

Dynamic Model for COTS Glue Code Development and COTS Integration

Wook K. Kim and Jongmoon Baik

**Department of Computer Science
University of Southern California
{wookkyuk,jobaik}@usc.edu**

1. Introduction

1.1 Problem Statement and Description

One of the most significant changes in the software development area is the trend of building systems incorporating pre-existing software, with special emphasis upon the use of commercial-off-the-shelf (COTS) software components. COTS describes software commercially available as stand-alone products and which offer specific functionality needed by a larger system into which they might be incorporated. The purpose of using COTS is to lower overall development costs with involving less development time by taking advantage of existing, market-proven, and vendor supported products. But we have no control over the functionality, performance, and evolution of COTS products since their Black-Box nature. Besides, most COTS products are not designed to inter-operate with each other and most COTS vendors do not support glue code (sometimes called glueware and binding code). So, most software development teams that use COTS components have difficulties in estimating effort and schedule for COTS glue code development and integration of them into application systems.

Without the glue code, the components would be un-integratable and COTS-based systems can be difficult to comprehend, less evolvable than intended, and less reliable than the original components [4].

Most software development teams have also problems in deciding when they start to develop the glue code and to integrate them into system. It highly depends on some factors such as number of COTS components, requirement specification, and availability of COT component required for the developing system.

1.2 Purpose of Study

The engineering of COTS-based systems continues to involve significant technical risk. A good indicator of the as-yet unresolved difficulties involved in building COTS-based systems is the “glue code” used to integrate components. This code is often ad hoc and

brittle, but it is needed to repair mismatched assumptions that are exhibited by the components being integrated.

The main objective of this study is to understand how glue code development process and COTS integration process affect each other and analyze how they have an effect on schedule and efforts of glue code development and integration of components into the developing system throughout the lifecycle.

1.3 Purpose of Model Building

In order to resolve the problems mentioned in the previous section, we will implement dynamic simulation model for glue code development and COTS integration. Using this model, we will simulate how integration process is affected by various factors such as number of COTS, percentage of updated and new COTS, and requirement specification. Especially, we will calibrate COCOTS component parameters (COTS Product Maturity, COTS Supplier Product Extension Willingness, COTS Product Interface Complexity, COTS Supplier Product Support, COTS Supplier Provided Training and Documentation) [1]. We will analyze staffing levels according to concurrency profiles between glue code development and application development (or custom component development) and simulate how various starting points of glue code development have an effect on system integration process productivity. And, we will analyze the impact of new parameters such as ratio of new and updated COTS component and number of COTS component. Finally, we will suggest several efficient scenarios for glue code development and integration process.

2. Background

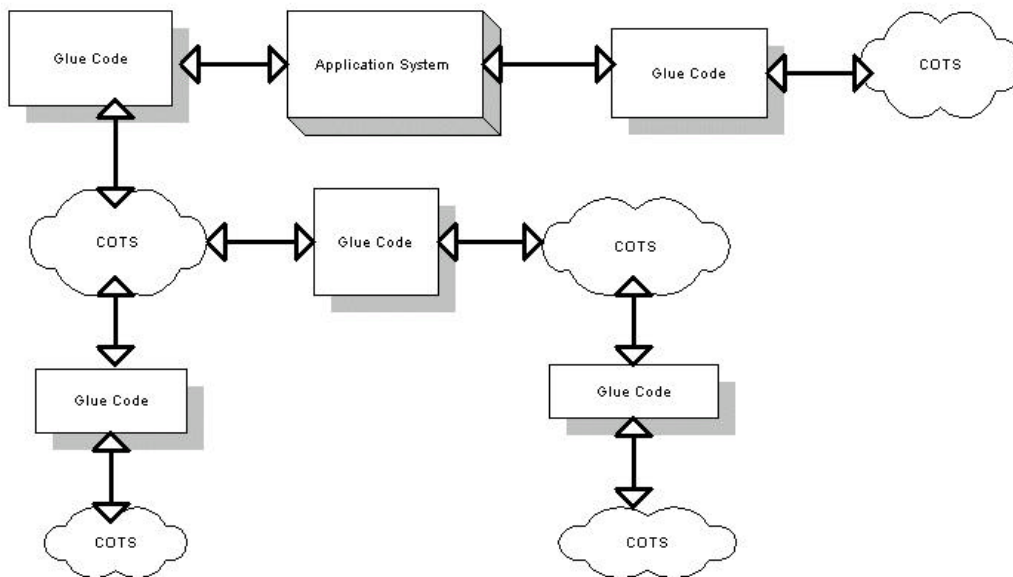
2.1 System Description

The system dynamic model we built consists of four sub-models: application development, glue code development, COTS component parameter, and human resource models. Glue code for COTS components is the new code needed to get a COTS product integrated into a larger system. It is usually clearly defined as connected to the COTS component itself, acting more as a “bridge” between the COTS component and the system into which it is being integrated [1]. It can be code needed to connect a COTS component either to higher-level system code, or to other COTS components used in the system as shown in Figure 2-1.

Glue code is considered as one of following: 1) any code required to facilitate information or data exchange between the COTS component and the application, 2) any code needed to "hook" the COTS component into the application, even though it may not necessarily facilitate data exchange, and 3) any code needed to provide functionality that was originally intended to be provided by the COTS component, AND which must interact with that COTS component [1].

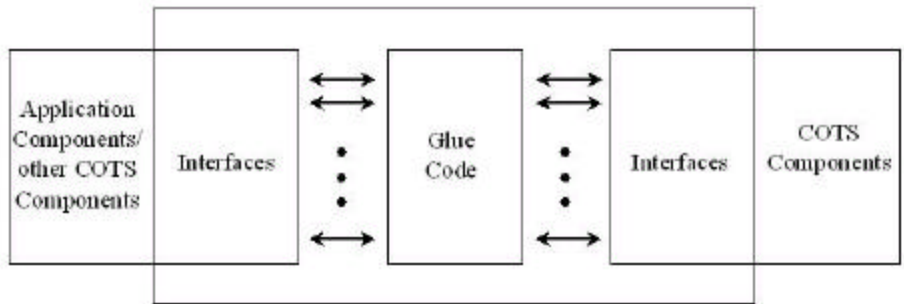
COCOTS parameters for the Glue code development model are divided into three categories. First, *Personnel Drivers* represent characteristics of COTS integrator personnel. Second, *Application/System Drivers* represent characteristics of the system into which COTS is being integrated. Third, *COTS Component Drivers* are the most important drivers because they represent characteristics of the integrated COTS component itself, so we will simulate and verify our simulation model based on these parameters. These *COTS Component Drivers* are divided as COTS Product Maturity, COTS Supplier Product Extension Willingness, COTS Product Interface Complexity, COTS Supplier Product Support and COTS Supplier Provided Training and Documentation.

Figure 2-1 represents overall COTS-based system diagram. COTS can be multiple different modules and glue code modules are needed for connecting COTS with application system or between COTS. Although COTS can be integrated into application system directly, many of COTS also can be integrated into another COTS.



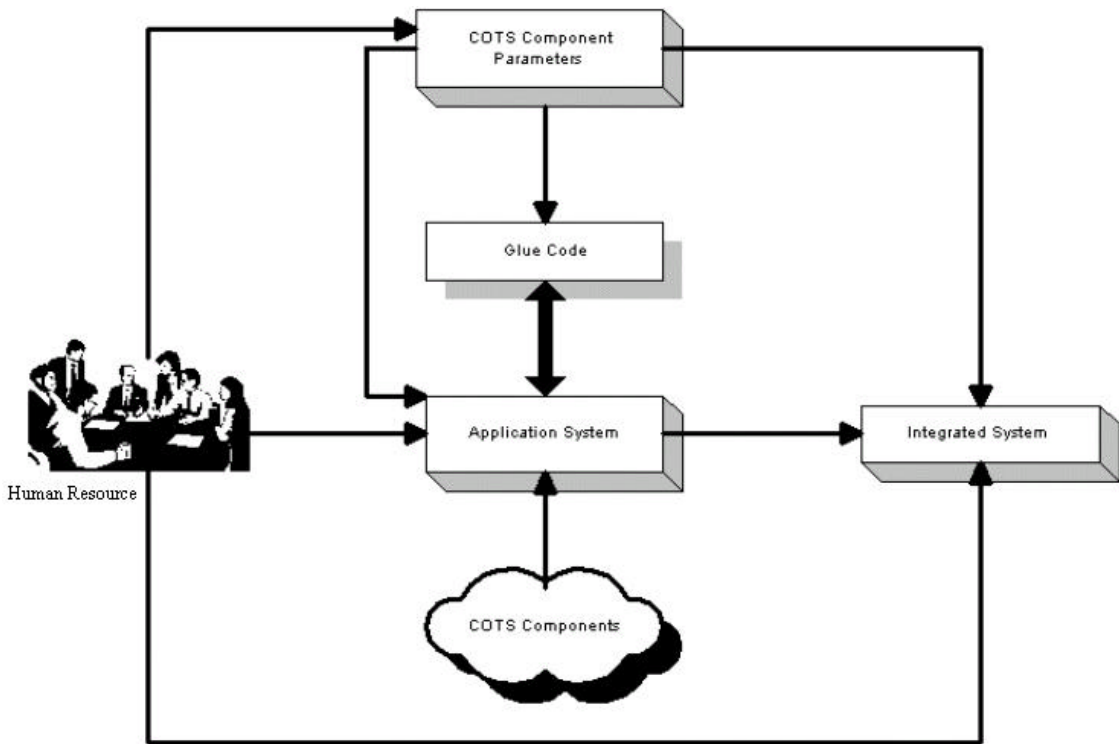
[Figure 2-1 Overview Diagram]

As mentioned earlier, COTS component is integrated with application component or another COTS component. Figure 2-2 adopted from [1] shows glue codes are used for connecting interfaces of COTS component and application component. Here, glue code development process is determined by interfaces of these two components. If the nature of the interfaces of the COTS component and the application component is complex, it is more difficult to develop glue code for those components.



