

Architecture Development Process Dynamics in MBASE

Cyrus Fakharzadeh

Center for Software Engineering

University of Southern California

Nikunj Mehta

Center for Software Engineering

University of Southern California

Architecture Development Process Dynamics in MBASE

Introduction

Many models of software processes focus on issues concerning specific aspects of a software development life cycle. In this report, we describe a process model for architecture development in the Model-Based Architecting and Software Engineering (MBASE) approach. This approach involves creation of four kinds of models i.e. product, process, property and success models, and their integration in an ongoing fashion. The architecture of a system forms the blue print for building the system and consists of the most important abstractions that address global concerns. Architecture is centric to many modern life cycles such as Rational Unified Software Development Process (USDP) and MBASE.

The MBASE approach comprises four distinct phases, each delimited by a milestone or an anchor point. Early phases of MBASE consist of identifying the requirements for the system and defining the system architecture. During each phase the architecture of a system is successively refined to cover a larger system scope. These anchor points involve consensus building among the system stakeholders for commitment to move forward.

Our model studies the dynamics of architecting processes in MBASE during its early phases i.e. Inception and Elaboration. The model also studies the impact of RAD factors such as collaboration and prototyping on the process of architecting. The models described here show that initial completion rates for the requirements identification and architecture development activities significantly impact the number of approved items. Prototyping factors such as IKIWISI and collaboration also significantly affect the rates of completion and approval. The model also describes a declining curve for staffing analysts and a linear growth for architecting and design personnel. The model behavior is similar to Rational USDP in terms of the effort distribution curves of the process.

The purpose of this study is to understand the dynamics of architecture development processes. This understanding will help for better planning of future projects. The model would be useful for the designers of CS 577 and similar courses at USC that use the MBASE approach for developing software. The model has been calibrated for the CS 577 projects using effort data reported by the student team members. It is possible to extend the calibration to non-CS 577 projects by appropriately calibrating the model for resource factors such as productivity and completion durations of various tasks.

Background

Software process modeling involves studying various aspects of a software development life cycle. These life cycle models can be broadly categorized as sequential and concurrent. Various models of sequential life cycles have been created that study various issues such as hiring policies, effects of inspection and effects of staff learning. Modern software development life cycles have focused on concurrent activities and involve parallel execution of various software development tasks. The Spiral model [Boehm 81] of software development is an example of such a life cycle and involves successive refinements of the software models through incremental growth. MBASE (Model-Based Architecting and Software Engineering) is a hybrid life cycle approach which involves four phases each with a possibly different growth mode i.e. sequential or evolutionary.

The MBASE approach involves creation and integration of product, process, property and success models. MBASE is also an architecture centric approach and the architecture is constantly integrated with other models as describe before. The architecture of a system is a blue print that identifies the most important abstractions of the system that address global system concerns. An architecture proves to be a starting point for the process models and supports the required property models.

The process of architecting is still not a very well understood one and involves varying degrees of method, theft and intuition. However, it is possible to model the concurrence relations among requirements and architecture activities. The process involves constant integration of the various models and requires that QA and coordination with other activities be performed on an ongoing rather than discrete basis.

The Product development model by Ford and Sterman provides a model for concurrent product development that considers the effect of inter-phase and intra-phase concurrency. Appendix D shows the model we have produced. Appendix E shows the source code for our model.

The Architecture Development Process Model

Our model simulates the two major activities of the front end of a software development project namely requirements elicitation and architecture design. It also models the prototyping tasks in the development process as a supporting activity. The two activities are modeled separately in the model and a generic process structure is chosen to describe the internal dynamics of each activity with customizations performed to accommodate each activity's characteristics. The resource allocation in a concurrent model cannot be described by a static relationship with progress. Instead the required resources are often dictated by resource availability. Once hired, these resources are dynamically allocated to various tasks based on the backlog of tasks of that kind. The performance of this system model is measured in effort levels and the number of items produced. The three sub systems of this model are process structure, resources and performance. Top level subsystem interactions are described in the Figure 1. The flow of products between project activities is described in the project network diagram in Figure 2.

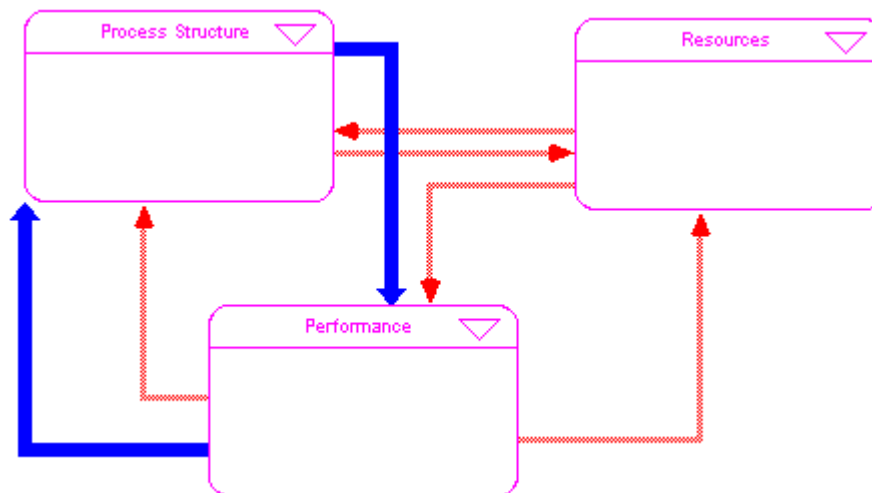


Figure 1 Subsystem interactions

Links between activities of the project are distinguished as carrying products of an activity or returning errors from the following activity. The inter-activity interactions arise from the following:

- Requirements activity produces requirement descriptions that are available to the architecture activity
- Architecture activity produces artifacts that are used in downstream activities but these downstream activities are not modeled.
- Architecture design reveals errors in the requirements definition which cause rework in the previous phase
- Prototyping activity is driven by both requirements and architecture activities and serves as a fast track route for discovering the details that would otherwise require more effort and time.

- Once prototyping is performed, it uncovers information that is required to better describe and understand the products of each of the two principal activities. Prototyping is a supporting activity and does not by itself consume or produce any artifacts.

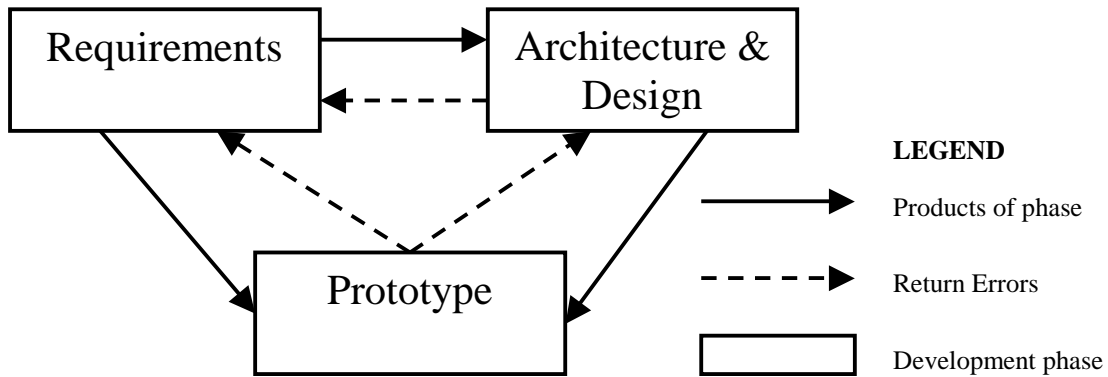


Figure 2 Activity Network Diagram

Reference behavior

The table below shows the average hours spent in each activity for each of the corresponding activities in the LCO, LCA, and RLCA phases.

Table 1. CSCI 577 data for effort distribution and completions

	LCO	%	LCA	%	RLCA	%	Total LCA	%	Overall	%	Adjusted
Management	230.8	40%	235.3	41%	112.2	19%	347.525	60%	578.33	38%	
Environment	110.1	60%	17.76	10%	54.6	30%	72.36	40%	182.44	12%	
Requirements	173.9	53%	103.3	32%	48.2	15%	151.505	47%	325.39	22%	50%
Architecture & Design	101	45%	63.7	28%	61.4	27%	125.1	55%	226.07	15%	34%
Implementation	35.52	36%	30.13	30%	33.3	34%	63.43	64%	98.95	7%	16%
Assessment	34.76	35%	40.61	40%	25	25%	65.61	65%	100.37	7%	
Total	686	45%	490.8	32%	334.7	22%	825.53	55%	1511.6	100%	100%

The following assumptions of the classification of activities and examples of each are given below.

1. Management: Client Interaction, Team meetings, E-mail, Telephone and Other Interactions; Planning and Control as well as Product Review; Transition Planning; Training; User Documentation; Customer Deliverables.
2. Environment: Tools Training, Quality Assurance, and other types of tasks not defined
3. Requirements: Updates to the Operational Concept Description or System and Software Requirements Definition;

4. Architecture & Design: Design Creation and Design Development; COTS Assessment, Tailoring and Integration
5. Implementation: Prototyping; Code Generation and Code Development
6. Assessment: Updates to the Feasibility Rationale; Unit Testing and Integration Testing; Test Specification

The columns for percentages (%) represent the percentages spent in that corresponding activity for the given activity for a given phase(LCO, LCA, or RLCA). The Adjusted column represents the adjusted percentages given that our model only addresses Requirements, Architecture & Design as well as Implementation. The adjusted value for Requirements were computed using the formula below.

$$(Overall[Requirements] + 0.5*Overall[Assessment])$$

$$Overall[Requirements] + Overall[Architecture \& Design] + Overall[Implementation] + Overall[Assessment]$$

The adjusted value for Architecture & Design was computed using the formula below.

$$(Overall[Architecture \& Design] + 0.3*Overall[Assessment])$$

$$Overall[Requirements] + Overall[Architecture \& Design] + Overall[Implementation] + Overall[Assessment]$$

The adjusted value for Implementation was computed using the formula below.

$$(Overall[Implementation] + 0.2*Overall[Assessment])$$

$$Overall[Requirements] + Overall[Architecture \& Design] + Overall[Implementation] + Overall[Assessment]$$

MBASE and USDP profiles

The model is based on the assumptions that the projects are completed in a fixed time frame and that schedule is an independent variable. This assumption is necessary so that the model can be calibrated against a set of projects from the CSCI 577 courses. However, the assumption does not prevent calibration to another project that has other preset schedule conditions.

Size parameters used in the model are relative to the overall project size. Thus, it is possible to calibrate the model with different projects that have had distinct staffing characteristics. For example, it is possible to have 4 very hard working people which produce more artifacts than those produced by members of a slow

paced 5 member team. This assumption makes it possible to calibrate a model for process improvement from an external point of view.

Model development

Modeling a concurrent process requires a good understanding the relations of the various concurrent activities of the project. Our model is based on the product development project model by Ford and Sterman, which describes the concurrency relationships that constrain the sequencing of tasks as the effects of and interactions with resources. This model identifies a generic process structure that can be used to describe the iteration and completion of artifacts and the generation and correction of errors in each activity. A generic structure used for the various activities simplifies the construction of the model as well as help in understanding it. This model is also extensible and in future other activities such as coding can also be integrated into the system.

