N92-19426

p- 25

SOFTWARE TECHNOLOGY INSERTION:
A STUDY OF SUCCESS FACTORS

Submitted to:
The Fifteenth Annual Software Engineering Workshop
November 28-29, 1990
NASA/Goddard Space Flight Center
Greenbelt, MD

by

Tom Lydon
Raytheon MSD
50 Apple Hill Drive
Tewksbury, MA 01876

Managing software development in large organizations has become increasingly difficult due to constantly increasing technical complexity, stricter government standards, a shortage of experienced software engineers, competitive pressure for improved productivity and quality, the need to co-develop hardware and software together, and the rapid changes in both hardware and software technology.

The "software factory" approach to software development minimizes risks while maximizing productivity and quality through standardization, automation, and training. However, in practice, this approach is relatively inflexible when adopting new software technologies. How can a large multi-project software engineering organization increase the likelihood of successful **software technology insertion (STI)**, especially in a standardized engineering environment?

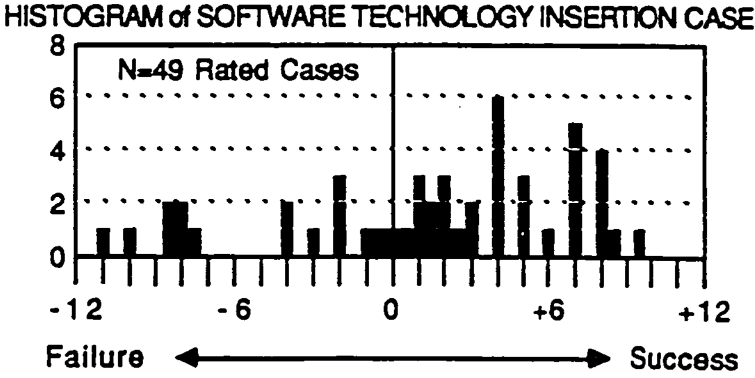


Figure 1 - Distribution of scores from 49 rated STI Cases

In an attempt to correlate various success factors with levels of success, 59 cases of "new software technology insertion" in thirteen recent projects at a large U.S. Defense electronics contractor were identified and categorized according to several criteria. The relative success or failure of 49 of these cases (see **Figure 1**) was determined by having key project personnel (Lead Engineer, Dept Manager, and tool supporters) rate 6 aspects (added together for total rating) of the software technology insertion results. Maximum success was scored as +12, and maximum failure as -12 on the rating scale. *The histogram in Figure 1 illustrates the distribution of scores from the 49 rated cases.*

There were 21 different **new software technologies** studied, most of them new **tools or methods**, including (in approximate lifecycle order):

- The use of DoD-STD-2167 or 2167A
- Structured analysis CASE tools
- Rapid-Prototyping in requirements or design
- In-House requirements traceability tool
- In-House program design language (PDL) tools
- Reusable Software in design or coding
- The use of Ada® as an implementation language
- The use of M68020 assembly language
- Microprocessor Development Stations (MDS) for integration testing
- In-House configuration management (CM), source code control tool
- Workstation-based engineering documentation tools
- The use of workstations as primary development platforms

Though meaningful statistical correlations were not possible due to the limited sample size, ratings were compiled and empirically compared with several technology factors measured for each STI case, including:

- Technology Type (Competence-Enhancing or -Destroying)
- Support Type (In-House or External)
- Maturity of the Technology (Young, Mature, and Old)
- Project Size (SLOC)
- Prior Expectations (for success or failure)
- Reasons (for using the new software technology)
- Methods (of inserting the new software technology)
- Perceived Time Savings
- Perceived Labor Savings
- Perceived Computer Cost Savings
- Perceived Quality Improvement
- Met Expectations? (for success or failure)

A closer look at the **"Top Eleven"** cases of successful STI (ratings ≥ +7), and the **"Bottom Seven"** cases of unsuccessful STI (ratings ≤ -7) shows that:

1. Perceived **Time Savings** and perceived **Labor Savings** are the most significant real indicators of successful or unsuccessful STI.