[Figure 2-2 Glue Code Configuration Diagram]

Figure 2-3 represents overall system configuration of COTS integration and glue code development process. COTS component parameters affect glue code development process and some of the parameters also affect application code development process. Nature of COTS component has an effect on application system development. However, nothing can affect COTS component because of the black-box nature. Integrated system is made up from application system and glue code. Glue code component and COTS component determines the integration productivity. We will implement our simulation model based on this system configuration model.



[Figure 2-3 System Configuration of COTS glue code development and integration]

2.2 System Reference Behavior

It is very important to observe reference behavior patterns in order to characterize the dynamic phenomenon occurred in our simulation model.

1. The productivity rates of glue code development tend to be smaller than that of application development since glue code development is constrained by the architecture of application system.
2. System integration productivity rate tends to be decreased if number of COTS is increased. However, number of COTS component does not affect glue code development. For example, let's suppose that two application systems involved in COTS integration project. Glue code requirements for two applications are each 1000 SLOC. But one application system integrates one COTS component and the other application system integrates 10 COTS components. In this case, glue code development productivity for two cases is same. However, integration productivity for 10 COTS is smaller than that of one COTS.

	Number of COTS	Total Glue Code SLOC	Glue Code per a COTS
Case A	1	1000	1000
Case B	10	1000	100

[Table 2-1 Number of COTS analysis]

- **Glue Code Development Schedule:** Case A = Case B
 - **Integration Schedule:** Case A < Case B
3. Glue code development rate is affected by the percentage of new and updated COTS component. If the percentage of updated COTS component is increased, glue code development rate will be increased because glue code also can be updated from the previous version.
 4. Effort profiles of glue code development tend to be smaller than that of application development because requirements of glue code development is smaller than that of requirements of application development and this behavior is observed from COCOTS Data Collection Program from USC-Center for Software Engineering [6].

2.3 Assumptions and Underlying Rationale

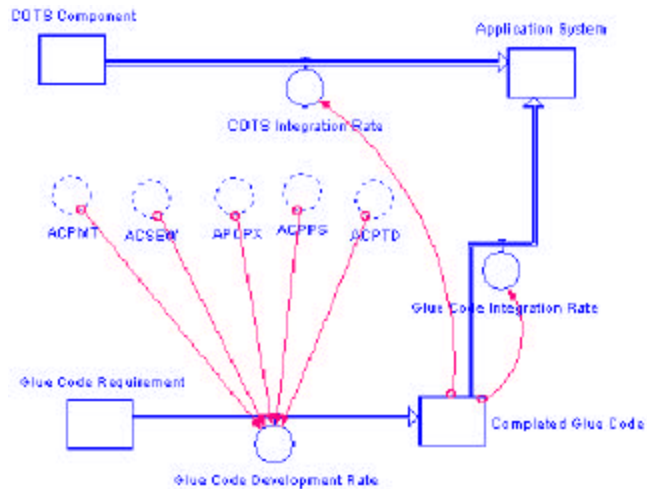
In order to develop the simulation model, we made some assumptions.

1. This simulation model is focused on the glue code development and their effect to the integration process, so application development and workforce effort is briefly simulated.
2. Parameters, which are not COTS component drivers, are set to be nominal value.
3. Because of following reasons, the final integrated system will be composed of glue code and application code except for COTS component.
 - Because of the black-box nature of COTS component, we are not able to know the size of the COTS component.
 - In case of that COTS will be upgraded in the future, COTS component is considered as a separate module.
4. Starting point of glue code development and COTS integration is based on the completion ratio of application and glue code development. For example, if the percentage of completed application reaches at X% of the total application requirements, the glue code development process is started. As the same way, if the percentage of developed glue code reaches at X% of the total glue code requirements, the integration process is started concurrently.
5. As mentioned earlier, many scenarios for starting points of the integration and glue code development will be presented in order to determine the most efficient starting point for the system integration with COTS component.
6. For simplicity, we assume that this simulation model does not allow feedbacks to the outside of the system boundary such as re-evaluation of the COTS products and feedback to the requirements definition for COTS and feedback to the COTS product selection process.

3. Model Development

3.1 Modeling Process

Since our simulation model is different from traditional application development model, the simulation process is specifically tailored to accommodate COTS product integration and it causes a set of assumptions and constraints quite different from traditional application development. The simulation model supports concurrent developing activities for glue code development and application development and integration between COTS component and application component.



[Figure 3-1 Static COTS simulation model]

Our first simulation model shown in Figure 3-1 was a static model of glue code development. We defined five parameters for COTS component. These were the only factors, which have an effect on the glue code development. Only one COTS component is simulated and integrated in the simulation model. But current model is dynamically modeled with glue code development and application development. They are concurrently developed and integrated. According to the completed rate of application, the production rate of glue code is determined.

Our first intention of this model when we start this project is to find dynamic effects of COTS component parameters. But we concluded that COTS component parameters do not have any dynamic aspects. So, our objective is changed to find dynamic effects between glue code development and integration process caused by the static COTS component parameters.

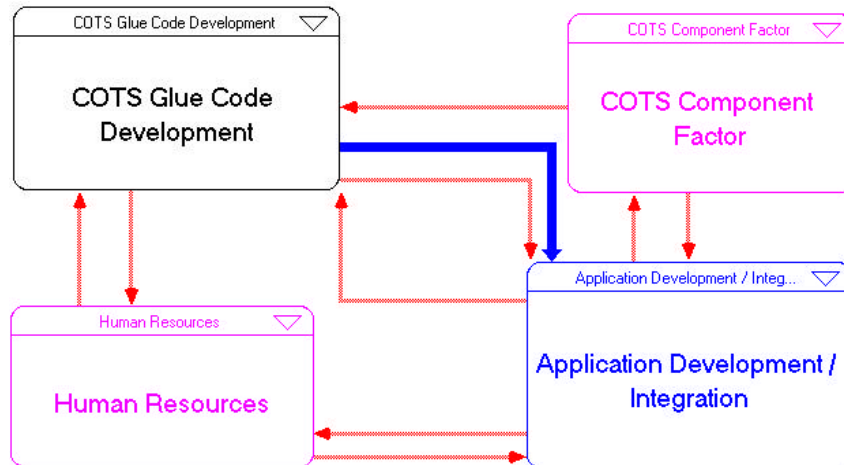
This simulation model consists of four sub-models, which are glue code development, application system development and integration, COTS component parameters, and human resource. We will discuss those sub-models in details in the next section.

3.2 Data Acquisition

Data for this simulation is acquired from COCOTS Data Collection Program from USC-Center for Software Engineering. For convenience, they are providing for download copies of the COCOTS research overview statement, the COCOTS data collection instrument, and the standard confidentiality agreement USC-CSE enters into with most of their data suppliers. Data points contain COTS related project regarding glue code development. Based on 13 project data among 20 projects from COCOTS database, we built this simulation model.

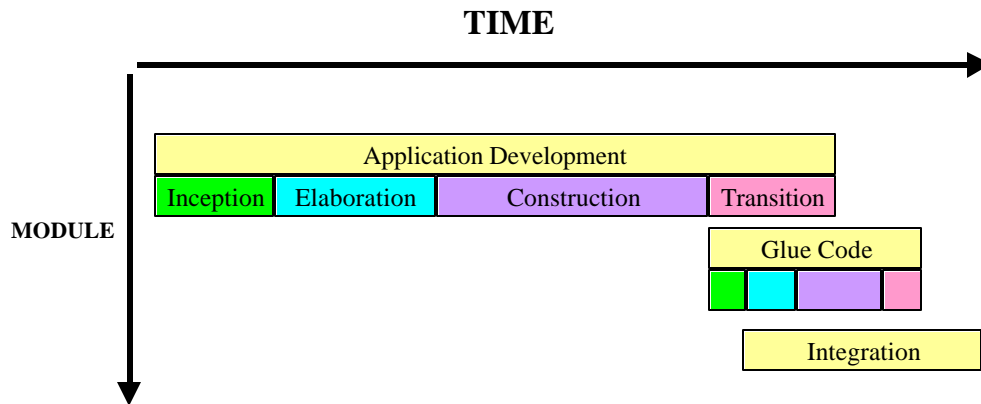
4. Model Description

As mentioned earlier, the simulation model for COTS integration and glue code development consists of four sub models. These four models are correlated to each other as followed Figure 4-1. The details of sub-models will be explained in the next sections.



[Figure 4-1 High-Level Design]

Figure 4-2 represents model timeframe. The model consists of three different development processes. Starting point of glue code development is depends on the application development and integration is depends on the glue code development.