This model was already calibrated for the hardware development and some understanding of the domain is required to model the process using this model. Many concepts such as concurrency constraints and average completion duration have to be mapped to the software process domain. Such mappings were created on the basis of analogies between software and hardware. The initial completion duration is equivalent to the time required to complete the first version of an artifact. Similarly the iteration duration is the time required for completing the next minor version of an artifact. QA activities involve removal of found defects from within the same phase. Coordination ensures that there is coordination among the two phases. Accordingly task productivity were determined on an empirical basis in terms of the percentage of work in an activity that is completed in a week per person. This quantity is fairly easily estimated for the CS 577 projects. Similarly average completion duration for each task is identified for the various tasks in both activities. The dynamic concurrence relations were modeled based on the authors understanding of the MBASE approach.

Another challenge was the accurate allocation of resources to the tasks to be performed. Many projects that use MBASE are RAD in nature. This means that resources are scarce and are usually capable of performing a variety of tasks. It is important to allocate the available resources in such a way that dead time and dead effort are minimized. This can only be performed by designing a closed loop system that allocates resources based on the number of pending tasks.

This was achieved by creating a model with only one activity and then use the arraying feature in iThink to create a multi-phase model. The single dimension model can be used to analyze and debug the flows within one phase. The inter-phase relations can then be modeled through a calibration of the two activities together.

The calibration of a life cycle model is quite sensitive to the project size. However, the early phases of the life cycle do not reveal precise size numbers. That makes it difficult to apply the available measures available from CS 577 data besides also increasing the number of input variables to the system. It is also very difficult to normalize for size in the UML terms as well as MBASE artifacts. Hence, we decided to calibrate the model as a function of the relative size of the project. Every project is considered to be 100 units in size with the performance calibration done in terms of the percentage completion. This also allows us to calibrate the model for comparison with the USDP effort profile.

The data for our model was acquired from the CSCI 577a (Fall 1998) and CSCI 577b (Spring 1999) Weekly Effort Reports. The data is listed as follows:

- Effort
- Size (documentation)
- Use Cases

- Requirements

For the size of the documents, number of pages were recorded in addition to the number of Use Cases in the documents. For requirements, the number of Nominal, Off-Nominal and Quality Attributes were collected for teams. The teams considered were those that had completed through Transition. The data collected is shown in the Appendix.

The problems encountered were that the data was not sufficiently specific enough in helping us to calibrate our model correctly. Much of the data needed was not collected for each team. For instance, there was no defect data collected for the classes during the LCO, LCA, and RLCA times. Also, the data for CSCI 577a (Fall 1999) was still being formed at the time of our study. This makes the task of validating the model a future activity for research.

We adjusted the parameters of our model such that our output would match the percentages for Requirements, Architecture & Design, and Implementation. Appendix A shows the complete percentages for all of the activities in CSCI 577a and 577b.

Model Description

The architecture development process model is based on the Ford-Sterman product development model. Process elements are organized in the form of a phase activity dichotomy, so that both activities namely requirements elicitation and architecture design are based on the tasks involving initial completion, coordination, QA and iteration. All the rates and levels of the generic process structure are arrayed in two dimensions i.e. activity and tasks.

There are two applicable activities namely requirements and architecture. Tasks are initial completion, quality assurance, coordination and iteration. More details on this model can be obtained from the Ford-Sterman model.

One of the additions of our model is the level *Identified*. This level represents the mental models of the involved stakeholders and a growth based on certain process characteristic patterns. Figure 5 shows how the mental model continues to grow during the project. An example of the mental model is how requirements come to the minds of customers. As the project progresses, the number of ideas reduces gradually. On the other hand number of new ideas in architecture increases gradually.

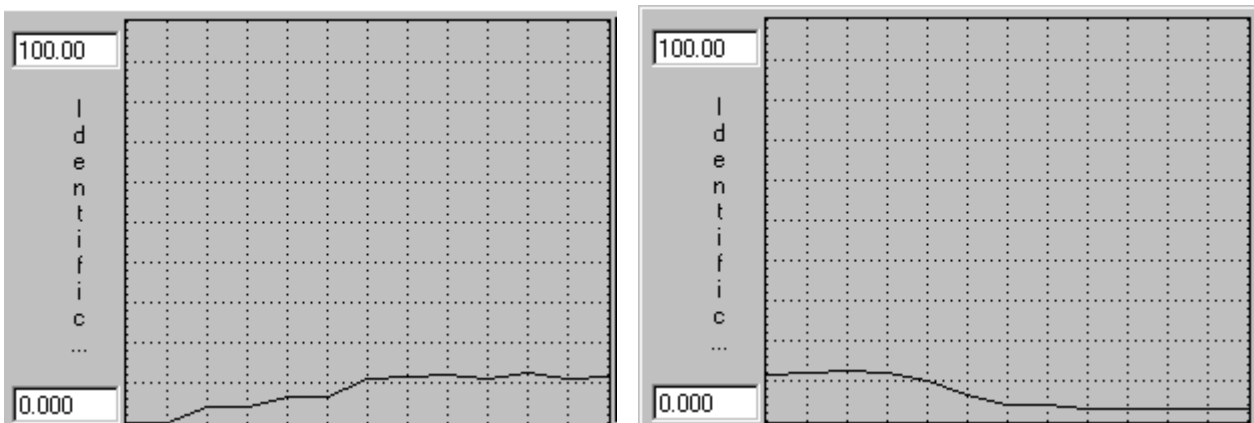


Figure 3 Identification Rate for (a) Requirements Elicitation and (b) Architecture Design

Internal concurrence constraint models the dynamic concurrence relations within individual activity whereas external concurrence constraint models the inter-activity dependencies. Internal concurrency for the two activities is modeled as shown in Figure 4.

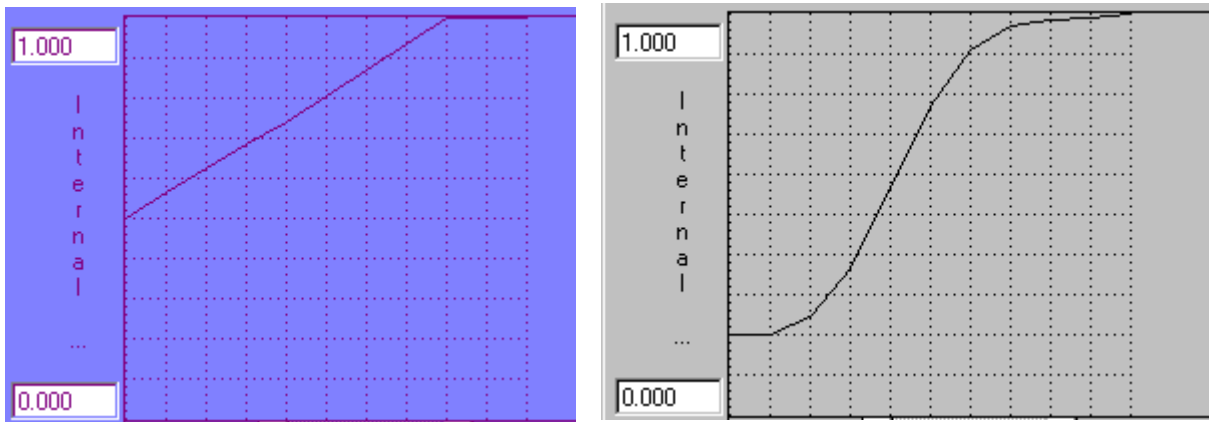


Figure 4 Internal Concurrence Relations for (a) Requirements Elicitation (b) Architecture Design

The concurrence for requirements elicitation indicates that it experiences a fairly high rate of independence on its state of completion. The large y-intercept implies that a large number of requirements are initially available for completion. At the same time all requirements are available for completion once 80% of the requirements are identified.

The architecture concurrence is in the form of an S-shape curve. This indicates that there is a significant dependence of architecture on the requirements. An initial completion level of 20% indicates that to some extent the architecture is known in advance before the project is begun. Most RAD projects are of this nature and the higher the stability of the RAD methodology in that project, the higher is the y-intercept. For example, a project that involves generation of new forms of reports from a standard SQL database that provides standard report design tools would have an extremely high y-intercept.

The external concurrence constraint relationship describes the inter-phase dependencies. In this model, the only inter-activity dependency is that between the requirements and architecture activity. It is modeled as shown in Figure 5. The S-shaped curve starting at the origin indicates that the architecture design can start only after some requirements elicitation is performed. However, the steep incline of the curve indicates that once some information is available about the requirements, the rest of the architecture can be designed quickly.

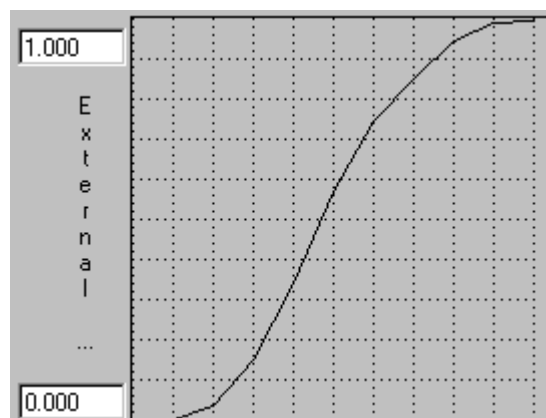


Figure 5 External Concurrence Constraint for Architecture Design Activity

The rate of iteration is determined by both the resource constraints on this rate as well as the minimum activity duration for iteration. One of our additions to this model is to study the effect of prototyping on the *Iter_Rate*. The rate is increased by the amount of time saved by prototyping which is given by:

$$\text{Iter_Rate} = \min(\text{Resource_Constraint}[\text{Activity}, \text{Iteration}], \\ \text{to_be_Iterated}[\text{Activity}] / \text{Average_Duration}[\text{Activity}, \text{Iteration}] \\ - \text{Reduction_due_to_Proto_Rate}[\text{Activity}])$$

Reduction_due_to_Proto_Rate is controlled by *Prototyping_Rate* and *Prototyping_Gain*. *Prototyping_Gain* is an empirical constant which affects requirements and the architecture activities differently. The reduction is calculated as:

$$\text{Reduction_due_to_Proto_Rate} = \text{Prototyping_Gain}[\text{Activity}] * \\ \text{Prototyping_Rate}$$

A separate prototyping chain is created to handle the demand for prototyping and create the prototypes based on resource and process constraints. *Proto_Need_Rate* models the demand for prototyping as determined by project specific criteria such as *Prototyping_Risk_Factors*. Not all the items in the main loop are to be prototyped, only a fraction are required. This produces the following equation

$$\text{Proto_Need_Rate} = \text{to_be_Iterated}[\text{Req}] * \text{frac_Prototypable}[\text{Req}] * \\ (\text{Prototyping_Risk_Factors}[\text{IKIWISI}] + \\ \text{Prototyping_Risk_Factors}[\text{Collaboration}]) + \\ \text{to_be_Iterated}[\text{Arch_Des}] * \\ \text{frac_Prototypable}[\text{Arch_Des}] * \\ (\text{Prototyping_Risk_Factors}[\text{Precedentedness}])$$

As seen from the equation only a few risk factors such as IKIWISI, Collaboration and Precedentedness are modeled. The risk factors are used in the same way as COCOMO cost drivers so that lower risk would reduce the need for prototyping whereas higher risks would increase the prototyping need rate.