2. Though users often complain about increased computer costs, **saving computer cost** is not an indicator of STI success, because it is not usually a goal or a motivator for the use of new technology.

3. Perceived **Quality Improvement** is a strong indicator of STI success, but not an indicator of STI failure.

4. Even in successful STI cases, users' **Prior Expectations** about what a new technology can/cannot do are not managed effectively.

In addition to the success ratings, **on-site structured interviews** were used to profile each new technology, and collect other qualitative information that was used to clarify and complete the data.

Tushman[1] describes new technology types as: (1) **competence-enhancing** - incrementally different, building on existing know-how, and substituting for older technologies without rendering their skills obsolete, or (2) **competence-destroying** - fundamentally different, requiring new skills, abilities, and knowledge for use. The main types of technology support are: (1) **In-House**, where the supporters work in the same organization as the users, or (2) **Outside**, where the supporters work in a different organization than the users.

A sample of the distribution of successful STI cases over these two combined factors (technology type and support type) is show in **Figure 2:**

Ratings of New Technology Types Across Two Dimensions

	IN-HOUSE Support	OUTSIDE Support
Competence ENHANCING	#Total= 16 #Rated= 13 Tot Rating= 47.0 Median= +5.0 Ave Rating= +3.6 BEST	#Total= 9 #Rated= 8 Tot Rating= 11.5 Median= +4.0 Ave Rating= +1.4 OK
Competence DESTROYING	#Total= 11 #Rated= 8 Tot Rating= -2.0 Median= +1.5 Ave Rating= -0.2 Poor	#Total= 23 #Rated= 20 Tot Rating= 14.5 Median= +0.7 Ave Rating= +0.7 Marginal

RATING SCALE: +12 = Maximum Success, -12 = Maximum Failure

Figure 2 - Distribution of Success/Failure across two factors

The new software technologies that had the **most successful** STI experience (though across a very limited set of cases) are summarized below:

#Cases	Ave Rating	New Software Technology
1	+9.5	In-House Automated Build Tool
2	+7.5	Microprocessor Development Stations for Integration
6	+4.8	In-House Software Problem Reporting Tool
3	+3.3	In-House Configuration Management (CM) Tool
7	+2.1	In-House Program Design Language Tool

The new software technologies that had the **least successful** STI experience are:

#Cases	Ave Rating	New Software Technology
1	-11.0	In-House Automated Code Documentation Tool
2	-8.8	Workstation-based Engineering Documentation Tool
4	-4.8	Workstation-based CASE Tool for Req'ts and Design

Among the overall conclusions from the study are:

1. Saving **schedule time** and **labor costs** are necessary and sufficient conditions for successful STI

2. **Improving quality** is a necessary, but not sufficient condition for successful Software Technology Insertion (STI)

3. Success with new software technology insertion (STI) is much greater for **competence-enhancing** than for competence-destroying technologies

4. Success with STI is somewhat greater for **in-house** supported technologies than for outside supported technologies

5. Success with STI is greater for **mature** technologies than for either young or old technologies (mature is >1 year after release, <5 years after release)

6. Success with STI is greater when users' expectations about "new technology" are controlled to avoid expecting too much -- exceeding users' expectations is not necessary for successful STI, but <u>not</u> meeting expections (i.e., disappointing them) is a sufficient condition for failure

7. Success with STI can be increased when there is **synergy** between multiple new technologies, such as Ada and workstations

These and other results and conclusions, along with some recommendations for large software development organizations, will be covered at the workshop.

Reference [1] Tushman, M., and Anderson, P., "Technological Discontinuities and Organizational Environments", Administrative Science Quarterly, Sept 1986.

