DESIGN OF INFORMATION SYSTEMS: SIMULATING THE EFFECTIVENESS OF KNOWLEDGE TRANSFER THROUGHOUT THE SYSTEM ANALYSIS PHASE.

Peter Otto
potto@csc.albany.edu

Salvatore Belardo
s.belardo@albany.edu

University at Albany, State University of New York
School of Business
1400 Washington Avenue,
Albany, New York 12222

Abstract
One of the critical issues determining the successful development of information systems is what might be described as the communication gap between the user group and the IS development group that occurs during the system analysis phase. This gap, we contend, stems from the lack of understanding of one another's domain knowledge without for example, a common vocabulary it is difficult to exchange knowledge. This article presents a system dynamics model that can simulate various knowledge exchange scenarios, to help develop strategies for an optimal knowledge transfer between these two groups. Knowledge exchange scenarios are a function of both the quantity (e.g. vocabulary) and quality (e.g. timeliness) of the knowledge possessed by each group. The paper concludes with a demonstration of different policy implementations and their impact on the effectiveness of the knowledge transfer during the systems analysis phase.

Keywords: Knowledge transfer, Knowledge quality, Communication in system design

Introduction

From the beginning of the computer era systems analysts have been confronted with the challenge of developing systems that satisfactorily meet user needs. All too often systems developed at considerable expense are not used, because the resulting system does not meet the user's expectations and requirements. It is generally agreed that one of the most significant factors in this failure is ineffective communication between the user group and the development group. Traditionally the development team has considerable knowledge of IS but is not necessarily familiar with the knowledge domain of the client. Conversely, the user group may have superficial knowledge of IS but not sufficient to judge, which systems, technologies, or development options would meet their needs. That the development and user groups typically have different mental models, and so a statement that is perfectly understandable and meaningful to one party is either not absorbed by the other or worse, is misinterpreted. Thus, the quality of the knowledge transfer is deficient. Senge (1990) cogently noted that surfacing and testing mental models is essential to learning. As a result of a deficient knowledge transfer, the information system typically does not perform as expected, is not operational at a specified time or cannot be used as intended.

In order to understand the knowledge transfer problem, we first examine the nature of knowledge in the developmental context and then introduce a set of ideas and concepts, which provide guidance on how best to structure teams to overcome these problems. Ideal knowledge transfer strategies can vary substantially depending upon the characteristics of both the development team and the user group. While such knowledge transfer strategies are difficult to frame, a system dynamics model could provide guidance as to what knowledge transfer strategy is most appropriate given the knowledge possessed by the development and user teams as well as the nature of the project environment.

One of the underlying premises of our work is that the degree of conformity between the developer's and users tacit knowledge for the other's domain has a substantial and critical impact on the nature and effectiveness of overall knowledge transfer. We assume that individuals possess both tacit knowledge and explicit knowledge. Explicit knowledge is what a person can articulate such as the personnel policy of a firm. Tacit knowledge entails information that is difficult to express, formalize, or share. Tacit knowledge in contrast to explicit knowledge, which is conscious and can be put into words. An individual experiences tacit knowledge as intuition, rather than as a body of facts or instruction sets he or she is conscious of having and can explain to others. Tacit knowledge is "knowing how" while explicit knowledge is "knowing that" (Lubit 2001). Tacit knowledge therefore is a mental model used by an individual to process information produced by others or to absorb information from observations.

Depending upon one's mental model(s), the explicit knowledge provided by one party could be interpreted by another as intended or could be badly misinterpreted determined by the receiving party's tacit knowledge. Therefore, the levels of tacit knowledge each side has of the other's domain affects the correctness and rapidity of the explicit knowledge transfer. Belardo et al. (2002) discuss the nature of the knowledge transfer problem and provides the excellent example that amplifies the importance of both explicit and tacit knowledge.

Our use of knowledge is in the context of Bloom's taxonomy (1956). Bloom's taxonomy consists of three domains – cognitive, affective and motor skills. Since we are concerned here with what a person knows and how efficiently and effectively a person can learn from others, we will focus on the cognitive domain. The cognitive domain defines in ascending order six levels of knowledge, beginning at the very lowest level vocabulary and progressing to the highest level, evaluation. The cognitive domain then is essentially a scale that can be used to evaluate the level of explicit knowledge possessed by an individual.

Dynamics of Knowledge Transfer

In situations where knowledge transfer is applied to a relatively stable technology with a slowly changing environment, the problems are more easily managed. This is true both, because knowledge accumulated during past interactions is more likely to be still relevant as well as, because the time pressure to complete the current system design before it is outdated will be minimized. An example of this situation would be developing a database to monitor the production of furniture, a relatively stable market with established supplier relationships and a more or less predictable product life cycle (Belardo et al. 2002).

An example where the knowledge transfer is more dynamic would be the design of a system to monitor the marketing of personal computers. The fact that the analyst had experience a number of years ago in developing an information system for mainframe computers, may provide little knowledge of current relevance. Likewise, from the users perspective, the marketing manager's pervious experience with an early electronic marketing application may be largely outdated by rapid evolution of the Internet (Belardo et al. 2002). Thus, the pressure on the knowledge transfer process are increased by the highly time dependent nature of the market as well as the short half-life of Internet innovations.

It is generally agreed that in more dynamic environments, the rates of knowledge acquisition as described by the learning curves of the design team and the user group become increasingly relevant. Among the factors, which influence the shape of these learning curves, are group size, tacit knowledge, and relevant explicit knowledge from previous experience.

Objective of the Study

The main objective of this paper is to build a system dynamics model that represents the dynamics of the knowledge transfer rate during the analysis stage of information systems development. The model is aimed at testing different policies in order to establish a target profile for tacit and explicit knowledge for the user and developer teams. While the composition of teams is understood in a general way by successful practitioners, a major contribution of this paper is to provide a framework for a more micro view of current and targeted knowledge levels for a user and developer team.

The system dynamics model focuses on the impact of tacit knowledge on the rate of acquisition of explicit knowledge by the developer and the user team. Using a model to determine a desired knowledge level is especially important when a dynamic design environment requires that the design process be compressed. Belardo et al. (2002) concludes that such a model can be used to make mid-course corrections in team composition. For example, by adding a senior member to the team, when there is a threat of not completing the project in time, we can evaluate the impact of knowledge acquisition from the team, which then determines the effectiveness. Using a system dynamics tool could help to optimize strategies about how to compose a team for the information system analysis project.

Problem statement

To create effective information systems, the design team must acquire a firm understanding of the user's expectations and requirements. On the other hand, the user should also be able to understand what the technical considerations and constraints for the information systems are. In order to do this there must be an efficient and effective transfer of knowledge between the two groups. The more efficient and effective the knowledge transfer is the more successful will the information system be.

Without a good understanding of the required knowledge levels between the teams during the information systems development, there is a likelihood that the team is not able to finish the project in time or costs are not kept in budget. One of the critical issues determining the rate of knowledge transfer between the teams hence, the success rate of an information system, is the composition of tacit and explicit knowledge within the user and developer team.