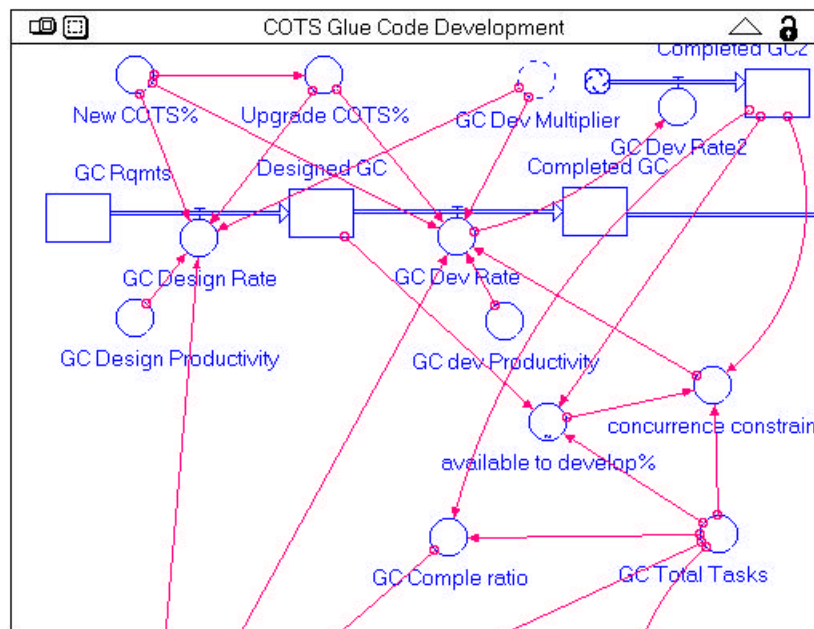
[Figure 4-2 Model Timeframe]

4.1 Glue Code Development Module

This module represents glue code development model from the system. The simulation model is represented in the Figure 4.3. The development process of glue code consists of three levels. The completed glue code is added to completed application code for integration. Integration process is represented in the application development sub-model. In this model, COTS components are divided into new COTS component and upgraded COTS component. In case of the upgraded COTS component, glue code is modified

from previous version. So, glue code development productivity is higher at integrating updated COTS component than integrating new COTS component. Results of sensitivity analysis of glue code development process based on the percentage of new and upgraded COTS is explained next section. Concurrency between design phase and implementation phase is explained.

***Caution:** When changing *GC_Rqmts* to other value, *GC_Total_Tasks* also has to be changed as the same value.



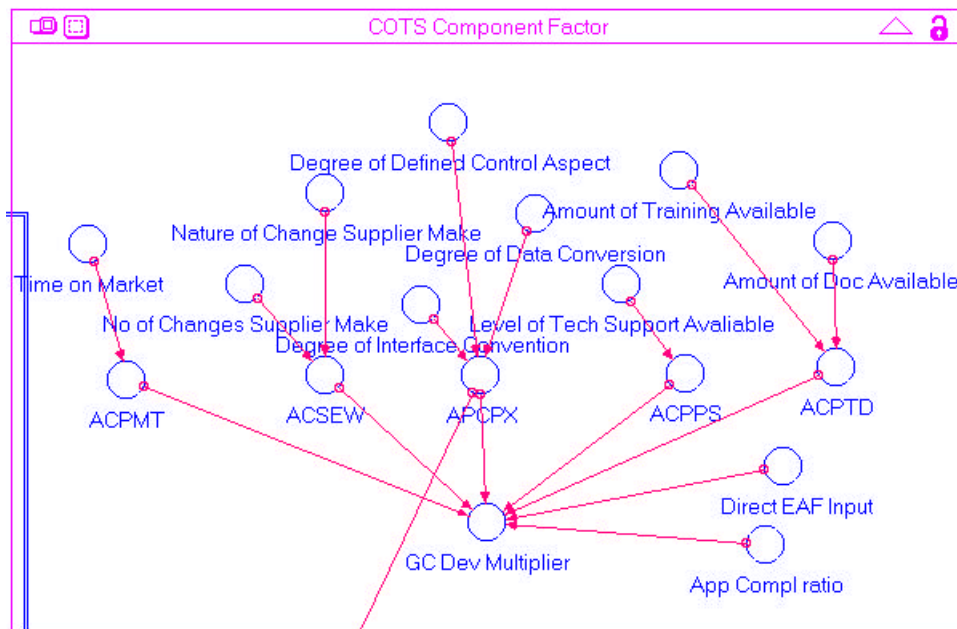
[Figure 4-3 COTS Glue Code Development]

- **Upgraded COTS%:** The percentage of upgraded COTS component. In this case, Glue code will be modified from previous version. So development rate is higher than integrating new COTS component
- **New COTS%:** The percentage of totally new COTS component
- **GC Rqmts:** Glue Code Requirements as SLOC
- **Completed_GC:** This level represents SLOC that have been implemented.
- **Completed_GC2:** This level represents SLOC that have been implemented. Difference from Completed_GC is that this value does not flow to the integration phase. It is for calculating cumulated completed glue code.
- **GC_Dev_Rate:** Glue code implementation rate per month. If there is no tasks that can be developed (i.e. constrained by process concurrence), then development rate = 0
- **GC_Dev_Rate2:** Same value with GC_Dev_Rate
- **Designed_GC:** This level represents SLOC that have been designed.
- **GC_Design_Rate:** Glue code design rate per month
- **GC_Dev_Rate:** If there is no tasks that can be developed (i.e. constrained by process concurrence), then development rate = 0.

- **concurrency_constraint:** The constraint flag is set based on whether there are tasks that can currently be developed per the concurrency relationship (1 = constrained)
- **GC_Comple_ratio:** Percentage of completed glue code. This parameter affects to the integration process and determines glue code staffing level.
- **GC_Design_Productivity:** The nominal glue code design productivity as SLOC/person-month.
- **GC_dev_Productivity:** The nominal glue code implementing productivity as SLOC/person-month.
- **GC_Total_Tasks:** Glue code tasks to be specified and developed.
- **available_to_develop%:** This concurrency relationship describes the percent of tasks that are available to be developed as a function of the tasks specified to-date.

4.2 COTS Component Factor Module

This module represents COTS component factor module from the system. GC_Dev_Multiplier is calculated by multiplying five COCOTS parameters, which are ACPMT, ACSEW, APCPX, ACPPS, ACPTD. App_Compl_ratio is used as an overhead to glue code development from application development. Especially, 'Direct EAF Input' can be used for determining EAF value directly without considering COTS parameter data and this feature can be used for verification of theoretical parameter values. .



[Figure 4-4 COTS Component Factor]

- **COTS Component Driver Specifications**

- 1) **ACPMT (COTS Product Maturity)** – This parameter represents COTS product maturity. The value of this parameter is estimated by time on market of the product.
- 2) **ACSEW (COTS Supplier Product Extension)**– COTS supplier’s willingness to change features of COTS component is specified. It is estimated by observing number of changes supplier make into the COTS component and the complexity of the change.
- 3) **APCPX (COTS Product Interface Complexity)**– It represents interface complexity of COTS product. If the interface of COTS component is complex, it is difficult to integrate COTS component into application system. The degree of the complexity is calculated by Table 4-1 [1].
- 4) **ACPPS (COTS Supplier Product Support)**– COTS supplier’s technical support is represented by this parameter. It contains technical supports for the integration team during the development, either directly from the component suppliers or through third parties.
- 5) **ACPTD (COTS Supplier Provided Training and Documentation)** – Provided training and documentation from COTS supplier is numerated. It is calculated by estimating the period of training and coverage of COTS product within documentation.

Complexity Elements	Very Low (point value = 1)	Low (point value = 2)	Nominal (point value = 3)	High (point value = 4)	Very High (point value = 5)	Corresponding Points
Interface Conventions (e.g., naming, relevant usage scenarios, service signature, service order)	N/A	Nearly all API conventions are clear and consistent.	Most API conventions are clear and consistent.	Few API conventions are clear and consistent.	API conventions are non-existent.	
Control Aspects (e.g., consistent and clear error handling/recovery)	N/A	Nearly all control aspects are well defined and consistently applied.	Most control aspects are well defined and consistently applied.	Few control aspects are well defined and consistently applied.	No control aspects are well defined and consistently applied.	
Data (e.g., conversion, number/range typing)	No data conversion required.	Little data conversion required and standard data types used.	Some data conversion required and standard data types used.	Significant data conversion required and/or use of non-standard data types.	Extensive data conversion required and/or use of non-standard data types.	

Total Point Score = _____

[Table 4-1 Interface Complexity Criteria]

Table 4-2 represents guidelines for determining model inputs for COTS component parameters. *COCOTS Data* represents definitions for each category from COCOTS. *Model Input* is calibrated value for iThink input parameters within the simulation.

