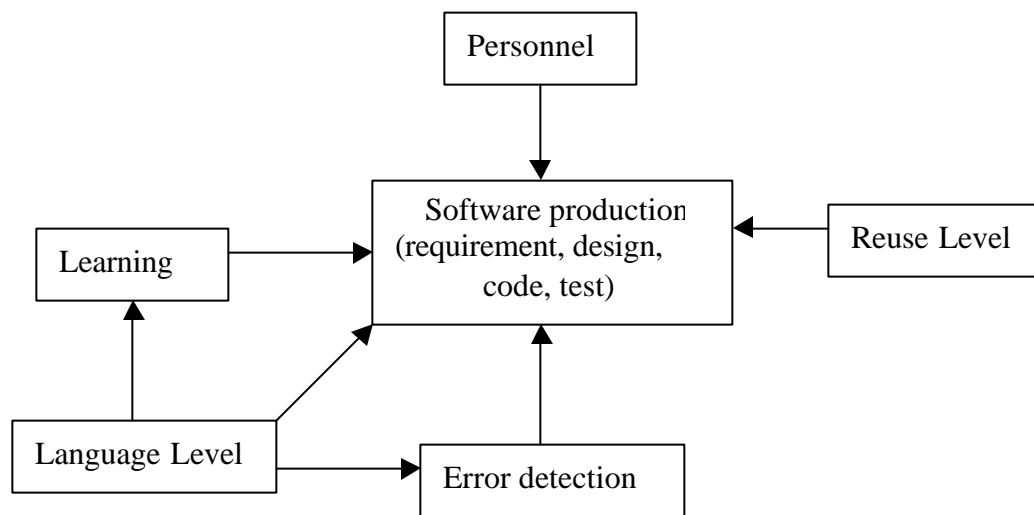


Simulation Report
Reuse and High Level Languages

1. Introduction

Reducing the schedule of a software development project becomes more important than reducing its cost. The purpose of this project is to test CORADMO parameter hypotheses of re-use and very high-level languages (RVHL). This parameter affect software schedule and cost roughly. By using different kind of programming language and reuse level, it will affect the schedule and effort. The purpose of this model is to study the language and reuse effects on a software development project. To understand language effects on software process, it will also benefit for management planning strategy.

2. Background



See attach papers for reference behavior.

Different languages are directly related to the effort of software development process, such as the design, coding, and testing. High-level programming language comes with source code library, which make the code generation more efficiencies. It also benefits the test process, since the source code library had been build and inspected. Different language level does affect the error level in each phase, because the richer the language the more you have test or verify as you build. However, the richer of the language, the more training is involved. Reusing assets also reduces human efforts and increase productivity on each phase.

3. Model Development

The modeling approach is to develop a model, which can be used to understand language effects, and reuse assets on software development process.

First is to understand each of the software development phases and their specific functions. There are four phases used on this model, requirements, design, coding, and approve phases. Each phase is used COCOMO phase effort estimation data, requirements is 16%, design is 25%, coding is 39%, and approve is 20%. A personnel profile also added into the process, this represents the size of the project personnel. This creates a model of software production structure.

From the Verner-Tate study, "Estimating Size and Effort in Fourth-Generation Development," which found out more information on language effects on effort. Use these data for different language level, effort ratio 1:6.7 for 3GL to 4GL, and ratio 1:3 for Assembly to 3GL. This creates language level effect for controlling the development of productivity and efforts in each phase.

Next interview with Mr. Donald J. Reifer, he suggested that by using high-level language, it requires more training. In terms of training, this is using a learning curve to represent, where 90% learning curve for 3GL, and 80% learning curve for 4GL. This learning curve then integrated into the model. There are two learning curves used in the model, Log-Linear and DeJong learning curve. User can only select one of the learning curves at a time for the simulation. Mr. Reifer also suggested error rates for different languages, such as 6 errors per KSLOC on Ada, and 10 errors per KSLOC on Fortran. These error rates have been applied on the design and code phases according to the language level using.

A reuse structure has been added into the model. This will reduce the project size flows into the software development phases, when reuse cases are applied. This reduces efforts by specifying the percentage of reuse use on the project.

For the personnel profile, communication overhead factor has been added. This will solve the problem when infinite people added can finish the project in zero time. Instantly, adding more people increases the communication overhead and reduces productivity.

A level for reuse code has been added, which can monitor the actual reuse level during the project development.

Data acquisition and reading, the sources of data are from books and journals below:

- Verner J, Tate G, Estimating size and effort in fourth generation development, IEEE Software, July 1988, pp. 15-22

- Jeffrey S. Poulin, Measuring Software Reuse : principles, practices, and economic models, Addison-Wesley, 1997

