The Successful Diffusion of Innovations:

Guidance for Software Development Organizations

Using the Personal Software Process as an example of an innovative information technology, the authors performed a field study of developers using the PSP approach on software development projects in industry. Their analysis of the results offers practical guidance on how software development organizations should support the diffusion of innovations into successful practice.

Gina C. Green, *Baylor University*

Alan R. Hevner, *University of South Florida*

How can we improve the probability of successfully introducing innovative software development techniques in software development organizations? This question has intrigued and confounded researchers in the software engineering field for many years. Several promising new tools and techniques have improved software development, but most are either not widely adopted or are often quickly abandoned. Researchers have noted this issue's importance in studies on Computer Aided Software Engineering tools[1–3] and on object-oriented,[4] user-centered,[5] and formal development techniques.[6] Here, we focus on why this happens—and how we can prevent it.

Our motivation is to better understand the phenomenon of innovative information technology diffusion so that we can better inform software development managers on how to ensure the successful use and diffusion of beneficial software development techniques in their organizations. Solutions to the problems of software development must include the human (peopleware and good management practices) as well as technical dimensions. Thus, a basic premise is that an effective strategy for IT diffusion must integrate technical research in software engineering with relevant behavioral research in the information systems field.

A large body of research on the diffusion and use of IT in organizations exists. Here, we expand the concept of IT to include software development techniques that can be innovations to the community of software developers. Thus, we can study software development practices under the guidance of well-established organizational theory. We hope to integrate behavioral IS research on IT diffusion with software engineering research on it by developing a comprehensive research model.

A Research Model of IT Diffusion

Our model (and the research study based on it) shows that two key factors influence software development IT diffusion success: developer involvement in the implementation process and characteristics of the environment into which the techniques are in-

0740-7459/00/$10.00 © 2000 IEEE

troduced (see Figure 1). Furthermore, the importance of these two factors is due in part to their influences on software developers' perceptions of control. (The Software Engineering Institute Special Report on this research includes a full discussion of the model and the background research literature supporting it.[7])

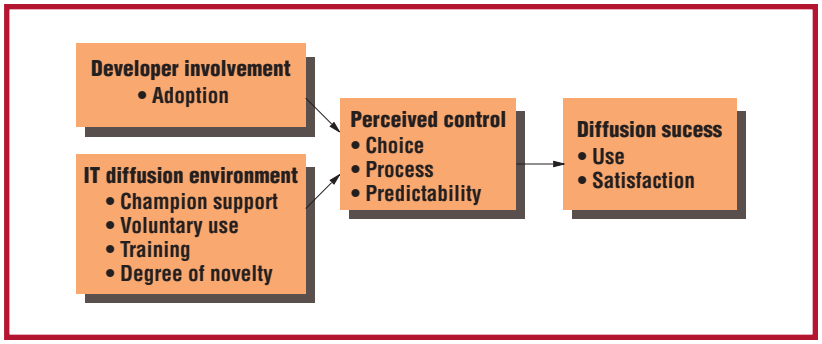

Figure 1. Our research model for IT diffusion.

Diffusion Success

A key measure of the successful diffusion of IT in an organization is its *use*.[8] However, when you mandate the use of IT, which often occurs when you introduce new software development technologies, *satisfaction* is sometimes a more appropriate measure of success.

Patrick Chau observed that in many software development innovation scenarios, the decision to adopt the innovation was made at an organizational level, and software developers were usually required to implement the decision.[2] However, he notes that the issue of whether or not the software developer enjoyed using the innovation—that is, the developer's satisfaction with the innovation—was of critical importance to productivity. Thus, satisfaction with the IT is a second diffusion success variable in the model.

Perceived Control

The construct of perceived control comes from social psychology research. Individuals are said to seek control because they need to know the causes and consequences of their and others' behaviors.[9] Perceived control is usually defined in terms of three dimensions: *choice*, *process*, and *predictability*.

The decisional or choice dimension refers to an individual who can choose among various possible actions. Early research on control equated more choice with more control. Subsequent research, however, found that too many choices could overwhelm the individual and hence reduce perceived control.[10] Still, increased control through choice can enhance performance on learning tasks and reduce stress. We examine perceived choice as the degree to which software developers feel they have control over deciding when it is beneficial or appropriate to use an IT innovation.

