Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2006 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2006

An Empirical Investigation of a General System Development Model

Timothy Burns
Ramapo College of New Jersey

Robb Klashner
New Jersey Institute of New Jersey

Fadi Deek
College of Science and Liberal Arts- New Jersey Institute of Technology

An Empirical Investigation of a General System Development Model

Timothy Burns
Ramapo College of New Jersey
tburns1@ramapo.edu

Robb Klashner
New Jersey Institute of New Jersey
klashner@njit.edu

Fadi P. Deek
New Jersey Institute of New Jersey
fadi.deek@njit.edu

ABSTRACT

Over the years a multitude of Information Systems development methodologies have emerged. While many of these methodologies have shown promise, prior research has shown that system development is a highly circumstantial process, and that no one methodology will be optimal for every context of every project. Research has also shown that system development practitioners have been using an ad hoc approach to modify formal methodologies in order to create a better fit for their circumstances.

This paper presents a more formal approach. A new model is described, based on the principles of general systems theory, which can be used to adapt existing information system development methodologies through the identification of general isomorphic properties.

The results are reported from an experiment that was conducted to measure the model in terms of developer satisfaction with the finished product, the development process, and the problem solving process. Also, the results of an evaluation by expert judges on finished products created using the model are shown. These results show that there is promise in the new model and that more research needs to be conducted.

Keywords

Information Systems Development, Information System Development Methodologies, Software Engineering

INTRODUCTION

This research is motivated by the current state of software development projects. There is little doubt that there is a software crisis (Brooks 1987). According to a study of 280,000 development projects conducted by the research firm "The Standish Group" in 2000, twenty three percent of all software development projects are cancelled before they finish and forty nine percent of software development projects are completed late, over budget, or do not meet the requirements of the users and/or the organization in which they are implemented. Only about one in four (twenty eight percent) of all software development projects are considered successful (SOFTWAREmag.com 2001, Whiting 1998, Standishgroup.com 1994).

Research has also shown that there is a divide between the IS development practices being developed and taught in academia and those used by practitioners (Fitzgerald 1997, 2003, Burns and Klashner 2005). There is a plethora of IS development methodologies and approaches, all of which claim to be the "silver bullet" (Brooks 1987) to the software crisis. In many cases they have the potential to be the silver bullets if used in the right environment. However, research has shown that none is the silver bullet all the time (Cockburn 2002, Fitzgerald 2003).

The goal of this research is to offer a solution to this problem in the form of a model. It is the aim of this model to be inclusive of the various IS development methodologies and to serve as a guide to IS development.

Method Tailoring

A methodology in IS development has a common definition suggesting that it is a recommended collection of phases, procedures, rules, techniques, tools, documentation, management, and training used to develop a system (Avison and Fitzgerald 2003, Cockburn 2002, Hoffer, George, and Valacich 2005). There have been significant advances and changes to

methodological techniques over the last 30 years and those changes can be characterized into specific eras (Avison and Fitzgerald 2003, Fowler 2002).

The advent of codified methodologies (e.g., Waterfall) marked the early methodology period. During this era many new methodologies were introduced. Methodologies from this era can be classified according to significant approaches such as structured (Yourdon and Constantine 1979), data-oriented, process-oriented, prototyping (Naumann and Jenkins 1982), participative (Mumford 1981), and object-oriented (Coad and Yourdon 1991).

People argue that in recent years we have entered a post-methodology era wherein researchers and practitioners are questioning the older methodological philosophies (Avison and Fitzgerald 2003, Fowler 2002). Most of the serious criticism of the prior approaches is that they are bureaucratic and labor intensive or "heavy" methodologies (Fowler 2002). In response to this, new methodologies introduced in "post-methodology" period are considered as lightweight or agile methodologies because of the lack of bureaucratic contingencies inherent to the heavy methodologies (Fowler 2002). These agile methodologies are considered by many in this postmodern era to be "amethodological" (i.e., a negative construct connoting not methodological) (Truex, Baskerville, and Travis 2000). 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, and 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).

