

**Effects of Knowledge Reuse
on the
Spacecraft Development Process ,**

by

E s t h e r S . D u t t o n

Submitted to the Department of Aeronautics and Astronautics
on May 9, 1997 in partial fulfillment of the
requirements for the Degree of
Master of Science in,
Aeronautics and Astronautics

at the

Massachusetts Institute of Technology

June 1997. .

© 1996 Esther S. Dutton.
All rights reserved.

The author hereby grants to MIT permission to reproduce
and to distribute publicly paper and electronic
copies of this thesis document in whole or in part.

Signature of Author: _____
Department of Aeronautics and Astronautics
May 9, 1997

Certified by: _____
Edward F. Crawley
Head of Aeronautics and Astronautics Department
Thesis Supervisor

Accepted by: _____
Daniel E. Hastings
Professor of Aeronautics and Astronautics
Chairman, Department Committee on Graduate Students

Effects of Knowledge Reuse on the Spacecraft Development Process

by
Esther S. Dutton

Submitted to the Department of Aeronautics and Astronautics
on May 9, 1997 in partial fulfillment of the
requirements for the Degree of
Master of Science in
Aeronautics and Astronautics

ABSTRACT

An experimental study was carried out to explore proposed knowledge management practices for the spacecraft development process at the Jet Propulsion Laboratory (JPL). The study was part of the Develop New Products (DNP) Project, which reengineered the process for spacecraft development at JPL. The reengineered process is called the DNP Process. The process used in this experiment was based on the NASA Program/Project Life Cycle Process Flow and refined to include methodologies employed by the DNP Process.

A system model of this DNP Process was created using Foresight, a commercially available suite of tools designed for rapid prototyping of complex systems. The objectives of this experiment were to simulate the spacecraft development process and determine the effects of knowledge management on development time. A method of quantifying knowledge creation and reuse in spacecraft projects was developed and programmed into the DNP Process model. The model was also designed to encounter and solve project problems, scale for spacecraft complexity, and use a specialized workforce.

Although DNP is pursuing several methods of reducing development time, this study emphasizes the practice of knowledge management. Other parameters, such as cost, were modeled but not analyzed. Such parameters were included only to create a comprehensive model that could be used in future studies.

Simulations were run on the model to determine the effects of reusing knowledge on project development time. The simulations were performed using spacecraft of different complexities and with different amounts of knowledge available for reuse. One set of simulations was run using a Monte Carlo scenario based on random inputs, The second set was run using test cases of five current flight projects.

The results from the simulations showed that increasing available reusable knowledge decreases spacecraft development time in complex projects. Less complex project did not appear to benefit significantly from knowledge availability. Five current flight projects tested, however, showed significant savings in development time could be realized by reusing knowledge.

The results of this, experiment suggest that knowledge management is a worthwhile investment for the Jet Propulsion Laboratory.

Thesis Supervisor: Edward F. Crawley
Title: Head of Aeronautics and Astronautics Department

BIOGRAPHICAL NOTE

"Esther S. Dutton" is a student in the MIT Engineering Internship Program (EIP). The EIP is a five-year BS/MS program in which three summers and one semester are spent at a company. This thesis was researched, performed, and documented during an eight month internship at the Jet Propulsion Laboratory in Pasadena, California.

Table of Contents

1.0 Introduction	1
2.0 Objectives	1
2.1 Problem	2
2.2 Hypothesis	2
2.3 Objectives	2
3.0 Background	2
3.1 Reengineering	2
3.2 Knowledge Management	3
3.3 Develop New Products Reengineering	4
3.4 DNP Processes	5
4.0 Approach	6
5.0 Develop New Products Process Model	7
5.1 Processes	9
5.1.1 Systems Engineering	11
5.1.1.1 Mission Feasibility	11
5.1.1.2 Mission Definition	12
5.1.1.3 System Definition	13
5.1.2 Subsystem Design	14
5.1.2.1 Preliminary Design	14
5.1.2.2 Final Design	15
5.1.3 Model Integration and Test	15
5.1.4 Build, Assemble, and Test	16
5.2 Reviews	17
5.3 Model Logic	19
5.3.1 Inputs and Outputs	19
5.3.2 Resources	21
5.3.2.1 Workforce	21
5.3.2.2 Knowledge	21
5.3.3 Problem Occurrence Logic	21
5.3.3.1 Problem Generation	22
5.3.3.2 Problem Solving	23
5.4 Formulas--and Assumptions	23
5.4.1 Project Complexity	24
5.4.2 Development and Production Cost	25
5.4.3 Equations for Total Cost	26
5.4.4 Workforce Cost	27
5.4.5 Knowledge Availability	27
6.0 Simulations	28
6.1 Random Inputs	28
6.2 Test Cases	29
7.0 Data Confidence	29
8.0 Data Collection	29

9.0 Results and Discussion	0...00.....	31
9.1 Process Duration vs. Knowledge Availability		32
9.2 Process Duration vs. Knowledge Scenario		34
10.0 Conclusions		38
11.0 Suggestions for Further Research		39
12.0 Acknowledgments00.....	39
13.0 References	"".....	39
14.0 Appendixes		40
Appendix A.....	40
Appendix B.....	41
Appendix C	42
Appendix D	43



List of Figures

1. DNP High Level Process Map	5
2. Experimental Approach	6
3. Develop New Products Model	8
4. DNP Processes	9
5. Process Hierarchy	10
6. Model Input Screen	19
7. Model Output and Monitor Screen	20
8. Problem Loops	22
9. Assumptions	23
10. Complexity Parameter Analysis	24
11. Spacecraft/Vehicle Level Cost Model	26
12. Knowledge Spectrum	29
13. Process Duration vs. Concurrent Projects	32
14. Process Duration vs. Previous Projects	33
15. Average Process Duration vs. Complexity Factor	33
16. Case One: Pathfinder Project	35
17. Case Two: Mars Global Surveyor Project	36
18. Case Three: First Generation Microspacecraft	36
19. Case Four: Seawinds	37
20. Case Five: State of Health Monitor (Micro Devices Laboratory Instrument)	38

List of Tables

1. Explanation of Terms in Subprocess Tables	10
2. Mission Feasibility Subprocesses	12
3. Mission Definition Subprocesses	13
4. System Definition Subprocesses	13
5. Preliminary Design Subprocesses	14
6. Final Design Subprocesses	15
7. Model Integration and Test Subprocesses	16
8. Build, Assemble, and Test Subprocesses	16
9. Major Reviews	18
10. Random Problem Generation	22
11. Project Complexity Factor	25
12. Random Inputs	28
13. Knowledge Scenarios	29
14. Test Cases	30
15. Test Matrix A	31
16. Test Matrix B	31

1.0 Introduction

The Jet Propulsion Laboratory's (JPL) Develop New Products (DNP) Reengineering Team was formed to design a new spacecraft development process in response to NASA's "faster, better, cheaper" goals for flight projects of the future. DNP is creating a self-titled process incorporates new initiatives into well-proven processes. The goals of the DNP Process are to produce spacecraft in less time, at lower risk, and for less money than ever before.

Three initiatives set the DNP Process apart from the old way of doing things. These include 1) practicing better knowledge management, 2) reducing long lead times, and 3) employing model based design. Separate process teams within DNP were tasked with implementing each of these three initiatives. Of these teams, the Data and Information Management (D&IM) Team was charged with creating the infrastructure for knowledge management at JPL. The knowledge management infrastructure will allow JPL flight projects to better reuse knowledge created by previous and current Projects.

Previous research has shown reusing knowledge increases the quality and significantly reduces cycle time for product development. However, this research was conducted on commercial projects such as cellular phones (Motorola) and automobiles (Chrysler). No research currently exists on knowledge management's effects on spacecraft development. Therefore, the DNP team had no market research available to support a knowledge management initiative. Although the DNP team felt knowledge management would help to achieve their goals, they were unsure if its impact on cost and schedule would be significant.

Professor Edward Crawley of MIT was called in as a consultant to the DNP team in mid-1 996. While at JPL, he expressed the need for a set of metrics to measure the success of the DNP Process. In order to determine if initiatives were successful, JPL needed a way to compare the new process to the old. He also suggested JPL do a preliminary study would predict how well the DNP Process would meet its goals.

Crawley's suggestions led to an aggressive modeling effort of the DNP Process. The first utilization of this model was the study of knowledge management, which this thesis documents.