The process or behavioral view of control refers to an individual's ability to take direct action on the environment to influence an event. It represents the ability to change or escape an unpleasant situation or to determine the sequencing of events. However, external factors—such as availability of resources, managerial constraints, constraints imposed by the IT, and so forth—could inhibit this ability. Studies have found that greater managerial behavior control results in higher performance on IS development teams. In their survey of 79 software designers representing 41 development teams, John Henderson and Soonchul Lee found that teams with greater levels of managerial behavioral control had higher levels of performance.[11] We examine perceived process control as the degree to which software developers feel that their ability to perform software development tasks when using the IT is free from organizational or IT-related constraints.

The predictability dimension of control refers to knowing what events will occur and when—not necessarily to controlling the event itself. Enhancing an individual's level of predictability can decrease stress and increase satisfaction.[10] We examine predictability as the degree to which software developers are able to predict software development outcomes when using the IT innovation.

We thus suggest the following hypotheses:

H1: As software developers' perceptions of choice in when to use a new IT increase, their level of satisfaction with the new IT will increase.

H2: As software developers' perceptions of choice in when to use a new IT increase, their level of use of the new IT will increase.

H3: As software developers' perceptions of process control in how to use a new IT increase, their level of satisfaction with the new IT will increase.

H4: As software developers' perceptions of process control in how to use a new IT

Most research in user involvement focuses on IT end users as the population of interest. However, we focus on software developer involvement.

increase, their level of use of the new IT will increase.

H5: As software developers' perceptions of predictability when using a new IT increase, their level of satisfaction with the new IT will increase.

H6: As software developers' perceptions of predictability when using a new IT increase, their level of use of the new IT will increase.

Developer Involvement

Most research in user involvement focuses on IT end users as the population of interest. However, we focus on software developer involvement in the *adoption* of the software development technique in the developer's organization. The more involved a software developer is in this decision process, the more likely the developer is to perceive greater levels of choice in when to apply the IT. Thus, we propose the following hypothesis:

H7: As software developers are more involved in the new IT adoption process, their perceived choice in when to use the new IT will increase.

IT Diffusion Environment

The IT diffusion environment construct includes various individual, organizational, and environmental characteristics surrounding the use of the software development technique. Our research model identifies four characteristics of particular importance when diffusing innovations: the degree of *champion support*, the degree of *voluntary use*, the effectiveness of *training*, and the degree of *novelty*.

Champion support

An IT champion is a "manager who actively and vigorously promotes [his or her] personal vision for using IT, pushing [a] project over or around approval and implementation hurdles."[12] Champion support consists of two aspects: controlling resources needed for system development and use, and signaling clues to encourage certain behaviors. In his field study, Juhani Iivari found such support to be significantly and positively associated with CASE usage.[3] The degree to which a champion can influence decisions can influence the degree to which developers feel free to make their own decisions. Thus,

H8: As software developers' perceptions of degree of champion support for a new IT increase, their perceived choice in when to use the new IT will decrease.

Support for the relationship in H8 is not an argument against champion support. On the contrary, several studies have found champion support to be beneficial in increasing IT use.[2,3] However, developers might see it as limiting their choices.

Voluntary use

Voluntary use is "the degree to which use of the innovation is perceived as being voluntary, or of free will."[13] Gary Moore and Izak Benbasat emphasize the importance of this variable in their study of innovation diffusion, because organizational policy can mandate or discourage using innovations within organizations. We examine perceived voluntary use as the degree to which software developers feel that they have control over whether or not to adopt an IT innovation in their organization. Because mandated adoption or use of an IT takes away the freedom of choice from IT users, we propose the following hypothesis:

H9: As software developers' perceptions of voluntary use for a new IT increase, their perceived choice in when to use the new IT will increase.

Furthermore, we expect higher levels of perceptions of voluntariness will lead to higher perceptions of software developer process control over how the new IT is used in software development tasks. Thus

H10: As software developers' perceptions of voluntary use for a new IT increase, their perceived process control over how to use the new IT will increase.