Time on Market

	Very Low	Low	Nominal	High	Very High
Model Input	0	0.3	1	1.5	2
COCOTS Data	Pre-release	6 months	1 year	1.5 year	2 year

Number of changes suppliers make

	Very Low	Low	Nominal	High	Very High
Model Input	1	3	5	7	9
COCOTS Data	No changes	3 changes	5 changes	7 changes	9 changes

Nature of changes suppliers make

	Very Low	Low	Nominal	High	Very High
Model Input	0	0.5	1	1.5	2
COCOTS Data		Minor Changes			Major changes

Level of Tech Support Available

	Very Low	Low	Nominal	High	Very High
Model Input	1	2	3	4	5
COCOTS Data	unsupported	Telephone	Help desk	Trained support	consulting

Amount of Document Available

	Very Low	Low	Nominal	High	Very High
Model Input	0	¼	2/4	¾	1
COCOTS Data	No documents	¼ of needed	2/4 of needed	¾ of needed	All needed

Degree of Interface Convention

	Very Low	Low	Nominal	High	Very High
Model Input	0	1	2	3	4
COCOTS Data	Not applicable	All API consistent	Most API consistent	Few API consistent	API nonexistent

Degree of Defined Control Aspect

	Very Low	Low	Nominal	High	Very High
Model Input	0	2	3	4	5
COCOTS Data	Not applicable	All consistent	Most consistent	Few consistent	nonexistent

Degree of Data Conversion

	Very Low	Low	Nominal	High	Very High
Model Input	1	2	3	4	5
COCOTS Data	Not required	Little data conversion required	Some data conversion required	Significant data conversion required	extensive data conversion required

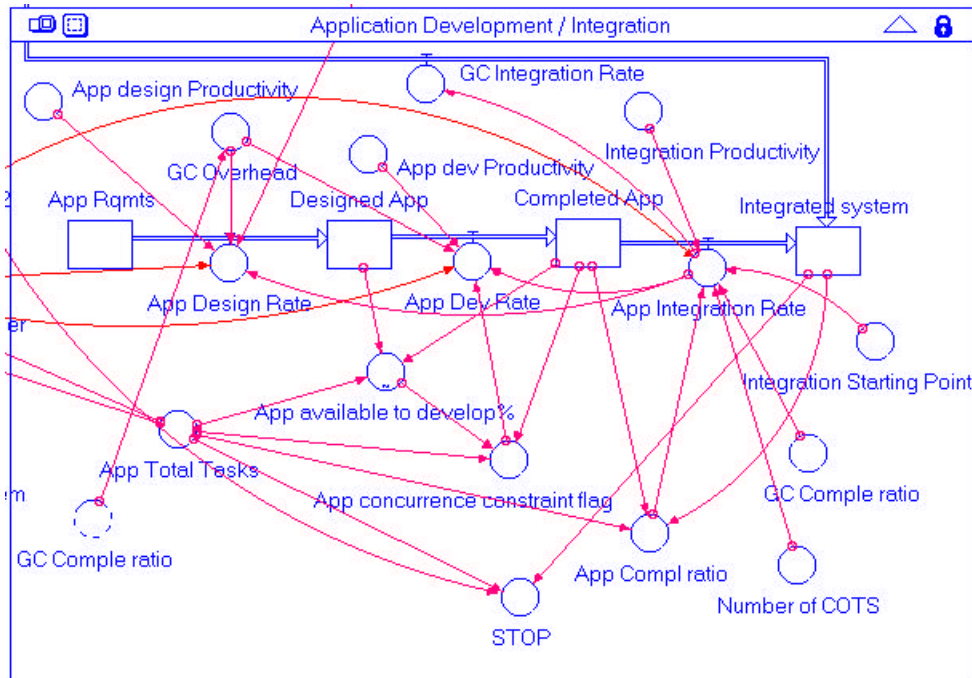
[Table 4-2 Model input Guidelines for COTS Parameters]

4.3 Application Development / Integration Module

This model represents application development and integration model from the system. Application development model consists of three levels. They are same with the glue code development. Completed application is flow to the integrated system with the completed glue code from previous section. When integrating COTS component, number of COTS components is an important factor to determine integration process. If number of COTS is higher, then integration process is slower. It is explained in section 2.2. Results from sensitivity analysis of integration process based on the number of COTS will be followed in next section.

One of most important parameters in this sub-model is *Integration_Starting_Point*. This parameter is for determining the stating point of integration process based on the glue code development process. For example, if this parameter is 0.5, then integration process is started when the glue code development process completed 50 % of glue code requirements. We will simulation this model with various different percentages in order to determine the most efficient starting point of the integration process. *GC_Overhead* represents glue code completion rate. If percentage of completed glue code is bigger, application development productivity is higher.

***Caution:** When changing *App_Rqmts* to other value, *App_Total_Tasks* also has to be changed as the same value.



[Figure 4-5 Application Development / Integration]

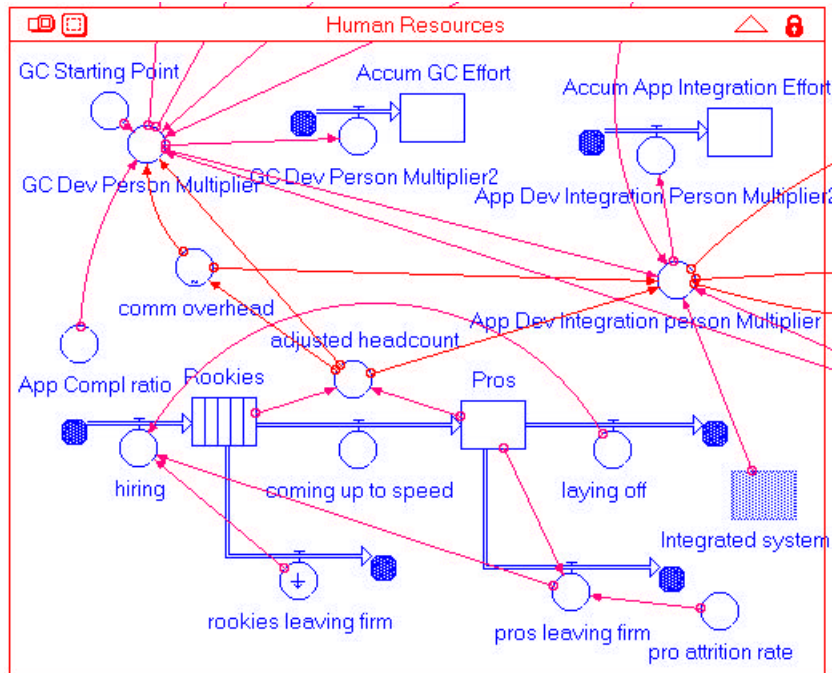
- **App_Rqmts:** This level represents the application requirements as SLOC left to be implemented.

- **Completed_App:** This level represents SLOC that have been implemented.
- **Designed_App:** This level represents SLOC that have been designed.
- **App_Design_Rate:** Application design rate per month.
- **App_Dev_Rate:** Application implementation rate per month.
- **App_Integration_Rate:** Application integration rate per month
- **Integrated_system:** This level represents SLOC that have been integrated. It consists of application code and glue code.
- **App_Compl_ratio:** Percentage of completed application. This parameter affects to the integration process.
- **App_concurrence_constraint_flag:** The constraint flag is set based on whether there are tasks that can currently be developed per the concurrence relationship (1 = constrained).
- **App_design_Productivity:** The nominal application design productivity as SLOC/person-month.
- **App_dev_Productivity:** The nominal application implementing productivity as SLOC/person-month.
- **App_Total_Tasks:** Application tasks to be specified and developed.
- **Integration_Productivity:** The nominal application integration productivity as SLOC/person-month.
- **Number_of_COTS:** Number of COTS component to be integrated.
- **App_available_to_develop%:** This concurrence relationship describes the percent of tasks that are available to be developed as a function of the tasks specified to-date.
- **GC_Overhead:** This value is based on the glue code completion rate
- **Integration_Starting_point:** Integration process is started when X % of glue code is completed.

4.4 Human Resources

This sub-model represents human resources from the system. This structure builds on the two-tier infrastructure with the category of rookie and pros. A conveyor is used to represent the Rookies in the organization. Rookie attrition is used to depict leakage flow. From the Figure 4-6, staffing level for glue code development and application development is separated as ‘GC Dev Person Multiplier’ and ‘App Dev Integration person Multiplier’. The staffing level is determined by requirements levels of glue code and application development.

The most important parameter of this sub-model is ‘GC Starting Point’. This parameter is for determining the starting point of glue code development based on the application development. For example, if this parameter is 0.5, then glue code development is started when the application development process completed 50 % of application requirements. We will simulation this model with various different percentages in order to determine the most efficient starting point of the glue code development.



[Figure 4-6 Human Resources]

- **App_Dev_Integration_person_Multiplier:** This parameter represents number of personnel available for application development and integration. Communication overhead is calculated here. If glue code development is finished, then personnel from glue code development is added to here.
- **App_Dev_Integration_person_Multiplier2:** The value of this parameter is same to App_Dev_Integration_person_Multiplier
- **Accum_App_Integration_Effort:** Accumulated efforts for application development
- **GC_Dev_Person_Multiplier:** This parameter represents number of personnel available for glue code development. Communication overhead is calculated here. If glue code development is finished, then every personnel is used for application development and integration.
- **GC_Dev_Person_Multiplier2:** The value of this parameter is same to GC_Dev_Person_Multiplier
- **Accum_GC_Effort:** Accumulated efforts for glue code development
- **GC_Starting_Point:** Glue code development process is started when X % of application is completed.
- **comm_overhead:** Percent of time spent communicating with other team members as a function of team size.
- **Pros:** Fully productive employees.
- **pro_attrition_rate:** 10% of all Pros leave each month.
- **coming_up_to_speed:** It takes 5 months before Rookies come up to speed as Pros.
- **laying_off:** Layoffs happen at one moment in time. The number you choose will be laid off and then the Slider will reset until you choose to layoff again.