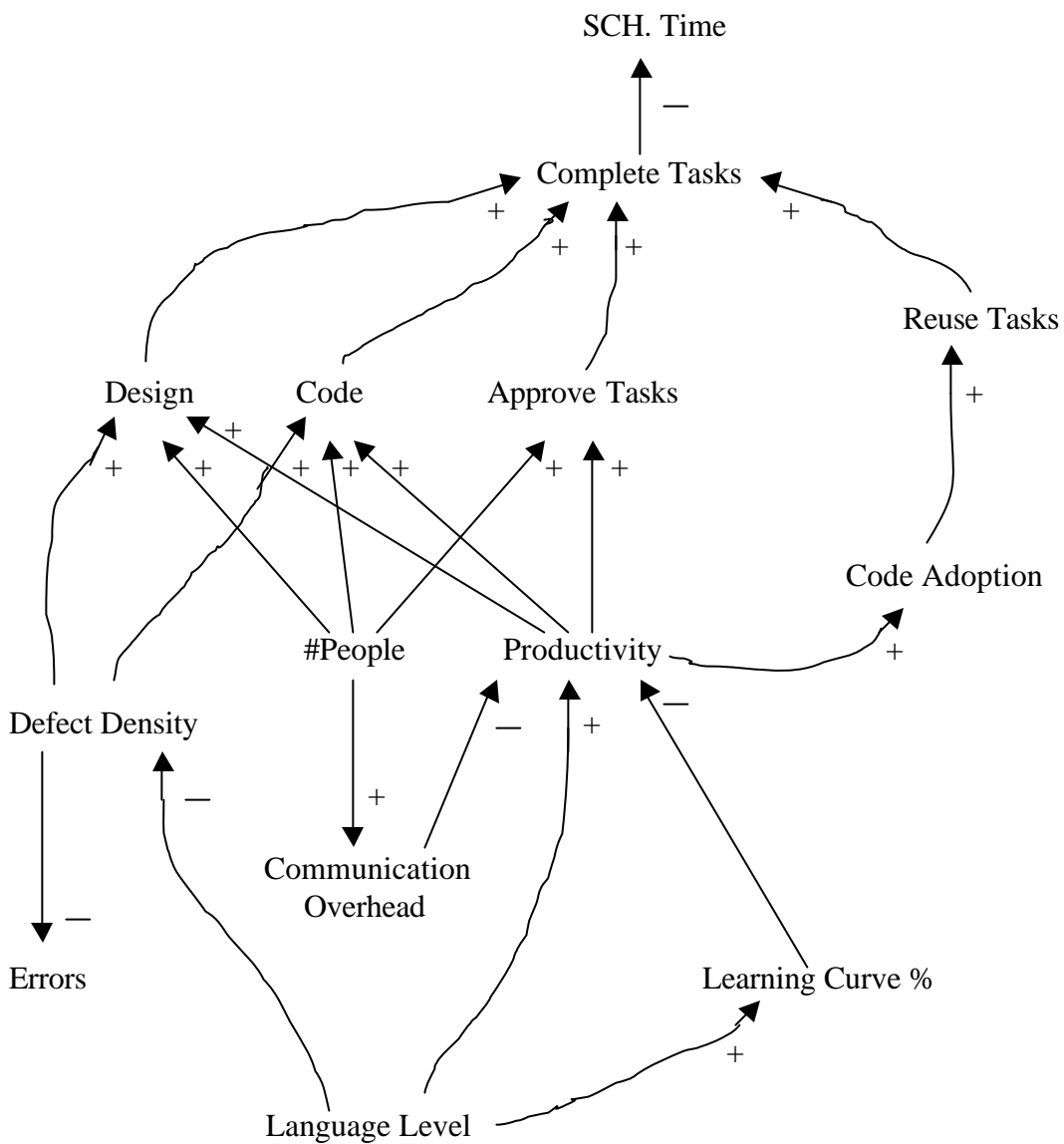
- Donald J. Reifer, Practical Software Reuse, 1997

- 4GL data (Verner-Tate results)