The *Prototyping_rate* represents the rate at which prototypes are developed and provide the supply side of the prototyping chain. Resource constraints and prototyping period determine this rate. The equation for *Prototyping_rate* is given as:

$$\text{Prototyping_rate} = \min(\text{Prototyping_Resource_Constraint}, \\ \text{Prototyping_required} / \text{Prototyping_Period})$$

The average prototyping duration is affected by *Prototyping_Factors* which model how quickly prototypes can be created and behave similarly as COCOMO cost drivers. Currently only Tools, Assets and Experience of the prototyping personnel are employed as schedule drivers for prototyping so that lower factors would increase the prototyping period whereas higher factors would decrease the prototyping period. The equation for *Prototyping_Period* is given by:

$$\text{Prototyping_Period} = \text{Average_Prototyping_Period} / \\ (\text{Prototyping_Factors}[\text{Tools}] * \\ \text{Prototyping_Factors}[\text{Experience}] * \\ \text{Prototyping_Factors}[\text{Assets}])$$

Resource allocation is performed dynamically from among the available resources on the basis of the demand from the various project tasks. All the tasks that require resources are identified and the backlog is calculated using productivity of those individual tasks. Now the work required in each task as a fraction of total backlog provides a measure of the fraction of resources required for that task. This information is updated at every unit of time. *Approved* and *to_be_Coordinated* items both are counted towards QA work. For the sake of calibrating the model for CS 577 scenario, a constant staffing level is assumed, since no hiring is usually done during the semester.

The feedback loop diagram for the model is shown in the Figure 6. It shows the reinforcement feedback loop caused by initial basework when more items are completed as some items get completed. Another reinforcement loop is created due to prototyping where as more initial work is completed, need for prototyping increases and thus the iteration rate increases as a result.

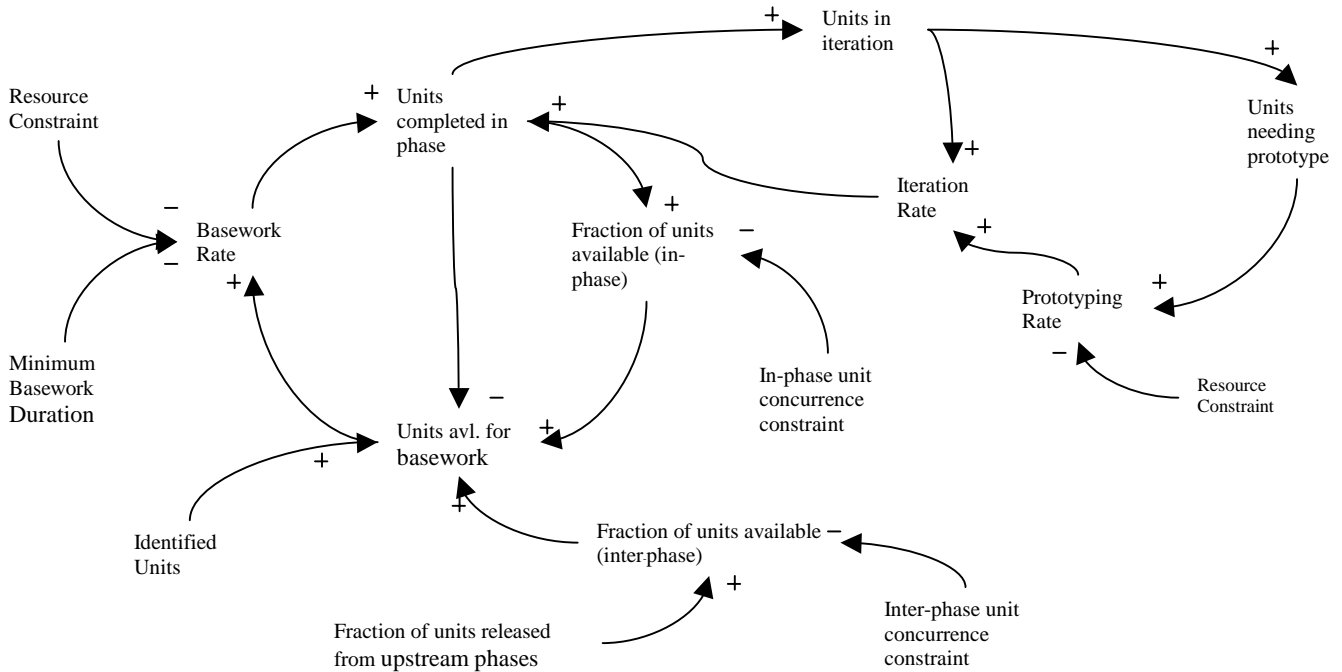


Figure 6 Main Causal Loop

The approach for verification and validation is to accurately set the parameters of the model using the data collected from CSCI 577.

Model Verification and Validation

The complete testing matrix is shown in the table from Appendix C. In that table, the values were changed one by one holding the rest of them the same. Using sensitivity analysis, the Prototyping Factor could be adjusted to increase/decrease the Prototyping Effort percentage. The next step would be a more extensive approach to model verification and validation with expert review.

Model Application and Transition

It was found that the model was most sensitive to changes in the Task Productivity corresponding to the Initial Completion of Architecture & Design. By using higher values, the percentages for Requirements and Architecture & Design could be reversed. Appendix F shows the model runs.

As we have seen above, the model behavior is predicated on the data collected from CS 577 projects. The data available for the project was not sufficient for complete modeling. The model is based on our understanding of MBASE and relies heavily on our judgement, as some of the data required for our model has not been collected from the CS 577 projects so far. As a further measure, these judgements can be validated against expert opinion and other estimates can be predicated on better data collection efforts.

Some other issues pertaining to transfer of this model to other projects and applications of MBASE and USDP would require model recalibration. This is because the model assumes a 16 week project with 4 personnel and uses productivity measures that come from the application of MBASE to CS 577 projects. Certain extensions to the model can also help applying other CORADMO opportunity tree factors to study the effect of prototyping. It is also possible to extend the model so that resource allocations are somewhat more controllable.

Conclusions and Recommendations for future work

The architecture development process model successfully models the activities during the initial phases of MBASE. This model is able to replicate the effort profiles for requirements and architecture/design activities based on a concurrent development model and a dynamic resource allocation scheme. It is our understanding that initial completion rates have a significant bearing on the rate of completion of the project artifacts. We also demonstrate the possibility of modeling a project in relative size terms instead of requiring a SLOC or FP size.

Our model also provides a starting point to model many of the RAD opportunity factors to understanding the effects of RAD techniques on software life cycles. In essence, this model can be a nice test bed for designing the CORADMO model. It can also be used to study staffing patterns for RAD projects.

Some extensions of this model would make it extensible to non-CS 577 projects by a proper calibration for the new process or approach. As future work, it is possible to study the creation and removal of defects in the model. Currently these are statically chosen and no probabilistic approach has been taken. The effect of peer-reviews and walkthroughs on the defect rates can also serve as a major addition to the model.

We have learned some lessons learned from the use of iThink Analyst for this continuous process model. Its support for arrays is a little difficult to use, for example its uniform treatment of array elements makes special cases difficult to specify. Marshalling information into arrays and connecting one dimension to another also pose a problem.

Other lessons have been learned in terms of the data collection and analysis effort. We find that more the variables being modeled more is the data needed; it easily takes off into a combinatorial explosion. Besides, not all data can be collected in advance. Often it is the case that once you start modeling, you find new forms of data to be collected. So it is difficult to work with an existing set of data where new kinds of data can not be obtained directly from an existing database. Also the starting point for the generation of artifacts is usually the weakest link in the model. Since this point also decides the behavior of all subsequent flows, it is possible that the first level or source can significantly alter the model behavior. The best strategy is to start from an insensitive source.

Appendices

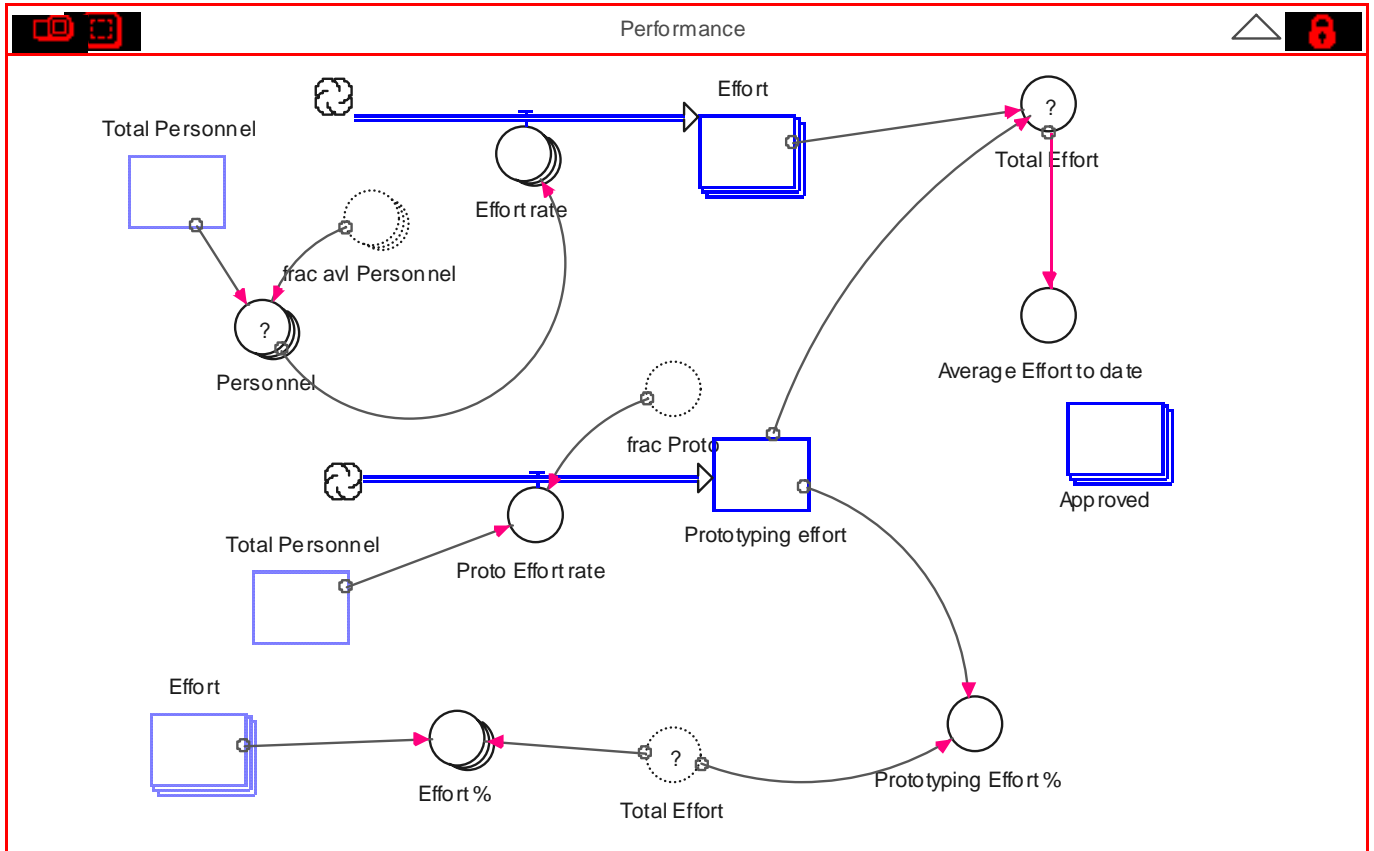
Appendix A. Activities in LCO, LCA, and RLCA