The research has pointed out three distinct areas of concern:

1. No one methodology can claim to be the best methodology for every software project. (Cockburn 2002, Fitzgerald 2003)
2. Present day developers are becoming increasingly discouraged with the traditional methodologies and their shortcomings. (Avison and Fitzgerald 2003)
3. Those developers that have continued to use the traditional methodologies have modified and adapted those methodologies to meet the specificities of the project. (Fitzgerald 1997, 2003).

One concept that has shown much promise is the idea of tailoring a methodology to the actual project development context (Fitzgerald 2003). 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) (Brinkkemper 1996, Fitzgerald 2003). With this technique, a methodology is constructed from a repository of "existing discrete predefined and pre-tested method fragments." (Fitzgerald 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, Ramesh, Lyytinen, and Tolvanen 2004). Also, the burden of selecting the correct fragment falls upon the analyst, possibly aided by a method engineering tool (Truex and Avison 2003). Method engineering tool development is a problematic procedure (Fitzgerald 2003), thus, evolution of software development methodologies using fragments is problematic.

Both contingency factors and ME techniques have had little success in practical industry applications. (Fitzgerald 2003, Rossi et al 2004). Ad hoc methodology tailoring, however, has been used for many years by practitioners in industry (Fitzgerald 1997). Many popular methodologies such as the Rational Unified Process (Jacobson, Booch, and Rumbaugh 1999) nearly mandate tailoring through modification and adaptation in order for them to work in a specific context.

THE MODEL

Theoretical Foundation

The contribution of this research is a general system development model. Models are used to abstract and represent phenomena (Hevner, March, Park, and Ram 2004, March and Smith 1995). The model introduced in this research will draw its theoretical foundation from the General Systems Theory. General systems theory is interdisciplinary and used to study

systems as a whole. It focuses on the complexity and the interdependence of the parts of a system in order to construct a common model. "General Systems Theory is a name which has come into use to describe a level of theoretical model-building which lies somewhere between the highly generalized constructions of pure mathematics and the specific theories of the specialized disciplines." (Boulding 1956)

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. General systems theory focuses on the arrangement of and relationships between the parts of a system that allow them to function as a whole (holism). "It is necessary to study not only parts and processes in isolation, but also to solve the decisive problems found in organization and order unifying them, resulting from dynamic interaction of parts, and making the behavior of the parts different when studied in isolation or within the whole..." (von Bertalanffy 1969).

One of the goals of general systems theory was to find common ground upon which scientific study could be conducted across several disciplines. Von Bertalanffy felt that this could be accomplished not by breaking entities down, but by finding laws that were common among the various fields. "A unitary conception of the world may be based, not upon the possibly futile and certainly farfetched hope finally to reduce all levels of reality to the level of physics, but rather on the isomorphy of the laws in different fields" (von Bertalanffy 1969).

If we consider the hundreds of IS development methodologies (Fitzgerald 2003) as "different disciplines" within the spectrum of IS development methodologies, then perhaps general systems theory holds the key to their unification. General systems theory: "…aims to point out similarities in the theoretical constructions of different disciplines, where they exist, and to develop theoretical models having applicability to at least two different fields of study" (Boulding 1956).

Definition of the Model

The model is called a "Collaborative, Hierarchical, and Incremental Problem-Solving" model (CHIPS). Like general systems theory, the intent of CHIPS is to provide a unification model across the disciplines of its domain. In the case of the CHIPS model, the "disciplines" refer to the plethora of system development methodologies and approaches. Also like general systems theory, the CHIPS model seeks to unify the disciplines, not by breaking the disciplines down, but by discovering the concepts that are isomorphic across the disciplines (von Bertalanffy 1969).

In order to properly define the CHIPS "model" the CHIPS "constructs" will first be defined. Constructs are the basic language of concepts from which phenomena can be characterized (March and Smith 1995). These concepts can then be combined into higher order constructions (i.e. models) used to describe tasks, situations, or artifacts (March and Smith 1995).