- COCOMO phase efforts estimate

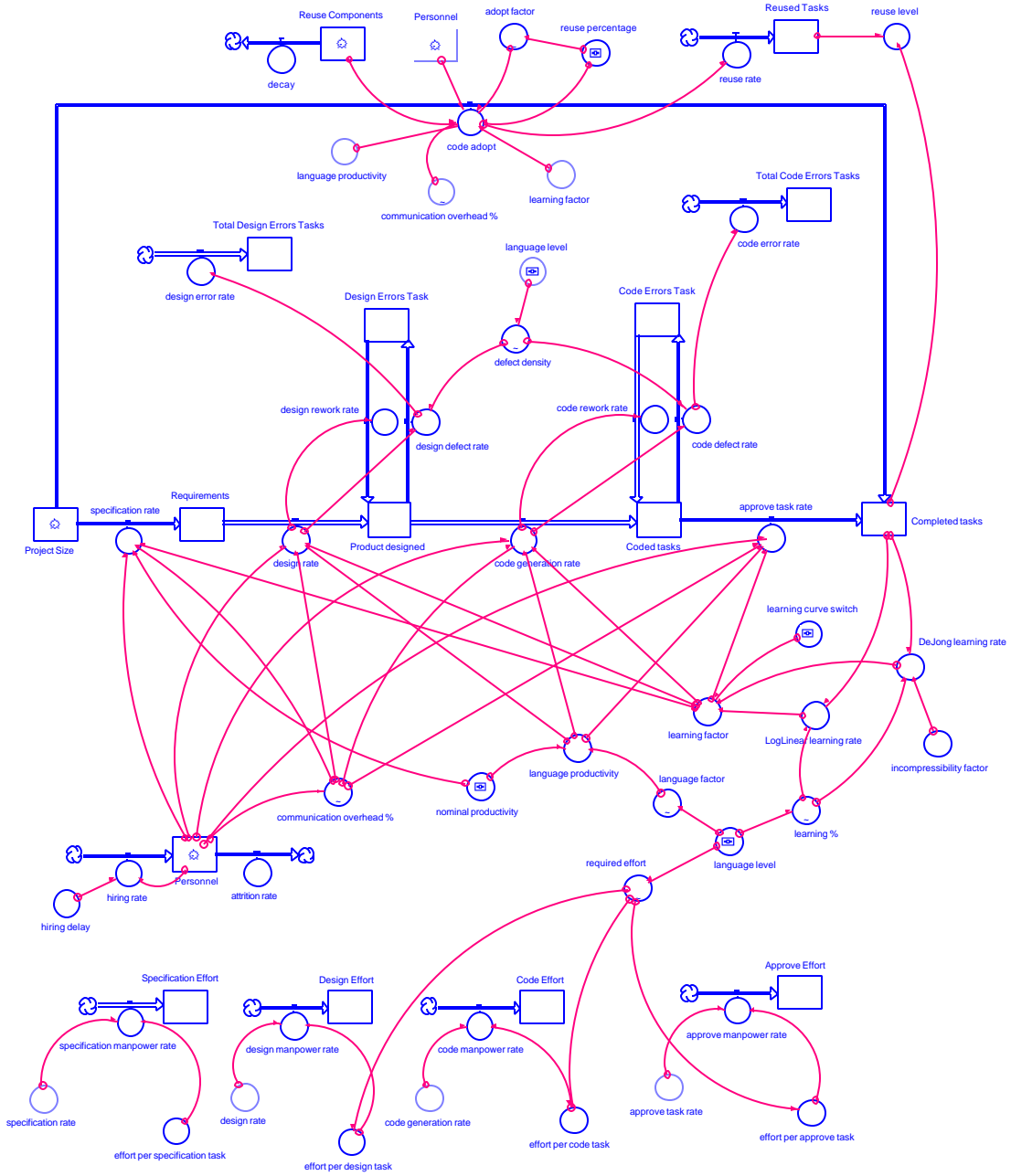
4. Model Description

Casual Loop Diagram

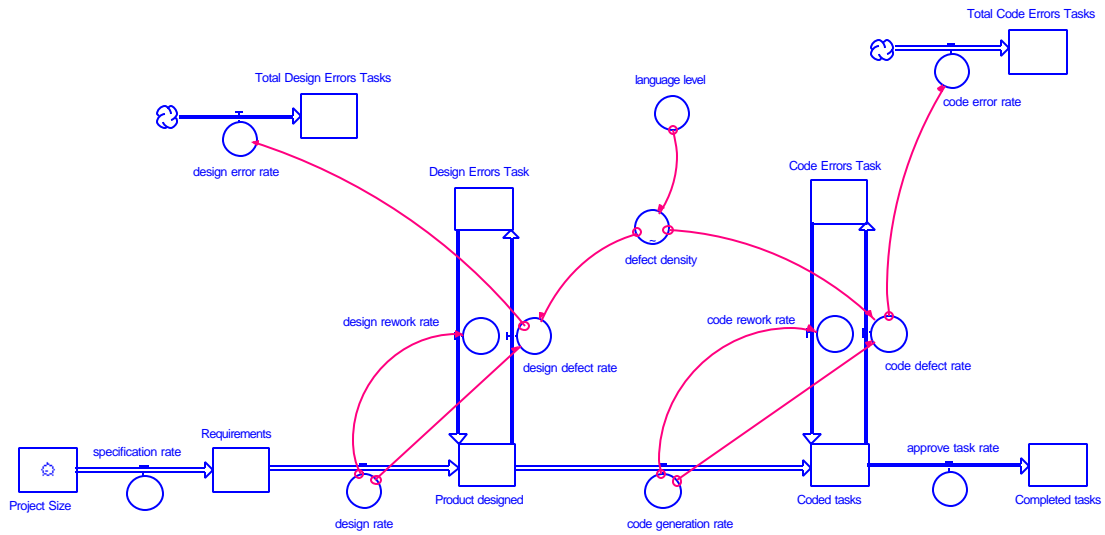


The about diagram show the language level affected defect density, productivity and learning curve %. Increase language level will decrease defect density and less error occurs. Higher language level also increases the productivity. But increase language level will increase learning curve %, which result a slower learning rate. By increasing a larger number of human resource, it increase communication overhead, thru decrease productivity. Less defect density, a certain amount of people and high productivity will increase design, code and approve tasks rate. Higher productivity also increases code adoption and increase code reuse. Since design, code, approve tasks, and code reuse are all increase, it is more efficiency to complete tasks. A higher complete tasks rate reduces the schedule time.