Training

The availability and effectiveness of training represent crucial factors in the successful diffusion of software development innovations. Training on the IT is an essential prerequisite to putting the IT to use. However, a key consideration is the context

in which training is conducted. Connie Gersick and J. Richard Hackman in their research on habit-forming note that the most effective time to introduce an intervention (or a change in the way of doing things) is when the individual or group is at a "breakpoint" in work.[14] When groups must change their norms and routines while simultaneously performing the routine, the effect is, at best, distracting and unhelpful. Gersick and Hackman argue that developers must give their full concentration to acquiring the new habits and learning the new techniques. We expect more effective training will lead to greater familiarity with the IT, thus enhancing the software developers' ability to predict their performance on software development tasks when using the IT:

> ***H11:*** *As software developers' perceptions of training effectiveness on a new IT increase, their perceived predictability when using a new IT will increase.*

Novelty

Degree of novelty refers to the extent to which learning and using the IT innovation represent new experiences to the user. Chau refers to this as the implementation gap,[2] where the gap is the difference between current skills and knowledge and the skills and knowledge the new IT requires. In a field survey, he found a significant, negative relationship between the degree of novelty and IT acceptance. Put in the context of the adoption of software development techniques, a technique (or suite of techniques) that represents a radical change to software developers can lead to a work environment that is uncertain, unclear, and thus less predictable than an environment in which more familiar techniques are used. This reasoning leads to the following hypothesis:

> ***H12:*** *As software developers' perceptions of the degree of novelty of the new IT increases, their perceived predictability when using a new IT will decrease.*

Research Field Study

To evaluate our research model, we performed an extensive field study of industrial software developers. We used a field survey because using data from practicing software developers in multiple organizations increases the external validity of the study's results. Thus, we can more confidently apply the results across a wider population of software developers and software development organizations. (A copy of the full survey we used in this study is in the SEI Special Report.[7])

The innovative IT we studied is the Personal Software Process approach. The PSP approach is a "self-improvement process to control, manage, and improve the way [software engineers] work."[15] It is not a software tool in the traditional sense of an IT. Rather, it is a disciplined approach to support software developers in the measurement and analysis of their work to improve their effectiveness as software engineers. Software developers who use the PSP approach apply quality principles such as defect elimination and incremental development to improve the quality of the software products they deliver. Individual programmers apply these principles; thus, the degree to which organizations benefit from the PSP approach directly depends on the individual software developers' adoption and use of the PSP.

Through a cooperative research and development agreement between the SEI, the University of South Florida, and Baylor University, we were able to identify software development organizations that had individuals trained in the PSP by SEI trainers. We designed and validated a research survey questionnaire through pretesting and a thorough pilot study.

The population of interest was software developers who have adopted a new software development technique (the PSP) within the timeframe of three months to two years. We specified a three-month minimum timeframe to ensure that the subjects had adequate time to work with the PSP and form a stable judgment. The two-year maximum timeframe ensured that the software development technique qualified as an innovation. The sampling frame consisted of US and international software development organizations with employees trained in or using the PSP within the last two years.

We distributed a total of 174 surveys between November 1997 and January 1999. Of these, we received 81 completed surveys, resulting in a response rate of 47%. Of the 81 completed surveys, eight respondents indicated that their use of the PSP was less than

When groups must change their norms and routines while simultaneously performing the routine, the effect is, at best, distracting and unhelpful.

Table 1

Summary of Demographic Statistics

Variable	Value
Gender	
Male	79%
Female	21%
Age	Mean = 33.2
	Range = 21 to 60
Position	
Analyst	10%
Designer	20%
Manager	6%
Programmer	51%
Project manager	13%
Highest degree held	
High school	3%
Bachelors	51%
Masters	43%
PhD	3%
Years of IS experience	Mean = 8.1
	Range = 1 to 25
Months of PSP experience	Mean = 12.2
	Range = 3 to 24
Origin of respondent (organization)	
US	78%
International	22%

Table 2

Descriptive Data on PSP Use

Variable	Value
Number of PSP projects	Mean = 2.4
	Range = 0 to 18
How PSP is used	
Critical projects only	36%
Noncritical projects only	25%
Mixture of critical and noncritical projects	39%
Percentage of projects using the PSP	
None	10%
1 to 25%	26%
26 to 50%	13%
51 to 75%	14%
Less than 75%	37%
Types of projects using the PSP	
Proof of concept only	31%
Small projects only	10%
Mixture of large and small projects	28%
Large projects only	7%
All projects	24%
Project phases where PSP used	
Requirements specification	32%
Analysis	40%
Design	80%
Implementation	92%
Testing	65%
Maintenance	24%

three months; one indicated that his use of the PSP was longer than two years. The resulting usable sample size was therefore 72.