The constructs of the CHIPS model include five classes of elements that are common across all system development projects. The five classes are: problems, solutions, people, problem solving mechanisms, and environmental concerns. Defined in depth, the five constructs would be:

1. Problems - CHIPS assumes that all IS development projects are made of a series of tasks that must be completed and problems that must be solved in order to complete the project (Ginat 2002). A task can be defined as a function to be performed or an objective (American Heritage Dictionary 2000). A problem can be defined as the difference between a goal state and the current state of the system (Hevner et al 2004). For this model, the two terms are synonymous and are thus both defined as a problem. The problems can fall into one of three sub-classes:
    a. Solved – Problems that are known and solved.
    b. Open – Problems that are known but have not yet been completely solved. In accordance with the concept of hierarchic order prevalent in general systems theory (von Bertalanffy 1969), many of the problems in a system development project are hierarchical in nature. A hierarchy is a collection of parts with ordered asymmetric relationships inside a whole (Ahl and Allen 1996). A problem may be made up of one or several subordinate problems. To successfully solve a problem, the subordinate problems must also be solved.
    c. Unknown – Problems that the developers do not know about yet, but assume will appear.
2. Solutions – A solution is the answer to or disposition of a problem (American Heritage Dictionary 2000). It can be in the form of an action, a methodology fragment, a tool, a process, a product, etc. The solutions can fall into one of four categories:
    a. Used Solutions – Solutions that have already been used in solving a project problem.

   b. Viable solutions - Solutions that can potentially be used to solve open problems for this project. This category overlaps with the used solutions category because potential viable solutions can come from the class of solutions that have already been used to solve a problem.

   c. Solutions Not Needed – these are the solutions that do not apply to this project.

3. People. – These are the stakeholders (i.e. a person or organization that has a legitimate interest in a project or entity) of the development project There are three classes of stakeholders:

   a. Primary Stakeholders – these are the people that have a direct interest in the project such as users, developers, consultants, vendors, etc.

   b. Secondary Stakeholders – these are people who have a less direct interest in the project. For instance, the manager of the department where the user of the system resides or the CEO of the organization.

   c. Tertiary Stakeholders – These are people with a very small interest in the project. This could potentially include everyone in the world.

4. Problem Solving Mechanisms – These are the tools, procedures, processes, etc. that are used to do the following: (Deek et al 1999)

   a. Define and understand problems

   b. Plan solutions to problems

   c. Implement solutions

   d. Verify and present the results

5. Environment - In accordance with general systems theory, the CHIPS model operates within an environment. The environment defines the circumstances, constraints, rules, laws, etc. of the organization, project, people, technology, etc. Simon discusses an inner environment, an outer environment, and the interface between the two that meets certain desired goals (Simon 1996). The outer environment consists of external forces and effects that act on the artifact. The inner environment is the set of components that make up the artifact and their relationships (i.e. the organization) of the artifact.

Using the constructs of the CHIPS model, the model can now be defined. The CHIPS model has the following phases, (as demonstrated in figure 1), that are repeated in an iterative manner throughout the life of the project:

The "Describe" phase is used to understand the current state of the project. It is a knowledge producing activity (March and Smith 1995). It includes analyzing the current environment (in terms of the five constructs) and 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 cycle, and adding to the list any new problems that can be identified. The list of open problems is then broken down into sub-problems, which are prioritized and arranged in a hierarchy.

The "Problem Solve" phase is used to solve the highest priority problem in the list of open problems. If the problem is something simple, for instance a task that needs to be completed, then it can immediately pass to the next phase. However, if the problem is complex, then a problem–solving technique must be applied in order to collaboratively find a solution to the problem. The solution to the problem may be a methodology fragment. For instance, it may be determined that the best solution at this phase would be to build a prototype or to create an ER diagram.