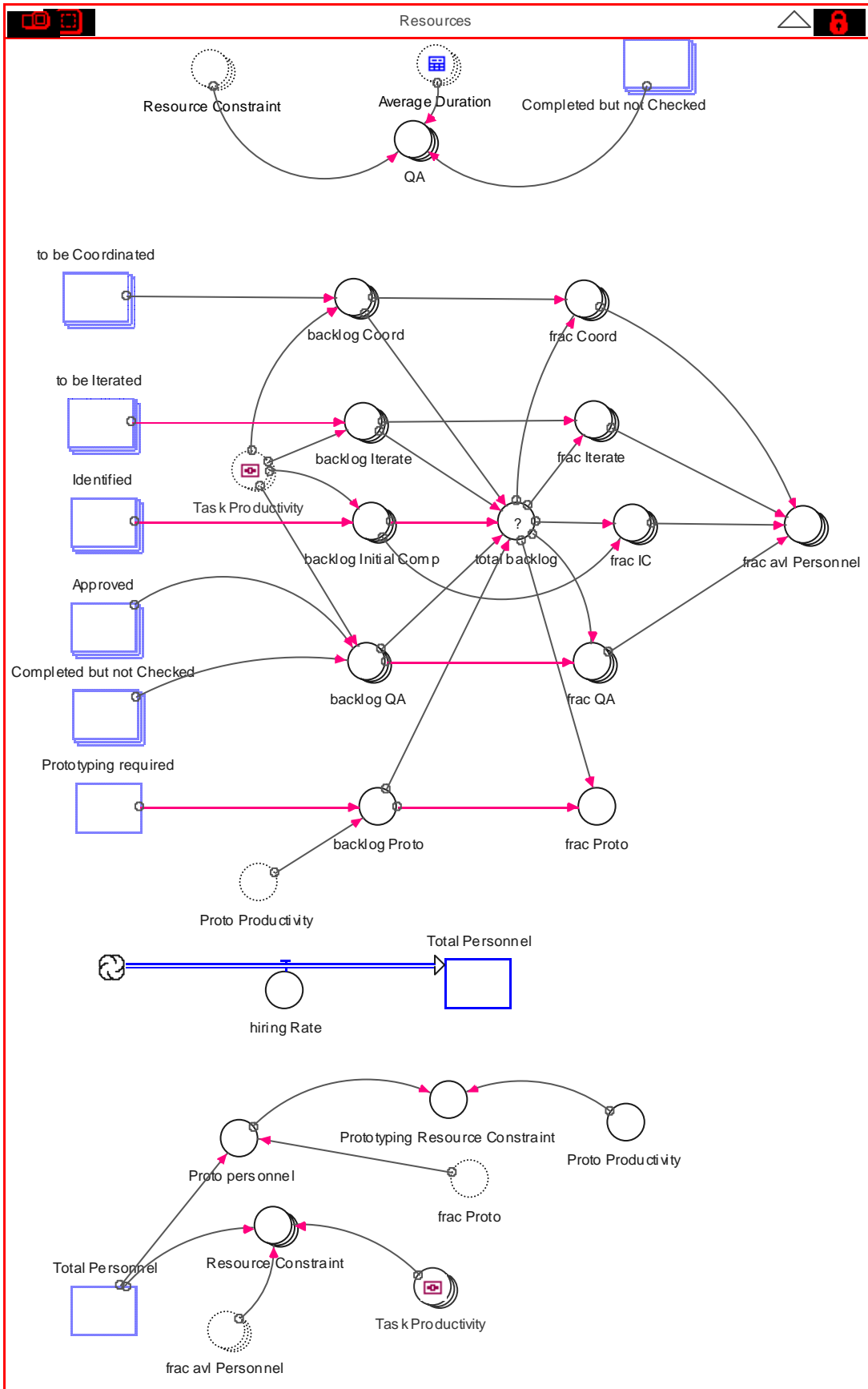
Table 2: Activities in LCO and LCA

	Total LCO (Hours)	Total LCA (Hours)	Total LCO%	Total LCA%	Average LCO (Hours)	Average LCA (Hours)	Average LCO%	Average LCA%	Total Hours	% Time
Tool Training/Learning:	1972.8	190.75	14.38	1.94	98.64	9.54	14.38	1.94	2163.55	9.2%
Application Research:	1142.65	320.45	8.33	3.26	57.13	16.02	8.33	3.26	1463.1	6.2%
Client Interaction:	630.6	399.55	4.6	4.07	31.53	19.98	4.6	4.07	1030.15	4.4%
Team meetings:	2149.4	1941.05	15.67	19.77	107.47	97.05	15.67	19.77	4090.45	17.4%
WinWin Usage:	1358.65	120.55	9.9	1.23	67.93	6.03	9.9	1.23	1479.2	6.3%
Email:	987.13	808.05	7.19	8.23	49.36	40.4	7.19	8.23	1795.18	7.6%
Telephone and Other Interactions:	343.6	252.91	2.5	2.58	17.18	12.65	2.5	2.58	596.51	2.5%
Operational Concept Description:	938	910.43	6.84	9.27	46.9	45.52	6.84	9.27	1848.43	7.9%
System and Software Requirements Definition:	708.2	735.33	5.16	7.49	35.41	36.77	5.16	7.49	1443.53	6.1%
System and Software Architecture Description:	876.8	953.6	6.39	9.71	43.84	47.68	6.39	9.71	1830.4	7.8%
Feasibility Rationale:	548.65	578.25	4	5.89	27.43	28.91	4	5.89	1126.9	4.8%
Life Cycle Plan:	513.8	797.5	3.74	8.12	25.69	39.88	3.74	8.12	1311.3	5.6%
Product Review and Integration:	146.5	234	1.07	2.38	7.33	11.7	1.07	2.38	380.5	1.6%
Prototyping	710.45	602.5	5.18	6.14	35.52	30.13	5.18	6.14	1312.95	5.6%
Planning and Control:	326.8	104.4	2.38	1.06	16.34	5.22	2.38	1.06	431.2	1.8%
Architecture Review Board preparation:	137.5	702.5	1	7.16	6.88	35.13	1	7.16	840	3.6%
Other 1	175.5	154.7	1.28	1.58	8.78	7.74	1.28	1.58	330.2	1.4%
Other 2	50.25	9.5	0.37	0.1	2.51	0.48	0.37	0.1	59.75	0.3%
Other 3	3	0	0.02	0	0.15	0	0.02	0	3	0.0%
TOTAL	13720.28	9816.02	100	100	686.01	490.8	100	100	23536.3	100.0%

Table 3: Activities in RLCA

	Total RLCA(Hours)	Total RLCA%	Average RLCA(Hours)	Average RLCA%	Total Hours	% Time
Tool Training/Learning:	311.15	15.20	51.86	15.20	311.15	15.2%
Application Research:	96.25	4.70	16.04	4.70	96.25	4.7%
Client Interaction:	81.55	3.99	13.59	3.99	81.55	4.0%
Team meetings:	195.90	9.57	32.65	9.57	195.9	9.6%
Email:	262.20	12.81	43.70	12.81	262.2	12.8%
Telephone and Other Interactions:	32.40	1.58	5.40	1.58	32.4	1.6%
Updates to Operational Concept Description:	109.50	5.35	18.25	5.35	109.5	5.4%
Updates to System and Software Requirements Definition:	118.50	5.79	19.75	5.79	118.5	5.8%
Updates to Feasibility Rationale:	47.75	2.33	7.96	2.33	47.75	2.3%
Design Creation or Modification:	171.25	8.37	28.54	8.37	171.25	8.4%
Design Review/Inspection:	61.10	2.99	10.18	2.99	61.1	3.0%
Code Generation or Modification:	81.50	3.98	13.58	3.98	81.5	4.0%
Code Review/Inspection:	9.10	0.44	1.52	0.44	9.1	0.4%
Unit Testing:	8.00	0.39	1.33	0.39	8	0.4%
Integration Testing:	6.00	0.29	1.00	0.29	6	0.3%
Prototyping	109.00	5.33	18.17	5.33	109	5.3%
COTS Assessment:	31.25	1.53	5.21	1.53	31.25	1.5%
COTS Tailoring:	4.50	0.22	0.75	0.22	4.5	0.2%
COTS Integration:	4.20	0.21	0.70	0.21	4.2	0.2%
Planning and Control:	106.35	5.20	17.73	5.20	106.35	5.2%
Project Review preparation:	35.65	1.74	5.94	1.74	35.65	1.7%
Quality Assurance:	38.00	1.86	6.33	1.86	38	1.9%
Transition Planning, Preparation and Execution:	8.50	0.42	1.42	0.42	8.5	0.4%
Training:	4.00	0.20	0.67	0.20	4	0.2%
User Documentation:	3.50	0.17	0.58	0.17	3.5	0.2%
Customer Deliverables:	4.30	0.21	0.72	0.21	4.3	0.2%
Test Specification:	88.50	4.32	14.75	4.32	88.5	4.3%
Other 1	8.50	0.42	1.42	0.42	8.5	0.4%
Other 2	8.00	0.39	1.33	0.39	8	0.4%
TOTALS	2046.40	100.00	341.07	100.00	2046.40	100.0%





Appendix E: Source Code of Model

Performance

$Approved[Activity](t) = Approved[Activity](t - dt) + (Approval_Rate[Activity] - Coord_due_to_downstream_QA[Activity]) * dt$

INIT $Approved[Activity] = 0$

DOCUMENT: Number of items that have been approved for release to the next phase. units

INFLOWS:

$Approval_Rate[Activity]$ (IN SECTOR: Process Structure)

OUTFLOWS:

$Coord_due_to_downstream_QA[Activity]$ (IN SECTOR: Process Structure)

$Effort[Activity](t) = Effort[Activity](t - dt) + (Effort_rate[Activity]) * dt$

INIT $Effort[Activity] = 0$

DOCUMENT: Amount of effort expended in the particular phase. This includes dead and wasted effort. person-weeks

INFLOWS:

$Effort_rate[Activity] = Personnel[Activity]$

DOCUMENT: Rate at which effort is spent on the particular phase. persons

$Prototyping_effort(t) = Prototyping_effort(t - dt) + (Proto_Effort_rate) * dt$

INIT $Prototyping_effort = 0$

DOCUMENT: Effort spent on prototyping. person weeks

INFLOWS:

$Proto_Effort_rate = Total_Personnel * frac_Proto$

DOCUMENT: Rate at which effort is spent on prototyping. persons

$Average_Effort_to_date = Total_Effort / time$

DOCUMENT: The average effort expended on the project on a weekly basis. person-weeks/weeks

$Effort_ \% [Activity] = Effort[Activity] / Total_Effort * 100$

DOCUMENT: percentage of the total effort spent in a particular phase. dimensionless

$Personnel[Activity] = Total_Personnel * ARRAYSUM(frac_avl_Personnel[Activity, *])$

DOCUMENT: Number of personnel working on a particular phase. This is only for observation purposes.
Persons

$$\text{Prototyping_Effort_}\% = \text{Prototyping_effort}/\text{Total_Effort}*100$$

DOCUMENT: Fraction of the total effort spent on prototyping. dimensionless

$$\text{Total_Effort} = \text{ARRAYSUM}(\text{Effort}[*]) + \text{Prototyping_effort}$$