13 Software Projects

Project ID#	Language	SLOC	Current Phase
1	Assembly Fortran C	49200 6400 2900	Integration Test
2	C	9100	Design & Code
3	Assembly C	4000 4500	Maintenance
4	C	8500	Integration Test
5	Assembly	10850	Design & Code
6	Assembly	7100	System Test
7	Assembly	16000	Integration Test
8	Fortran	n/a	Cancelled
9	Ada	25000	Design
10	Ada	n/a	Integration Test
11	Assembly	4850	Integration Test
12	C	13700	Maintenance
13	Assembly	18000	Design & Code

21 New Software Technologies

(most of them new tools, methods, languages)

A - The use of Ada® as an implementation language
B - In-House automated build tool(s)
C - In-House automated code documentation tool
D - In-House program design language (PDL) tools
E - In-House metrics tools for automatic data collection
G - In-House standard test reporting tool based on RDBMS
I - Workstation-based engineering documentation tool
J - The use of M68020 assembly language
K - Microprocessor Development Stations (MDS) for integration testing
L - In-House project scheduling and reporting tool
M - In-House configuration management (CM), source code control tool
N - In-House Vax/Unix documentation package using troff
P - In-House Software Problem Reporting Tool based on RDBMS
R - Rapid-Prototyping in requirements or design
S - Structured analysis graphical CASE tool
T - Structured analysis graphical CASE tool
U - Reusable Software in design or coding
W - The use of workstations as primary development platforms
X - Workstation-based engineering documentation tool
Y - In-House requirements traceability tool
Z - The use of DoD-STD-2167 or 2167A

Project/Technology Matrix

New Technology ID

Project ID#	A	B	C	D	E	G	I	J	K	L	M	N	P	R	S	T	U	W	X	Y	Z
1	●			●							●		●			●			●		
2					○																
3		●		○							●		●	●			●				
4						●															
5				○			○						●							○	○
6				●						○		○	●						●		●
7			●	●			●						●		●						●
8															●						
9	●				●					●							●				●
10	●			●			●						●		●			●			●
11				●				●	●		●			●						●	●
12								●	●					●							
13	○						●							○	●						●

59 STI Cases: ● Rated ○ Not Rated

Measuring Perceived STI Success

- For each STI Case, 6 **Questions** were asked of:
  - (1) Lead Engineer (Project/Matrix)
  - (2) Dept Manager (Functional/Matrix)

For Each STI Case: Statement (Agree or Disagree?)	**Agree.....Disagree**				
	+2	**+1**	**0**	**-1**	**-2**
1. I would use the new method/tool again		√			
2. The new method/tool saved **schedule time**			√		
3. The new method/tool saved **labor cost**	√				
4. The new method/tool saved **computer cost**					√
5. The new method/tool **improved quality**		√			
6. The new method/tool met my expectations				√	

- **Total Rating** for each STI Case is **sum** (example =+1)
  i.e., maximum = +12, minimum = -12

(Note: Questions not weighted)

References

[1] Tushman, M., and Anderson, P., "Technological Discontinuities and Organizational Environments", Administrative Science Quarterly, Sept 1986.

[2] Scacchi, W., and Babcock, J., "Understanding Software Technology Transfer", MCC Technical Report STP-309-87, October 1987.

Acknowledgements

• To the Raytheon Company, for helping to sponsor this work at M.I.T. during 1989-1990.

• To Dr. Ralph Katz, for his technical (and non-technical) advice on this thesis work.

• To the M.I.T. Management of Technology Program, in which the work was performed.

VIEWGRAPH MATERIALS

FOR THE

R. LYDON PRESENTATION

The Fifteenth Annual Software Engineering Workshop
NASA/Goddard Space Flight Center, Greenbelt, MD
November 28-29, 1990

Software Technology Insertion: A Study of Success Factors

Tom Lydon
Raytheon MSD, Mailstop T3ML19
50 Apple Hill Drive, Tewksbury, MA 01876

Software Technology Insertion (STI)

Software Technology Insertion	"New" Software Technology + Opportunity to Insert
"New" Software Technology	Tool or Method that is unfamiliar to the majority of a Project Team, usually replacing a more familiar one
Opportunity to Insert	A software development activity on a new (most likely) or ongoing (less likely) software project