Audience

The model is aimed at managers who have responsibility for information system development and implementation. These managers work in industry or consulting companies and have strategic and tactical responsibilities for information systems, which include securing needed investments and resources, and making sure that all IT efforts are consistent with the business objectives of the firm. A broader target audience would include information systems developers and users. Here, the model could be used to support managers who must articulate information system development task specific strategic recommendations.

Model Purpose

The purpose of the model is to simulate the outcome of different policies to evaluate the quantity and quality of explicit and tacit knowledge in user and developer teams. The model could also be used to monitor the progress for improving both the quantity and quality of explicit knowledge over time. Simulating the dynamics of the team composition will give managers a tool for assessing initially the potential for effective knowledge transfer for a particular choice of individuals in the user and developer team.

Modelling Structure

Key questions addressed in this paper are:

I. How do individual levels of tacit and explicit knowledge influence the efficacy and effectiveness of knowledge transfer and consequently the quality of the information system?

II. What insights concerning team composition strategies can the model provide?

With regard to the first question, the proposed model should be able to predict the effect of changes from initial values of tacit, and explicit knowledge with regard to the quality of the information system.

With regard to the second question, the model should provide the manager with insights about the effect of different strategies concerning the best way to compose an information development team.

Theoretical Framework

Figure 1 shows the causal loop diagram for the network of variables that affect knowledge acquisition, the effectiveness of the teams, and as a result the quality of information systems development.

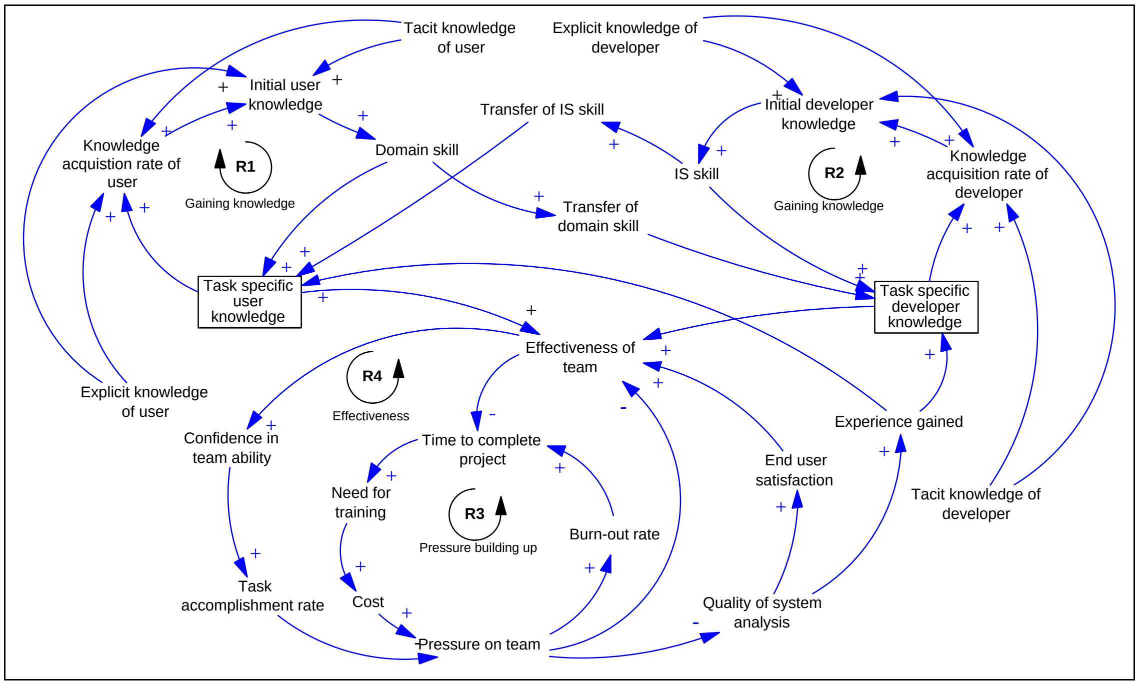


Figure I – Causal feedback structure

Description of the Causal loop diagram and Feedback Loops

There are many reinforcing loops in the causal loop diagram shown in figure 1. The two reinforcing knowledge loops (R1 and R2), determine the effectiveness of the team, which could have opposite effects depending on the initial degree of tacit and explicit knowledge of either group. This means that insufficient tacit and explicit knowledge can inhibit the transfer of knowledge from one team to the other, which will gradually reduce the effectiveness of the system analysis team. Another strong reinforcing loop, which influences the quality of the information system, and user satisfaction is R3 “Pressure building up”. If time to complete the project goes up, the pressure on the team builds up, thus causes a burn-out effect that can dramatically reduce the effectiveness of the team.

Dynamic Hypothesis

The effectiveness of a system design team is related not only to the initial explicit knowledge that the team will bring to the project but also to the tacit knowledge possessed by the members of the team. Narrowly conceived decisions about the team composition could lead to unexpected behavior and greatly influence the effectiveness of the team. The following figure visualizes the dynamic hypothesis.

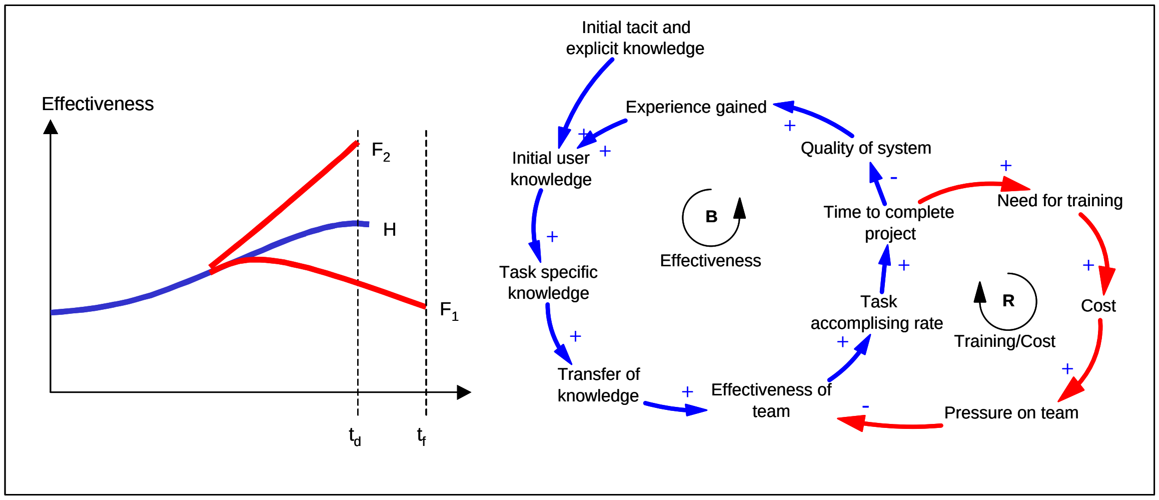


Figure 2 – Diagram for Dynamic Hypothesis

The diagram on the left side of figure 2 shows the reference mode and the expected best and worst-case behavior in the information systems development project. The diagram on the right side represents the causal loops, which causes the behavior of the system. A fear, or worst-case scenario, as represented in the diagram with the sloop F_1, is when the initial level of tacit and explicit knowledge is low and, as a consequence, causes low effectiveness. As a result of low tacit and explicit knowledge, the team will not meet the anticipated time (t_d) to finish the project but end the project later (t_t). Another fear, represented with the sloop F_2, is when the team realizes that the project needs mid-term correction, which could be accomplished either by adding more people to the team or by going through some training.