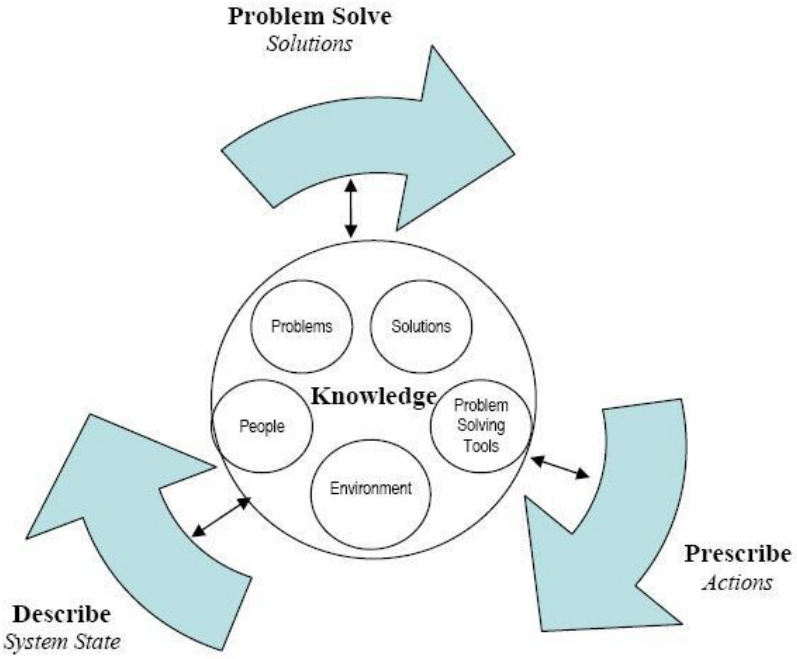

Figure 1 – A General System Development Model

The final phase is to "Prescribe". This is a knowledge using activity (March and Smith 1995). Using the knowledge gained during the previous two phases the next course of action is prescribed. The problem that is solved by completing the action is now marked as a solved problem and the solution is recorded in the knowledge base.

Advantages of the CHIPS Model

Table 1 lists several popular IS development methodologies and approaches. While the table does not provide an exhaustive list of the hundreds of available methodologies (Fitzgerald 2003), it does point of the differences, advantages and disadvantages off the ones listed. Every software development project is unique in terms of culture, people, length, etc. (Cockburn 2002). For this reason, methodologies should be "custom fit" to a project depending on contingencies (Cockburn 2002, Fitzgerald 2003). Research has also shown that no one software methodology is best in all circumstances (Cockburn 2002, Fitzgerald 2003). Circumstances may even change during the life of a project that could affect the methodology that should be used at a particular phase of the project.

The goal of the CHIPS model is to provide a means by which the best possible IS development methodology (or fragment thereof) is utilized at the best possible time, based on the context, contingencies, and constraints of the project. We have previously defined two formalized approaches that are prior attempts to fulfill those requirements; the contingency factors approach (where a methodology is chosen for a project based on a set of contingent factors) (Avison 2003, Fitzgerald 2003) and method engineering (where methodologies are decomposed into a repository of methodology fragments and then re-assembled into met-methodologies of optimum fragments using a special tool) (Brinkkemper 1996). There is also one more approach that is informal in nature. That is the ad hoc methodology tailoring that takes place by practitioners in industry as they adapt methodologies to match their circumstances (Fitzgerald 2003). However, it is the goal of this research to present a more formalized model than this ad hoc methodology.

The inadequacies of the contingency factors approach are apparent (Fitzgerald 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 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 little regard for the interrelationships of those parts.

CHIPS is holistic, anti-reductionistic, and anti-mechanistic. It seeks to integrate I.S. development methodologies by identifying and capitalizing on their isomorphic characteristics. In an open system, negative entropy occurs through a process of decision making and feedback (von Bertalanffy 1969). Therefore it stands to reason that a model that focuses on problem solving would be one that leads to higher levels of order over time.

A Sample Implementation of the CHIPS Model

The potential of the CHIPS model can be demonstrated by analyzing a fictitious development project. Using the CHIPS model, the project is initiated by executing the first define phase. A list of open problems is created, decomposed, and arranged in a hierarchy. As a result the highest priority problem that is identified is to select a base system development methodology for the project.