This thesis documents the approach, modeling, simulation, and data collection methods of the experiment. It also provides background on the DNP reengineering effort at JPL. Results and discussion are followed by suggestions for further research.

2.0 Objectives

2.1 Problem

A number of knowledge transfer problems at JPL contribute to long development time. These problems force engineers to do work from scratch, rather than make use of previous work. Although JPL's flight projects are unique, many are related in a broad sense. Therefore, information can be shared between them. For example, a start-up spacecraft intending to travel to Mars could reuse many of the requirements for past flight projects went to Mars, for the planetary parameters will be the same.

Project isolation is one of the major knowledge transfer problems. Little knowledge is shared between projects developed concurrently. Networks are not lab-wide, and thus information must be transferred by email attachment or by floppy disk. Although the use of the World Wide Web has increased electronic information transfer, projects do not post documents consistently or comprehensively. Often separate projects use separate databases to store the same data and produce redundant documentation. And the most commonly used form of engineering documentation - the memo - is poorly tracked. Some projects had made an effort to keep hardcopies on file, but cataloging is poor and accessibility is unpredictable.

Another major problem is poor archiving and cataloging of previous projects. Engineers are often unable to find documents generated by other projects. Catalogs exist, but many are out of date or not electronically accessible. Information cannot be reused if it cannot be found. Previous research shows engineers will typically search for 15 minutes before giving up and doing the work from scratch. In this way, engineers "reinvent the wheel," duplicating previous efforts.

The D&IM Team was formed to solve these problems and others related to knowledge transfer at JPL. Using the benchmarked practices of successful industry partners, the team sought to develop a methodology for corporate knowledge management. This experiment attempted to gain a priori knowledge of whether the methodology would be successful. The question to be answered was: Will better management of JPL's corporate knowledge reduce spacecraft development time?

2.2 Hypothesis

By archiving, packaging, and reusing knowledge, JPL can reduce the cycle time of spacecraft development."

2.3 Objectives

The experimental objective was to assess the impact of knowledge reuse on spacecraft development time. A secondary objective was to produce a comprehensive, flexible model other DNP teams could use to test their methodologies. In order to meet the secondary objective, the DNP model was designed to meet the following requirements:

1. Perform realistic iterations based on a problem occurrence logic
2. Have ability to vary the following (for scalability):
 - a. complexity of project
 - b. type of project
 - c. number of concurrent projects
 - d. number of previous projects
 - e. knowledge reuse scenario

The first requirement will assure the model follows a realistic development cycle. Processes are often iterated upon to solve problems as they arise. Different types of problems may require different process iterations. The second requirement will assure the model can be scaled for different types of projects. The complexity of the project refers to how technically or logistically difficult it is to design. The type of project is earth orbiter, interplanetary, or instrument. The number of concurrent projects are those developed at the same time as the modeled project. Previous projects are those were design prior to the modeled project. The **knowledge reuse scenario** is the amount of reusable information the project is expected to have. An example of a knowledge scenario would be "all reusable" if the project expects to draw a considerable amount of information from a previous project.

3.0 Background

This section will provide background on the Develop New Products Program at JPL and its relationship to the reengineering movement of the 90's.

3.1 Reengineering

Reengineering is a radical change occurs when a company redesigns work for significant improvements in productivity. Michael Hammer and James Champy, authors of Reengineering the Corporation, formally define **reengineering** as: "the fundamental rethinking and radical redesign of business systems to achieve dramatic improvements in critical, contemporary measures of performance, such as cost, quality, service, and speed."

The challenge for reengineering teams at JPL is to significantly improve cost, quality, and speed simultaneously. Although JPL engineers have been well-trained at Hammer's reengineering seminars,

understanding technically how to reengineer does not ensure success. Effectively using principles of change management does ensure success. Managing resistance at JPL is the most difficult part of reengineering - and the part least receptive to an engineering method. Addressing the human challenge early allows companies more time to accept the change. Unfortunately, JPL early efforts were focused more on identifying innovations, with minor emphasis on human factors. Recent outreach efforts, such as the Reengineering Times newsletter and DNP web site, have helped JPL employees better understand reengineering and prepare for change.

Early successes have fueled the growth of the reengineering movement in America. At Ford Motor Company, reengineering improved invoice processing so it required 75% fewer people and more accurate financial information was produced. A credit issuance process at IBM Credit Corporation used to take two weeks now takes only four hours. Taco Bell reconfigured their restaurants, increasing peak capacity for a top unit from \$400 per hour to over \$1500 per hour.

Reengineered processes are simpler than those they replace. Several jobs might be combined into one. The number of checks and controls might be reduced, Work is performed where it makes most sense and workers are empowered to make more decisions themselves. Centralized and decentralized operations are combined in new productive ways. Information technology (such as, knowledge-based, expert systems) is often employed in the design of these new processes.

An example of a process is getting gas for a car. In this case, the purpose of the process is to fill the tank and pay for the gas. The process begins with pulling into the station, and ends with receiving a receipt and leaving. The customer has the money and has come to buy gas, and the station is the supplier.

The process steps are the activities the customer and the station personnel do to complete the transaction. This simple example is a business process. Business processes are sets of activities transform a set of inputs into a set of outputs for the customer or another process.

Metrics of success are important elements of reengineering. They are necessary to know how well reengineering is working, and where the company is headed. Successful reengineering efforts constantly assess themselves and improve all dimensions of their companies. Metrics are the cornerstone of assessment and the foundation for any business improvement.

A metric is a standard measure to assess performance in a particular area. Metrics are the basis for a successful process management system and programs for continuous improvement. Customer focus and performance standards take the form of metrics assess a companies's ability to meet customer needs and business objectives.

3.2 Knowledge Management

Knowledge Management is an evolving business trend many companies are still unfamiliar with. It has been billed as a critical tool for the 21st-century corporation and been the subject of a recent onslaught of books, magazine articles, conferences, business-school classes, World Wide Web sites and even an emerging executive position (Chief Knowledge Officer). Leading companies such as Skandia Insurance Co. Ltd. have launched major knowledge management initiatives which many others are benchmarking. The confusion's understandable. Even experts and practitioners disagree on something as fundamental as exactly what to call concerted efforts to capture, organize and share what employees know. Such efforts are often referred to as managing "intellectual capital; "intellectual assets" or "knowledge resources."

Proponents generally agree about why it's important, Because of downsizing, constant change, companies feel more pressure than ever to maintain a well-informed workforce, boost productivity and gain competitive advantage. By creating a comprehensive and easily accessible organizational archive, knowledge management helps meet all those goals. But questions continue to evolve along with the trend itself. Is it possible to manage something as intangible as knowledge? How do you determine its value?

Knowledge management projects vary widely, but typical efforts are intended to retain and organize employee expertise, making it easily available anywhere, any time. Tools for preserving and sharing knowledge range from newsletters to Lotus Notes, World Wide Web sites to in-house workshops, books to best practices. Hughes Space & Communications Co.'s Knowledge Highway includes a lessons-learned database with hypertext links to directories of human expertise, published materials and other information.

Proponents say effective knowledge management pays off in fewer mistakes, less redundancy,

quicker problem solving, better decision making, reduced research and development costs, and improved products. Leif Edvinsson, Skandia's vice president remarked, "There is a need to build a bridge between the old ways of doing things and the new ways. Intellectual capital is bridge," Skandia claims its knowledge management efforts reduced the start-up time for opening a corporate office in Mexico from seven years to six months.

To meet the demands of a faster, better, cheaper business environment, JPL needs to have information available in formats will last a long time and are economical to upkeep. Projects need to reuse ideas and products and move towards capabilities-driven design. The need to exchange information more quickly and easily is key to this effort. JPL seeks to benchmark other companies and create a stable and productive knowledge management architecture.

3.3 Develop New Products Reengineering

In 1995, House and Senate Republicans proposed a 36% cut in **NASA spending** by the year 2000. To prepare for this, Dan Goldin's strategy over the last few years has been to absorb a 36% cut through fiscal year (FY) 2000 while maintaining funding stability through FY 1997. He hoped by continued stability in FY 1997 NASA centers could restructure carefully to assure safe and achievable cost savings. Part of this restructuring effort would include eliminating low-priority support functions, and non-essential programs.