After training, the effectiveness of the team would increase, however, costs would exceed initial estimates. Higher costs could also lead to higher pressure on the team, which would result in a "burn-out effect", where the effectiveness of the team will decrease, causing a similar effect as in the F_1 scenario. The "best case or hope scenario" (H) represents the dynamic where a system analysis team has the necessary level of tacit and explicit knowledge to finish the project in the desired time (t_d) without pressure building up, and without cost overrun.

Stock and Flow Diagram

The stock and flow diagram is separated into different sectors as shown in figure 3 and 4. Each of the stock and flow is part of the conceptual system dynamics model, which is represented in figure 5.

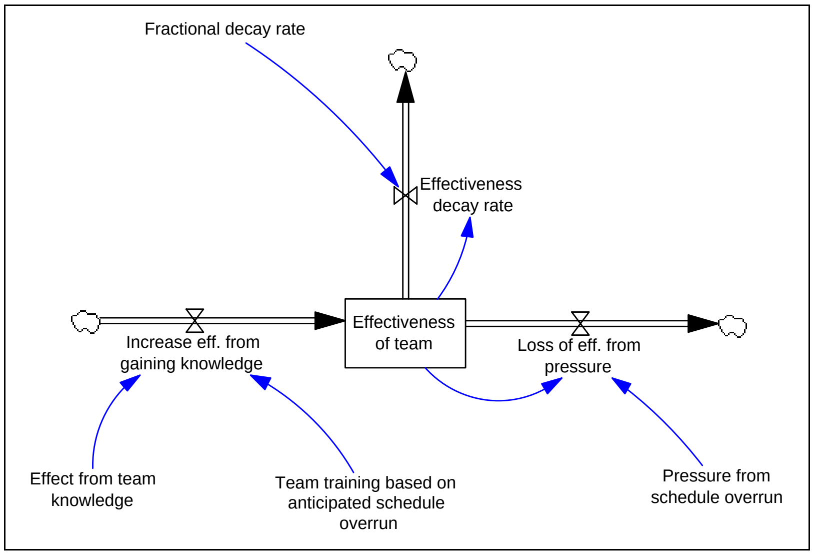

Fractional decay rate
Effectiveness decay rate
Effectiveness of team
Increase eff. from gaining knowledge
Loss of eff. from pressure
Effect from team knowledge
Team training based on anticipated schedule overrun
Pressure from schedule overrun

Figure 3 – Stock and Flow Determining the effectiveness of the team.

The effectiveness of the team is determined by the task specific knowledge of the user and developer teams, aggregated in the variable “Effect of team knowledge”. Increasing the effectiveness of the team could either be a result of higher levels of knowledge, based upon higher initial values of tacit or explicit knowledge, or could be due to training. Losing effectiveness is a result of pressure, which builds up when the expected date to complete the project is not met. The fractional decay rate is a constant representing a normal loss of effectiveness due to exhaustion in team-based projects.

The next stock and flow diagram (figure 4) represents the accumulation of task specific user and developer knowledge, which is based on initial values of tacit and explicit knowledge.

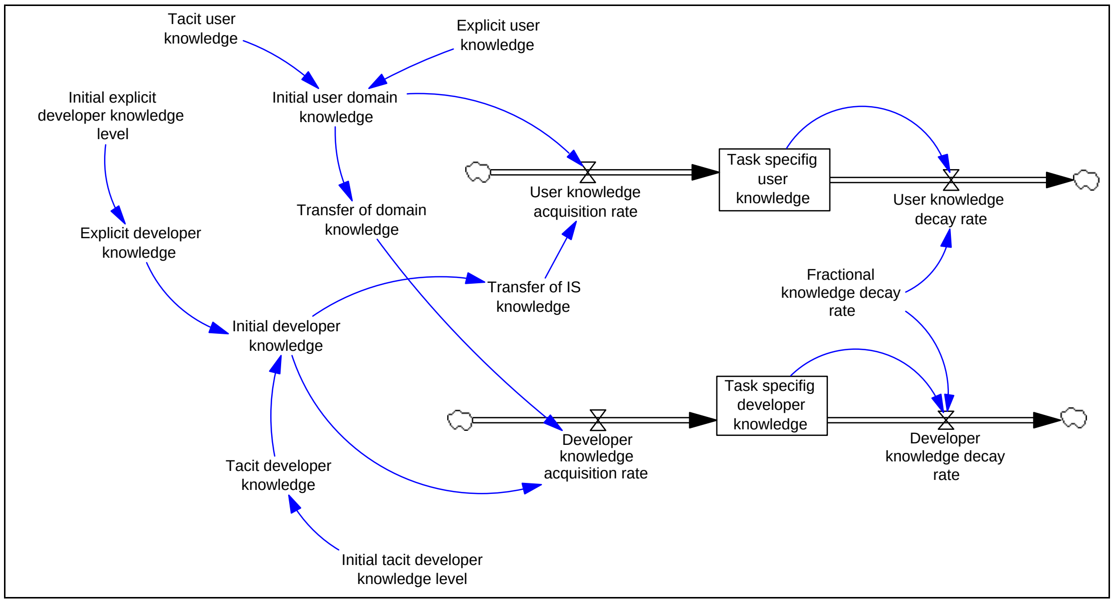

Tacit user knowledge
Explicit user knowledge
Initial explicit developer knowledge level
Initial user domain knowledge
Task specifig user knowledge
User knowledge acquisition rate
User knowledge decay rate
Transfer of domain knowledge
Explicit developer knowledge
Transfer of IS knowledge
Fractional knowledge decay rate
Initial developer knowledge
Task specifig developer knowledge
Developer knowledge acquisition rate
Developer knowledge decay rate
Tacit developer knowledge
Initial tacit developer knowledge level

Figure 4 – Stock and flow for task specific knowledge

The initial values of tacit and explicit knowledge determine the domain and IS knowledge of the two teams. The acquisition rate of knowledge depends on the degree of initial knowledge and the transfer rate between the developer and the user team. Outflow is determined by a fractional knowledge decay rate over time, which in essence represents the rate of losing knowledge. The remaining stock and flow structure of the system dynamics model is represented in figure 5.