- **pros_leaving_firm:** The number of Pros leaving the firm each month is a percentage of the total number of Pros at the firm. For the course of the simulation this percentage is assumed to be 10%.
- **Rookies:** For the first 5 months at the firm, all new hires are thought of as Rookies. After 5 months they have either left or graduated to become Pros. During this 5 month training period, a Rookie is thought to be able to produce about 50% of the work of a Pro.
- **rookies_leaving_firm:** The number of Rookies who leave each month. It is assumed that you wish to replace each one that leaves.
- **adjusted_headcount:** The adjusted headcount accounts for the fact that Rookies are only half as productive as Pros.

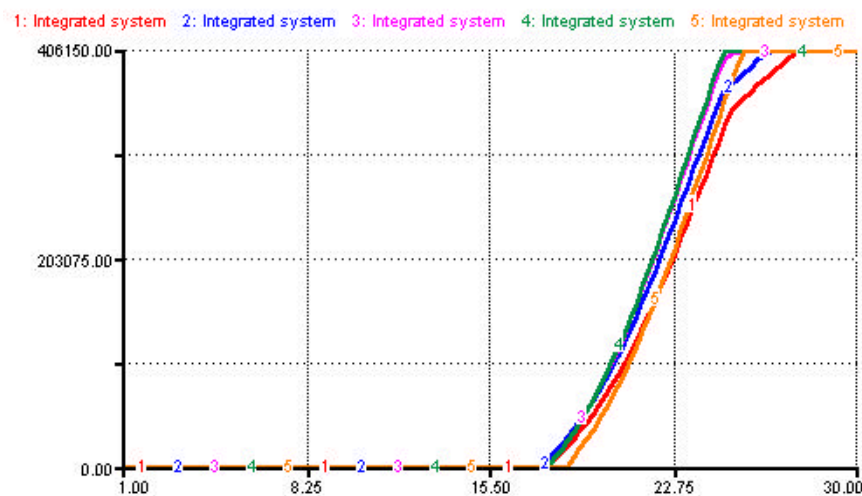
5. Model Verification and Validation

When deciding how to validate this simulation model, I used two approaches. The first approach I use for validating this simulation model is to test with actual data from projects that have already completed and format the data such that it could be entered into this simulation model. The second approach I use for validating this simulation model is to test important parameters by sensitivity test.

5.1 Sensitivity Analysis

(1). Sensitivity Analysis of ‘GC_Starting_Point’

As mentioned earlier, this parameter determines glue code development starting point. To start glue code development, parts of application system should be implemented because glue code is developed based on the application system. We determine glue code development starting point by the percentage of completed application system. Figure 5-1 represents integrated system by various starting point.

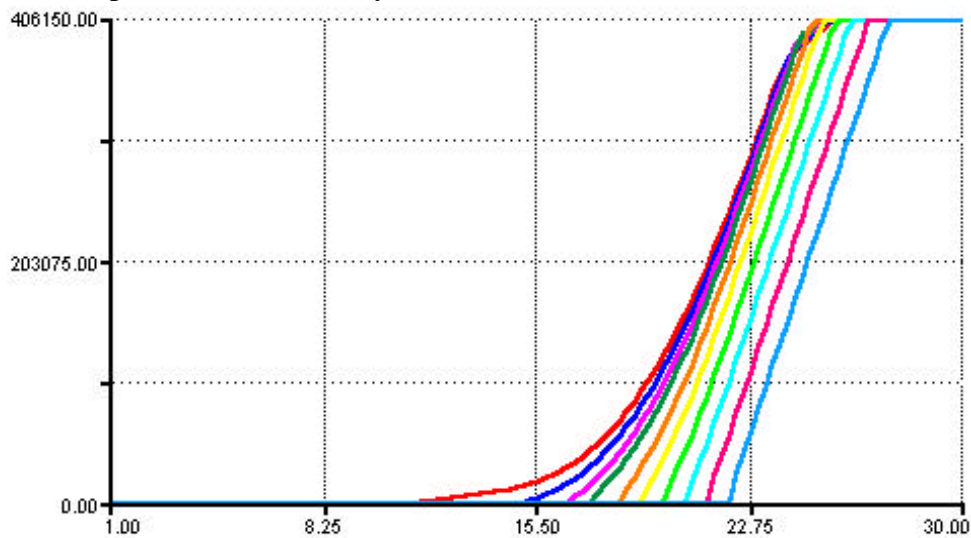


[Figure 5-1 Integrated System when Starting point of glue code development = 60%, 70%, 80%, 90% 100% of application development completion]

According to the Figure 5-1, to start glue code development when 80-90% of application is completed has the biggest schedule reduce.

(2). Sensitivity Analysis of ‘Integration_Starting_Point’

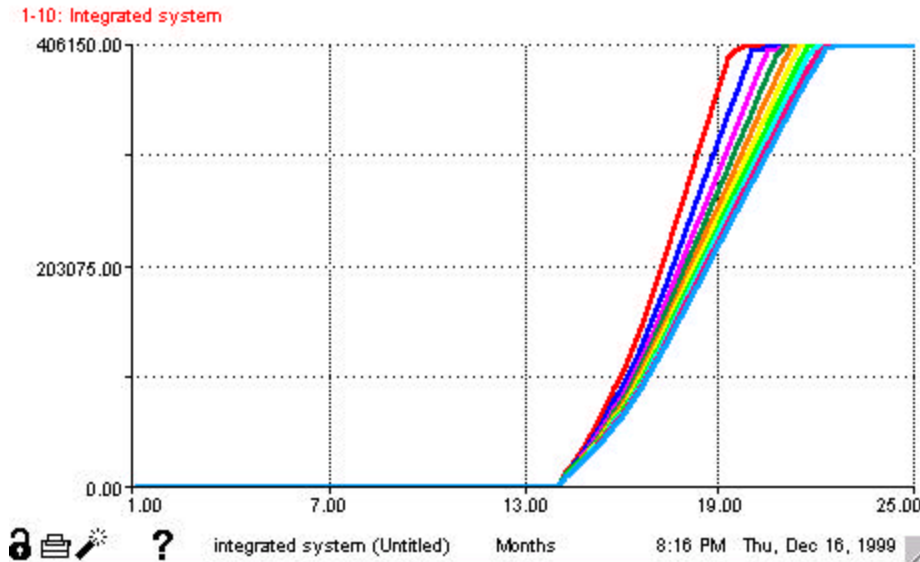
This test is for determining starting point of integration process. As mentioned earlier, integration process is started based on the completed glue code development. According to Figure 5-2, if integration process starts after 60% of glue code is completed or above, the schedule delay is higher. But if integration process is start before that point, there is very small schedule delay. Specifically, if the integration process start when 30-40% of glue code developed, the schedule delay is the smallest.



[Figure 5-2 Integrated System when Starting point of integration = 0% - 100% of glue code development completion]

(3). Sensitivity Analysis of ‘Number of COTS’

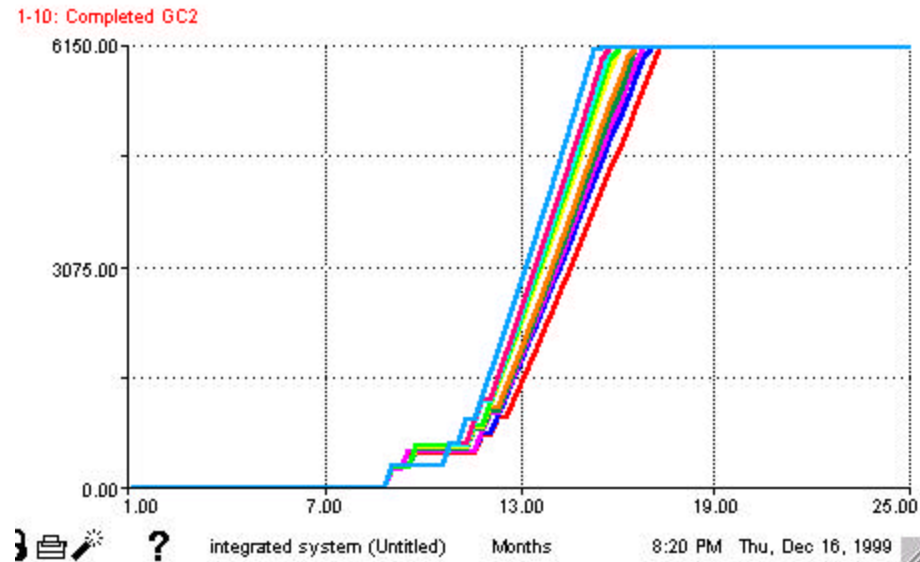
Although glue code requirements are same, if numbers of COTS component are different, the integration productivity is different. This sensitivity analysis is based on various total COTS component number. According to the following graph, integration process is delayed based on the number of COTS. But every other process has the same patterns and values.