In the face of reduced resources JPL must find ways to maximize scientific return while minimizing cost. Since cost is directly related to launch mass, it is clear JPL can no longer build large spacecraft such as Cassini and Galileo. JPL's is preparing for a future of mini-spacecraft, with mini-instruments or single instruments significantly reduce launch mass. The DNP Process will enable JPL to perform smaller, more advanced, unique space exploration missions.

Large missions are vital to JPL, however, and thus will not disappear "immediately. Large missions contribute funds to improve the laboratory's infrastructure. Smaller projects cannot afford to contribute significantly to such improvements JPL is not prepared today to, support a multitude of small projects. It must make the transition quickly, however, to meet growing public and customer expectations for better, faster, cheaper spacecraft.

Many of DNP's methods are based on the work of Michael Hammer, who is widely recognized as an expert in reengineering. Cheryl Currid of JPL identified seven principles of reengineering based on the work of Hammer and others. She lists them as 1) organize work around results, not tasks, 2) capture data only one time--when it is first created, 3) allow. decision points where work is performed, 4) incorporate controls into information processing, 5) make people who use a process do the work, 6) work in parallel instead of sequentially, then integrate results, and 7) treat geographically dispersed resources as one.

These seven principles guide the" DNP effort, whose goal is to reduce project development cycle-time by 50% and in turn reduce cost by 30%. **Cycle time is defined** as the time from **pre-proposal** studies to launch readiness (or delivery to customer). The cost goal is limited to the cost incurred during this cycle time. **DNP seeks to** accomplish these two goals with no reduction in performance and no increase in risk.

Moreover, DNP seeks to achieve their goals without a routine need for heroic performance by the spacecraft engineers. The first spacecraft to attempt a better, faster, cheaper approach was the Pathfinder project, which is part of the Discovery Program. It took 3 years from funding onset to launch with a relatively low budget of \$171 million dollars, The Pathfinder project, however, was infamous for its heroic hours worked by its engineers.', ,

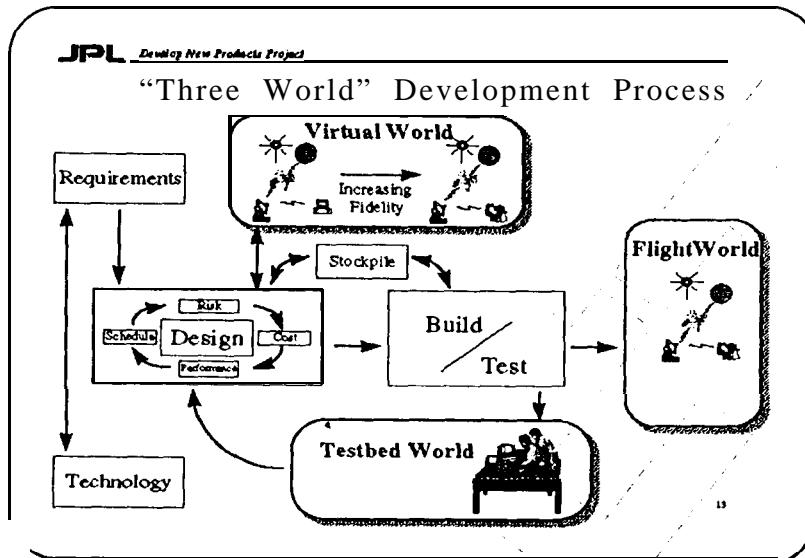


Figure 1: DNP High Level Process Map

The DNP goal is to define, document, and "institutionalize a set of processes and supporting tools to design and develop JPL missions. DNP has the responsibility not only to reform spacecraft design, but to provide a basis for a long term, continuous improvement process" It must provide opportunities for improvement as new processes and technologies evolve.

DNP seeks to capture the best ideas, tools, and methodologies currently in use JPL and add a suite of new tools and methodologies. These new tools will be either developed in-house or acquired from successful industry partners. In order to add new tools, it will be necessary to refine the laboratory's infrastructure. The Project Design Center and Design Hub are two new computing and communication centers house and maintain new tools in a distributed design environment. Institutionally-maintained facilities and tools are key in DNP's process reengineering effort.

In order to aid the reengineering process, DNP is setting up relationships with industry partners such as Lockheed Martin, universities such as MIT, and government agencies such as The Aerospace Corporation. These relationships will bring outside opinions and best practice strategies to the DNP teams.

3.4 DNP Processes

The DNP Process spans the entire life cycle of a project, from the time a new idea is conceived until the end of the mission. It is divided into four concurrently operating processes: PPIC, MSD, DBAT, and VIVO. These processes are designed to produce a high fidelity product definition by the start of development phase and enable a development time frame which is 1/2 the time.

Project Planning, Implementing, and Closing (PPIC) provides project level direction and resources to all processes. PPIC combines inputs for the other processes to construct a project plan which is used for project tracking metrics. PPIC provides project management, allowing DNP Processes to perform in an efficient work environment: Project Managers are given the tools and techniques necessary to plan and manage their projects. PPIC ensures the implementation phase with a well-defined approach.

The Mission and System Design (MSD) process provides the tools, processes, and expertise to accomplish Concept Development (CD), Mission System Engineering (MSE), and Scenario Development (SD). MSD provides a nurturing, accessible environment for new missions. It encourages missions to take advantage of new technology in order to attain high benefits with low cost.

The DBAT process "Designs, Builds, Assembles, and Tests" models, hardware, and software, It provides design and manufacturing processes and institutionally supported tools. DBAT supports the

Design Hub, a multi-disciplinary, multi-project design facility. DBAT is also implementing the strategic stockpile initiative, intended to reduce long lead times for critical hardware.

The Verify, Integrate, Verify, and Operate (VIVO) process provides a continuous testing environment throughout the life cycle of the project, VIVO provides the functions of system integration and test, mission operations development, and mission operations. Mission operations, however, is not an area undergoing reengineering through the DNP effort.

4.0 Approach

The five steps in the experiment approach were 1) define process interfaces, 2) program model, 3) populate model, 4) simulate and 5) extract metrics. An outline of the approach is shown in Figure 2.

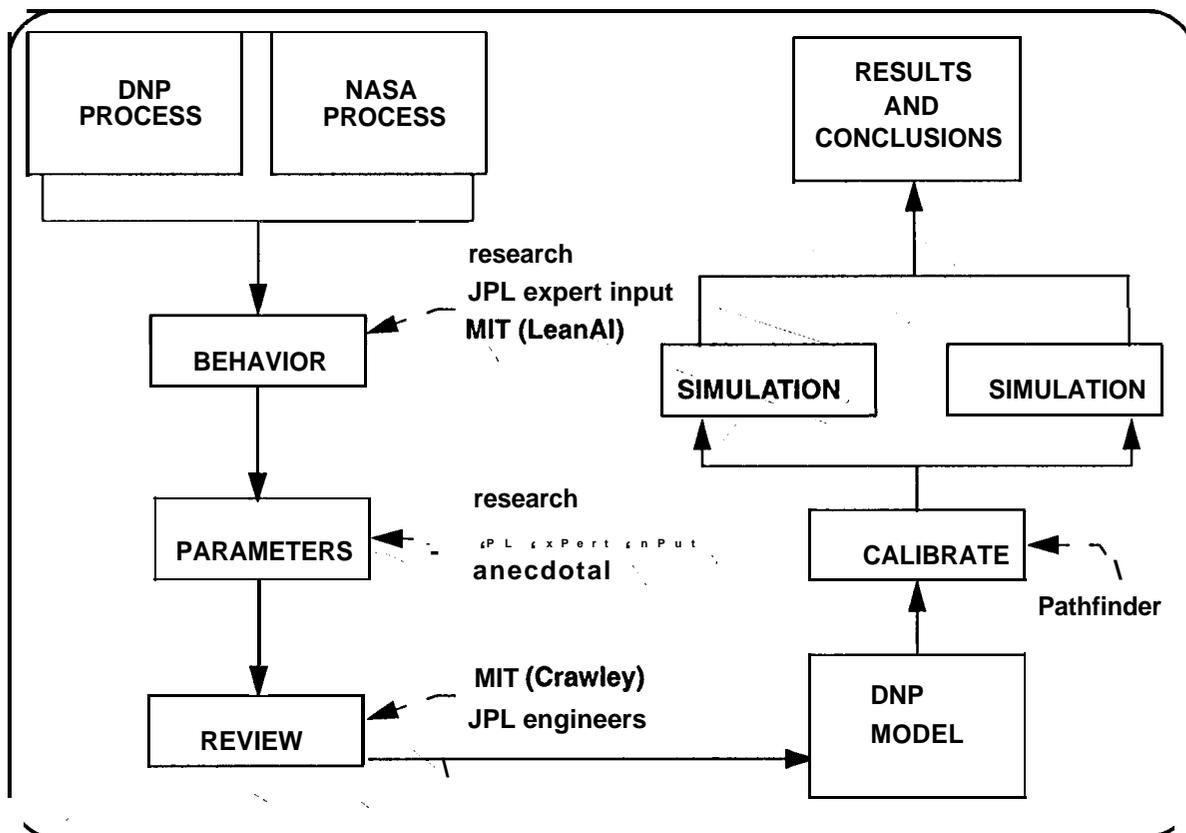


Figure 2: Experimental Approach

The first step was to define the interfaces on the life cycle process maps. The NASA Program and Project Life Cycle process map was the baseline and JPL's processes were mapped to it. Each process is represented by a block. The lines between the blocks represent the information flow, or the interface. In order to understand how knowledge reuse was to be accomplished, it was first necessary to know what information is passed along these interfaces.