Complete Model Diagram

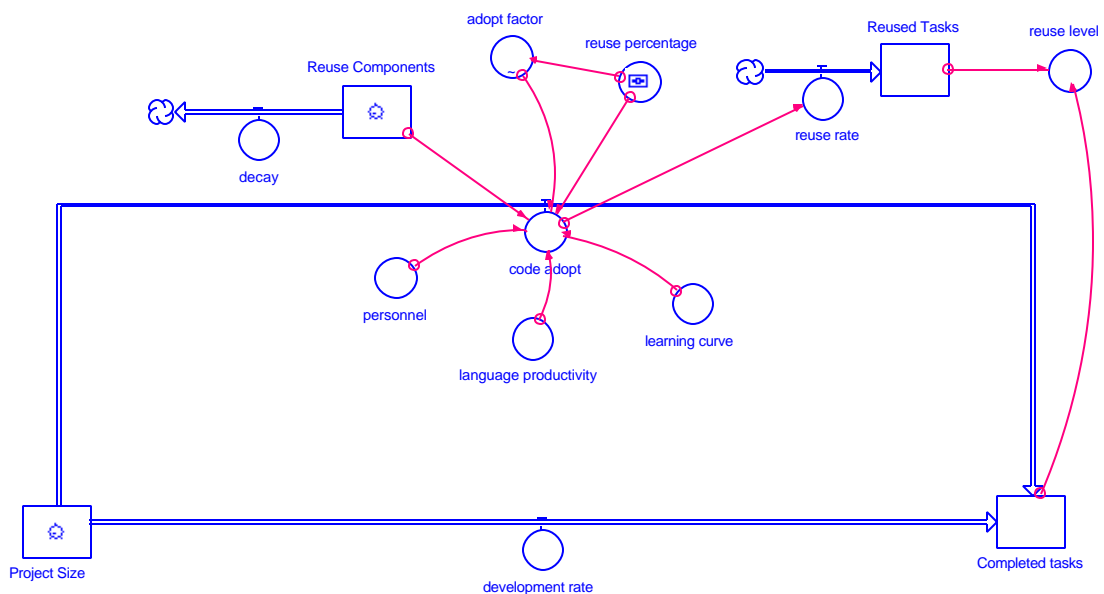


Software Production Phases Diagram



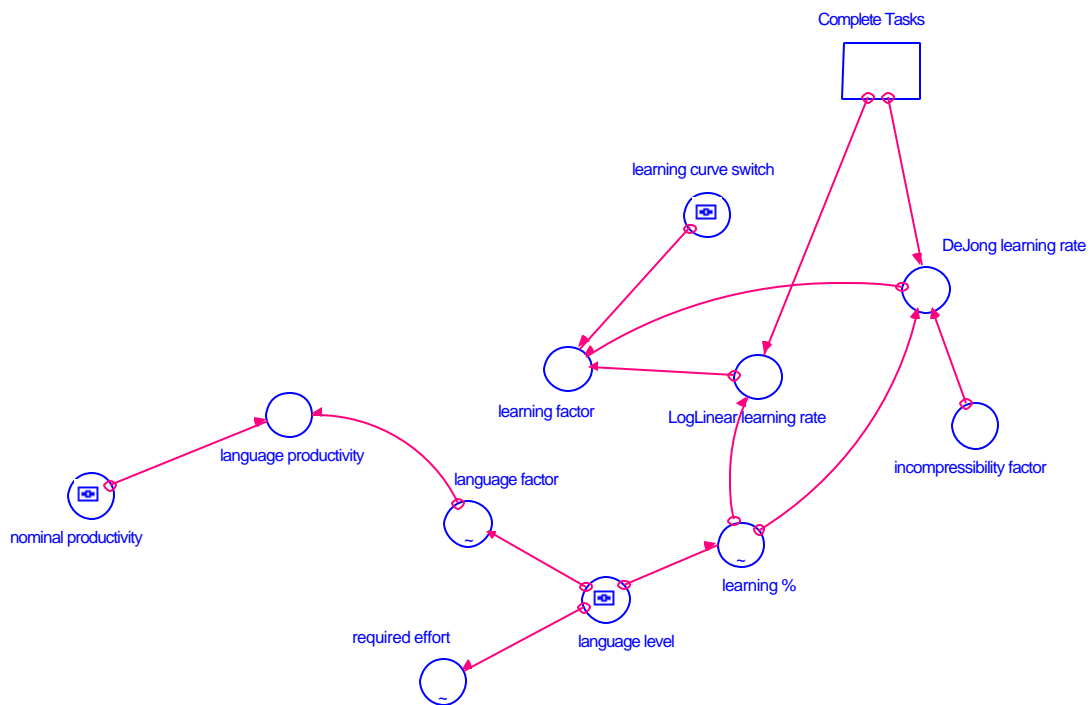
A software development starts with a project size, then, it flows into requirements, design, code, and approve phases to complete tasks. The flow rate on each phases are determine by (communication overhead*personnel*productivity / phase efforts). It has error detection on design and code phases, and it requires reworks to flow back into the product line. The error rate depends on the defect density, which is controlled by language level. Assume an error found it requires 3 times more effort to rework and fix the error. Rework rate is (normal flow rate / 0.3)

Reuse Structure Diagram



A reuse structure reduces the project size when there is any reuse percentage specified. The reuse percentage is used to specify reuse portion in the project. The reuse components act like a library to provide reuse assets. When there is no reuse component available, reuse could not be activate even the reuse percentage had been specified. The reuse components have a decay rate due to the change of needs or old components. The adopt factor use to adjust the adopt reuse tasks rate. Adoption rate is (personnel*productivity*learning curve* reuse percentage / adopt factor)
 The reuse level is used to check the actual reuse in the project.
 [Reuse Level = (reuse tasks/complete tasks)*100%]

Language and Learning Curve Diagram



Language level controls language factor where language productivity use language factor to modify nominal productivity.

[Language productivity = nominal productivity / (1-language factor/100)]

Language level also affects the required effort, which is used to calculate the effort on each phase.

There are two learning curves used, and they can be choice by using the learning curve switch. Learning curve switch set to 0 = Log-Linear learning curve. Learning curve switch set to 1 = DeJong learning curve. The learning curves require learning percentage, which is also controlled by language level.

