies in practice: A field study", The Information Systems J. 7, 3, 201–212 1997.

Fitzgerald, B., Russo, N., O'Kane, T., "Software Method Tailoring at Morotola", Communications of the ACM, 46, 4, 64-70 2003.

Fowler, M., "The New Methodology", WWW http://martinfowler.com/articles/newMethodology.html, Accessed May 18, 2005, 8pm.

Hansen, B., Jacobsen, D., Kautz, K. "Systems Development Methodologies in Practice", in Proceedings of the Information Systems Development Conference, Melbourne, Australia, August 25-27, 2003.

Hevner, A., March, ST, Park, J., Ram, S. "Design Science Research in Information Systems", MIS Quarterly (28:1) March 2004, pp. 75-105.

Highsmith, J., "Adaptive Software Development - A Collaborative Approach to Managing Complex Systems", Dorset House Publishing, New York, NY 2000.

Hoffer,J., George, J., Valacich,J., "Modern Systems Analysis & Design", Sixth Edition, Prentice Hall, 2010.

Howells, J., 'Tacit Knowledge, Innovation, and Technology Transfer", Technology Management and Strategic Management, Vol. 8, No. 2, 1996

Kuhn, A., "The Logic of Social Systems", San Francisco: Jossey-Bass. 1974.

Lippert, S. K., Anandarajan, M. "Academic vs. practitioner systems: Planning and analysis". Association for Computing Machinery. Communications ofthe ACM, 47(9), 91. 2004.

March, S., Smith, G, "Design and Natural Science Research on Information Technology." Decision Support Systems 15 (1995): 251 - 266. 1995.

McGregor, J., "Mix and Match", in Journal of Object Technology, vol. 7 no. 4, July-August 2008, pp 7 - 16

Newell, A., Simon, H., "Human Problem Solving", Prentice Hall 1972.

Riemenschneider, C., Hardgrave, B., "Explaining Software Development Tool Use with The Technology Acceptance Model", Journal of Computer Information Systems 41 (4), 2001, pp. 1-8.

Rogers, E., "The Diffusion of Innovations, Fourth ed., Free Press, New York, NY, 1995.

Rosnow, R. L., & Rosenthal, R. (2005). "Beginning behavioral research: A conceptual primer" (5th ed.). Upper Saddle River, NJ: Prentice Hall.

Rossi, M., Tolvanen, J.-P., Ramesh, B., Lyytinen, K., Kaipala, J., "Method Rationale in Method Engineering", Proceedings of the 33rd Hawaii International Conference on System Sciences. 2000.

Simon, H., The Sciences of the Artificial, Third Edition. Cambridge, MA, MIT Press 1996.

Truex, D., Baskerville, R., Travis. J., "Amethodical Systems Development: The Deferred Meaning of Systems Development Methods", Journal of Accounting, Management, and Information Technologies, Tech 10. pp 53-79 2000.

Truex, D., Avison, D., "Method Engineering: Reflections on the Past and Ways Forward", Ninth Americas Conference on Information Systems, 2003.

Venkatesh, V., & Davis, F. D., "A theoretical extension of the technology acceptance model: Four longitudinal field studies", Management Science, (46:2), 186-204 2000.

von Bertalanffy, L., "Kritische theorie der Formbildung", Borntraeger. 1928.

von Bertalanffy, L. General System Theory, Braziler, New York, 1969.