A spreadsheet was created listed each process, its input, and its output. The spreadsheet also documented the tools (such as a word processor) used by the process and the form the inputs/outputs took (such as a report or set of raw data). Defining the interfaces showed what information was necessary to perform a process. It also illustrated the best methods of information storage. Traditionally, most outputs took the form of documents which were placed under strict configuration-control." By defining the interfaces, it became clear most information would be better passed in the form, for example, of electronic spread-

sheets rather than hardcopy documents.

The interface spreadsheet was used as a guide to program the model, the second step in the approach. For each information input/output identified, a data type was created in the model. The model was programmed to pass along information in the form of data types from process to process. A process could not begin until all the necessary information was received. And if an input was refined, and received again by a process, the process repeated and produced an updated output.

Programming the model took several weeks. It consisted of decomposing the high level life cycle process into elements and subprocesses and defining their behavior. The behavior of a process is the mechanics it performs to accomplish a set of tasks. Process behaviors were gathered from research, input from experts at JPL, and systems dynamics guidance from Sean Morgan of the MIT Lean Aircraft Initiative.

Each subprocess was defined as either a state transition diagram or a mini-specification, which are two programming elements in the Foresight software tool. A state transition diagram describes the behavior of a subprocess using a diagram of states. A mini-specification describes behavior in much the same way as a C-program. Figure 3 shows the hierarchical breakdown of the model. The details of the model are explained in section 5.0. It is an elaborate C code' mimics the process of spacecraft development.

After the model was programmed, the next step was to populate it with statistics drawn from past projects. Baseline time and workforce parameters were programmed into each process. Workforce and time estimates were averaged based on program size. Data was gathered from research at the JPL archives and input from experts at JPL. These parameters could be multiplied by a figure of merit the represented complexity, A simple project had a figure of merit of 1- and thus used the baseline time and workforce values. A more complicated project might have a figure of merit of 2- and thus used twice the workforce and took twice as long as baseline. Examples of projects and figures of merit are shown in Table 11.

After parameters were entered into the model process, it underwent a formal review at JPL by a committee of system and subsystem engineers. It also underwent an informal review by Professor Edward Crawley at MIT. After the review, model development was frozen and calibration began, The model was calibrated to Pathfinder Project cost and schedule data.

The fourth step in the approach was to run simulations of the life cycle model. These simulations are detailed in section 6.0.

The final step was to extract metrics from the process, model and simulations. The relationships among time, cost, workforce, problems, and knowledge variables were graphed. The metrics drawn from these relationships are detailed in section 5.2.

5.0 Develop New Products Process Model

The Develop New Projects Process Model developed for this experiment is based on both the NASA Program/Project Life Cycle Process Flow and DNP Process maps. The top-level Foresight representation of the DNP Model is shown in Figure 3. The dotted lines show examples of data passed between the processes. Each process, block in Figure 3 has a hierarchy of processes beneath it. The hierarchy structure is shown in Figure 5.

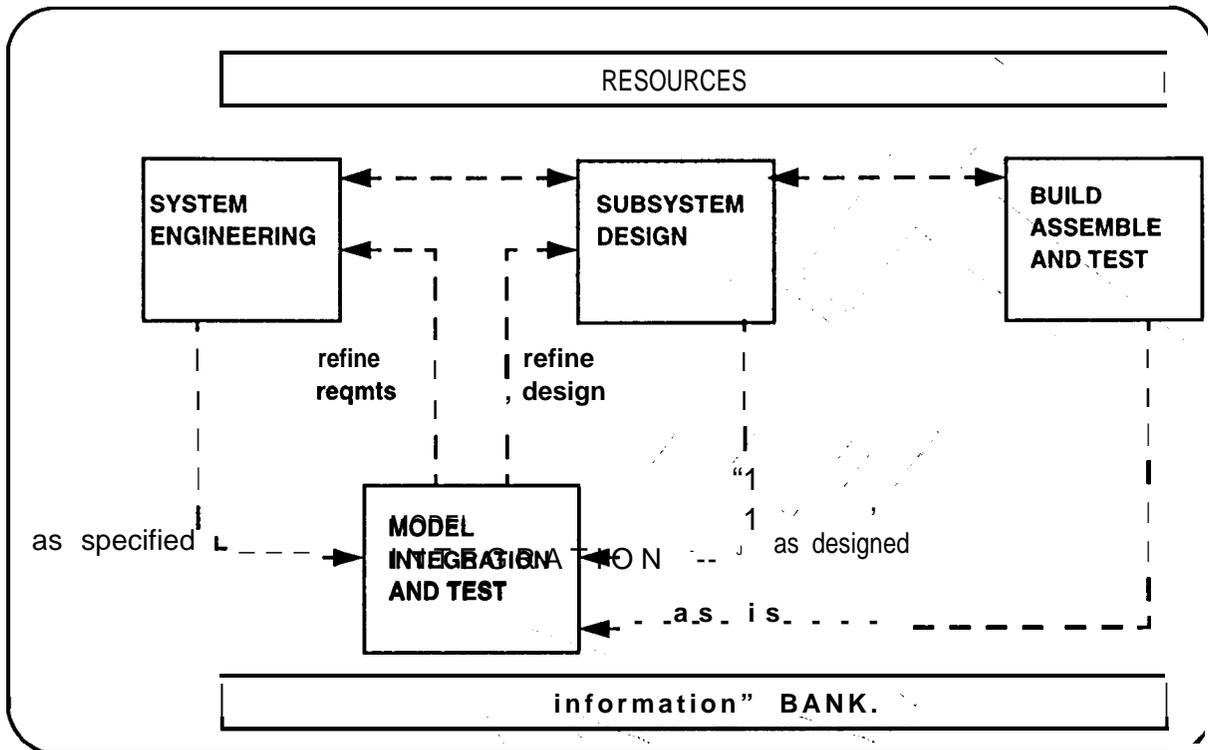


Figure 3: Develop New Products Model

The DNP model shown in Figure 3 is a generic map of the spacecraft development process. The four high level processes shown do not exactly correlate with DNP's four main processes. DNP Processes were defined by JPL and thus are specific to the laboratory's way of doing business. In order to generalize the model for use outside of JPL, the generic process names of System Engineering, Subsystem Design, etc. were chosen. The DNP Processes, however, can be mapped on the model as shown in Figure 4. The DBAT, MSD, and VIVO processes overlap several generic processes in the model. PPIC is represented as a resource manager only because it has several functions are outside the scope of this model, such as the development of project tracking metrics.