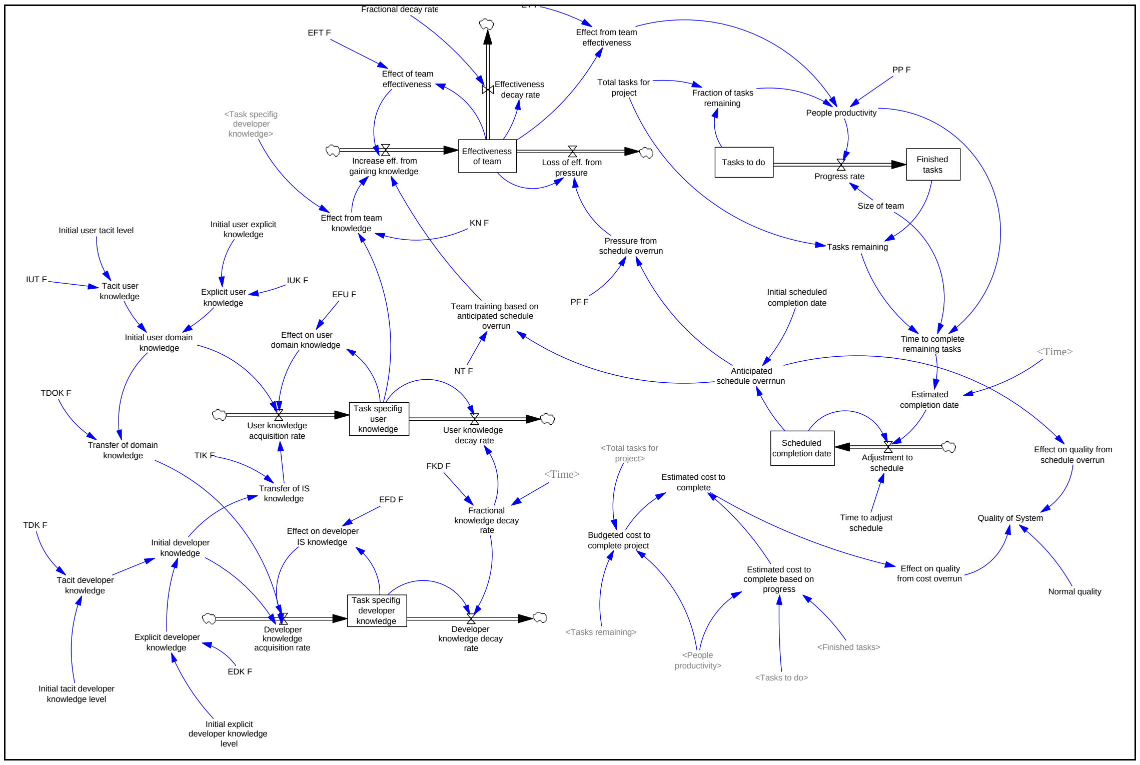

Fractional decay rate
EFT F
Effect of team effectiveness
Effectiveness decay rate
Effect from team effectiveness
Total tasks for project
Fraction of tasks remaining
PP F
People productivity
<Task specifig developer knowledge>
Increase eff. from gaining knowledge
Effectiveness of team
Loss of eff. from pressure
Tasks to do
Progress rate
Finished tasks
Size of team
Effect from team knowledge
KN F
Pressure from schedule overrun
Tasks remaining
Initial user tacit level
Initial user explicit knowledge
IUT F
Tacit user knowledge
Explicit user knowledge
IUK F
EFU F
Initial user domain knowledge
Effect on user domain knowledge
Team training based on anticipated schedule overrun
PF F
Initial scheduled completion date
Time to complete remaining tasks
<Time>
NT F
Anticipated schedule overrnun
Estimated completion date
TDOK F
Task specifig user knowledge
User knowledge acquisition rate
User knowledge decay rate
Scheduled completion date
Adjustment to schedule
Effect on quality from schedule overrun
Transfer of domain knowledge
TIK F
FKD F
<Total tasks for project>
<Time>
Estimated cost to complete
Transfer of IS knowledge
EFD F
Fractional knowledge decay rate
Time to adjust schedule
Quality of System
TDK F
Effect on developer IS knowledge
Budgeted cost to complete project
Initial developer knowledge
Estimated cost to complete based on progress
Effect on quality from cost overrun
Tacit developer knowledge
Normal quality
Task specifig developer knowledge
Explicit developer knowledge
Developer knowledge acquisition rate
Developer knowledge decay rate
<Tasks remaining>
<Finished tasks>
<People productivity>
EDK F
<Tasks to do>
Initial tacit developer knowledge level
Initial explicit developer knowledge level

Figure 5 – Overview of the model

The stock and flow structure in figures 3 and 4 represent the build block of the model, the remaining stock and flow representation is based upon generic type structures, which are often used in system dynamics models for project management. For example, the stock and flow structure of “task to do” and “unfinished tasks”, as well as “scheduled completion rate” are generic model formulations.

Input Variable Description

The model has two input variables, which determine the level of tacit and explicit knowledge of the user team and the developer team.

Explicit knowledge; is operationalized along four quality descriptors; accuracy, completeness, timeliness, and consistency. These quality descriptors have been discussed extensively in the information quality literature (e.g. Ballou and Pazer 1985, Wang and Strong 1996) as useful measures for the quality of explicit knowledge (Huang et al., 1999).

For the purpose of this study and for the sake of simplicity, the variable “explicit knowledge” is an aggregated value to represent the quality of knowledge with a three-point ordinal scale of high, medium, and low. The input value in the model for a high explicit knowledge level is between 0.7 and 0.9. Input value for medium explicit knowledge level is from 0.55 – 0.69, and low from 0.3 – 0.54.

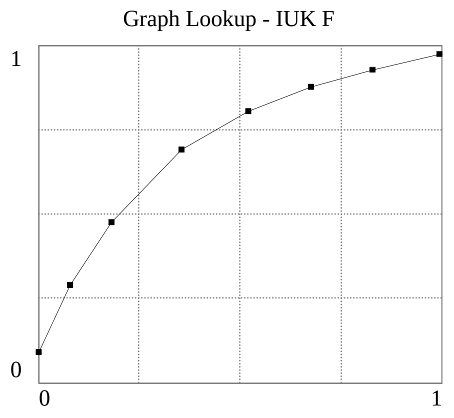


Figure 6 – Lookup table for explicit knowledge

Figure 6 shows the table function for the input variable “explicit knowledge”. It is suggested that explicit knowledge, e.g. accuracy, completeness, timeliness, and consistency, are easier to gain and learn. Therefore, the effect from explicit knowledge on the variable “initial knowledge” is different from the table function of tacit knowledge.

The second input variable is tacit knowledge. ***Tacit knowledge***; as stated before, is operationalized as the mental model used by an individual to process explicit knowledge, information or data provided by others as well as the ability to effectively assimilate observations of the relevant environment (Belardo et al. 2002). The lookup table for the variable "tacit knowledge" - as represented in figure 7 - is based on the assumption that tacit knowledge is much harder to gain, thus the shape of the curve in the table function is not so steep as for the explicit knowledge table function.