The CHIPS model now mandates that the developers execute the problem-solving phase. The project is one that requires a methodology that provides a very high level of visibility to management and to the users. After applying several problem solving techniques, the highly visible waterfall methodology is chosen. The development staff is familiar with the Rational Unified Process (RUP) (Jacobson, Booch, and Rumbaugh 1999) and there is a strong chance that the system requirements will undergo several changes throughout the life of the project, so the developers would also like to integrate some RUP features into the project.

A plan is laid out that includes the typical phases of the waterfall methodology. The initial SDLC phase (system investigation) is then completed using only the method fragments inherent to the waterfall approach. Throughout this phase and all subsequent others the project iterates through the define, problem solve, and prescribe activities of the CHIPS process.

During the analysis phase, data collection is performed using the traditional methods (i.e. interviews, surveys, questionnaires, etc.). However, instead of using traditional modeling techniques (i.e. data flow diagrams, flowcharts, E-R diagrams, etc.) to abstract the data collected, RUP modeling techniques are used. For instance, class diagrams, state diagrams, and activity diagrams would be used to demonstrate the system.

The project eventually progresses to the design phase. Both the logical and physical designs are accomplished through the employment of RUP techniques to model the new system's features and the development team's understandings and agreements. Using RUP, the team designs key objects and classes of objects in the new system. This process involves consideration of the problem domain, the operating environment, and the user interface. The problem domain involves the classes of objects related to solving a problem or realizing an opportunity.

During the design phase, the development team also needs to consider the sequence of events (scenarios) that must happen for the system to function correctly. This sequence of events can be diagrammed in a sequence diagram. The RUP concepts of iteration and incrementation can also be implemented, as the system design is refined through continuously adding more detail to the UML diagrams.

Eventually the project progresses to the implementation phase. Using RUP, the object model begun during analysis and completed during design, is turned into a set of interacting objects in a system. Object-oriented programming languages are used to create classes of objects that correspond to the models (or CASE tools such as Rational Rose are employed).

Using the RUP paradigm, the initial implementation is evaluated by users and improved. Additional objects and scenarios are added incrementally as the project iterates through this phase. Eventually a complete, tested, and approved system is available for use. Once the system enters the maintenance phase, the project reverts back to the traditional waterfall approach, with the exception that the project continues to follow the define, problem solve, prescribe cycle of CHIPS.

Methodology	Characteristics	Advantages	Disadvantages
Waterfall	Project is divided into distinct phases	Ability to quantify an abstract subject. Especially useful to management.	Costly if requirements change. No product until late in the process
Prototype	Develop an initial model of the system for user feedback	Allows the user to provide feedback before a large investment made	Lack of documentation and poor project visibility
Spiral	Successive refinement of requirements and design	Incorporates risk	Can incur endless fiddling
Joint Application Development	Workshops where users and I.S. people meet to define system	Brings important participants of a project together to hash out the details	Can be costly and time consuming
Rapid Application Development	Development life cycle designed for faster development and higher quality	Quicker development and less development costs	Higher risk of low quality systems.
OO	Decompose a problem into objects that encapsulate data and behavior	Better modeling, analysis & design. Improved communication	Increased development time
Extreme Programming (XP)	Short cycles, incremental planning, flexible schedule	Increased customer involvement, teamwork, & communication	Code centered vs. design centered. Lack of documentation
Crystal	People-centric, ultra-light, shrink to fit	Takes into account current cultural conditions	Requires constant fine tuning.
Scrum	Lightweight, iterative, incremental process	Increased productivity and adaptability	Poor resource estimating. Limited to small teams.
Feature Driven Development	Design and Build by feature Model-driven short-iteration process.	More planning then rest of agile methodologies	Increased complexity
DSDM	Works under time constraints of RAD	Increased user satisfaction. RAD time and cost savings	Requires special training to use (not free). Complex.
Adaptive	Based on complex adaptive systems. Emergence vs. determinism	Adaptive to change.	Poor resource estimating.
Open Source	Collaborative Development	Increased quality and reliability.	Adapting the process to business/commercial applications