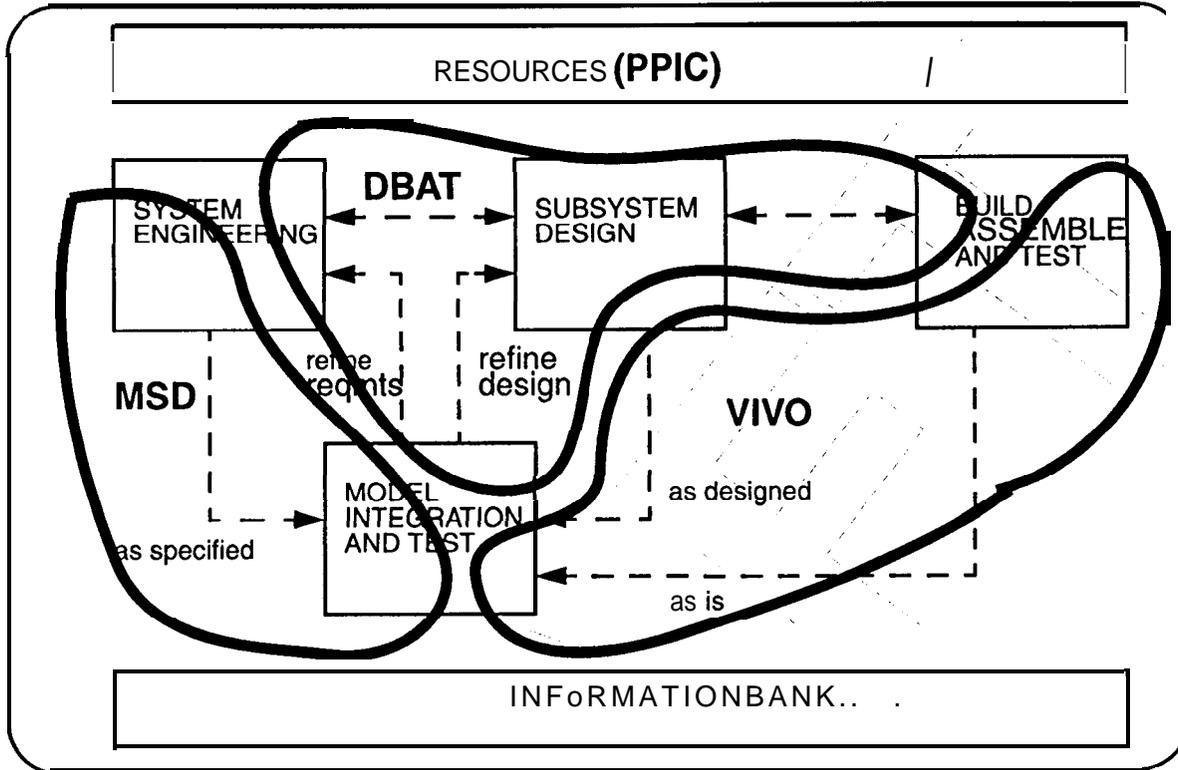


Figure 4: DNP Processes shown with DNP model

5.1 Processes.

A process is a set of interrelated activities and supporting resources transform inputs into outputs. The high level processes in the DNP model include, Systems Engineering, Subsystem Design, Build/Assemble/Test, and Model Integration and Test. Two of the high level processes contain process elements within them. Elements are groups of subprocesses with a similar goal. Three elements are found within Systems Engineering: Mission Feasibility, Mission Definition, and System Definition. Preliminary Design and Final Design are elements of Subsystem Design. Model Integration and Test and Build, Assemble, and Test have no elements grouped beneath them. They have only a number of subprocesses. The hierarchy of high level processes, elements, and subprocesses is shown in Figure 5.

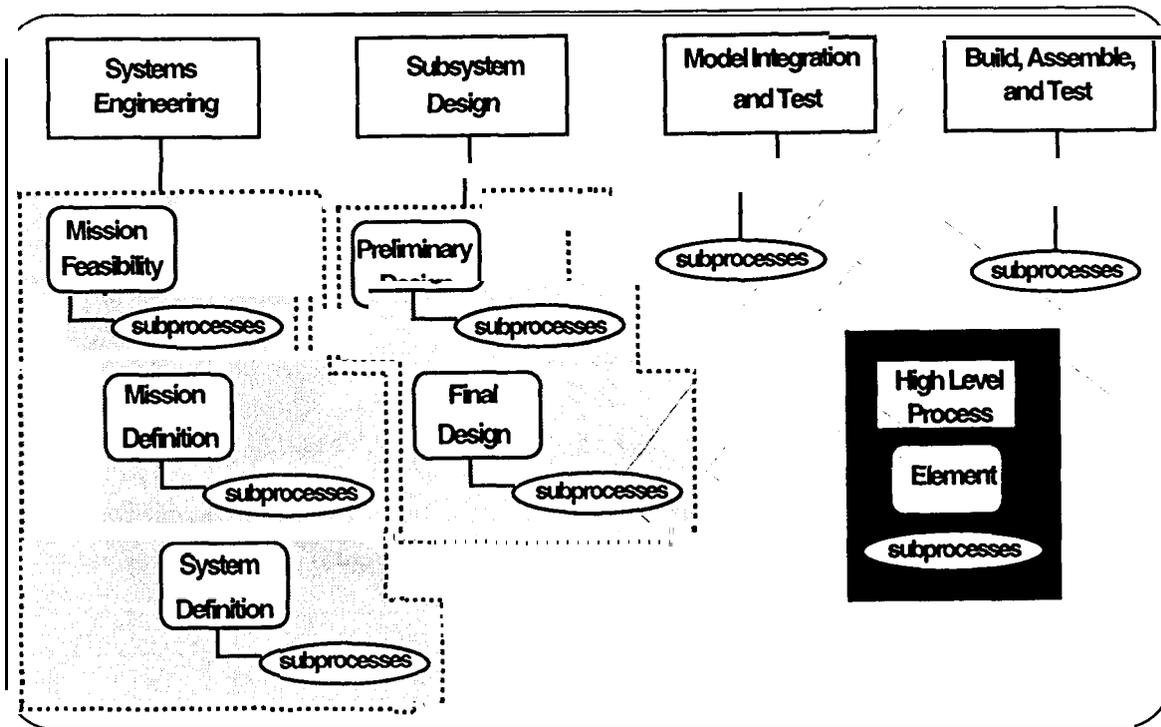


Figure 5: Process Hierarchy

The following section will explain in detail the parameters assigned to each subprocess in the model. Parameters describe and define the work being done in the subprocess. The information listed in the subprocess description tables is explained in Table 1.

Table 1: Explanation of Terms in Subprocess Tables

Subprocess	A specific subprocess within the main processor element
Priority	The relative importance of a subprocess; workforce is provided first to high priority subprocesses
Min	The minimum number of personnel needed to complete a subprocess; process will not begin until minimum is acquired
Max.	The maximum number of personnel to complete a subprocess; anymore personnel would be superfluous
Duration	Duration of subprocess; total estimated time of main process divided by number of subprocesses
Concept	Percentage of subprocess work requiring a "guru"
Detail	Percentage of subprocess work requiring a generalist
Grunt	Percentage of subprocess work requiring a specialist

The priority parameter is used to distribute the workforce among the subprocesses. Priority 1 is the highest and priority 5 is the lowest. No restriction was placed on the priority of the processes, therefore subprocesses of the same priority could be found in the same element or high level process. Higher priority

subprocesses will receive personnel allocations first, and lower priority subprocesses have to wait until personnel are available before they can begin.

The parameters “min” and “max” describe the workforce needed to perform the subprocess. The duration refers to the time, in days, it would take to perform the subprocess. Workforce and duration parameters are those for the simplest-case project. These parameters were multiplied by a complexity factor in order to scale the model for different projects. This complexity factor is described in Section 5.2.4.1.

The parameters “concept”, “detail”, and “grunt” refer to the type of work performed by the subprocesses. A subprocess typically performs a percentage of each type of work. These work types were created to model a specialized workforce in spacecraft development, where different types of engineers are needed for different types of work. Three types of engineers were defined in the model to perform the three aforementioned types of work. If a specific engineer was not available for the type of work needed, the process had to wait until engineer became available.

Concept work must be accomplished by a “guru” - a person with considerable experience and knowledge contributes significant insight to a project. Concept level work is typically done during the planning, definition, and preliminary design stages of a project. An example of concept level work is the development of mission requirements.

In the early stages of project development, the foresight of a guru can help make trades and assign margins will prevent major problems later. Concept level work must be done by a guru, and a guru cannot be replaced by any number of generalists or specialists. JPL has actually identified a dozen or so gurus in the laboratory who are largely responsible for the success of many flight projects.

Detail level work is accomplished by a “generalist” - a person with experience on a particular process. A generalist maybe competent and versed in several spacecraft subsystems, and design methods, but may not have the years of experience or broad authority a guru has. Unlike gurus, generalists comprise the majority of the workforce. Detail level work is done at all stages of the project and is varied in purpose. Detailed design work includes creating interface control documents or tracking engineering change requests. Generalists cannot be replaced by any number of specialists or gurus.