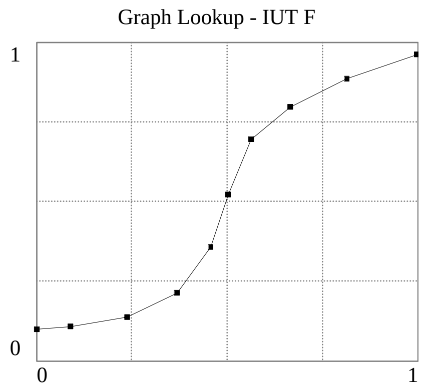
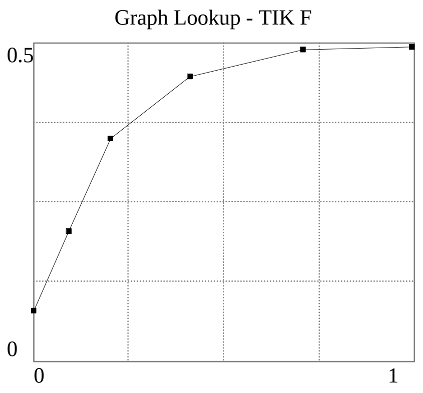


Figure 7 – Lookup table for tacit knowledge

The lookup tables for the developer team and the user team are identical for both the initial level of explicit as well as tacit knowledge.

Knowledge Transfer Variables
An initial level of user or developer knowledge determines the rate, with which each team can transfer their knowledge to the other team. For example, a developer team with high tacit and explicit knowledge is able to better communicate and understand the needs of the user, thus the user team will easier learn the content of the developer's domain.



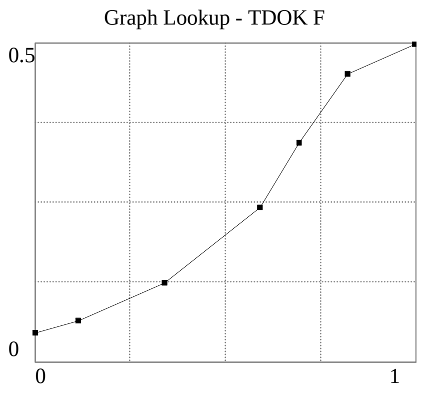


Figure 8 – Lookup table for the variables knowledge transfer

The graph on the left side of figure 8 represents the table function for the variable "transfer of IS knowledge". The curvature suggests that the developer team is better able to facilitate understanding and to optimize knowledge transfer for all situations. On the other hand, the graph on the right side, which is the table function for the variable "transfer of domain knowledge", suggests that the transfer of knowledge from the domain of the user team needs a higher degree of comprehension on the part of the system developer.

Model Behavior

In Time and Budget Policy

The first policy simulation is an example of a situation where at the project start each member of the user and developer team has a rather high level of tacit and explicit knowledge, and experience from previous information systems development (input value for the model is: tacit = 0.7, explicit = 0.8). The model then simulates the effectiveness of the team to fulfill 42 tasks in a given time frame of 50 months. The number of tasks, the time to complete the project, and the cost for developing an information system are arbitrarily and do not represent real conditions. However, it is not our objective to apply real data to represent cost and timing for the development of an information system but to simulate the effect of knowledge transfer based upon initial levels of tacit and explicit knowledge.

Results from the first policy simulation

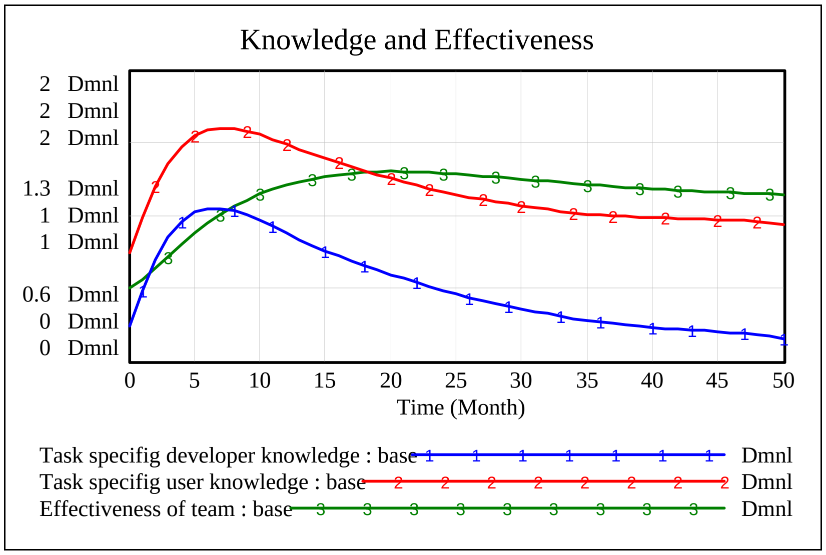


Figure 8 – Graph for effectiveness and task specific knowledge

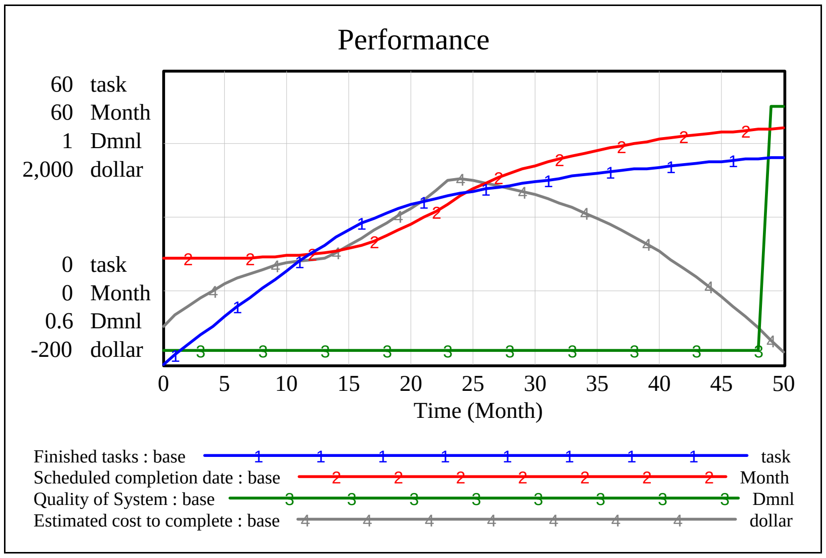


Figure 9 - Graph for team performance

Observation

Task specific developer knowledge is lower compared to the user knowledge. In addition the level of developer knowledge declines faster than that of the user. There could be two reasons. First, the rate of knowledge transfer from the user team is not high enough to bring the level of task specific developer to a sufficient level. Second, since the transfer of user knowledge, which is determined by the function table, is not the same as for the user team, the decay of task specific developer knowledge is steeper.

The performance of the team, based on the initial values, is sufficient to finish the 42 tasks in time and budget, and achieve the desired level of quality.

Medium knowledge level policy

The second policy tested is based on medium levels of tacit and explicit knowledge on both sides of the teams. Input values are 0.5 for tacit and 0.6 for explicit knowledge.