DOCUMENT: Total effort spent on the project. peron-weeks

Process Structure

$$\begin{aligned} \text{Completed_but_not_Checked}[\text{Activity}](t) = & \text{Completed_but_not_Checked}[\text{Activity}](t - dt) + \\ & (\text{Iter_Rate}[\text{Activity}] + \text{Initial_Completion_Rate}[\text{Activity}] - \text{Discover_Intraphase_Defect_Rate}[\text{Activity}] - \\ & \text{Discover_Interphase_Defect_Rate}[\text{Activity}] - \text{Approval_Rate}[\text{Activity}]) * dt \end{aligned}$$

$$\text{INIT Completed_but_not_Checked}[\text{Activity}] = .2$$

DOCUMENT: Number of items that have been completed but cannot be considered to be checked. These items have been recently completed and need verification. units

INFLOWS:

$$\begin{aligned} \text{Iter_Rate}[\text{Activity}] = & \min(\text{Resource_Constraint}[\text{Activity}, \text{Iteration}], \\ & \text{to_be_Iterated}[\text{Activity}]/\text{Average_Duration}[\text{Activity}, \text{Iteration}] - \text{Reduction_due_to_Proto_Rate}[\text{Activity}]) \end{aligned}$$

DOCUMENT: Rate at which iteration can be performed on items to remove internal defects. This rate is affected by the speed at which prototyping can be performed. units/week

$$\begin{aligned} \text{Initial_Completion_Rate}[\text{Activity}] = \\ \min(\text{Resource_Constraint}[\text{Activity}, \text{Initial_Comp}], \text{Avl_for_Initial_Completion}[\text{Activity}]/\text{Average_Duration} \\ [\text{Activity}, \text{Initial_Comp}]) \end{aligned}$$

DOCUMENT: Rate at which items can be preliminarily processed to an initial completion. units/week

OUTFLOWS:

$$\begin{aligned} \text{Discover_Intraphase_Defect_Rate}[\text{Activity}] = \\ \text{QA}[\text{Activity}] * \text{prob_Discover_Def}[\text{Activity}] * \text{prob_Intraphase_Def}[\text{Activity}] \end{aligned}$$

DOCUMENT: The rate at which defects are detected within a phase. e.g defects of incompleteness and incorrectness. These can be removed through refinement and iteration. units/week.

$$\begin{aligned} \text{Discover_Interphase_Defect_Rate}[\text{Activity}] = & \text{QA}[\text{Activity}] * \text{prob_Interphase_Def}[\text{Activity}] * (1 - \\ & \text{prob_Intraphase_Def}[\text{Activity}]) * \text{prob_Discover_Def}[\text{Activity}] \end{aligned}$$

DOCUMENT: Rate at which defects are found in downstream phases leading to rework in the previous phases. Once the rework is completed, these items might need more iteration. units/week.

$$\begin{aligned} \text{Approval_Rate}[\text{Activity}] = & \text{QA}[\text{Activity}] - \text{Discover_Interphase_Defect_Rate}[\text{Activity}] - \\ & \text{Discover_Intraphase_Defect_Rate}[\text{Activity}] \end{aligned}$$

DOCUMENT: Rate at which items are released. units/week

Identified[Activity](t) = Identified[Activity](t - dt) + (Identification_Rate[Activity] - Initial_Completion_Rate[Activity]) * dt

INIT Identified[Activity] = 0

INFLOWS:

Identification_Rate[Activity] = time

OUTFLOWS:

Initial_Completion_Rate[Activity] =
min(Resource_Constraint[Activity,Initial_Comp],Avl_for_Initial_Completion[Activity]/Average_Duration [Activity,Initial_Comp])

DOCUMENT: Rate at which items can be preliminarily processed to an initial completion. units/week

Prototyping_required(t) = Prototyping_required(t - dt) + (Prototype_Need_Rate - Prototyping_Rate) * dt

INIT Prototyping_required = 0

DOCUMENT: A measure of the need to perform prototyping. This depends on the risks involved in the project as well as the number of requirements and architectural elements identified. units

INFLOWS:

Prototype_Need_Rate =
To_be_Iterated[Req]*frac_Prototypable[Req]*(Prototyping_Risk_Factors[IKIWISI]+Prototyping_Risk_Factors[Collaboration])+to_be_Iterated[Arch_Des]*frac_Prototypable[Arch_Des]*(Prototyping_Risk_Factors[Precedentedness])

DOCUMENT: Rate at which new prototyping needs come up. units/week

OUTFLOWS:

Prototyping_Rate = min(Prototyping_Resource_Constraint,Prototyping_required/Prototyping_Period)

DOCUMENT: Rate at which prototyping is performed. units/week

to_be_Coordinated[Activity](t) = to_be_Coordinated[Activity](t - dt) + (Discover_Interphase_Defect_Rate[Activity] + Coord_due_to_downstream_QA[Activity] - Coord_Rate[Activity]) * dt

INIT to_be_Coordinated[Activity] = 0

DOCUMENT: Number of items that require inter-phase coordination to be performed. units

INFLOWS:

Discover_Interphase_Defect_Rate[Activity] = QA[Activity]*prob_Interphase_Def[Activity]*(1-prob_Intraphase_Def[Activity])*prob_Discover_Def[Activity]

DOCUMENT: Rate at which defects are found in downstream phases leading to rework in the previous phases. Once the rework is completed, these items might need more iteration. units/week.

$$\text{Coord_due_to_downstream_QA}[\text{Activity}] = \text{QA}[\text{Activity}] * \text{DefRel}[\text{Activity}]$$

DOCUMENT: Amount of coordination that has to be performed between this phase and the successive phases due to defects. units/week

OUTFLOWS:

$$\text{Coord_Rate}[\text{Activity}] = \min(\text{Resource_Constraint}[\text{Activity}, \text{Coord}], \text{to_be_Coordinated}[\text{Activity}] / \text{Average_Duration}[\text{Activity}, \text{Coord}])$$

DOCUMENT: Rate at which coordination is performed. This depends on the kinds of techniques used for coordination such as email, meetings and traceability tools. units/week

$$\text{to_be_Iterated}[\text{Activity}](t) = \text{to_be_Iterated}[\text{Activity}](t - dt) + (\text{Discover_Intraphase_Defect_Rate}[\text{Activity}] + \text{Coord_Rate}[\text{Activity}] - \text{Iter_Rate}[\text{Activity}]) * dt$$

$$\text{INIT to_be_Iterated}[\text{Activity}] = 0$$

DOCUMENT: Number of units that require iteration. The iteration is required to remove internal defects

INFLOWS:

$$\text{prob_Intraphase_Defect_Rate}[\text{Activity}] = \text{QA}[\text{Activity}] * \text{prob_Discover_Def}[\text{Activity}] * \text{prob_Intraphase_Def}[\text{Activity}]$$

DOCUMENT: The rate at which defects are detected within a phase. e.g defects of incompleteness and incorrectness. These can be removed through refinement and iteration. units/week.

$$\text{Coord_Rate}[\text{Activity}] = \min(\text{Resource_Constraint}[\text{Activity}, \text{Coord}], \text{to_be_Coordinated}[\text{Activity}] / \text{Average_Duration}[\text{Activity}, \text{Coord}])$$

DOCUMENT: Rate at which coordination is performed. This depends on the kinds of techniques used for coordination such as email, meetings and traceability tools. units/week

OUTFLOWS:

$$\text{Iter_Rate}[\text{Activity}] = \min(\text{Resource_Constraint}[\text{Activity}, \text{Iteration}], \text{to_be_Iterated}[\text{Activity}] / \text{Average_Duration}[\text{Activity}, \text{Iteration}] - \text{Reduction_due_to_Proto_Rate}[\text{Activity}])$$

DOCUMENT: Rate at which iteration can be performed on items to remove internal defects. This rate is affected by the speed at which prototyping can be performed. units/week

$$\text{Average_Duration}[\text{Req}, \text{Iteration}] = 0.5$$

DOCUMENT: Duration of each requirement iteration. weeks

$$\text{Average_Duration}[\text{Req}, \text{Coord}] = 1$$

DOCUMENT: Duration of each requirement coordination. weeks

$$\text{Average_Duration}[\text{Req}, \text{Quality_Assurance}] = 0.75$$

DOCUMENT: Duration of each performing quality assurance on requirements. weeks

$$\text{Average_Duration}[\text{Req}, \text{Initial_Comp}] = 0.25$$

DOCUMENT: Duration of initial completion for each requirement. weeks

$$\text{Average_Duration}[\text{Arch_Des}, \text{Iteration}] = 1$$

DOCUMENT: Duration of each architecture and design iteration. weeks

$$\text{Average_Duration}[\text{Arch_Des}, \text{Coord}] = 1$$

DOCUMENT: Duration of each coordination performed for requirements and architecture. weeks

$$\text{Average_Duration}[\text{Arch_Des}, \text{Quality_Assurance}] = 0.5$$

DOCUMENT: Duration of each performing quality assurance on architecture and design. weeks

$$\text{Average_Duration}[\text{Arch_Des}, \text{Initial_Comp}] = .75$$

DOCUMENT: Duration of initial completion for architecture and design. weeks

$$\text{Average_Prototyping_Period} = 2$$

DOCUMENT: Average time period required for each prototyping iteration. weeks

$$\text{Avl_for_Initial_Completion}[\text{Activity}] = \text{Max}(0, \text{Total_Available}[\text{Activity}] - (\text{to_be_Coordinated}[\text{Activity}] + \text{Considered_Satisfactory}[\text{Activity}] + \text{to_be_Iterated}[\text{Activity}]))$$

DOCUMENT: Number of items available for initial completion. units/week

$$\text{Considered_Satisfactory}[\text{Activity}] = \text{Approved}[\text{Activity}] + \text{Completed_but_not_Checked}[\text{Activity}]$$

DOCUMENT: number of items considered satisfactory and forms the sum of completed and approved items. units

$$\text{DefRel}[\text{Activity}] = (1 - \text{prob_Discover_Def}[\text{Activity}]) * (\text{prob_Interphase_Def}[\text{Activity}] + \text{prob_Intraphase_Def}[\text{Activity}])$$

DOCUMENT: Fraction of defects released to the next phase. dimensionless

$$\text{External_Prec_Constraint}[\text{Activity}] = \text{Prev_Phase_Completion}[\text{Activity}]$$

DOCUMENT: External precedence constraint on the work for the current phase. dimensionless

$$\text{Fraction_Completed_and_Approved}[\text{Activity}] = \text{Considered_Satisfactory}[\text{Activity}] / \text{Total_Size}$$

DOCUMENT: Fraction of the total units of development that have been completed or approved. dimensionless

$$\text{frac_Prototypable}[\text{Req}] = .15$$

$$\text{frac_Prototypable}[\text{Arch_Des}] = .1$$

$$\text{Internal_Prec_Constraint}[\text{Activity}] = \text{Fraction_Completed_and_Approved}[\text{Activity}]$$

DOCUMENT: Internal Precedence constraint on the completion of work in the phase based on the amount of work already completed or available for completion. dimensionless

$$\text{Prev_Phase_Completion}[\text{Req}] = 100 + 0 * \text{Fraction_Completed_and_Approved}[\text{Req}]$$

DOCUMENT: Number of items from the requirements phase that have been completed or approved. This is required to determine the external concurrence in the project. units

$$\text{Prev_Phase_Completion}[\text{Arch_Des}] = \text{Fraction_Completed_and_Approved}[\text{Req}]$$