Grunt work is accomplished by a “specialist” - a person with a specific talent or responsibility. A specialist might be a CAD programmer, a Foresight modeler, or a thermal blanket seamstress. “Grunt” work is specialized work such as programming system models or manufacturing components. Specialists typically have focused skills gurus and generalists do not have. Therefore grunt work cannot be performed by a guru or generalist.

The following sections describe the high level processes, elements, and subprocesses in the model. For more information on the subprocess definitions, see the NASA Systems Engineering Process for Programs and Projects (JSC 49040).

5.1.1 Systems Engineering

Systems engineering is a technical process and a management process must be performed throughout the life cycle of a project. System engineering is most active during the mission and system definition phases of a project. During this time, systems engineers define the requirements for the system.

During preliminary and final design, system engineers resolve interface problems, do trade-off analyzes, and assist in verifying performance. During production, system engineering verifies system capability and maintains the system baseline.

This experiment models the mission feasibility, mission definition, and system definition processes of system engineering. Systems engineering processes take place during preliminary and final design and production are not independently modeled, but their functions are included in the design and production phase models.

5.1.1.1 Mission Feasibility

Mission Feasibility is the process of understanding the science problem and identifying mission solutions. The primary product is an assessment of how difficult or how costly it is to achieve goals within mission constraints. In Mission Feasibility, engineers avoid committing to a particular approach too early

because of the changing nature of science objectives. By refining mission objectives and constraints Mission Feasibility lays the groundwork for understanding what is needed in a mission. Table 2 summarizes the subprocesses and parameters within Mission Feasibility.

Table 2: Mission Feasibility Subprocesses

Subprocess	Priority	Min	Max	Duration	Concept	Detail	Grunt
Refine user needs and objectives	3	3	7	3	0.20	0.40	0.40
Refine constraints and assumptions	3	3	7	3	0.20	0.40	0.40
Develop top level mission requirements	1	3	7	3	0.20	0.40	0.40
Develop top level functional mission concept	2	3	7	3	0.20	0.40	0.40
Develop evaluational criteria	3	3	7	3	0.20	0.40	0.40
Flowdown top level system requirements	2	3	7	3	0.20	0.40	0.40
Develop feasible system concept	2	3	7	3	0.20	0.40	0.40
Allocate requirements	1	3	7	3	0.20	0.40	0.40
Analyze and evaluate	3	3	7	3	0.10	0.40	0.50
Synthesize & downselect	3	3	7	3	0.10	0.40	0.50
	1=high			“(days)	%	%	%

5.1.1.2 Mission Definition

Mission Definition identifies the technologies and procedures needed to accomplish science objectives. The focus is on optimizing the mission while still exploring alternative concepts. Primary products are mission_ and system requirements and top-level system architecture. The first steps of Mission Definition focus on mission functions, later, steps concentrate on mission scenarios, aborts and contingencies, and mission preparations. Table 3 summarizes Mission Definition.

Table 3: Mission Definition Subprocesses

Subprocess	Priority	Min	Max	Duration	Concept	Detail	Grunt
Refine top level mission requirements	2	5	10	3	0.20	0.40	0.40
Refine mission concepts and operations	3	5	10	5	0.20	0.40	0.40
Develop and refine evaluation criteria	5	5	10	5	0.20	0.40	0.40
Define and refine system requirements	1	5	10	5	0.20	T	
Develop alternative system architecture and concepts	5	5	10	5	0.20	0.40	0.40
Allocate requirements to segments and elements	3	5	10	3	0.20	T	
Analyze and evaluate architecture and concepts	3	5	10	5	0.10	0.40	0.50
Synthesize & downselect	3	5	10	5	0.50	0.50	0.10
	1=high			(days)	%	%	%

5.1.1.3 System Definition

System Definition optimizes system elements to be acquired or developed. Risk mitigation efforts, such as early development of critical technology or procurement of long lead items, are started in System Definition. Program metrics are refined to estimate cost and schedule. Management preparations, such as preparation of engineering plans to support program control, are made for the Preliminary Design stage. System models are generated in the System Definition process and refined throughout the remainder of the life cycle. Table 4 summarizes System Definition.

Table 4: System Definition Subprocesses

Subprocess	Priority	Min	Max	Duration	Concept	Detail	Grunt
Mission and " - " requirements analysis	3	5	10	5	0.20	0.40	0.40
Develop system evaluational criteria	3	5	10	T		0.40	0.40
Generate models	1	3	7	5	0.10	0.20	0.70
Flowdown and refine " requirements "	1	5	10	5	0.20	0.40	0.40
	1=high			(days)	%	%	%

Table 4: System Definition Subprocesses

Subprocess	Priority	Min	Max	Duration	Concept	Detail	Grunt
Develop and refine prime item concepts	3	5	10	5	0.20	0.40	0.40
Refine models	3	5	10	5	0.10	0.20	0.70
Allocate requirements to end items	2	5	10	3	0.20	0.40	0.40
Evaluate and analyze system end item concepts	5	5	10	5	0.20	0.40	0.40
Select and synthesize optimal option	3	5	10	3	0.20	0.40	0.40
	1=high			(days)	%	%	%

5.1.2 Subsystem Design

Subsystem Design is the detailed design of the spacecraft system in which individual subsystems are developed. It includes the elements of preliminary design and final design. Typical spacecraft subsystems include propulsion, structure, attitude control, thermal, and power. Subsystem design typical includes the fabrication of engineering models and/or breadboards for verification testing prior to the Critical Design Review.

5.1.2.1 Preliminary Design

Preliminary Design produces a functionally complete design meets the mission requirements. The system is defined through implementation and interfaces among all subsystems are established. Engineering models may be developed and tested to optimize the design. Throughout Preliminary Design, integration efforts by systems engineering maintain the cohesiveness of the system. Table 5 summarizes Preliminary Design.

Table 5; Preliminary Design Subprocesses

Subprocess	Priority	Min	Max	Duration	Concept	Detail	Grunt
Analyze and refine requirements	1	10	20	3	0.10	0.45	0.45
Perform design analyses	2	10	20	10	0.33	0.33	0.33
Perform engineering development tests	3	10	20	10	0.00	0.40	0.60
Define interfaces	3	10	20	10	0.10	0.90	0.00
Perform preliminary design	1	10	20	20	0.33	0.33	0.33
Evaluate, verify, and validate design	2	10	20	10	0.33	0.33	0.33
	1=high			(days)	%	%	%

Table 5: Preliminary Design Subprocesses

Subprocess	Priority	Min	Max	Duration	Concept	Detail	Grunt
Complete plans and documents for qualification items	4	10	20	10	0.00	0.60	0.40
	1=high			(days)	%	%	%

5.1.2.2 Final Design

The primary product of Final Design is a cost effective design is ready to build, integrate, and test. Engineering models and qualification items are assembled and tested. Detailed test and verification plans are prepared. Table 6 summarizes Final Design.

Table 6: Final Design Subprocesses ,"

Subprocess	Priority	Min	Max	Duration	Concept	Detail	Grunt
Define and control detailed interfaces	3	10	20	40	0.10	0.80	0.10
Perform detailed design	1	10	20	40	0.33	0.33	0.33
Perform engineering tests	3	10	20	40	0.00	0.40	0.60
Fabricate and test qualification items	2	10	20	40	0.00	0.40	0.60
Evaluate, verify and validate design	2	10	20	40	0.33	0.33	0.33
Update models	2	3	7	5	0.10	0.20	0.70
Complete detail design and production plans	4	2	5	40	0.00	0.60	0.40
Refine models	2	3	7	5	0.10	0.20	0.70
	1=high			(days)	%	%	%

5.1.3 Model Integration and Test

Model Integration and Test (MIT) is a process developed by DNP and represents the model based design initiative. MIT is responsible for building system models, refining them to meet new requirements or specifications, iterating models to increase fidelity, and finalizing models for use during spacecraft operations. Table 7 summarizes model integration and test.

Table 7: Model Integration and Test Subprocesses

Subprocess	Priority	Min	Max	Duration	Concept	Detail	Grunt
Build model	1	1	3	1 1 5	0.00	0.10	0.90
Refine model	3	1	3	1 5	0.00	0.10	0.90
Finalize model	3	1	3	5	0.00	0.10	0.90
Iterate model	3	1	3	5	0.00	0.10	0.90
	1=high			(days)	%	%	%

5.1.4 Build, Assemble, and Test

The Build, Assemble, and Test process provides for the fabrication, assembly, verification, and validation of hardware and software. This process takes a design, developed in the Subsystem Design process, procures or builds the hardware and software, performs verification testing and delivers the final products for integration, system test, and operations. Table 8 summarizes Build, Assemble, and Test.