Log-Linear learning curve: [$y = a (x)^n$]

DeJong learning curve: [$y = a (M + (1-M) (x)^n$]

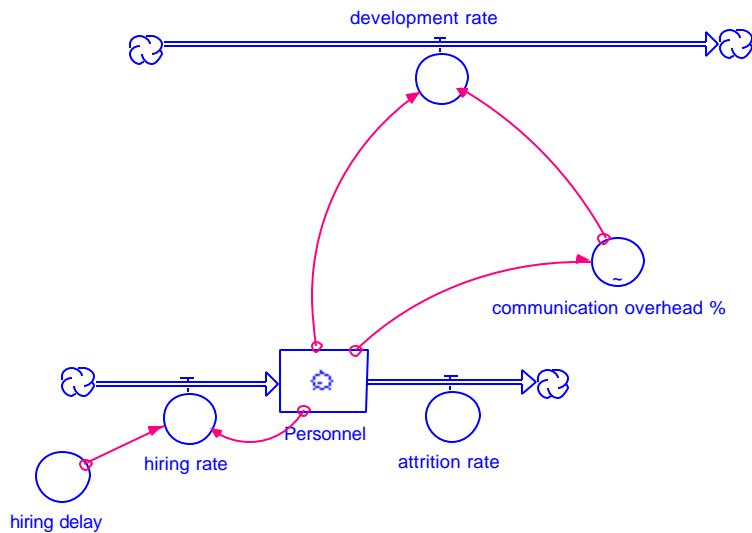
“a” is a constant represents the cost of the first unit.

“x” represents the complete tasks at that time.

“n” represents the slope of the learning curve, $slop = \log(\text{learning rate } \%) / \log(2)$

“M” is incompressibility factor for DeJong learning curve

Personnel Profile Diagram



This defines the human resource for the project. The hiring rate affect by hiring delay, and attrition rate represent personnel lost. Add more people, which will increase communication overhead. Personnel and communication overhead are one of the factors on the software development rate. Increase personnel will also increase communication overhead, which will decrease the software development rate.

Model inputs and outputs

Inputs	Units
Language level	Number of level
Project size	Functional-Point
Personnel	Number of Person
Hiring delay	Month
Attrition rate	People per month
Reuse percentage	%
Reuse components	Number of component
Decay rate	Components per month
Nominal productivity	Positive number
Learning curve switch	0 = Log-Linear, 1 = DeJong
Incompressibility factor	Number

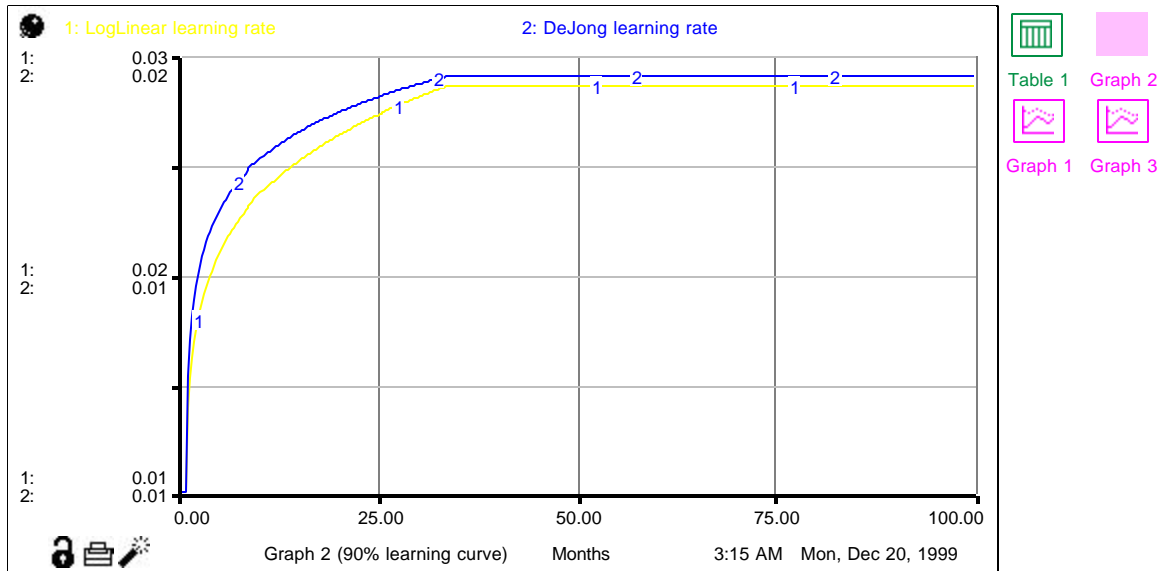
Outputs	Units
Schedule	Month
Complete tasks	Tasks
Reuse tasks	Tasks
Reuse level	%
Total design error tasks	Tasks
Total code error tasks	Tasks
Specification effort	Man-month
Design effort	Man-month
Code effort	Man-month
Approve tasks effort	Man-month