Results and Guidelines

Our research model is grounded in established theory in IS, social psychology, diffusion of innovations, and software engineering. The framework is a causal model, which shows relationships between theoretical constructs that affect the successful diffusion of software development techniques. Although data-analysis techniques cannot prove the theory depicted in the causal model, proof or support for the theory is argued on the basis of the theories upon which we constructed the model. However, the data analysis can shed light on whether the causal model is consistent with the data collected in the study.

By applying path analysis on the causal model, we can examine the direct effects of one variable on another, as well as indirect effects of one variable on another through one or more intervening (mediating) variables. Path analysis uses a series of multiple regressions, where path coefficients assess the strength of the relationships depicted in the causal model. Thus, we can use the magnitude, direction, and significance of the path coefficients to test the research hypotheses.

Table 1 summarizes the demographic information on survey respondents, and Table 2 presents descriptive data on PSP use. (The SEI Special Report contains a complete presentation of the data analyses.[7]) Table 3 summarizes the results of the hypotheses. One of the key findings is software developers' perceptions of control over their work when using an innovative software development technique significantly relates to that developer's satisfaction with the technique. Support of H1 and H5, as well as the significance of the relationship in H3, promote this finding. Thus, perceived control is established as an important factor in determining a software developer's satisfaction with a software development innovation.

Another significant finding is that there are manager-controlled factors that can enhance software developers' perceptions of control, thereby increasing the chances that the innovation will be diffused into software development practice. These factors include the software developers' involvement in the adoption process (H7 support), the degree of champion support for the IT (H8 support), and the degree to which use of the development tech-

nique is voluntary (H9 and H10 support).

Developer Perceived Control and IT Diffusion Success

One of the central constructs explored in this research is the software developers' perceptions of personal control over their work when using an IT innovation. Results of data analyses show that there is a direct relationship between software developers' perceptions of control and their satisfaction with using the software development technique. Specifically, this research has demonstrated that developers are more satisfied with using software development innovations if

- they have increased choice in when to use that innovation (H1),
- they have decreased process control in how to use that innovation (H3), and
- the innovation increases the predictability of their work (H5).

The second observation is quite interesting. The data suggest that the more personal control developers have over how they use PSP, the less satisfied they are using it. The more the organization emphasizes organizational standards and structure in its use of PSP, the more satisfied the developers are in using it. This result contradicts H3 and suggests there might be differences between IS personnel or tasks and the personnel or tasks studied in previous research that found a positive relationship between personal process control and satisfaction. IS tasks are complex and unstructured. By enforcing standards and structure, a software development technique can bring structure to a previously unstructured task environment, thereby reducing overall task complexity. Thus, contrary to current social psychology literature on perceived process and behavioral control, our research suggests that in a complex task environment such as software development, decreases in personal control as a result of the use of a disciplined software development technique can result in greater satisfaction.

This research also demonstrates that developers tend to subsequently use a software development technique when that technique provides more predictability in their software development work (H6). Thus,

Table 3

Summary of Hypothesis Support

Hypothesis	Dependent variable	Independent variable	Supported?
H1	Satisfaction	Choice	Yes
H3	Satisfaction	Process	No[1]
H5	Satisfaction	Predictability	Yes
H2	Use	Choice	No
H4	Use	Process	No
H6	Use	Predictability	Yes
H7	Choice	Involvement	Yes
H8	Choice	Champion	Yes
H9	Choice	Voluntariness	Yes
H10	Process	Voluntariness	Yes
H11	Predictability	Training	No
H12	Predictability	Novelty	No

1. The relationship is significant but in the opposite direction of the hypothesis.

H1 suggests that managers should provide as much freedom as possible—within organizational limits—to individual developers to decide when to apply innovative software development techniques.

H3, H5, and H6 provide support for using more disciplined engineering practices as a way to provide more organizational controls and predictability in the complex task of software development.