DOCUMENT: Number of items from the requirements phase that have been completed or approved. This is required to determine the external concurrence in the project. units

$$\text{prob_Discover_Def}[\text{Req}] = .85$$

DOCUMENT: Dimensionless probability that a requirement is defective and found in the requirement phase.

$$\text{prob_Discover_Def}[\text{Arch_Des}] = .95$$

DOCUMENT: Dimensionless probability that a architecture or design is defective and found in the architecture and design phase.

$$\text{prob_Interphase_Def}[\text{Req}] = .05$$

DOCUMENT: Dimensionless probability that a defect is found in the architecture phase.

$$\text{prob_Interphase_Def}[\text{Arch_Des}] = .1$$

DOCUMENT: Dimensionless probability that a defect is found to be defective in the next phase. In this case, there is no immediate phase that continues the work of the architecture phase and so a constant value of 0.2 is used.

$$\text{prob_Intraphase_Def}[\text{Req}] = .4$$

DOCUMENT: Dimensionless probability that a requirement is defective and is found to be so in the requirements phase.

$$\text{prob_Intraphase_Def}[\text{Arch_Des}] = .25$$

DOCUMENT: Dimensionless probability that an architecture unit is defective and is found to be so in the same phase.

$$\text{Prototyping_Factors}[\text{Tools}] = 1$$

DOCUMENT: The effort driver for prototyping tools that indicates the extent to which these tools can resolve critical risks through prototyping. Dimensionless. These factors are calibrated just like COCOMO and a nominal value of 1.0 is used. When the conditions are favorable, PF values are reduced and increased when unfavorable conditions exist.

$$\text{Prototyping_Factors}[\text{Experience}] = 1$$

DOCUMENT: The effort driver for prototyping experience that indicates the precedentedness of the prototyping effort. Dimensionless These factors are calibrated just like COCOMO and a nominal value of

1.0 is used. When the conditions are favorable, PF values are reduced and increased when unfavorable conditions exist.

$$\text{Prototyping_Factors[Assets]} = 1$$

DOCUMENT: The effort driver for prototyping assets that indicates the prepositioning of assets for prototyping. Dimensionless. These factors are calibrated just like COCOMO and a nominal value of 1.0 is used. When the conditions are favorable, PF values are reduced and increased when unfavorable conditions exist.

$$\text{Prototyping_Period} = \frac{\text{Average_Prototyping_Period}}{\text{Prototyping_Factors[Tools]} \cdot \text{Prototyping_Factors[Experience]} \cdot \text{Prototyping_Factors[Assets]}}$$

DOCUMENT: Nominal period required for performing prototyping. weeks

$$\text{Prototyping_Risk_Factors[IKIWISI]} = .5$$

DOCUMENT: Amount of extra prototyping to be done as a result of IKIWISI risk. This is measured in terms of a typical COCOMO effort driver form where 1.0 indicates nominal impact, a value less than 1 indicates less work and more than 1 indicates extra effort. dimensionless.

$$\text{Prototyping_Risk_Factors[Precedentedness]} = .2$$

DOCUMENT: Amount of extra prototyping to be done as a result of precedentedness of system risk. This is measured in terms of a typical COCOMO effort driver form where 1.0 indicates nominal impact, a value less than 1 indicates less work and more than 1 indicates extra effort. dimensionless.

$$\text{Prototyping_Risk_Factors[Collaboration]} = .5$$

DOCUMENT: Amount of extra prototyping to be done as a result of collaborative development risk. This is measured in terms of a typical COCOMO effort driver form where 1.0 indicates nominal impact, a value less than 1 indicates less work and more than 1 indicates extra effort. dimensionless.

$$\text{Reduction_due_to_Proto_Rate[Activity]} = \text{Req_Proto_Frac[Activity]} \cdot \text{Prototyping_Rate}$$

$$\text{Req_Proto_Frac[Req]} = 5$$

$$\text{Req_Proto_Frac[Arch_Des]} = 3$$

$$\text{Total_Available[Activity]} = \text{Total_Size} \cdot \min(\text{Internal_Prec_Constraint[Activity]}, \text{External_Prec_Constraint[Activity]})$$

DOCUMENT: Number of personnel available to perform various activities of a phase. persons

$$\text{Total_Size} = 100$$

DOCUMENT: Nominal size of the project in terms of development units. This is assumed as 100 so that all development units are measured in percentage rather than real terms

$$\text{External_Prec_Constraint[Activity]} = \text{Prev_Phase_Completion[Activity]}$$

DOCUMENT: External precedence constraint on the work for the current phase. dimensionless

Internal_Prec_Constraint[Activity] = Fraction_Completed_and_Approved[Activity]

DOCUMENT: Internal Precedence constraint on the completion of work in the phase based on the amount of work already completed or available for completion. dimensionless

Resources

Total_Personnel(t) = Total_Personnel(t - dt) + (hiring_Rate) * dt

INIT Total_Personnel = 3.5

DOCUMENT: Total personnel available for work in the engineering phase for product related work. persons

INFLOWS:

hiring_Rate = 0

backlog_Coord[Activity] = to_be_Coordinated[Activity]/Task_Productivity[Activity,Coord]

DOCUMENT: Backlog of coordination work. person-weeks

backlog_Initial_Comp[Activity] = Identified[Activity]/Task_Productivity[Activity,Initial_Comp]

backlog_Iterate[Activity] = to_be_Iterated[Activity]/Task_Productivity[Activity,Iteration]

DOCUMENT: Backlog of iteration work. person-weeks

backlog_Proto = Prototyping_required/Proto_Productivity

DOCUMENT: Backlog of prototyping work. person-weeks

backlog_QA[Activity] =
(Approved[Activity]+Completed_but_not_Checked[Activity])/Task_Productivity[Activity,Quality_Assurance]

DOCUMENT: Backlog of QA work. person-weeks

frac_avl_Personnel[Req,Iteration] = frac_Iterate[Req] +
0*(frac_IC[Req]+frac_Coord[Req]+frac_QA[Req])

frac_avl_Personnel[Req,Coord] = frac_Coord[Req] +0*(frac_IC[Req]+frac_Iterate[Req]+frac_QA[Req])

frac_avl_Personnel[Req,Quality_Assurance] =
frac_QA[Req]+0*(frac_IC[Req]+frac_Coord[Req]+frac_Iterate[Req])

frac_avl_Personnel[Req,Initial_Comp] =
frac_IC[Req]+0*(frac_Coord[Req]+frac_Iterate[Req]+frac_QA[Req])

DOCUMENT: Fraction of the total personnel available for the various activities in the different phases. dimensionless.

frac_avl_Personnel[Arch_Des,Iteration] =
frac_Iterate[Arch_Des]+0*(frac_IC[Arch_Des]+frac_Coord[Arch_Des]+frac_QA[Arch_Des])

$\text{frac_avl_Personnel}[\text{Arch_Des}, \text{Coord}] =$
 $\text{frac_Coord}[\text{Arch_Des}] + 0 * (\text{frac_IC}[\text{Arch_Des}] + \text{frac_Iterate}[\text{Arch_Des}] + \text{frac_QA}[\text{Arch_Des}])$

$\text{frac_avl_Personnel}[\text{Arch_Des}, \text{Quality_Assurance}] =$
 $\text{frac_QA}[\text{Arch_Des}] + 0 * (\text{frac_IC}[\text{Arch_Des}] + \text{frac_Coord}[\text{Arch_Des}] + \text{frac_Iterate}[\text{Arch_Des}])$

$\text{frac_avl_Personnel}[\text{Arch_Des}, \text{Initial_Comp}] =$
 $\text{frac_IC}[\text{Arch_Des}] + 0 * (\text{frac_Coord}[\text{Arch_Des}] + \text{frac_Iterate}[\text{Arch_Des}] + \text{frac_QA}[\text{Arch_Des}])$

$\text{frac_Coord}[\text{Activity}] = \text{if } (\text{total_backlog} < 0.1) \text{ then } 0 \text{ else } \text{backlog_Coord}[\text{Activity}] / \text{total_backlog}$

DOCUMENT: Fraction of the total effort to be expended in coordination. dimensionless

$\text{frac_IC}[\text{Activity}] = \text{if } (\text{total_backlog} < 0.1) \text{ then } 0 \text{ else } \text{backlog_Initial_Comp}[\text{Activity}] / \text{total_backlog}$

DOCUMENT: Fraction of the total effort to be expended in initial completion. dimensionless. This is assumed to be constant throughout the engineering phase

$\text{frac_Iterate}[\text{Activity}] = \text{if } (\text{total_backlog} < 0.1) \text{ then } 0 \text{ else } \text{backlog_Iterate}[\text{Activity}] / \text{total_backlog}$

DOCUMENT: Fraction of the total effort to be expended in iteration. dimensionless

$\text{frac_Proto} = \text{if } (\text{total_backlog} < 0.1) \text{ then } 0 \text{ else } \text{backlog_Proto} / \text{total_backlog}$

DOCUMENT: Fraction of the total effort to be expended in prototyping. dimensionless

$\text{frac_QA}[\text{Activity}] = \text{if } (\text{total_backlog} < 0.1) \text{ then } 0 \text{ else } \text{backlog_QA}[\text{Activity}] / \text{total_backlog}$

DOCUMENT: Fraction of the total effort to be expended in QA. dimensionless

$\text{Prototyping_Resource_Constraint} = \text{Proto_personnel} * \text{Proto_Productivity}$

DOCUMENT: The limit imposed on prototyping by available resources. units/week

$\text{Proto_personnel} = \text{Total_Personnel} * \text{frac_Proto}$

$\text{Proto_Productivity} = 2$

DOCUMENT: Productivity associated with prototyping activity. units/week

$\text{QA}[\text{Activity}] =$
 $\text{min}(\text{Resource_Constraint}[\text{Activity}, \text{Quality_Assurance}], \text{Completed_but_not_Checked}[\text{Activity}] / \text{Average_Duration}[\text{Activity}, \text{Quality_Assurance}])$

DOCUMENT: Quality assurance activities include configuration management and testing required to release the development unit to the next phase. units/ week

$\text{Resource_Constraint}[\text{Activity}, \text{Task}] =$
 $\text{frac_avl_Personnel}[\text{Activity}, \text{Task}] * \text{Total_Personnel} * \text{Task_Productivity}[\text{Activity}, \text{Task}]$

DOCUMENT: The amount of personnel effort available for performing an activity in a particular phase. units/week

$\text{Task_Productivity}[\text{Req}, \text{Iteration}] = 3$

DOCUMENT: Productivity for iterations on requirements. units/person/week

Task_Productivity[Req,Coord] = 6

DOCUMENT: Productivity for coordination of requirements. units/person/week

Task_Productivity[Req,Quality_Assurance] = 6

DOCUMENT: Productivity for quality assurance on requirements. units/person/week

Task_Productivity[Req,Initial_Comp] = 5

DOCUMENT: Productivity for initial completion of requirements. units/person/week

Task_Productivity[Arch_Des,Iteration] = 3

DOCUMENT: Productivity for iterations on architecture and design. units/person/week

Task_Productivity[Arch_Des,Coord] = 10

DOCUMENT: Productivity for coordination for architecture and design. units/person/week

Task_Productivity[Arch_Des,Quality_Assurance] = 12

DOCUMENT: Productivity for quality assurance of architecture and design. units/person/week