**[Figure 5-3 Integrated System at
Number of COTS = 1, 5.33, 9.67, 14, 18.3, 22.7, 27, 31.3, 35.7, 40]**

(4). Sensitivity Analysis of ‘New_COTS_%’

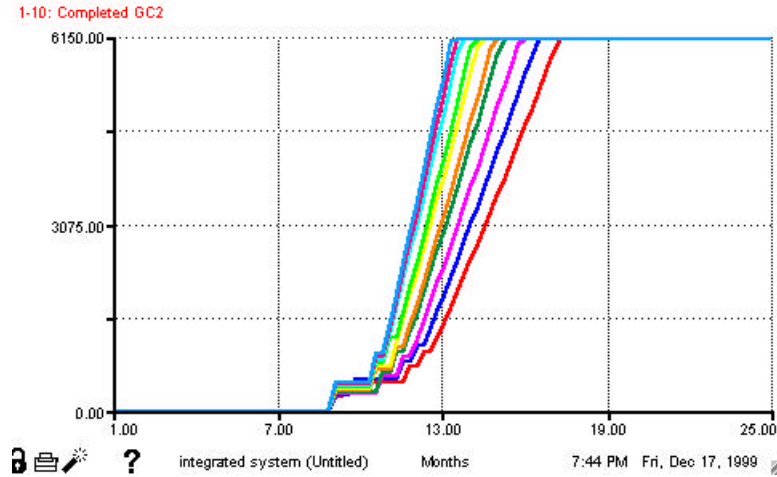
When COTS component is integrated, it can be a new COTS component or upgraded one from previous COTS component. If the COTS is the upgraded one, developers do not need to develop totally new glue code. They can upgrade glue code from previous version. I had sensitivity analysis based on the percentage of new COTS component. According to the following graph, completed glue code is delayed based on the percentage of new COTS. But every other process has the same patterns and values.



**[Figure 5-4 Completed Glue Code at
New_COTS % = 1, 0.889, 0.778, 0.667, 0.556, 0.444, 0.333, 0.222, 0.111, 0]**

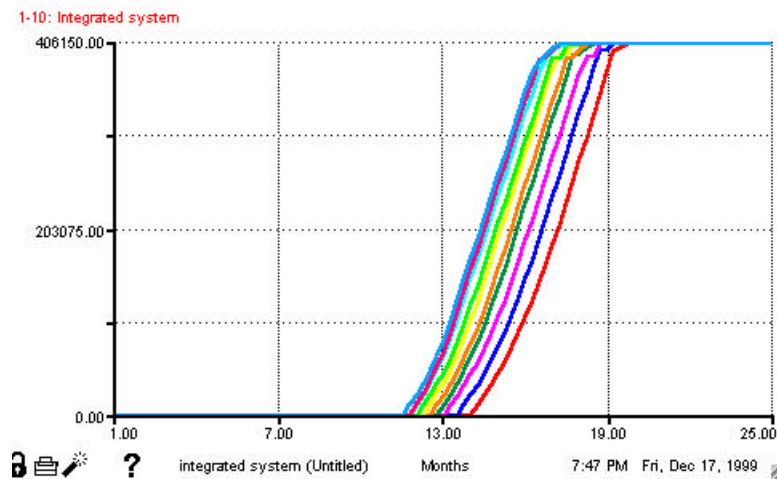
(5). Sensitivity Analysis of COTS Component Factor-ACPMT

This analysis represents schedule of completed glue code based on the various ACPMT values. ACPMT parameter is determined by time on market of the integrated COTS component. I tested with the value from 1 to 2 years of market time.



[Figure 5-5 Completed Glue Code when Time on Market is 1, 1.11, 1.22, 1.33, 1.44, 1.56, 1.67, 1.78, 1.89, 2 years]

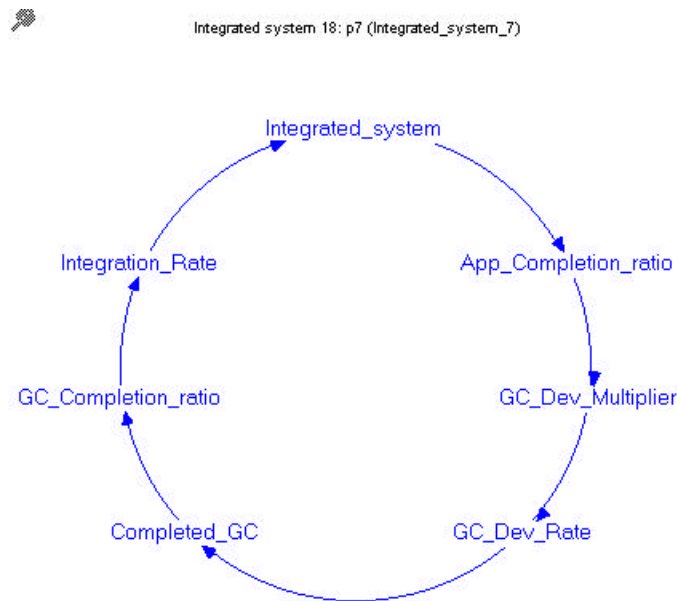
Figure 5-6 represents schedule of integrated system based on the various time on market values. Although COTS component drivers are calculating efforts of glue code development, according to the following graph, the drivers also have effects on the COTS integration process.



[Figure 5-6 Integrated System when Time on Market is 1, 1.11, 1.22, 1.33, 1.44, 1.56, 1.67, 1.78, 1.89, 2 years]

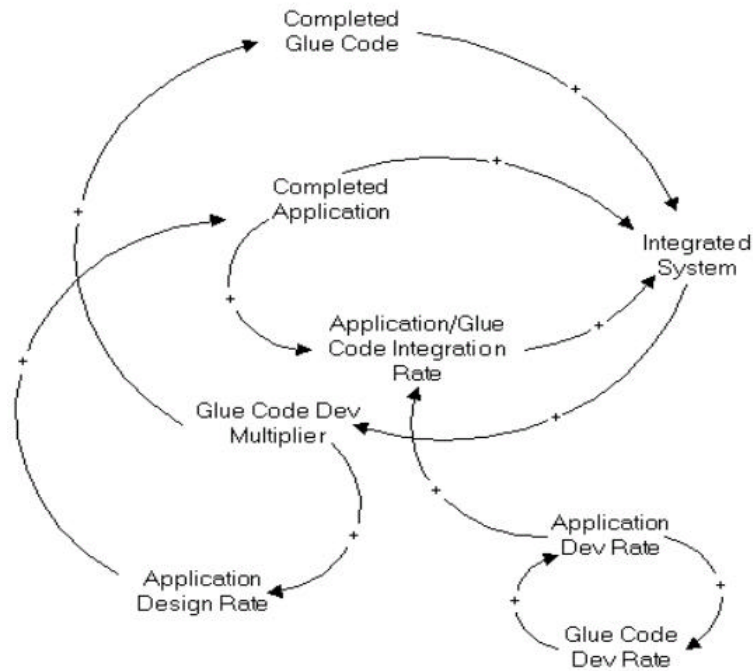
5.2 Causal Loop Diagrams

Although it is not always necessary to use a causal loop diagrams for systems characterized by a high degree of correlation, these diagrams are helpful for obtaining a common understanding of the system components and their relationships before advancing to the formality of flow diagrams. The less a system is structured by physical components and the more it depends on human interaction, then the higher the correlation and the more difficult it is to describe the structure of the system. In these kinds of systems, causal loop diagrams can be very helpful in clarifying the nature of the system [2].



[Figure 5-7 Feedback Loop(1)]

Figure 5-7 represents a feedback diagram, which is automatically generated from iThink. If *Integrated_system* is increased, *App_Completion_ratio* is also increased because *App_Completion_ratio* is calculated by completed application and completed integration. *App_Completion_ratio* is one of input data for *GC_Dev_Multiplier*, so if *App_Completion_ratio* is increased, *GC_Dev_Multiplier* is also increased and it causes to increase *GC_Dev_Rate*. And Higher *GC_Dev_Rate* makes higher *Completed_GC*. It also increases *GC_Completion_Ratio*. If Completed glue code is enough, *Integration_Rate* is also increased and it causes to increase *Integrated_system*. This loop is continued until the addition of completed glue code and completed application is same to the integrated system.



[Figure 5-8 Feedback Loop(1)]

Figure 5-8 represents a feedback diagram of our simulation model. This feedback loop consists of reinforcing (+) loops and counteracting (-) loops are hidden for simplicity.

Completed Glue Code increases *Integrated System* and *Integrated System* increase *Glue Code Dev Multiplier* according to the Figure 5-7. APCPX, one of *Glue Code Dev Multipliers* are affecting App Design Rate, so *Glue Code Dev Multipliers* can be said to increase *Application Design Rate*. And increased *Application Design Rate* also increases *Completed Application*. Increased *Completed Application* also increases *Integration Rate*. *Application Dev Rate* and *Glue Code Dev Rate* are promoting each other.