Table 8: Build, Assemble, and Test Subprocesses

Subprocess	Priority	Min	Max	Duration	Concept	Detail	Grunt
Ready production facilities	4	7	10	5	0.00	0.50	0.50
Fabricate and assemble end item	3	2	5	20	0.00	0.10	0.90
Complete end item , verification test prep ,	4	2	5	10	0.00	0.40	0.60
Complete plans and documents for end item	4	7	10	10	0.00	0.60	0.40
Test & verify end item	2	7	10	20	0.10	0.40	0.50
Assemble and integrate system	2	7	10	20	0.00	0.20	0.80
Complete test plans& documents for system	4	2	5	10	0.00	0.60	0.40
Complete plans and documents for system	4	2	5	10	0.00	0.60	0.40
Test and verify system ,	2	7	10	20	0.00	0.40	0.60
Refine model	3	3	7	5	0.10	0.20	0.70
Perform acceptance testing	3	7	10	20	0.00	0.20	0.80
	1=high			(days)	%	%	%

5.2 Reviews

Reviews were included in the model in order to comprehensively model the design process. A review is an extended process may span days or months depending on the stage of the project, the complexity of the system, and the readiness at the time of the review. Eight major reviews defined in the NASA Program/Project Life Cycle Process Flow are included in the model.

The Mission Concept Review (MCR) is an internal review typically occurs near the end of the Mission Feasibility process. The purpose of the MCR is to understand the science objectives, and examine the proposed missions concepts. The objectives of the review are to demonstrate mission objectives are understandable, demonstrate technical and programmatic feasibility of meeting the mission objectives, and ensure evaluation criteria are ready for mission analysis. A MCR is successful when the proposed mission has sufficient quality and merit to warrant further study in the Mission and System Design process.

The Mission Requirements Review (MRR) occurs following the identification of mission requirements in the Mission Definition process. Its purpose is to substantiate requirements and assess their readiness for external review. The objectives of the review are to confirm the mission concept satisfies the user needs and to confirm mission requirements support external support requirements. After successful completion of the MRR, the program submits the Mission Need Statement for customer approval.

The Mission Definition Review (MDR) occurs near the end of the Mission Definition process. The purpose of the MDR is to examine the system functional requirements and assure they will satisfy the mission. The objectives of the review are to establish the allocation of the functional system requirements is optimal, validate system requirements meet mission objectives, and identify technology risks and the plans for risk mitigation. A successful MDR leads to further development of the system design.

The System Requirements Review (SRR) occurs following the formation of the project team and evaluates their understanding of the mission and system. The objectives of the review are to confirm the system requirements meet the mission objectives and confirm the system specifications are sufficient to meet the project objectives. Successful completion of the SRR freezes project requirements.

The System Definition Review (SDR) occurs at the end of the system definition process. The purpose of the SDR is to examine the system architecture and requirements flowdown. The objectives of the SDR are to demonstrate the architecture is acceptable, requirements allocation is complete, and the system can be built within the constraints. After successful completion of the SDR, the project begins the design and acquisition of the end items.

The Preliminary Design Review (PDR) occurs after the Preliminary Design process and its purpose is to demonstrate the preliminary design meets all system requirements with acceptable risk. The PDR shows the optimal design option has been selected and all interfaces identified. The objectives of the PDR are to ensure all system requirements are complete and allocated, show the proposed design solution is expected to meet the functional and performance requirements. show the design is verifiable and poses no major problems to schedule or cost, and to support the next element of the Subsystem Design process. After successful completion of the review, engineering drawings, specifications, and interface control documents are approved.

The Critical Design Review (CDR) is held at the end of the Final Design process and its purpose is to present the complete system design and demonstrate technical problems have been resolved without compromising performance, reliability and safety. The CDR ensures the design maturity supports the onset of fabrication, verification and integration of hardware and software.

The Production Readiness Review (PRR) occurs prior to the start of manufacturing. The purpose of the PRR is to ensure production plans, facilities, and personnel are in place and ready to begin production. The objectives of the review are to demonstrate all engineering problems encountered during development are resolved, ensure the design documentation is adequate to support manufacturing, and ensure manufacturing plans and preparation are adequate to begin production. A successful PRR results in certification of production readiness by management.

Table 9: Major Reviews

Acronym	Review	Priority	Min	Max	Duration
MCR	Mission Concept Review	2	5/5	7/10	5
MRR	Mission Requirements Review	2	5/5	7/10	5
MDR	Mission Definition Review	2	5/5	7/10	5
SRR	System Requirements Review	2	5/5	7/10	5
SDR	System Definition Review	2	5/5	7/10	5
PDR	Preliminary Design Review	2	5/5	7/10	5
CDR	Critical Design Review	2	5/5	7/10	5
PRR	Production Readiness Review	2	5/5	7/10	5
		1=high	guru/ generalist	guru/ generalist	(days)

5.3 Model Logic

The following section explains the programming logic behind the DNP model. The logic was developed using Foresight software constructs and guidelines (see Appendix A). The section is divided into four parts, 1) inputs and outputs, 2) resources, 3) problem occurrence, and 4) formulas and assumptions. For more information on the logic of the Foresight modeling language, see the Foresight Users Guide.

5.3.1 Inputs and Outputs

The model accepts seven inputs in order to calculate complexity, cost, and available knowledge for each simulation. Some of the seven inputs are used for two purposes, i.e. to calculate cost and complexity. These inputs are entered by the user via the graphical interface shown in Figure 6. This graphical user interface was programmed using Altia Design software, a companion product to Foresight (see Appendix A).

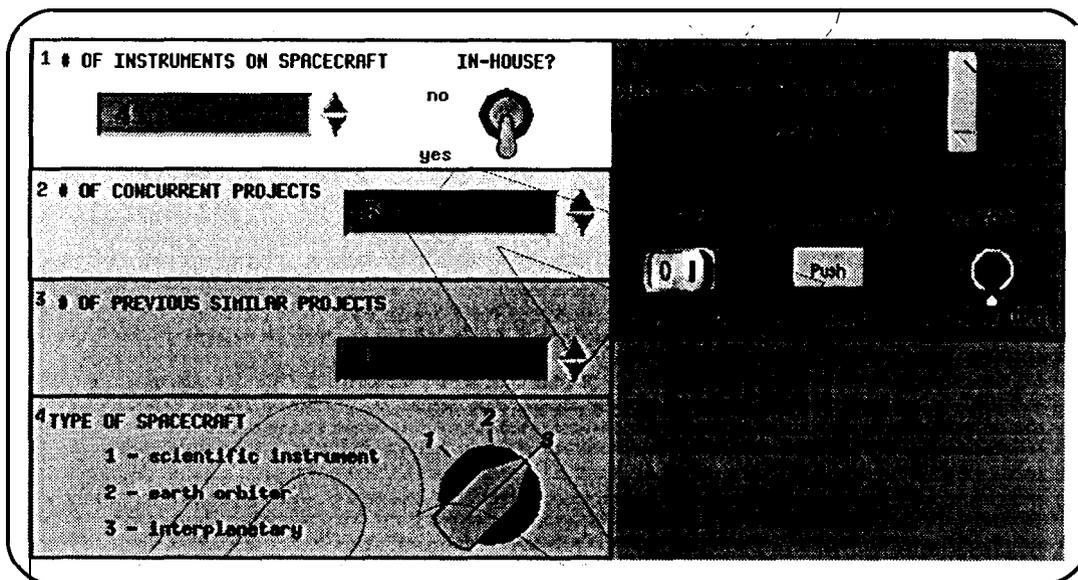


Figure 6: Model 'Input Screen

Inputs 1 -4 are used to calculate spacecraft complexity. The user inputs the number of scientific instruments on the spacecraft in box 1. Instruments are individual assemblies designed to carry out a specific scientific experiment. Whether to counts suite of instruments as one or many is left to the user. But it is suggested the user think in terms of interfaces. A suite of instruments in the same assembly with one interface to the bus can be considered a single instrument. A suite of instruments with multiple interfaces to the bus should be considered as multiple instruments.