Developer Involvement and Perceived Control

Results show that greater levels of developer involvement in the process of adopting a software development innovation are associated with enhanced feelings of perceived control (H7). Thus,

H7 encourages software development managers to involve software developers in IT adoption tasks such as identifying areas for improvement in software development, identifying tools and techniques on the market that may improve these software development areas, determining objectives of using the new innovations and assessing costs and benefits.

Diffusion Environment Factors and Perceived Control

Data analysis of H8 reveals that greater degrees of champion support for IT are associated with lowered levels of perceived control. Although traditional IS research has strongly advocated champion support for IT in organizations, this finding suggests that too much

Using the PSP and other innovations implies increased personal responsibility for continuous quality and productivity improvements.

push from an IT champion might adversely impact software developers' perceived control, potentially decreasing the chance for successfully diffusing the IT in the organization. This statement is not meant to discourage champion support for IT innovations. Although we didn't find a significant, direct relationship between champion support and IT diffusion success, previous IS studies have found that increases in champion support are directly associated with increases in IT use:[3]

> *The significance of H8 is that while potentially effective in increasing innovation use, increases in champion support should be augmented with other control reinforcement mechanisms, such as adequate training and increased levels of developer involvement, so that developers maintain a sense of personal control in their work.*

The degree to which adopting the software development innovation is voluntary increases both software developers' control over choice in deciding when to apply the PSP and their control over the process of how to adapt the PSP to development tasks. Support for H9 reinforces Moore and Benbasat's assertion that mandating the use of an IT takes away freedom of choice from IT users.[13] Support for H10 shows that the freedom to decide whether to adopt or use an IT can positively influence developers' perception of the freedom that they have over how best to accomplish their work. Thus,

> *Support for H9 and H10 suggest that where possible, IS managers should encourage rather than mandate adoption and use of a software development innovation, thereby creating a situation of voluntary use.*

There was a lack of support for H11 and H12, which relate training and degree of novelty to perceived control through their effects on the ability to predict software development work when using the software development technique innovation. Neither variable had a significant effect on predictability. The lack of support for these hypotheses suggests that there could be other factors not included in this study that better explain the ability to predict software development outcomes when using innovative techniques.

Diffusion Success: Satisfaction and Use

We found strong support (H1, H3, and H5) that perceived control by developers greatly influences their satisfaction with the innovation. However, the lack of support for H2 and H4 relating choice and process control to innovation use might raise questions as to whether increased perceived control also leads to increased use of an innovation. Thus, we ran an additional analysis to determine if the data demonstrates that increased PSP satisfaction leads to increased levels of PSP use. The analysis clearly finds a significant relationship between innovation satisfaction and use—as intuition would suggest. This result provides our final guideline:

> *Managerial efforts to increase software developers' satisfaction with an innovative IT will result in increased use of the IT.*

The model of innovative IT diffusion developed in this research identifies behavioral indicators of IT diffusion success. Our field study clearly demonstrates significant relationships between developer involvement, diffusion environment, and the mediating variables of perceived developer control with IT diffusion success. While traditional software engineering literature focuses more on objective indicators of IT success, such as error reduction and productivity improvements, these improvements in software development will not be realized without the incorporation and sustained use of the software development technique in practice.

In the future, we hope to further explore perceived control constructs. Using the PSP and other innovations implies increased personal responsibility for continuous quality and productivity improvements. Understanding and influencing the degree of personal control perceived by software developers is important when asking developers to take on more personal responsibility. Although our research is a step in this direction, additional analyses reported in the SEI

Special Report[7] suggest that there might be more relationships between variables that we have not fully examined.[7] We are currently exploring relationships among perceived control variables using more powerful structural equation modeling techniques.

In addition, the proposed model is not limited to the PSP innovation. Because the research model is grounded in well-established theory, we expect the model to hold true for a wide range of software development innovations. Future research should validate this by studying other IT innovations. We also need further research to develop more reliable measures of the adequacy of training on IT and the degree of newness of IT to the developer. Once such measures are developed, we'll retest H11 and H12 to see if there is, in fact, support for the relationships between training, novelty and predictability.

In practice, software developers might also be motivated to use an IT innovation because the IT can measurably improve their effectiveness. You measure effectiveness in terms of software quality (for example, the number of system defects) or productivity (for example, lines of code delivered per person-month). We plan to develop a set of hypotheses on the relationships between the IT diffusion environment and the developers' perceived IT quality and productivity impacts and the relationships between the developers' perceived IT impacts and IT diffusion success.