Successful STI

Perceived STI Success

User's sense of **Labor** Cost Savings +
User's sense of **Computer** Cost Savings +
User's sense of Elapsed **Time** Savings +
User's sense of **Quality** Improvement

Real STI Success

Measured **Labor** Cost Savings +
Measured **Computer** Cost Savings +
Measured Elapsed **Time** Savings +
Measured **Quality** Improvement

STI "Cases" Overview

STI Case	A single incident of STI on a single project, usually within a single development phase
59 STI Cases Identified	Across 13 different projects; from 1 to 7 STI Cases per project
49 STI Cases Rated for Perceived Success	Some of the 59 identified cases were not able to be rated
13 Different SW Projects	Some ongoing, some just completed; using Ada, C, Fortran, Assembly; ranging in size from 2900 to 49200 SLOC
21 Different SW Technologies	Most new tools, methods, languages (e.g., CASE, 2167A, Ada, Rapid-Proto, Reuse,...)

Software Technology Insertion: Success Factors

13 Software Projects

Project ID#	Language	SLOC	Current Phase
1	Assembly Fortran C	49200 6400 2900	Integration Test
2	C	9100	Design & Code
3	Assembly C	4000 4500	Maintenance
4	C	8500	Integration Test
5	Assembly	10850	Design & Code
6	Assembly	7100	System Test
7	Assembly	16000	Integration Test
8	Fortran	n/a	Cancelled
9	Ada	25000	Design
10	Ada	n/a	Integration Test
11	Assembly	4850	Integration Test
12	C	13700	Maintenance
13	Assembly	18000	Design & Code

21 New Software Technologies
(most of them new tools, methods, languages)

A - The use of Ada® as an implementation language
B - In-House automated build tool(s)
C - In-House automated code documentation tool
D - In-House program design language (PDL) tools
E - In-House metrics tools for automatic data collection
G - In-House standard test reporting tool based on RDBMS
I - Workstation-based engineering documentation tool
J - The use of M68020 assembly language
K - Microprocessor Development Stations (MDS) for integration testing
L - In-House project scheduling and reporting tool
M - In-House configuration management (CM), source code control tool
N - In-House Vax/Unix documentation package using troff
P - In-House Software Problem Reporting Tool based on RDBMS
R - Rapid-Prototyping in requirements or design
S - Structured analysis graphical CASE tool
T - Structured analysis graphical CASE tool
U - Reusable Software in design or coding
W - The use of workstations as primary development platforms
X - Workstation-based engineering documentation tool
Y - In-House requirements traceability tool
Z - The use of DoD-STD-2167 or 2167A

Project/Technology Matrix

New Technology ID

Project ID#	A	B	C	D	E	G	I	J	K	L	M	N	P	R	S	T	U	W	X	Y	Z
1	●			●							●		●			●			●		
2					○																
3		●		○							●		●	●			●				
4						●															
5				○			○						●							○	○
6				●						○		○	●						●		●
7			●	●			●						●		●						●
8															●						
9	●				●					●							●				●
10	●			●			●						●		●			●			●
11				●				●	●		●			●						●	●
12								●	●					●							
13	○						●							○	●						●

59 STI Cases: ● Rated ○ Not Rated

Other Measured Factors

- **Technology Type** (Competence-Enhancing or -Destroying)
- **Support Type** (In-House or External)
- **Maturity** of the Technology (Young, Mature, and Old)
- **Project Size** (SLOC)
- **Prior Expectations** (for success or failure)
- **Reasons** (for STI choice)
- **Methods** (of STI insertion)
- Perception of **Time** Savings
- Perception of **Labor** Savings
- Perception of **Computer** Cost Savings
- Perception of **Quality** Improvement
- **Result vs. Prior Expectations** (for success or failure)