Task_Productivity[Arch_Des,Initial_Comp] = 11

DOCUMENT: Productivity for initial completion of architecture and design. units/person/week

total_backlog =

ARRAYSUM(backlog_QA[*])+ARRAYSUM(backlog_Coord[*])+ARRAYSUM(backlog_Iterate[*])+AR
RAYSUM(backlog_Initial_Comp[*])+backlog_Proto

DOCUMENT: Total work backlog in person-weeks

Not in a sector

Appendix F: Graphs of Model Runs

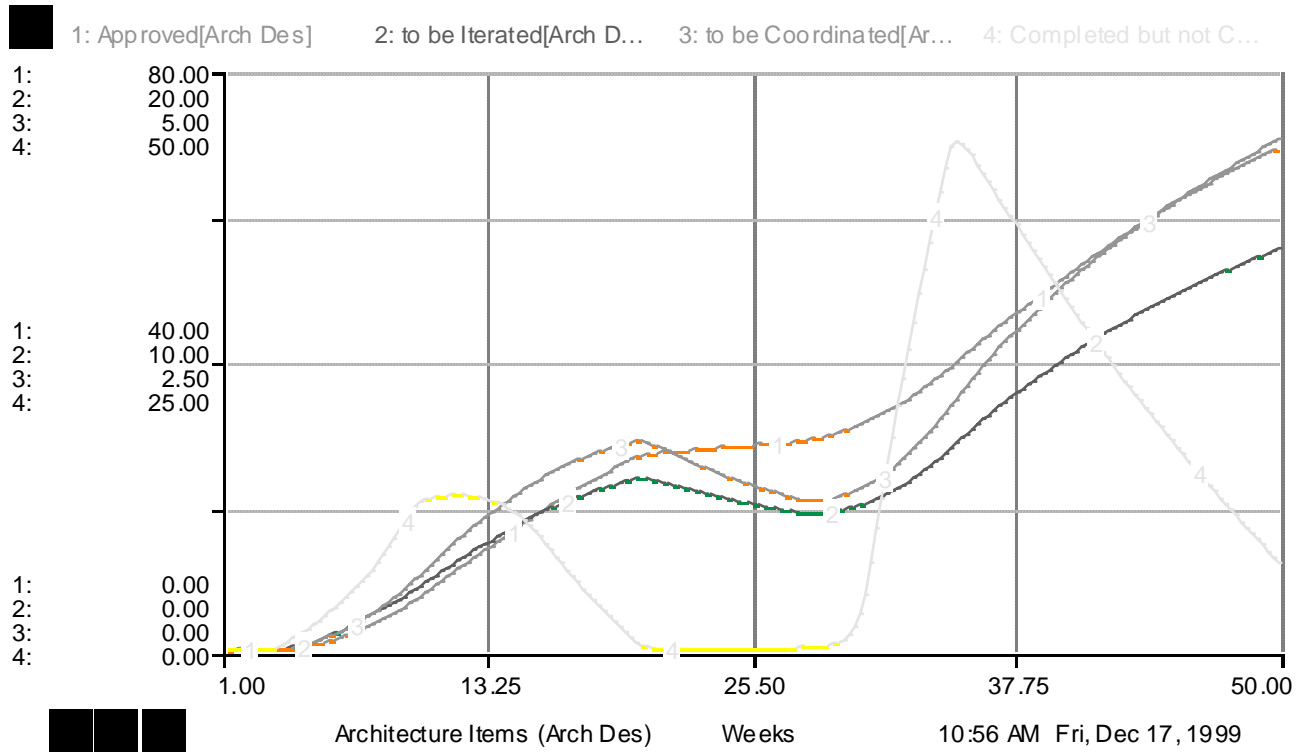


Figure 7: Architecture Items

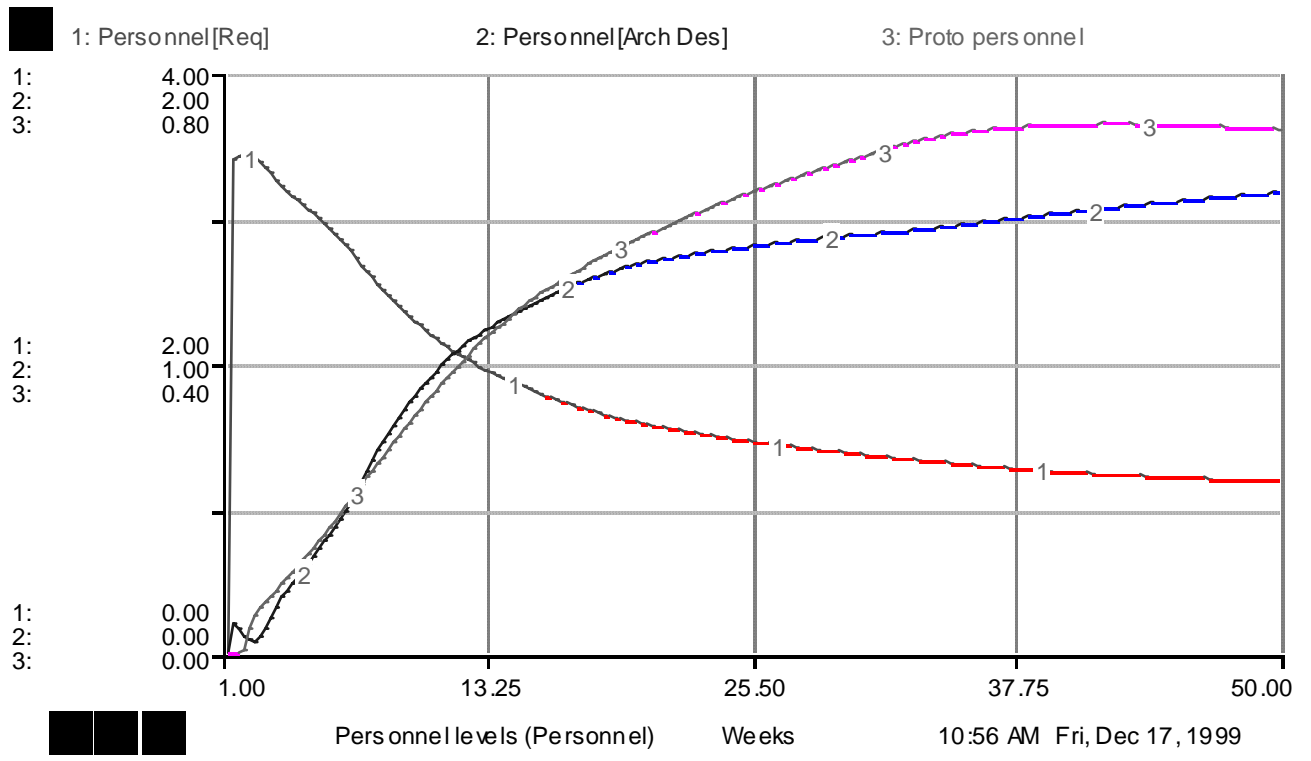