Acknowledgments

We acknowledge the invaluable support of the Software Engineering Institute in this research. In particular, Priscilla Fowler worked closely with us to identify and contact study participants. She was our lifeline to SEI resources and staff. We also thank the members of Gina Green's doctoral committee at the University of South Florida, Rosann Collins, Paul Cheney, and Carnot Nelson. Their insightful comments added immensely to this research.

References

1. W. Orlikowski, "CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development," *MIS Quarterly*, Vol. 17, No. 3, Sept. 1993, pp. 309–340.
2. P. Chau, "An Empirical Investigation of Factors Affecting the Acceptance of CASE by Systems Developers," *Information and Management*, Vol. 30, 1996, pp. 269–280.
3. J. Iivari, "Why Are CASE Tools Not Used?" *Comm. ACM*, Vol. 39, No. 10, Oct. 1996, pp. 94–103.
4. M. Fayad, W. Tsai, and M. Fulghum. "Transition to Object-Oriented Software Development," *Comm. ACM*, Vol. 39, No. 2, Feb. 1996, pp. 109–121.
5. M. Mantei and T. Teorey, "Incorporating Behavioral Techniques Into the Systems Development Life Cycle," *MIS Quarterly*, Vol. 13, No. 3, Sept. 1989, pp. 257–273.
6. C. Holloway and R. Butler, "Impediments to Industrial Use of Formal Methods," *Computer*, Vol. 29, No. 4, Apr. 1996, pp. 25–26.
7. G. Green and A. Hevner, *Perceived Control of Software Developers and Its Impact on the Successful Diffusion of Information Technology*, Special Report CMU/SEI-98-SR-013, Apr. 1999; www.sei.cmu.edu/publications/documents/98.reports/98sr013/98sr013abstract.html (current Nov. 2000).
8. E. Rogers, *Diffusion of Innovations*, Fourth Ed., Free Press, New York, 1995.
9. A. Baronas and M. Louis, "Restoring a Sense of Control During Implementation: How User Involvement Leads to System Acceptance," *MIS Quarterly*, Vol. 12, No. 1, Mar. 1988, pp. 111–124.
10. E. Langer, *The Psychology of Control*, Sage Publications, Thousand Oaks, Calif., 1983.
11. J. Henderson and S. Lee, "Managing IS Design Teams: A Control Theories Perspective," *Management Science*, Vol. 38, No. 6, June 1992, pp. 757–776.
12. C. Beath, "Supporting the Information Technology Champion," *MIS Quarterly*, Vol. 15, No. 3, Sept. 1991, pp. 355–372.
13. G. Moore and I. Benbasat, "Development of an Instrument to Measure the Perceptions of Adopting an Information Technology Innovation," *Information Systems Research*, Vol. 2, No. 3, Sept., 1991, pp. 192–222.
14. C. Gersick and J. Hackman, "Habitual Routines in Task-Performing Groups," *Organizational Behavior and Human Decision Processes*, Vol. 47, No. 1, Oct. 1990, pp. 65–97.
15. W. Humphrey, *A Discipline for Software Eng.*, Addison Wesley, Reading, Mass., 1995.

About the Authors

Gina C. Green is an assistant professor of Information Systems at Baylor University. Her research interests include the diffusion of innovations, systems development methods, project management, and complexity in computer-supported work. She holds an MSE and BS in computer science and a PhD in management information systems from the University of South Florida. She is a member of the ACM, SIGMIS, and Decision Sciences Institute. Contact her at Information Systems Dept., Hankamer School of Business, Baylor Univ., Waco, TX 76798; Gina_Green@baylor.edu.

Alan R. Hevner is a professor in the Information Systems and Decision Sciences Department in the College of Business Administration at the University of South Florida. He holds the Salomon Brothers/HRCP Chair of Distributed Technology. His areas of research interest include software engineering, distributed database systems, database system performance evaluation, and telecommunications. He has a PhD in computer science from Purdue University and is a member of the ACM, IEEE, Association for Information Systems, and Institute for Operations Research and Management Sciences. Contact him at Information Systems and Decision Sciences Dept., College of Business Administration, Univ. of South Florida, Tampa, FL 33620; Ahevner@coba.usf.edu.