APPENDIX ONE

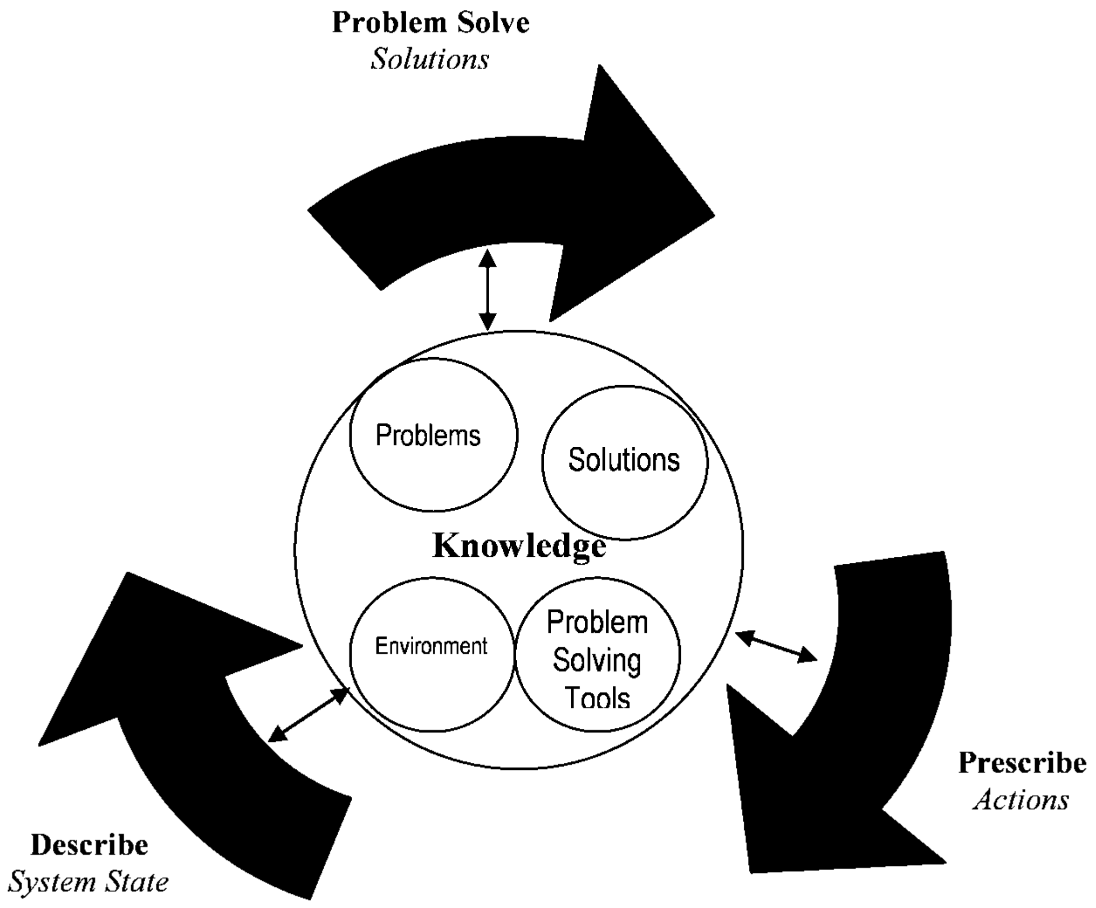

Figure 1 A Practitioner Based System Development Model.

APPENDIX TWO

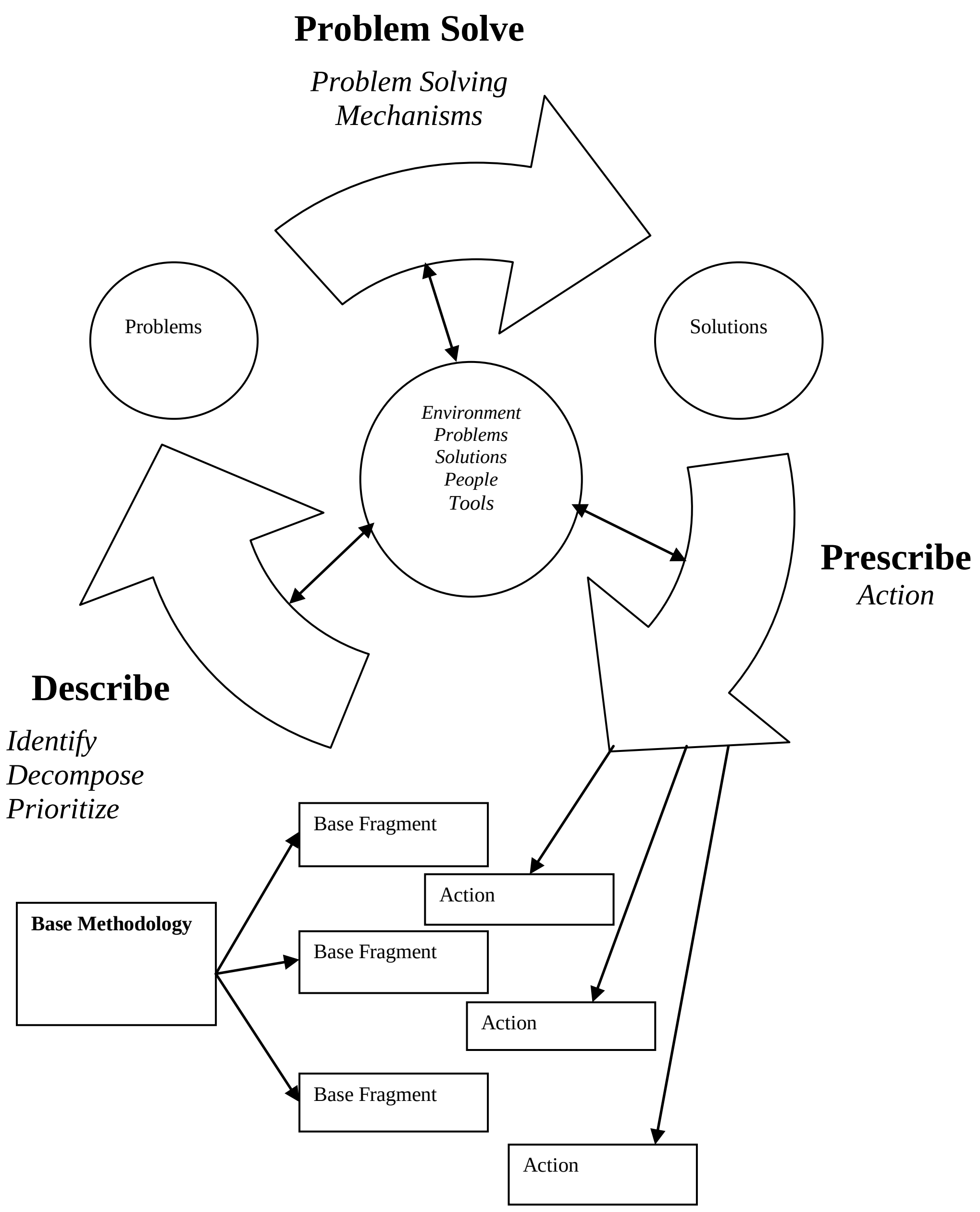

Figure 2 A sample walkthrough of the model.