Table 1 – Summary of Popular Software Development Methodologies/Approaches

RESEARCH METHODOLOGY

In order to evaluate the CHIPS IS development model a laboratory experiment was conducted. The experiment was based on a prior experiment that was conducted by Alavi (Alavi 1984). In the Alavi experiment the prototyping approach to system development was evaluated. The CHIPS experiment followed a similar procedure as the Alavi experiment in that graduate IS students were used to simulate system developers and MBA students were used to simulate business users. The system developers were divided into groups of development teams (3 or 4 students per team), with the teams randomly assigned to one of four treatments. A total of 13 teams and forty subjects participated in the experiment. Two MBA students played the role of business owners/users during the development process and then were expert judges for the finished development projects.

		Use of CHIPS	
Change To Requirements		No CHIPS & No Changes to Requirements	CHIPS & No Changes to Requirements
		No CHIPS & Late Stage Changes to Requirements	CHIPS & Late Stage Changes to Requirements

Table 2 – A 2 x 2 Experiment to Evaluate CHIPS

Table 2 demonstrates the design of the experiment. The two independent variables that were manipulated were the use of CHIPS (used vs. not used) and the introduction of requirements changes in late stages of the development process (changes vs. no changes). This design allowed the subjects to be assigned to one of four treatments:

1. Do not use CHIPS and have no changes to requirements.
2. Do not use CHIPS and have late stage requirements changes.
3. Use CHIPS and have no changes to requirements.
4. Use CHIPS and have late stage requirements changes.

The experiment measured four dependent variables:

1. Subjects' satisfaction with the finished design.
2. Subjects' satisfaction with their group's problem solving capabilities.
3. Subjects' satisfaction with the development process.
4. Expert judges' rating of finished design.

The task assigned to all teams was to design an e-commerce web site. The task involved only design; no program coding or implementation took place. All of the groups were given the same initial requirements of the web site. They were asked to design the web site and describe their design using graphical representations of the web pages, documentation, diagrams, and tables.

Before they began, , all of the subjects were given instruction in various problem solving techniques, but told that they were not required to use those techniques. Subjects assigned to use the CHIPS model were given additional instruction in the "describe", "problem solve", "prescribe" process of the model and were asked to periodically complete a specially designed worksheet that would monitor their progress. Subjects not assigned to use the CHIPS model were left on their own to decide what methodology (if any) to use in the development process and what techniques to use in monitoring their team's progress. The subjects were then given two weeks to complete the design. Two of the treatments were given a significant change to the requirements after the first week.

Once the subject teams completed and submitted their final projects, they were asked to complete a final post-test questionnaire that measured subjects' satisfaction with the development process, the finished design, and their group's problem solving capabilities. The final projects were then submitted to the expert judges for grading/ranking. Each judge

was given a subset of the projects to grade at first and then eventually the projects were rotated between the judges so that each judge graded each project. The judges were not allowed to confer with each other during the grading period so that judges would not influence each other's grading.

Results

Once the data had been collected, statistical tests were performed. The questionnaire was validated using Chronbach's alpha. The expert judges' reliability was validated by performing a bivariate Pearsons two tailed test of correlation and a paired samples t-test of significance. Tests of significance were then performed on the variables. The results are displayed in tables three, four, and five.

	Satisfaction with Final Design	Satisfaction with Problem Solving	Satisfaction with Development Process	Expert Judges Ranking
Non-CHIPS	6.04	5.68	5.87	87.75
CHIPS	6.29	6.04	6.11	88.21
Probability	0.139	0.05	0.119	0.45

Table 3 – Non-CHIPS versus CHIPS – Means and t-test probability

Scores measured on 7-point Likert Scales (7=High) for columns 2-4

Scores measures on scale of 1-100 for column 5

The difference between the means is statistically significant at $p<.05$

	Satisfaction with Final Design	Satisfaction with Problem Solving	Satisfaction with Development Process	Expert Judges Ranking
Non-CHIPS/No Change	6.00	5.58	5.75	89.50
CHIPS/No Change	6.43	6.40	6.29	83.50
Probability	0.12	0.004	0.038	0.818