5.3 Scenario Test

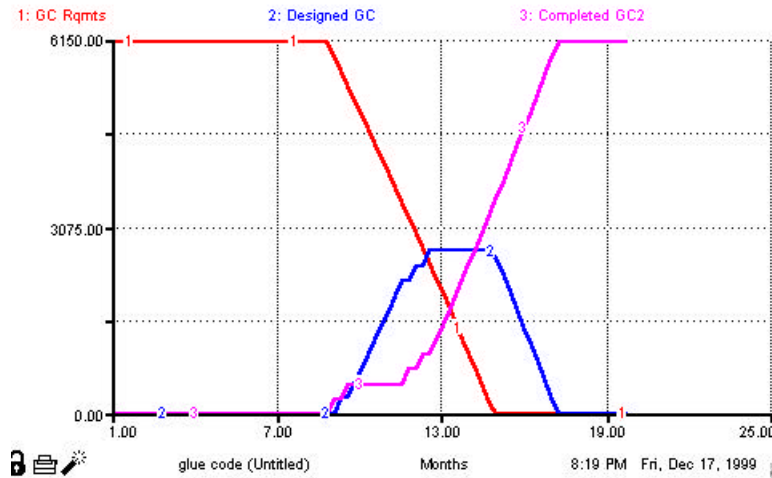
For this test, I generated three scenario cases. Basically I used COCOTS data explained in the previous chapter and I systematically varied the time-dependent constants and variables. More scenarios can be generated if required.

	Scenario #1	Scenario #2	Scenario #3
Glue Code Development			
GC Rqmts	6150 SLOC	25000 SLOC	10000 SLOC
New COTS %	100 %	70%	30 %
Upgraded COTS %	0 %	30%	70 %
GC Design Productivity	8 * 25 SLOC / PM	8 * 25 SLOC / PM	8 * 25 SLOC / PM
GC Dev Productivity	8 * 25 SLOC / PM	8 * 25 SLOC / PM	8 * 25 SLOC / PM
GC Total Tasks	6150 SLOC	25000 SLOC	10000 SLOC
COTS Component Factor			
Time on Market	1 (year)	0.3 (year)	3 (year)
Nature of Change Supplier Make	1 (nominal)	2 (high)	0.5 (low)
Number of Changes Supplier Make	5 (nominal)	7 (high)	3 (low)
Degree of Interface Convention	3 (nominal)	4 (high)	2 (low)
Degree of Defined Control Aspect	3 (nominal)	2 (low)	2 (low)
Degree of Data Conversion	3 (nominal)	2 (low)	2 (low)
Level of Tech Support Available	3 (nominal)	3 (normal)	3 (normal)
Amount of Training Available	½ (Half of needed)	¾ (¾ of needed)	1 (All of needed)
Amount of Doc Available	½ (Half of needed)	¼ (¼ of needed)	½ (Half of needed)
Application Dev / Integration			
App Rqmts	400000 SLOC	280000 SLOC	737000 SLOC
App Design Productivity	8 * 25 SLOC	8 * 25 SLOC	8 * 25 SLOC
App Dev Productivity	6 * 25 SLOC	6 * 25 SLOC	6 * 25 SLOC
Integration Productivity	20 * 25 SLOC	20 * 25 SLOC	20 * 25 SLOC
Number of COTS Component	1	7	4
App Total Tasks	40000 SLOC	280000 SLOC	737000 SLOC
Human Resources			
Rookies	10,10,10	10,10,10	10,10,10
Pros	30	30	30
Coming up to speed	5	5	5
Laying off	0	0	0
Pro Attrition Rate	0.1	0.1	0.1
Rookies leaving firm	0	0	0

[Table 5-1 Simulation Scenarios]

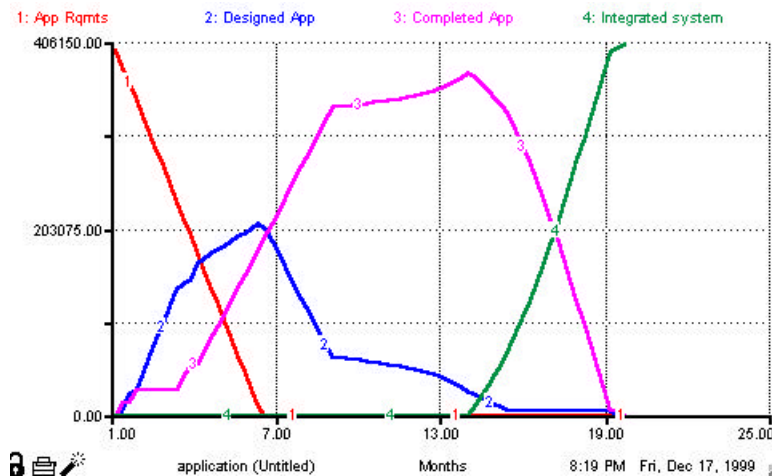
5.3.1 Scenario #1 Test Result

Data for this simulation is adapted from DSR project from COCOTS data. According to the following graphs, glue code development is started when parts of application is started to complete. And there is concurrency in design and implementation phases. When parts of glue code is started to complete, the integration process start. The integration graph has S-shaped graph. As for personal allocation, staffing pattern is affected by each development phases.

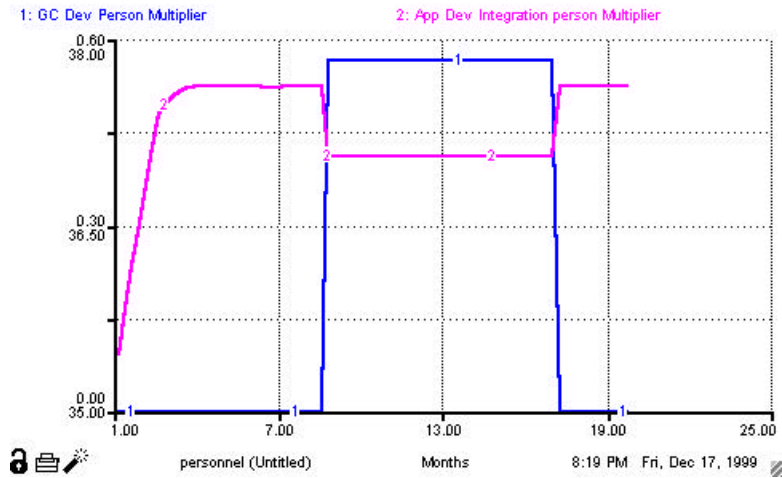


[Figure 5-9 Glue Code Development]

According to Figure 5-9 and 5-10, integration is started although glue code development is not finished. So integration can be finished just after glue code development is finished. Application development rate is reduced when the glue code development is processed because staff moves to the glue code development.

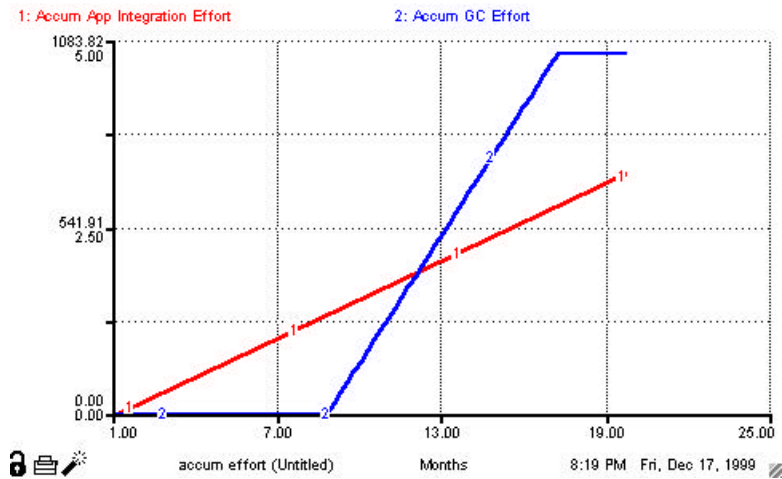


[Figure 5-10 Application Development and Integration]



[Figure 5-11 Staffing Pattern]

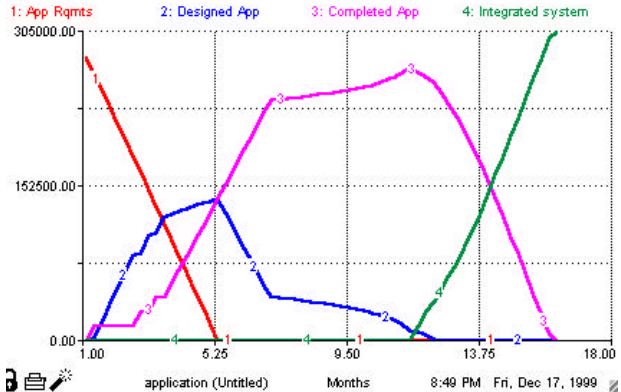
Figure 5-11 represents staffing pattern for glue code development and application development. If glue code development is started, staff moves from application development to glue code development. If glue code development is finished, staff goes back to the application development for integration process.



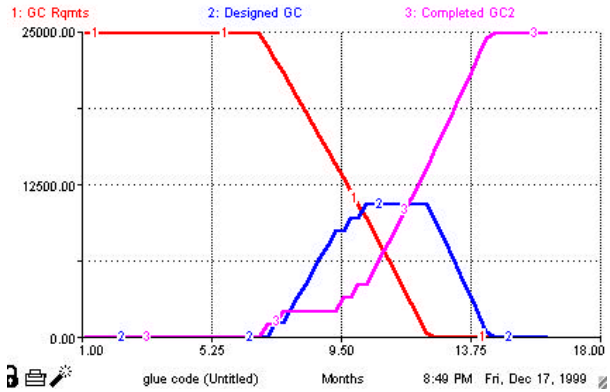
[Figure 5-12 Accumulated Effort]

Figure 5-12 represents accumulated efforts for glue code development and application development and integration. When the glue code is completed, staffing is moved to the application development process, so the glue code development effort does not increase. Effort difference between glue code development and application development is because of the initial requirements difference. Application development efforts contain integration efforts.