Results from the second policy simulation

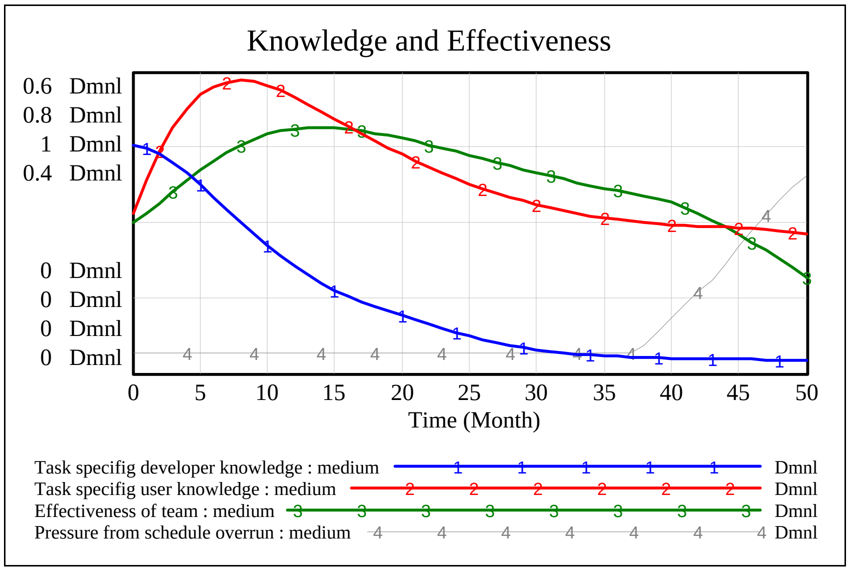


Figure 10 - Graph for effectiveness and task specific knowledge

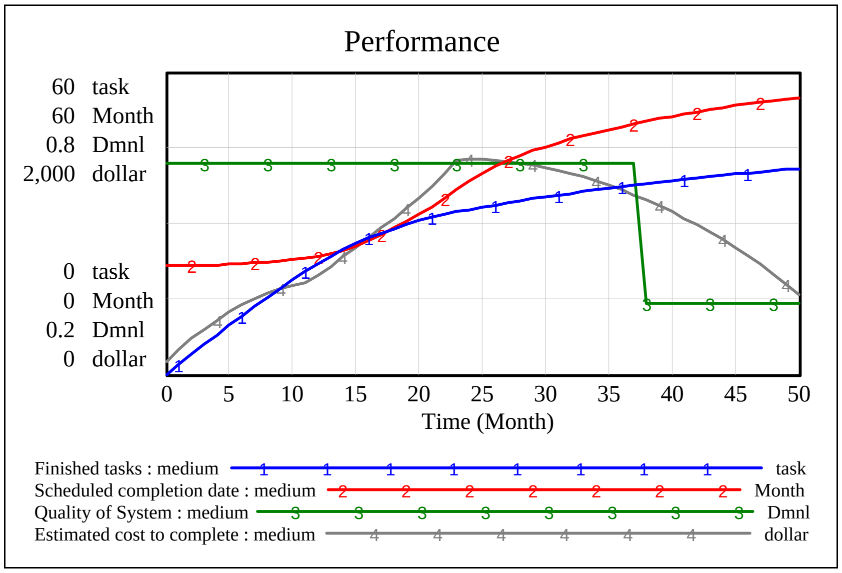


Figure 11 - Graph for team performance

Observation

Because of the medium levels of tacit and explicit knowledge as well as the decay rate (simulated as a value which decreases over time) the task specific knowledge of both teams is too low to complete the system in the scheduled time. Pressure is building up (graph 4, in figure 10) because of the number of unfinished tasks, and as a result, reducing the effectiveness of the team.

As a result of a lower level of effectiveness the system will not be completed in time and as we can see, the costs are higher than anticipated. Finally, the quality of the system is below expectations (as represented with the graph "Quality of system", shown in figure 11).

Policy to compensate lack of user knowledge

The third policy tries to compensate for low levels of tacit and explicit knowledge on the user side (initial values: tacit = 0.5, explicit = 0.5), with high levels of tacit and explicit knowledge on the developer side.

Results from third policy simulation

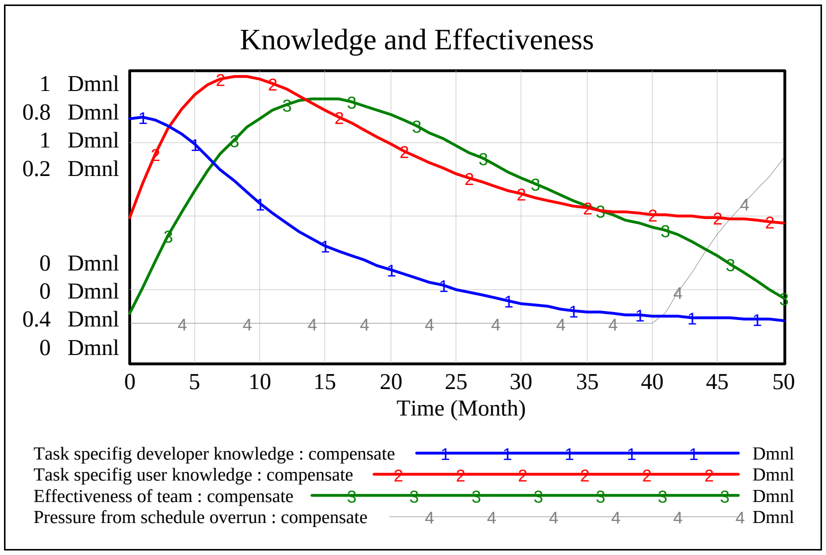


Figure 12 - Graph for effectiveness and task specific knowledge

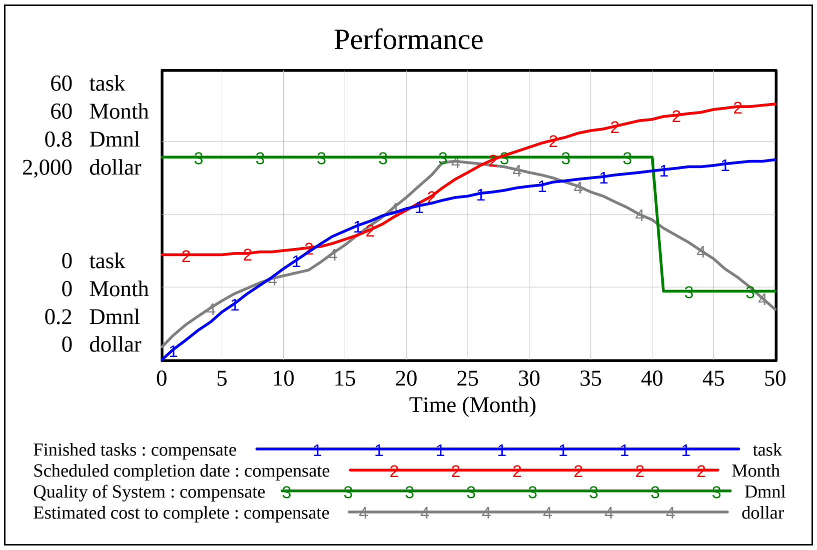


Figure 13 - Graph for team performance

Observation

Without an appropriate level of tacit and explicit knowledge on the user side, the quality of knowledge is too low to transfer the desired domain knowledge to the developer. Thus, the developer team is not able to comprehend the vocabulary and requirements from the user, which would be required to achieve a desired level of effectiveness.