Measuring Perceived STI Success

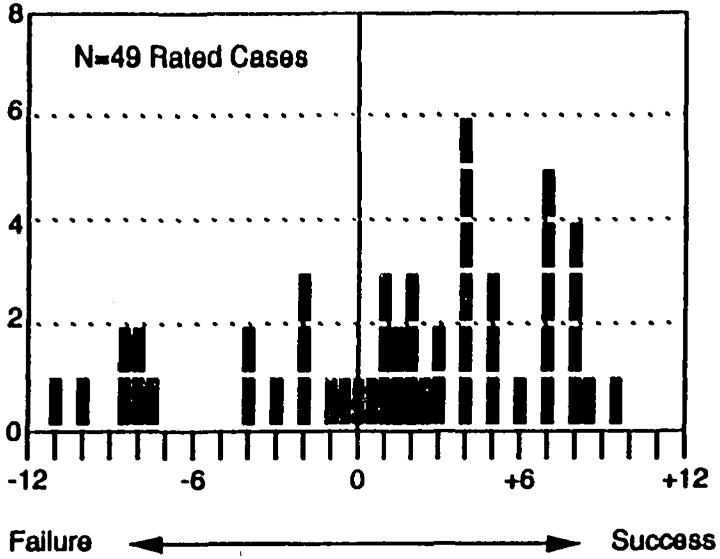
- For each STI Case, **6 Questions** were asked of:
  - (1) Lead Engineer (Project/Matrix)
  - (2) Dept Manager (Functional/Matrix)

For Each STI Case: **Statement** (Agree or Disagree?)	**Agree.....Disagree**				
	+2	**+1**	**0**	**-1**	**-2**
1. I would use the new method/tool again		✓			
2. The new method/tool saved **schedule time**			✓		
3. The new method/tool saved **labor cost**	✓				
4. The new method/tool saved **computer cost**					✓
5. The new method/tool **improved quality**		✓			
6. The new method/tool met my expectations				✓	

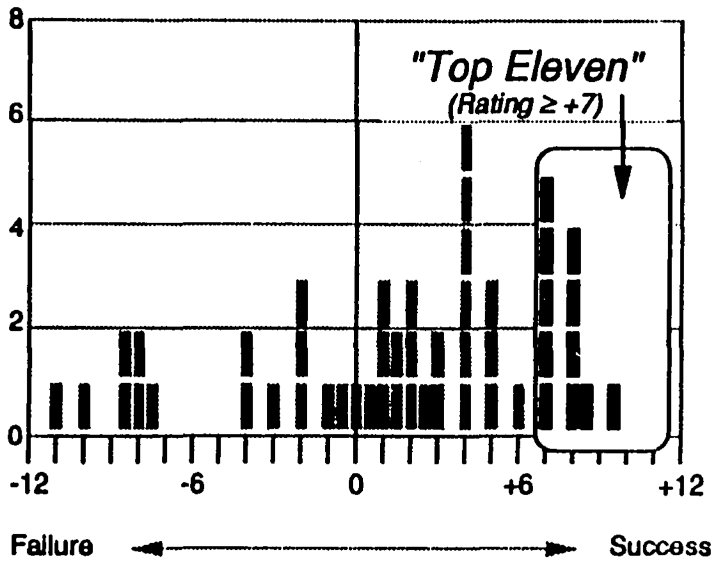
- **Total Rating** for each STI Case is **sum** (example =+1)
  i.e., maximum = +12, minimum = -12

(Note: Questions not weighted)

Histogram Of Software Technology Insertion Cases



Software Technology Insertion: Success Factors



Top Eleven STI Cases

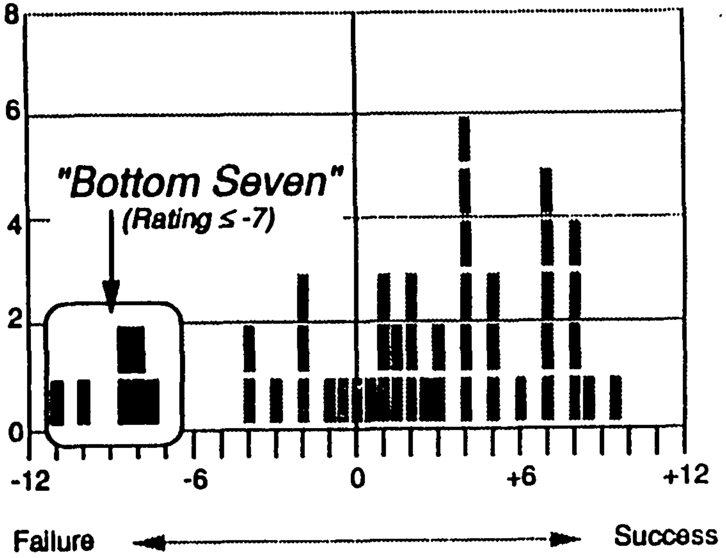
Main reasons for "success":