Table 4 – Non-CHIPS/No Change versus CHIPS/No Change – Means and t-test probability

	Satisfaction with Final Design	Satisfaction with Problem Solving	Satisfaction with Development Process	Expert Judges Ranking
Non-CHIPS/Change	6.10	5.80	6.00	86.00
CHIPS/Change	6.20	5.78	5.98	92.92
Probability	0.374	0.524	0.528	0.019

Table 5 – Non-CHIPS/Change versus CHIPS/Change – Means and t-test probability

DISCUSSION

The discussion begins with three limitations of the experiment. First, the sample size of this initial experiment was very small. Second, a laboratory experiment could in no way simulate all of the circumstances that would be involved in a typical system development project, and third, the subjects were asked only to perform design tasks. No implementation was performed. However, given those limitations, the results from the lab experiment did identify several areas were there were statistically significant differences between the use of the CHIPS model versus not using the CHIPS model, even though the CHIPS developers generally scored higher in almost all of the areas analyzed.

As for the first dependent variable, developer satisfaction with the solution, there were no significant differences between CHIPS and non-CHIPS developers. This held true regardless of whether late stage changes were introduced or not. However, the second dependent variable measured, developer satisfaction with their problem solving process did show some significant differences between CHIPS and non-CHIPS developers.

CHIPS developers, overall, were more satisfied with their problem solving process than those developers who did not use CHIPS. This would stand to reason as the CHIPS model is essentially a problem solving system. This rule also held true when there were no late stage changes introduced to the project. However, surprisingly, when there were late stage changes introduced to the project there was no significant difference between the way CHIP and non-CHIPS developers felt about their problem solving process.

There were also some significant differences measured in the third dependent variable, developer satisfaction with the development process. In particular, developers were more satisfied with the development process when there were no late stage changes and they used CHIPS as their development model. However there was no statistically significant difference with regard to developer satisfaction with the development process for CHIPS versus non-CHIPS users overall or if there were late stage changes introduced.

The last dependent variable that was measured was the expert judges rating of the finished projects. When there were late stages introduced to the project, there was a very significant difference in CHIPS developers versus non-CHIPS developers, with CHIPS developers scoring significantly higher. This would make sense as the essence of CHIPS is to make the developers better able to adapt to change and thus provide a better product when there is change late in the development process.

The results of this initial experiment, while promising, are significant for several reasons. First, mean scores were higher for CHIPS developers than they were for non-CHIPS developers for almost all of the dependent variables. This can tentatively be used as an indicator that CHIPS has a role as a legitimate model. Second, there were areas that showed a statistically significant improvement of CHIPS developers over non-CHIPS developers. Given the limitations of the experiment elaborated earlier, this could be an indicator that CHIPS has the potential to be a model that improves the development process. Finally, the results of this experiment give support to the argument that CHIPS can be used to integrate several system development methodologies, not by breaking those methodologies down into method fragments, but by identifying the isomorphic characteristics of those methodologies.

Future research is needed in several areas. First, additional lab experiments are needed to replicate this initial experiment, but with larger sample sizes. Second, field experiments are needed that will test the CHIPS model in a more realistic setting and against other popular methodologies and approaches. Finally, specific methodologies and instantiations of the CHIPS model need to be developed and evaluated accordingly.

REFERENCES

1. Abrahamsson, P., Warsta, J., Siponen, M., Ronkainen, J.(2003) “New Directions on Agile Methods: A Comparative Analysis”, IEEE.
2. Ahl, V., Allen, T. F. H.(1996) “Hierarchy theory, a vision, vocabulary and epistemology”, Columbia University Press.
3. Alavi, M.(1984) “An assessment of the prototyping approach to information systems development”, Communications of the ACM, Volume 27 Issue 6.
4. The American Heritage Dictionary of the English Language (2000), Fourth Edition, Houghton Mifflin Company.
5. Avison,D., Fitzgerald, G. (2002) ”Information Systems Development: Methodologies, Techniques and Tools”, Third Edition McGraw-Hill/Irwin.
6. Avison,D., Fitzgerald, G. (2003) "Where Now for Development Methodologies?", Communications of the ACM, 46,1 2, 79-82.