5. Model Verification and Validation

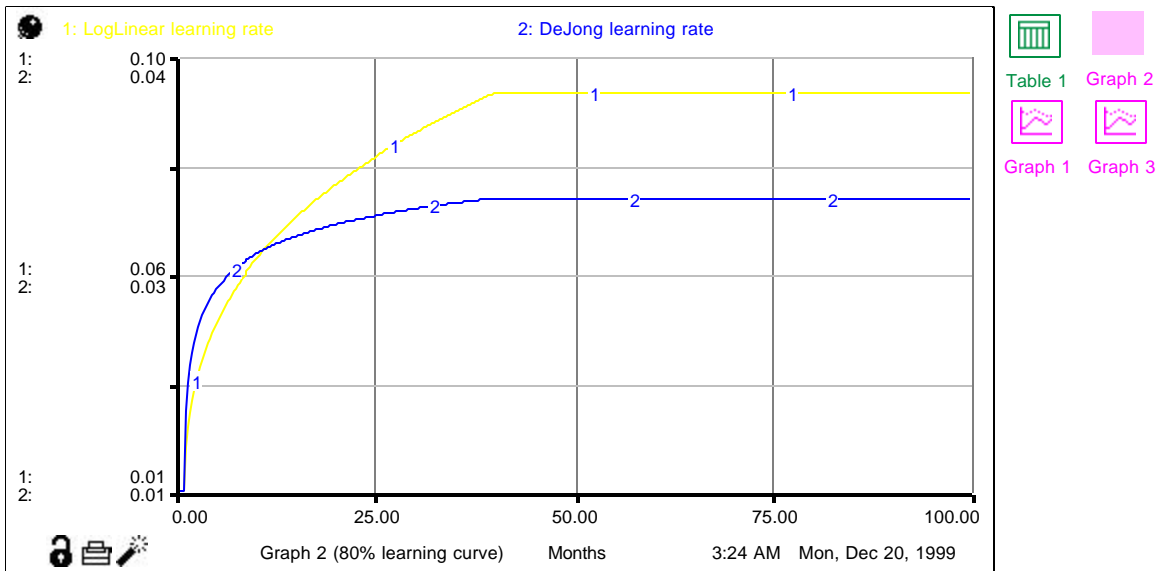
Nominal Case

Inputs	
Language level	3
Project size	1500
Personnel	10
Hiring delay	9999
Attrition rate	0
Reuse percentage	0
Reuse components	1000
Decay rate	1
Nominal productivity	1
Learning curve switch	1
Incompressibility factor	0.25

Test on learning curves



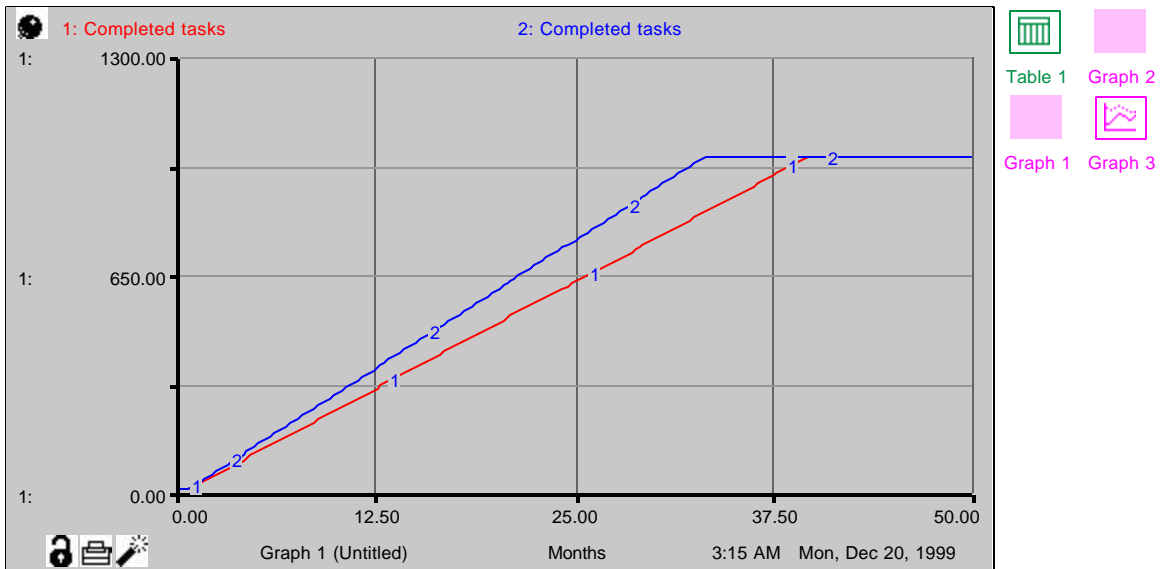
90% learning curve on Log-Linear and DeJong



80% learning curve on Log-Linear and DeJong

The 80% learning curve has higher learning rate than the 90% learning curve.

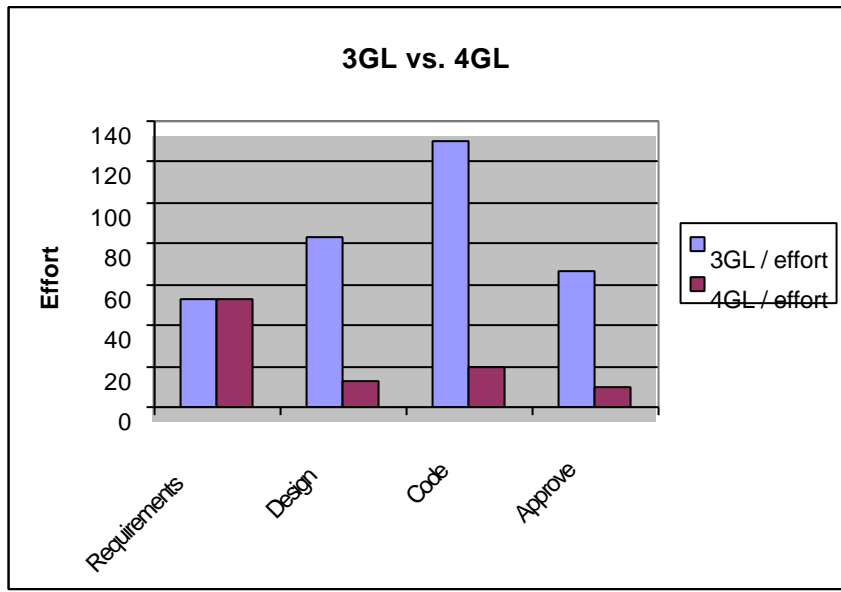
Test 1 – language level



3:15 AM Mon, Dec 20, 1999		Table 1 (Table.TXT)							
Months	31	32	33	34	35	36	37	38	39
1: Completed tasks	804.01	829.58	855.15	880.72	906.30	931.88	957.46	983.04	1,001.00
2: Completed tasks	964.86	995.52	1,001.00	1,001.00	1,001.00	1,001.00	1,001.00	1,001.00	1,001.00