This policy implies that we cannot compensate for a lack of user knowledge by having more senior people in the developer team with higher tacit and explicit knowledge. The user team must be able to transfer their domain knowledge to the developer team in order to achieve a level of effectiveness.

Insights from the Model

One insight from the system dynamics model is described in the previous policy, where low levels of user knowledge cannot be compensated with high levels of developer knowledge. Another insight from the model is based on testing the effect from a low tacit knowledge and high explicit knowledge on the user side. The results suggest that the level of tacit knowledge has a higher effect on the quality of knowledge as opposed to altering the scale for explicit knowledge.

Conclusion

The model described in this paper is a first attempt to capture the dynamics of knowledge transfer during the system analysis phase. The model can be helpful to determine a desired level of explicit and tacit knowledge for the user and developer group, necessary to form an ideal team capable of delivering a system analysis project in time.

A further expansion of the model is necessary to gain insights in the various knowledge exchange scenarios when the size and the composition of the team is changing during the system analysis phase. Because knowledge is an abstract concept, further research should also investigate the parameter values that we have used in the model to better quantify the ideas and concepts in the proposed system dynamics model.

References

Bloom, B. S. (1956). *Taxonomy of Educational Objectives*, David McKay Company, New York.

Huang K., L. Y., and Wang R. (1999). *Quality Information and Knowledge*, Prentice-Hall PRT, Upper Saddle River, NJ.

Lubit, R. (2001). “Tacit Knowledge and Knowledge Management: The Key to sustainable Competitive Advantage.” *Organizational Dynamics*, 29(3).

Pazer, D. P. B. a. H. L. (1985). “Modeling Data and Process Quality in Multi-Input, Multi-Output Information Systems.” *Management Science*, Vol. 31(No. 2), pp. 150-162.

Salvatore Belardo, D. P. B., and Harold L. Pazer. (2002). “Design of Information Systems: A Knowledge Quality Perspective.” , Paper under Review, Data on File.

Senge, P. M. (1990). *The Fifth Discipline: The Art and Practice of the Learning Organization*, Doubleday/Currency., New York.

Strong, R. W. a. D. (1996). “Beyond Accuracy: What Data Quality Means to Data Consumers.” *Journal of Management Information Systems*, Vol. 12(No. 4), pp. 5-34.