7. Boulding, K. E. (1956) "General Systems Theory -- The Skeleton of Science," Management Science, 2(3), 197-208.
8. Brinkkemper, S. (1996) ”Method Engineering: engineering of information systems development methods and tools” Elsevier Science B.V.
9. Brooks Jr., F. (1987) ”No Silver Bullet, Essence and Accidents of Software Engineering”, Computer Magazine.
10. Burns, T., Klashner, R. (2005) “A Cross-Collegiate Analysis of Software Development Course Content”, Proceedings of the 6th Conference on Information Technology Education, Newark, NJ, USA, pp. 333-337.
11. Coad, P. Yourdon, E. (1991) "Objected Oriented Analysis”, Yourdon Press Computing Series, NJ.
12. Cockburn, A. (2002) "Agile Software Development", Addison-Wesley.
13. Deek, F.P., Turoff, M., McHugh, J.(1999) "A Common Model for Problem Solving and Program Development", Journal of the IEEE Transactions on Education, Volume 42, Number 4., pp. 331-336.
14. Fitzgerald, B. (1997) "The use of systems development methodologies in practice: A field study”, The Information Systems J. 7, 3, 201–212.
15. Fitzgerald, B. (2003) "Software Method Tailoring at Motorola", Communications of the ACM, 46,4.64-70.
16. Fowler, M. (2002) "The New Methodology”, http://martinfowler.com/articles/newMethodology.html, Accessed May 18, 2005, 8pm.
17. Ginat, D. (2002) “On Varying Perspectives of Problem Decomposition”, SIGCSE ’02, February 27- March 3, 2002, Covington, KY, ACM.
18. Hevner, A., March, ST, Park, J., and Ram, S. (2004) “Design Science Research in Information Systems,”. MIS Quarterly (28:1), pp. 75-105.
19. Hoffer,J., George, J., Valacich, J. (2005) "Modern Systems Analysis & Design”, Fourth Edition, Addison-Wesley.
20. Jacobson, I., Booch, G., Rumbaugh, J. (1999) “The Unified Software Development Process”, Addison-Wesley.
21. March, S. and Smith, G, (1995) "Design and Natural Science Research on Information Technology." Decision Support Systems 15 (1995): 251 - 266.
22. Mumford, E. (1981) "Participative Systems Design: A Structure Method." Systems, Objectives, Solutions 1(1): 5-19.
23. Naumann, J. D., Jenkins, A.M. (1982) "Prototyping: The New Paradigm for Systems Development." MIS Quarterly 6(3): 29-44.
24. Rossi, M., Ramesh, B., Lyytinen, K., and Tolvanen, J. (2004) "Managing Evolutionary Method Engineering by Method Rationale," Journal of the AIS (5:9).
25. Simon, H. (1996) “The Sciences of the Artificial”, Third Edition. Cambridge, MA, MIT Press.
26. Truex, D., Baskerville, R., Travis. J. (2000) “Amethodical Systems Development: The Deferred Meaning of Systems Development Methods”, Journal of Accounting, Management, and Information Technologies, Tech 10. pp 53-79.
27. Truex, D., Avison, D. (2003) “Method Engineering: Reflections on the Past and Ways Forward”, Ninth Americas Conference on Information Systems.
28. von Bertalanffy, L. (1928) “Kritische theorie der Formbildung”, Borntraeger.
29. von Bertalanffy, L. (1969) General System Theory, Braziler, New York.
30. Whiting, R. (1998) “Development in Disarray”, Software Magazine, September, 20.
31. WWW, http://www.SOFTWAREmag.com/archive/2001feb/CollaborativeMgt.html, Accessed January 15, 2006, 3pm.
32. WWW, http://www.standishgroup.com/sample_research/chaos_1994_1.php, Accessed July 23, 2004, 11am.
33. Yourdon, E., Constantine, L.L (1979) "Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design, 1st edition.", Prentice-Hall.