5.3.2 Scenario #2 Test Result

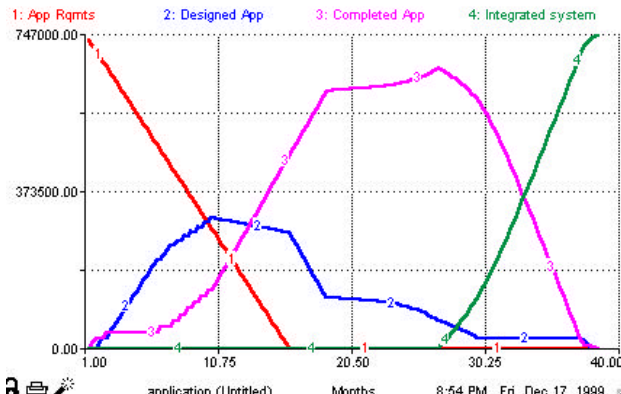


[Figure 5-13 Application Development and Integration]

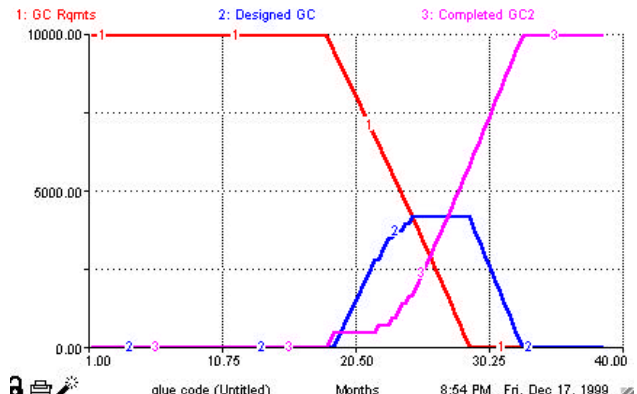


[Figure 5-14 Glue Code Development]

5.3.3 Scenario #3 Test Result



[Figure 5-15 Application Development and Integration]

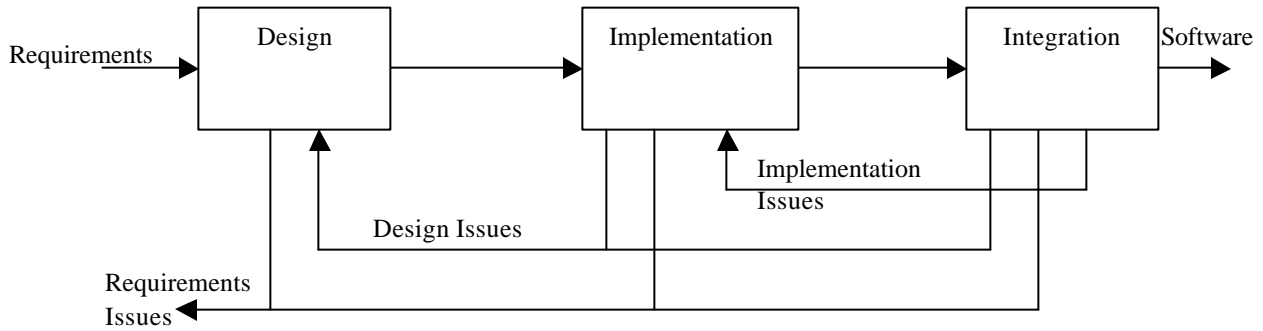


[Figure 5-16 Glue Code Development]

Figure 5-13 ~ Figure 5-16 represents test results of scenario #2 and #3 data from Table 5-1. According to the figures, there is a big schedule difference between the two cases. It is caused by the requirements difference from the two cases. Graphs of two scenarios are similar because we use the same starting points, which are determined in the sensitivity tests.

6. Model Application and Transition

The integration module from this glue code development and COTS integration simulation model can be applied to other integration process such as system integration containing hardware and software. Figure 6-1 and Table 6-1 represent applicable algorithm for integration process.



[Figure 6-1 Integration Process]

Cell	Design	Implementation	Integration
Entry	Approved requirements, changes, and development plan	Inspected and approved design and changes	Inspected and approved code and changes
Exit	Inspected and approved design and changes	Inspected and approved code and changes	Inspected, tested, and integrated software
Feedback/In	Design issues	Implementation issues	
Feedback/Out	Requirements issues	Requirements and design issues	Requirements, design and implementation issues
Task	Design	Implementation, inspection, and unit test	Integration, system test, system management
Measures	Resources, Product: Changes, Errors, Design, Document pages	Resources, Product: Changes, Errors, Code, Document pages	Resources, Product: Changes, Errors, Code, Document pages, Test suite

[Table 6-1 Integration Process]

7. Conclusions and Recommendations

One of the most interesting analyses of this simulation is to determine starting point of glue code development and integration. According to the sensitive analysis of previous section, glue code development should start when 80% of application (custom-based) components is completed and integration should start when 30% of glue code is completed. We extracts following conclusion regarding starting point of development.

Glue code development has to start in the ending of the application development and integration process has to start in the beginning of the glue code development.

This COTS integration simulation model was developed to prove the necessity of considering system dynamic relationships rather than rely on statistical correlation when developing glue code and integrating COTS component into system. Software planning and estimating tools such as COCOMO are statistical models, as opposed to system dynamics which incorporates nonlinear equations and feedback loops that describe causal influences on a project. Using these system dynamic relationships provides a realistic approach to modeling a project and facilitating an understanding of the various components.

The ability to interact with COTS characteristic parameters during the simulation helps to model true behavior of glue code and integration during a project. A project planner has the ability to perform many "What-If" scenarios quickly and study the trade-offs to alternative approaches to scheduling and staffing decisions. Before investing time and money and implementing a decision that may prove risky, a project planner can simulate the decision and analyze the results.

Although software COTS products are attempting to simulate the "plug-and-play" capability of the hardware world, in today's reality, software COTS products seldom plug into anything easily. Most products require some amount of adaptation to work harmoniously with the other commercial or custom components in the system. The typical solution is to adapt each COTS product through the use of "wrappers," "bridges," or other "glueware." It is important to note that adaptation does *not* imply modification of the COTS product [4]. However, adaptation can be a complex activity that requires technical expertise at the detailed system and specific COTS component levels.

This paper has illustrated how a system dynamics model and simulation of the COTS integration and glue code development can be used to assist in predicting when good software productivity levels will be achieved. Our model can be calibrated to a project environment taking into account integration productivity. Our sample runs have illustrated the importance of deciding starting point of integration and correlation of glue code and application development.

Integration with COTS software products requires adjustment and accommodations to the development approach and development process. Preparations must be made to start prototyping activities and integration activities immediately to use COTS product advantages and accelerate development. Additional resources must be allocated for late in the development cycle to provide maintenance and support to the software developers.

As future works, this simulation model can be enhanced for other COCOTS Sub-model such as Assessment, Tailoring, and Volatility. They include feedback to the COTS product selection process and feedback to the requirements definition for COTS products and re-evaluation of the COTS products.

8. References

- [1] <http://sunset.usc.edu/COCOTS/cocots.html>
- [2] *iThink Manual*, High Performance Systems, Inc., Hanover, NH.
- [3] Brownsword, L., Carney D., Oberndorf, T., "The Opportunities and Complexities of Applying COTS Components," *SEI interactive*, Vol 2, Issue 3, September 1999.
- [4] Fox, G., Marcom, S., Lantner, K., "A Software Development Process for COTS-Based Information System Infrastructure," *CROSSTALK*, March 1998.
- [5] Wallnau, K., Carney, D., Pollak, B., "How COTS Software Affects the Design of COTS-Intensive Systems," *SEI interactive*, Vol 1, Issue 1, June 1998.
- [6] Abts, C., Clark, B., "Project Level COTS Integration Experience Survey," USC-Center for Software Engineering, November 1999.
- [7] Abts, C., Boehm, B., "USC-CSE COTS Integration Cost Calculator V2.0 User Guide," USC-Center for Software Engineering, September 1997.
- [8] Breierova, L., Choudhari, M., "An Introduction to Sensitivity Analysis," MIT System Dynamics in Education Project, September 1996.
- [9] Carney, D., "Assembling Large Systems from COTS Components: Opportunities, Cautions, and Complexities," Software Engineering Institute, June 1997.
- [10] Sha, L., Goodenough, J., Pollak, B., "Simple Architecture: Meeting the Challenges of Using COTS in High-Reliability Systems", *CROSSTALK*, April 1998.
- [11] Ford, D., Sterman, J., "Dynamic Modeling of Product Development Processes," Massachusetts Institute of Technology, Cambridge, MA, January 1997.
- [12] Madachy, R., Boehm, B., *Software Process Dynamics*, IEEE Computer Society Press, 2000
- [13] Madachy, R., "CS599 Software Process Modeling Course Notes", USC-Center for Software Engineering, November 1999.