Figure 8: Personnel Levels

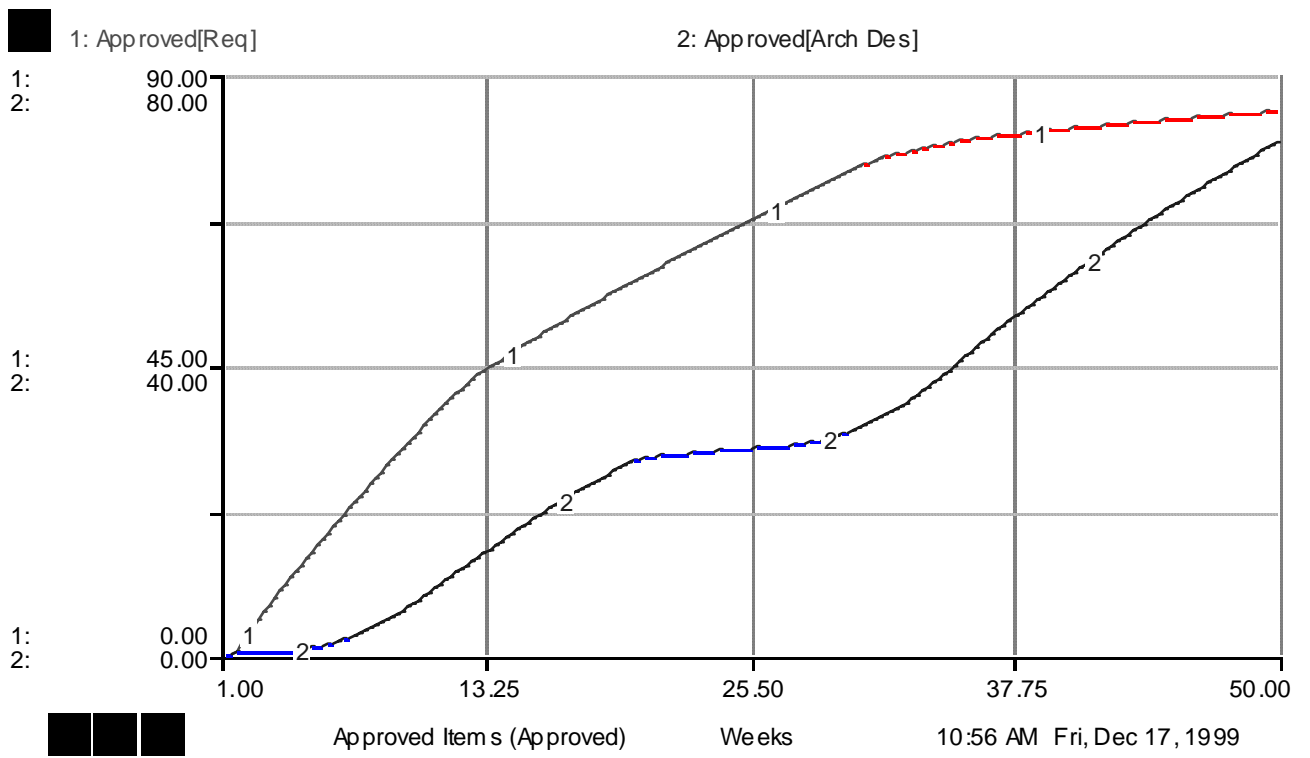


Figure 9: Approved Items

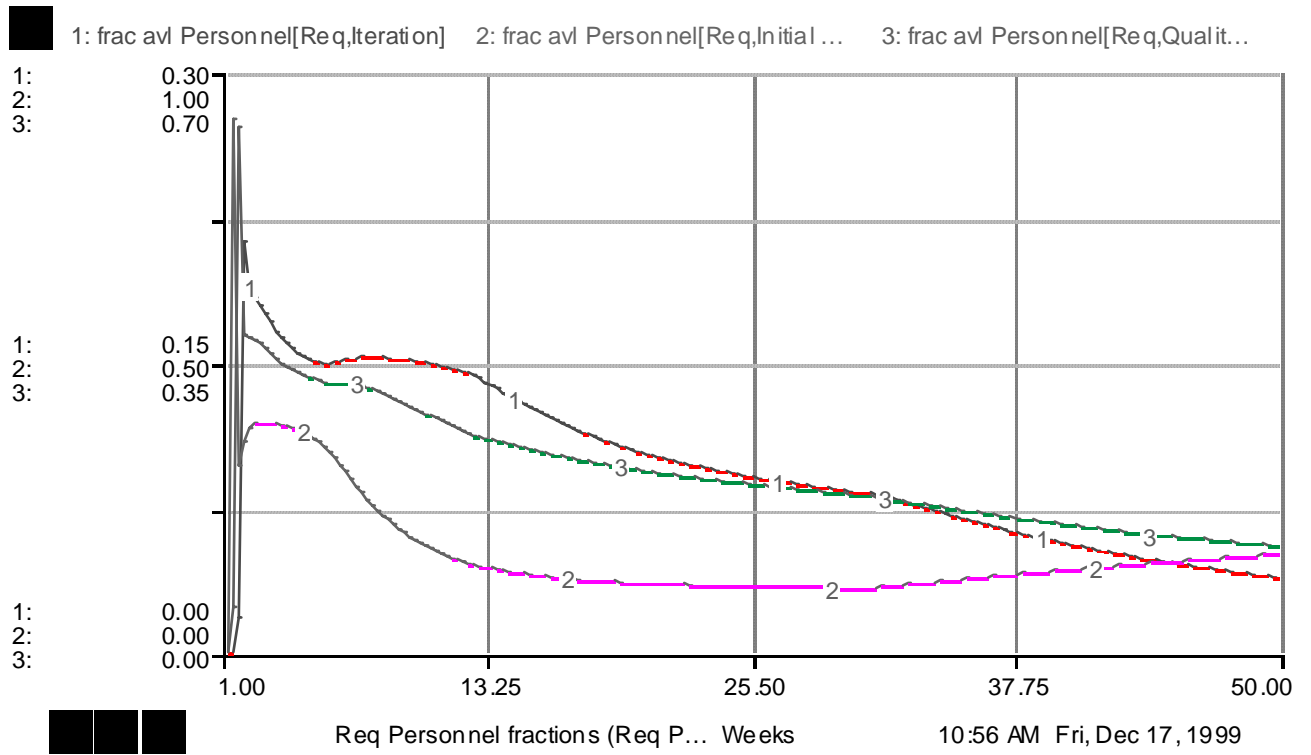


Figure 10: Required Personnel fractions

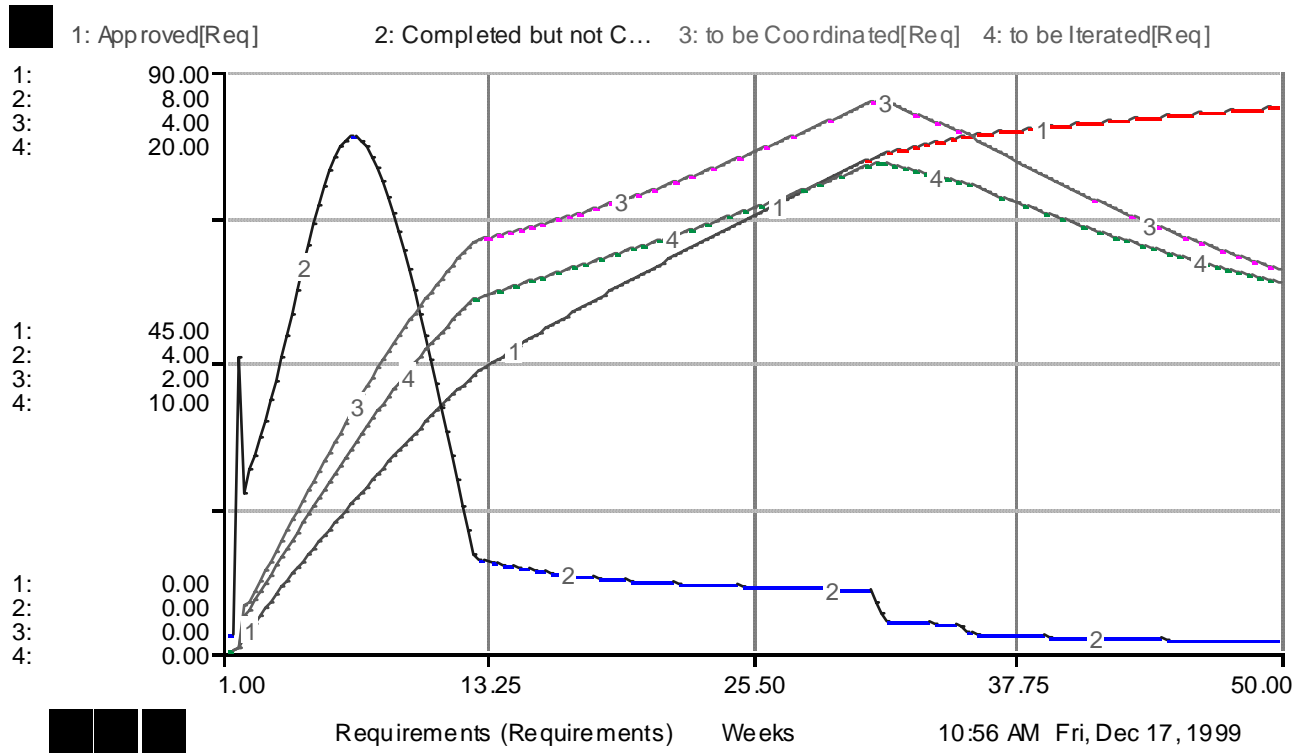


Figure 11: Requirements

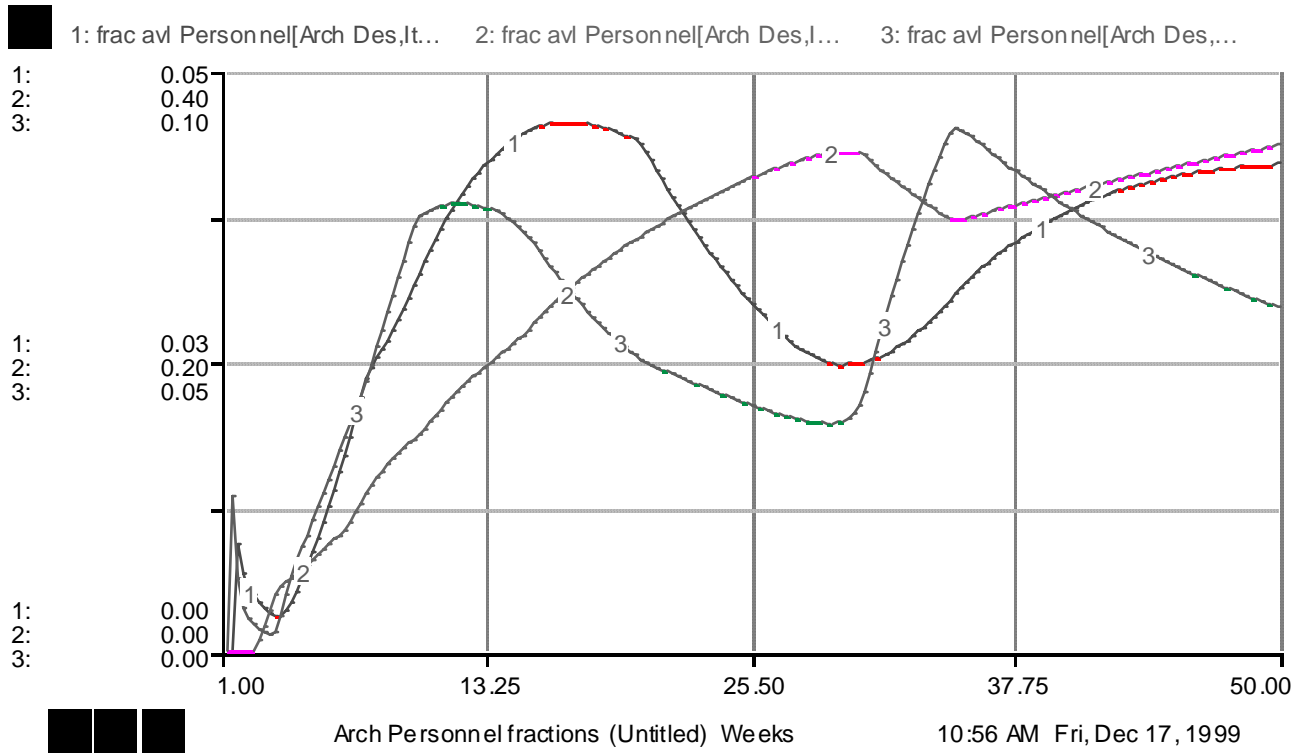


Figure 12: Architecture Personnel fractions

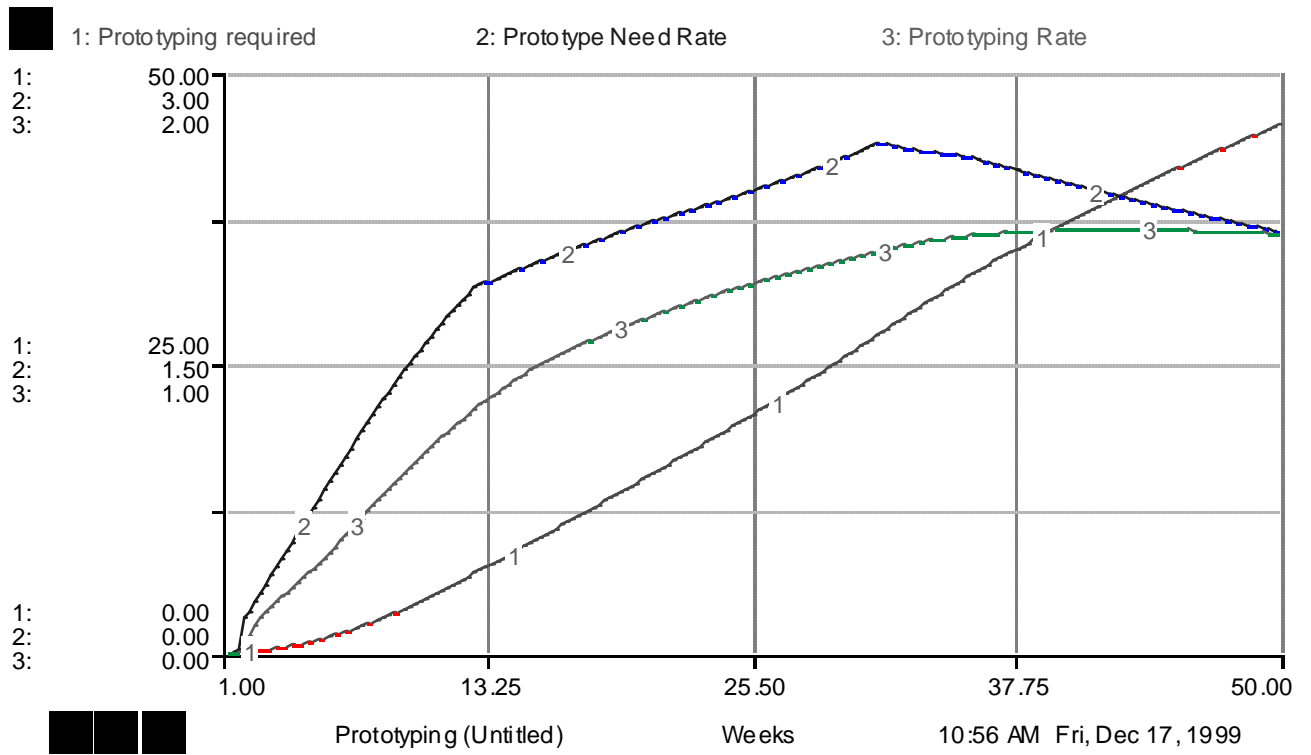


Figure 13: Prototyping

References

Ford, David and Sterman, John, "Dynamic Modeling of Product Development Processes", January 1997.