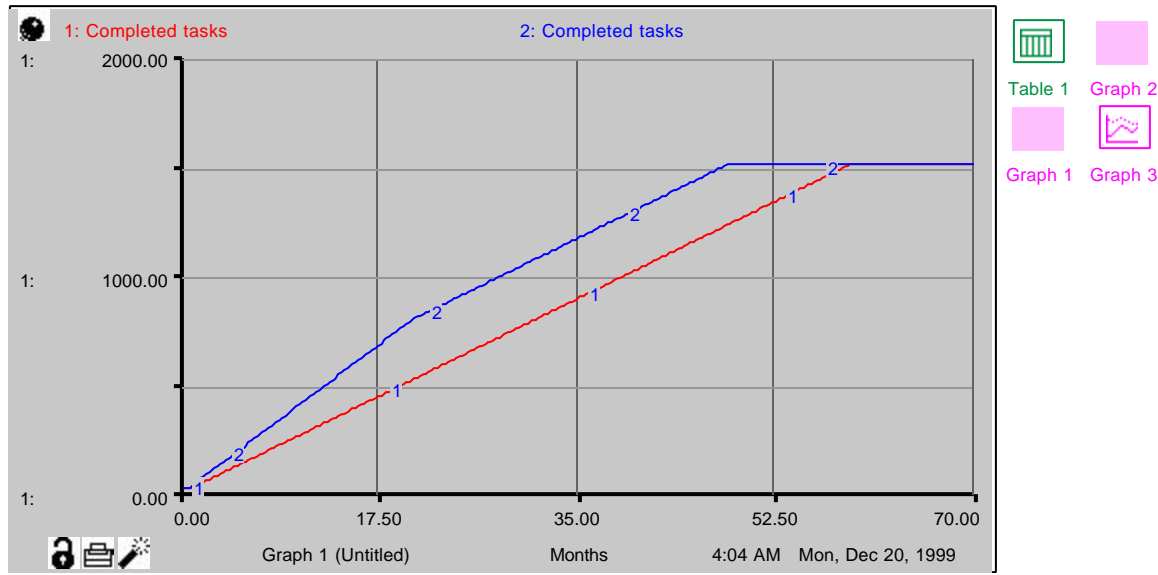
Curve 1, Third Generation Languages used → Run 1
 Curve 2, Fourth Generation Language used → Run 2

Inputs	Run 1	Run 2	Outputs	Run 1	Run 2
Language level	3	4	Schedule	39	33
Project size	1000	1000	Complete tasks	1001	1001
Personnel	10	10	Reuse tasks	0	0
Hiring delay	9999	9999	Reuse level	0	0
Attrition rate	0	0	Total design error tasks	14.9	8.9
Reuse percentage	0	0	Total code error tasks	9.9	6.0
Reuse components	1000	1000	Specification effort	53.28	53.28
Decay rate	1	1	Design effort	83.32	12.45
Nominal productivity	1	1	Code effort	129.99	19.42
Learning curve switch	1	1	Approve tasks effort	66.66	9.96
Incompressibility factor	0.25	0.25			



Higher language level which reduce the effort on design, code and approve phases, but it doesn't reduce effort on requirement phase. With 4GL, it reduce the schedule time to finish the project.

Test 2 – reuse level



Months	47	48	49	50	51	52	53	54	55	56
1: Completed tasks	1,213.11	1,238.69	1,264.28	1,289.87	1,315.46	1,341.05	1,366.64	1,392.24	1,417.83	1,443.43
2: Completed tasks	1,491.90	1,501.00	1,501.00	1,501.00	1,501.00	1,501.00	1,501.00	1,501.00	1,501.00	1,501.00

Curve 1, Third Generation Languages without reuse → Run 1
 Curve 2, Third Generation Language with 20% reuse → Run 2

Inputs	Run 1	Run 2	Outputs	Run 1	Run 2
Language level	3	3	Schedule	59	48
Project size	1500	1500	Complete tasks	1501	1501
Personnel	10	10	Reuse tasks	0	277.6
Hiring delay	9999	9999	Reuse level	0	18.5%
Attrition rate	0	0	Total design error tasks	22.4	18.2
Reuse percentage	0	20%	Total code error tasks	14.9	12.2
Reuse components	1000	1000	Specification effort	79.92	65.13
Decay rate	1	1	Design effort	124.99	101.85
Nominal productivity	1	1	Code effort	194.98	158.89
Learning curve switch	1	1	Approve tasks effort	99.99	81.48
Incompressibility factor	0.25	0.25			