- "Synergy" within a project (4)
- Critical need for a capability (3)
- "Synergy" between two technologies (2)
- Mature and powerful tool (1)

May or may not **"Save Computer Costs"** (+0.2)

"Met Expectations" (+0.5) not as critical as:

- "Save Time" (+1.8)
- "Save Labor" (+1.7)
- "Improve Quality" (+1.6)



Bottom Seven STI Cases

Main reasons for "failure":

- Immature technology (3)
- Interface problems (3)
- Technology not "needed" by LE (2)
- Wrong technical solution (1)

May or may not **"Improve Quality"** (-0.4)

"Save Computer Costs" (-1.1) not as critical as:

- "Save Time" (-1.8)
- "Save Labor" (-1.9)
- "Met Expectations" (-1.9)

Competence-Enhancing vs **Competence-Destroying**

Competence-Enhancing technology - major improvement in price/performance that builds on existing know-how; a substitute for older technology, but does not render old skills obsolete; increase efficiency.

Competence-Destroying technology - new way of making a given product; requires new skills, abilities, and knowledge for use; may combine previously discrete steps into continuous flow, or be a completely different process

Maturity of a New Software Technology

Young technology - Released < 1 year, or prior to 2nd major release (V1.x)

Mature technology - Released > 1 year, and after 2nd major release (V2.x+)

Old technology - Released > 5 years, or after end of formal support

Ratings of New Technology Types Across Two Factors

	IN-HOUSE Support	**OUTSIDE Support**	
Competence ENHANCING ("incremental")	#Total= 16 #Rated= 13 Tot Rating= 47.0 Median= +5.0 Mean Rating= +3.6 BEST	#Total= 9 #Rated= 8 Tot Rating= 11.5 Median= +4.0 Mean Rating= +1.4 OK	**Mean = +2.8**
Competence DESTROYING ("radical")	#Total= 11 #Rated= 8 Tot Rating= -2.0 Median= +1.5 Mean Rating= -0.2 Poor	#Total= 23 #Rated= 20 Tot Rating= 14.5 Median= +0.7 Mean Rating= +0.7 Marginal	**Mean = +0.4**
	Mean = +2.1	**Mean = +0.9**	

RATING SCALE: +12 = Maximum Success, -12 = Maximum Failure

Software Technology Insertion: Success Factors

Summary of Results

*(Focus on success **factors** rather than successful **technologies**)*
*(Focus on **perceived** rather than **real** STI success)*

- Saving **schedule time** and **labor costs** drive successful STI (obvious?)
- **Improving quality** is necessary, but not sufficient for successful STI
- Exceeding users' expectations not necessary for successful STI, but **not meeting expectations** is sufficient for failure (i.e., must control)
- Much greater success for **competence-enhancing** vs competence-destroying technologies
- Greater success for **mature** vs young or old technologies
- Somewhat greater success for **in-house** vs outside supported

Next Step:
Linking **Perceived Success** with **Real Success** via **Software Metrics Collection**

- Corporate-wide effort to implement automatic collection of software metrics as a by-product of development - **MSD** is Lead Division
- **10** current software metrics defined (similar to Mitre Metrics)
- Based mainly on **DoD-STD-2167A**
- AutoCollection in development for both **project-specific** and **process-level** (across multiple projects) software metrics

Overview of Raytheon MSD's Software Metrics Collection