Appendix: Model Formulation

```
Quality of System=
        Normal quality*Effect on quality from schedule
        overrun*Effect on quality from cost overrun
        ~       Dmnl
        ~               |

Scheduled completion date= INTEG (
        Adjustment to schedule,
                Estimated completion date)
        ~       Month
        ~               |

IUT F(
        [(0,0)-
        (1,1)],(0,0.100877),(0.0948012,0.122807),(0.253
        823,0.149123),(0.370031,0.214912\

                ),(0.458716,0.350877),(0.504587,0.52
        193),(0.562691,0.692982),(0.666667,0.79386),(0
        .813456\
                ,0.885965),(0.996942,0.960526))
        ~       Dmnl
        ~               |

TDOK F(
        [(0,0)-
        (1,0.5)],(0,0.0460526),(0.116208,0.0635965),(0.
        342508,0.122807),(0.590214,0.241228\

                ),(0.69419,0.342105),(0.82263,0.4495
        61),(0.996942,0.497807))
        ~       Dmnl
        ~               |

TDK F(
        [(0,0)-
        (1,1)],(0,0.100877),(0.0948012,0.122807),(0.253
        823,0.149123),(0.370031,0.214912\

                ),(0.458716,0.350877),(0.504587,0.52
        193),(0.562691,0.692982),(0.666667,0.79386),(0
        .813456\
                ,0.885965),(0.996942,0.960526))
        ~       Dmnl
        ~               |

Developer knowledge acquisition rate=
        Effect on developer IS knowledge*Initial
        developer knowledge*Transfer of domain
        knowledge
        ~       Dmnl
        ~               |

Tacit user knowledge=
        IUT F(Initial user tacit level)
        ~       Dmnl
        ~       Initial value: 0.6
        |

EDK F(
        [(0,0)-
        (1,1)],(0,0.0921053),(0.0550459,0.328947),(0.15
        5963,0.583333),(0.330275,0.776316\

                ),(0.504587,0.872807),(0.663609,0.91
        2281),(0.810398,0.942982),(0.993884,0.973684)
        )
        ~       Dmnl
        ~               |

Initial user tacit level=
        0.5
        ~       fraction
        ~               ~
                :SUPPLEMENTARY
        |

Transfer of domain knowledge=
        TDOK F(Initial user domain knowledge)
        ~       Dmnl
        ~               |

IUK F(
        [(0,0)-
        (1,1)],(0,0.0921053),(0.0795107,0.289474),(0.18
        3486,0.473684),(0.35474,0.688596\

                ),(0.522936,0.802632),(0.675841,0.87
        7193),(0.828746,0.925439),(0.993884,0.973684)
        )
        ~       Dmnl
        ~               |

Initial explicit developer knowledge level=
        0.8
        ~       fraction
        ~               |

Explicit user knowledge=
        IUK F(Initial user explicit knowledge)
        ~       Dmnl
        ~       0.8 as initial value
        |

Initial user explicit knowledge=
        0.8
        ~       fraction
        ~               |

TIK F(
        [(0,0)-
        (1,0.5)],(0,0.0789474),(0.0948012,0.203947),(0.
        201835,0.348684),(0.412844,0.445175\

                ),(0.70948,0.489035),(0.993884,0.493
        421))
        ~       Dmnl
        ~               |

Tacit developer knowledge=
        TDK F(Initial tacit developer knowledge level)
        ~       Dmnl
        ~               |

Initial tacit developer knowledge level=
        0.7
        ~       fraction
        ~               |

User knowledge acquisition rate=
        Effect on user domain knowledge*Initial user
        domain knowledge*Transfer of IS knowledge
        ~       Dmnl
        ~               |
Explicit developer knowledge=
        EDK F(Initial explicit developer knowledge
        level)
        ~       Dmnl
        ~               |

Transfer of IS knowledge=
        TIK F(Initial developer knowledge)
        ~       Dmnl
        ~               |

FKD F(
        [(0,0)-
        (50,0.5)],(0,0.140351),(4.89297,0.210526),(10.7
        034,0.260965),(17.4312,0.296053\

                ),(22.63,0.324561),(28.5933,0.348684
        ),(33.3333,0.366228),(38.2263,0.370614),(44.95
        41\
                ,0.377193),(50,0.394737))
        ~       Dmnl
        ~               |

Fractional knowledge decay rate=
        FKD F(Time)
        ~       fraction
        ~               |

Initial developer knowledge=
        Explicit developer knowledge*Tacit developer
        knowledge
        ~       Dmnl
        ~               |

EFT F(
        [(0,0)-
        (1.5,1.5)],(0,0.0921053),(0.0703364,0.157895),(
        0.174312,0.355263),(0.278287,0.809211\

                ),(0.41896,1.14474),(0.590214,1.3157
        9),(0.770642,1.38158),(0.993884,1.42105),(1.32
        569\
                ,1.46053),(1.48624,1.49342))
        ~       Dmnl
        ~               |

Initial user domain knowledge=
        Explicit user knowledge*Tacit user knowledge
        ~       Dmnl
        ~               |

EFD F(
        [(0,0)-
        (1,1)],(0,0.0745614),(0.0672783,0.223684),(0.14
        0673,0.438596),(0.373089,0.745614\

                ),(0.816514,0.969298),(0.993884,0.99
        5614))
        ~       Dmnl
        ~               |

Normal quality=
        0.95
        ~       fraction
        ~               |

Effect of team effectiveness=
        EFT F(Effectiveness of team)
        ~       Dmnl
        ~               |

Effect on developer IS knowledge=
        EFD F(Task specifig developer knowledge)
        ~       Dmnl
        ~               |

EFU F(
        [(0,0)-
        (1,1)],(0,0.0745614),(0.0672783,0.223684),(0.14
        0673,0.438596),(0.373089,0.745614\

                ),(0.816514,0.969298),(0.993884,0.99
        5614))
        ~       Dmnl
        ~               |

Effect on user domain knowledge=
        EFU F(Task specifig user knowledge)
        ~       Dmnl
        ~               |

Effect on quality from cost overrun=
        IF THEN ELSE(Estimated cost to complete >
        50, 0.65 , 1 )
        ~       Dmnl
        ~               |

Effect on quality from schedule overrun=
        IF THEN ELSE(Anticipated schedule overrnun
        > 0, 0.55 , 1 )
        ~       fraction
        ~               |

Estimated cost to complete based on progress=
        (Tasks to do/People productivity)*Finished tasks
        ~       task/fraction
        ~               |

Estimated cost to complete=
        Budgeted cost to complete project+Estimated
        cost to complete based on progress
        ~       fraction*task
        ~               |

Budgeted cost to complete project=
        Total tasks for project+Tasks remaining/People
        productivity
        ~       task/fraction
        ~               |

Effectiveness of team= INTEG (
        +"Increase eff. from gaining knowledge"-
        Effectiveness decay rate-"Loss of eff. from
        pressure"\
                ,
                0.5)
        ~       Dmnl
        ~               |

Pressure from schedule overrun=
        PF F(Anticipated schedule overrnun)
        ~       Dmnl
        ~               |

Anticipated schedule overrnun=
        Scheduled completion date-Initial scheduled
        completion date
        ~       Month
```

```
NT F(
	[(0,0)-
	(50,1)],(0,0.210526),(5.04587,0.368421),(13.608
	6,0.574561),(28.5933,0.807018)\
		,(49.6942,0.973684))
	~	Dmnl
	~		|

Tasks remaining=
	Total tasks for project-Finished tasks
	~	task
	~		|

PP F(
	[(0,0)-
	(1,1)],(0.0030581,0.0921053),(0.140673,0.23245
	6),(0.217125,0.464912),(0.321101\

		,0.820175),(0.492355,0.969298),(0.65
	4434,0.95614),(0.779817,0.824561),(0.862385,0
	.517544\

		),(0.95107,0.311404),(0.990826,0.280
	702))
	~	Dmnl
	~		|

Time to complete remaining tasks=
	IF THEN ELSE(Tasks remaining >0, Tasks
	remaining/(Size of team*People productivity),\
		 0)
	~	task/(Month*people)
	~		|

"Loss of eff. from pressure"=
	Pressure from schedule overrun*Effectiveness of
	team
	~	Dmnl
	~		|

PF F(
	[(0,0)-
	(20,1)],(0,0.0263158),(2.69113,0.127193),(6.177
	37,0.337719),(8.68502,0.662281\

		),(12.6606,0.846491),(17.1254,0.9385
	96),(19.8777,0.982456))
	~	Dmnl
	~		|

People productivity=
	PP F(Effect from team effectiveness*Fraction of
	tasks remaining)
	~	fraction
	~		|

Progress rate=
	People productivity*Size of team
	~	task/people
	~		|

Adjustment to schedule=
	(Estimated completion date-Scheduled
	completion date)/Time to adjust schedule
	~	task/months
	~		|

Finished tasks= INTEG (
	Progress rate,
		0)
	~	task
	~		|

Fraction of tasks remaining=
	Tasks to do/Total tasks for project
	~	Dmnl
	~		|

Total tasks for project=
	42
	~	task
	~		|

Tasks to do= INTEG (
	-Progress rate,
		Total tasks for project)
	~	task
	~		|

ET F(
	[(0,0)-
	(2,1)],(0,0.0438596),(0.0733945,0.232456),(0.26
	9113,0.403509),(0.617737,0.570175\

		),(1.11927,0.72807),(1.98777,0.99122
	8))
	~	Dmnl
	~		|

KN F(
	[(0,0)-
	(10,1)],(0,0.166667),(0.58104,0.328947),(1.2232
	4,0.486842),(2.47706,0.644737)\

		,(4.22018,0.75),(5.99388,0.833333),(
	7.95107,0.903509),(9.96942,0.97807))
	~	Dmnl
	~		|

Effect from team effectiveness=
	ET F(Effectiveness of team)
	~	Dmnl
	~		|

Effect from team knowledge=
	KN F(Task specifig developer knowledge+Task
	specifig user knowledge)
	~
	~		|

Developer knowledge decay rate=
	Fractional knowledge decay rate*Task specifig
	developer knowledge
	~	fraction
	~		|

Effectiveness decay rate=
	Fractional decay rate*Effectiveness of team
	~	Dmnl
	~		|

Estimated completion date=
	Time to complete remaining tasks+Time
	~	Month
	~		|

Fractional decay rate=
	0.12
	~	fraction
	~		|

Initial scheduled completion date=
	50
	~	Month
	~		|

Size of team=
	2
	~	people
	~		|

Task specifig developer knowledge= INTEG (
	Developer knowledge acquisition rate-Developer
	knowledge decay rate,
		Initial developer knowledge)
	~	Dmnl
	~		|

Task specifig user knowledge= INTEG (
	User knowledge acquisition rate-User knowledge
	decay rate,
		Initial user domain knowledge)
	~	Dmnl
	~		|

Time to adjust schedule=
	4
	~	Month

**************************************************
	.Control
**************************************************
	********~
		Simulation Control Parameters
	|

FINAL TIME  = 50
	~	Month
	~	The final time for the simulation.
	|

INITIAL TIME  = 0
	~	Month
	~	The initial time for the simulation.
	|

SAVEPER  = 1
	~	Month [0,?]
	~	The frequency with which output is
	stored.
	|

TIME STEP  = 1
	~	Month [0,?]
	~	The time step for the simulation.
	|
```

Back to the Top