Curve 1, 10 personnel → Run 1

Curve 2, 15 personnel → Run 2

Curve 3, 20 personnel → Run 3

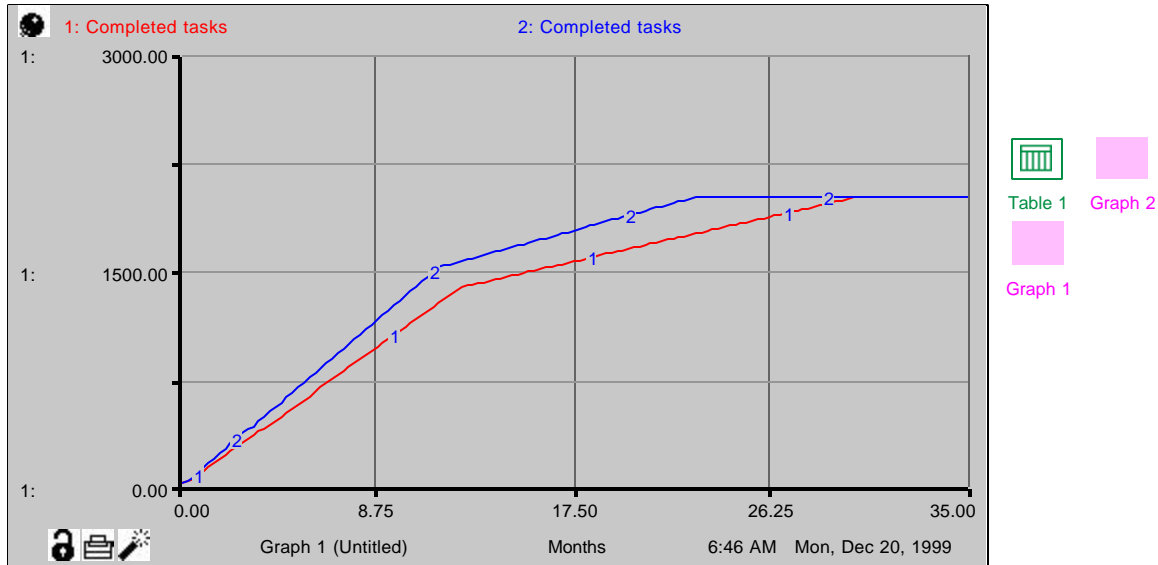
Curve 4, 30 personnel → Run 4

Inputs	Run 1	Run 2	Run 3	Run 4
Language level	4	4	4	4
Project size	1000	1000	1000	1000
Personnel	10	15	20	30
Hiring delay	9999	9999	9999	9999
Attrition rate	0	0	0	0
Reuse percentage	0	0	0	0
Reuse components	1000	1000	1000	1000
Decay rate	1	1	1	1
Nominal productivity	1	1	1	1
Learning curve switch	1	1	1	1
Incompressibility factor	0.25	0.25	0.25	0.25

Outputs	Run 1	Run 2	Run 3	Run 4
Schedule	33	24	20	22
Complete tasks	1001	1001	1001	1001
Reuse tasks	0	0	0	0
Reuse level	0	0	0	0
Total design error tasks	8.9	8.9	8.8	8.8
Total code error tasks	6.0	5.9	5.9	5.9
Specification effort	53.28	53.28	53.28	53.28
Design effort	12.45	12.45	12.45	12.45
Code effort	19.42	19.42	19.42	19.42
Approve tasks effort	9.96	9.96	9.96	9.96

Adding more people in the project, which can reduce the schedule, but too many personnel will create communication overhead. In run 3, it has 20 personnel and finishing the project in 20 months. But in run 4, 30 personnel make the project longer and finishing in 22 months.

6. Model Application and Transition



Months	21	22	23	24	25	26	27	28	29	30
1: Completed tasks	1,713.64	1,748.99	1,784.34	1,819.69	1,855.04	1,890.39	1,925.75	1,961.10	1,996.50	2,001.00
2: Completed tasks	1,960.31	2,001.00	2,001.00	2,001.00	2,001.00	2,001.00	2,001.00	2,001.00	2,001.00	2,001.00

Curve 1, 3GL with 50% reuse – run 1
 Curve 2, 4GL with 50% reuse – run 2

Inputs	Run 1	Run 2	Outputs	Run 1	Run 2
Language level	3	4	Schedule	30	22
Project size	2000	2000	Complete tasks	2001	2001
Personnel	15	15	Reuse tasks	954.1	1050.2
Hiring delay	9999	9999	Reuse level	47.68%	52.48%
Attrition rate	0	0	Total design error tasks	15.5	8.4
Reuse percentage	50%	50%	Total code error tasks	10.4	5.6
Reuse components	1000	1000	Specification effort	55.72	50.61
Decay rate	1	1	Design effort	87.15	11.83
Nominal productivity	1	1	Code effort	135.95	18.45
Learning curve switch	1	1	Approve tasks effort	69.72	9.46
Incompressibility factor	0.25	0.25			

A schedule estimates for development 2000 function points with 50% reuse. The high of the language, which finish the project faster in 22 months, the other finished in 30 months. Higher language level also has less errors rate on coding.

Limitations of the model

- This model can only test on 3GL and 4GL languages, because there is not enough data for other language levels such as error rate and learning curve.
- Reuse percentage cannot be set to 100%. It is impossible to reuse 100%, otherwise no software need to be develop.
- The model cannot fulfill the excite input reuse percentage, it has a little different reuse level.
- Personnel profile cannot exit 60 people.
- Setting the nominal productivity to infinity, which the project can finish in zero time.

7. Conclusions and Recommendations

A higher programming language would also consider as reuse, because it does more automatic code generation. This is like a library concept, which reuse the packaging code. In the real world, the reuse policy is not only for code reuse. Reuse also includes pattern reuse, design reuse, test case reuse...etc. In each case, they have different degree of levels and factors. This model builds the reuse as reduce the hold software size.

The modeling tools used to create this model is ithink 5.0 software from High Performance System, Inc. on PC Platform with Windows NT 4.0 operating system.

8. Appendices