The user also indicates in box 1 whether the project was performed mostly in-house (at JPL) or mostly out-of-house. An in-house project is one in which JPL manages and designs the spacecraft. Cassini is an example of an in-house project. Out-of-house projects are typically situations where JPL maintains management of the project, but the spacecraft is designed and manufactured by another NASA center or industrial partner. An example of an out-of-house project is Stardust, where JPL maintains project management, but Lockheed Martin Astronautics builds the spacecraft.

The user indicates in boxes 2 and 3 the number of concurrent and previous projects. Concurrent projects are those with a similar mission being developed at the same time as the modeled project. The definition of "similar mission" is left to the user. The definition is flexible because concurrent projects are modeled to be a source of modifiable knowledge only (see section 5.2.4.4). It is suggested, however, the

user think in terms of shared resources. For example, although the **Cassini** and **Galileo** missions were to different planets (Saturn and Jupiter respectively) their missions were similar in that they both had to deal with radiation at Jupiter. Therefore, the two missions could be considered concurrent because they could share the knowledge of radiation protection at Jupiter.

Previous projects, however, are subject to more stringent similarity criteria because they are the source of reusable knowledge. A previously developed project should have many similarities to the modeled project in order to be considered. An example would be the **Voyager II** spacecraft. In modeling this case, **Voyager 1** would be considered a previous project because it had a similar mission and bus and similar requirements.

The user selects a spacecraft type in box 4. Three choices are given: **scientific instrument**, **earth orbiter**, and **interplanetary spacecraft**. A scientific instrument is an instrument developed to be part of the payload of a spacecraft. An Earth orbiter is one follows an orbit about the **Earth** only. An Interplanetary spacecraft might be a orbiter, a probe, or a lander. Near-earth object and solar missions are considered to be interplanetary missions in the model.

Boxes 4 and 5 are used to determine spacecraft cost. The user inputs the dry mass (in **kg**) of the spacecraft in box 5. There is no limit on dry mass, but the model is not accurate, at dry masses of over 9,000 kg. Box 6 is used to assign a particular type of knowledge scenario to the simulation. The scenarios are explained in section 5.2.4.4.

The graphical user interface also includes a screen for model outputs and monitors. The screen, shown in Figure 7, was also created using **Altia Design** software. The estimated cost is the only output visible at the start of the simulation. The other outputs are monitored throughout the simulation and final values are shown at the end.

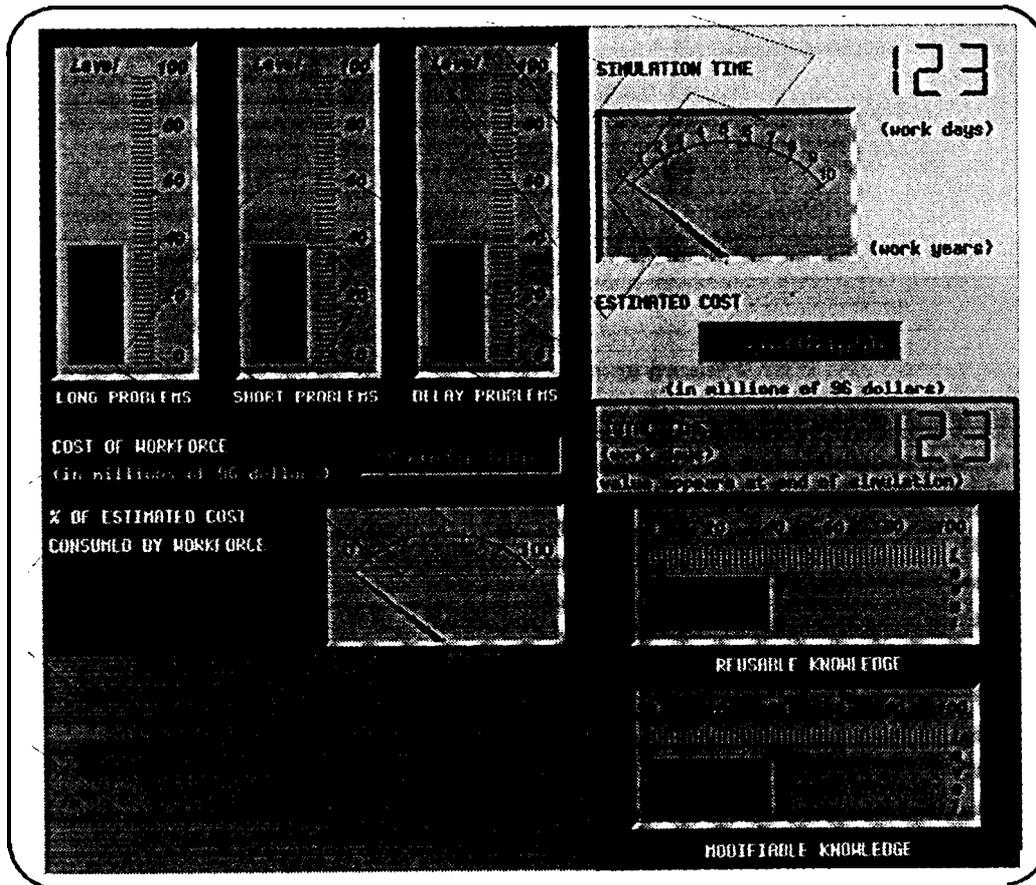


Figure 7: Model Output and Monitor Screen

The left hand column of the output screen shows three system monitors. The first block shows the number of problems, grouped by type, waiting to be solved. Problem occurrence is discussed in section 5.2.3. The second block shows the current cost of the **workforce**. This value starts at zero and is increased during the simulation. At the end of the simulation, the total workforce cost for the project is displayed. The third and bottom block shows the percentage of the estimated cost spent on the **workforce**. This value also starts at zero and is increased throughout the simulation.

The right hand column of the output screen shows two outputs and two monitors. The first block monitors simulation time in days and years. The second block outputs the estimated cost. The third block outputs the total development duration at the end of the simulation. The fourth and bottom block monitors the amount of reusable and modifiable knowledge available to the project.

The input and output screens are helpful for single-run simulations. Batch or Monte Carlo simulations can be run much faster by using a compiled version of the model. This compiled version uses an input file rather than the user interface which must be written prior to beginning the simulation.

5.3.2 Resources

To model constraints on spacecraft development, “resource” elements in Foresight were used. The resource element affects the behavior of a process in a number of ways. For example, a process will not execute if a required resource is being used by another process. Resource usage can be tracked as a function of time to help anticipate the requirements for the designed system. Two types of constraints were modeled as resources were **workforce** and **knowledge**.

5.3.2.1 Workforce

Three types of personnel were modeled: **guru**, **generalist**, and **specialist**. A **guru** was requested for concept level work, a **generalist** for detailed design work, and a **specialist** for “grunt” work. A limited number of engineers were made **available** at the start of the project based on typical **workforce** allocations at JPL, project size, complexity, and percentage of in-house or out-of-house development. Each type of engineer was tracked as a separate resource, e.g. a limited number of **gurus** were available. If a process requested a **guru** and one was not available, the concept level work on process was stalled. Foresight keeps track of workforce requests in an internal queue which **stacks** requests in order of priority. When a **guru** became available, it was allocated to the process with the highest priority requesting it. process would use the **guru** to complete its concept level work and then release the **guru** to work on another process.

5.3.2.2 Knowledge

Two types of knowledge were modeled: reusable and modifiable. A process realized greater time savings by using reusable knowledge, therefore it requested reusable knowledge first. If none were available, it then requested modifiable knowledge. If neither reusable nor modifiable knowledge is available, the process created its products from scratch. Knowledge resources and their use in DNP model are described in Section 5.2.4.4.

5.3.3 Problem Occurrence Logic

Every spacecraft development process will encounter problems. These problems range from major (a major **redirection** by NASA) to minor delays (documents are inaccurate and must be updated). In order to represent the development process well, the DNP model was designed to generate, encounter, and solve problems. The problem occurrence logic was developed by Lynne f? Cooper of JPL.

5.3.3.1 Problem Generation

Four types of problems were generated randomly by the DNP model: major, long loop, short loop, and delay. One major problem was generated each year of the life cycle simulation. Major problems cause a high level process or element to iterate back to the previous high level process or element. Long loop problems cause the model to iterate back to an earlier subprocess. Short loop problems cause the model to iterate back to the previous subprocess. Delays cause the subprocess to repeat itself. The iteration of major, long loop, short loop, and delay problems is illustrated in Figure 8.

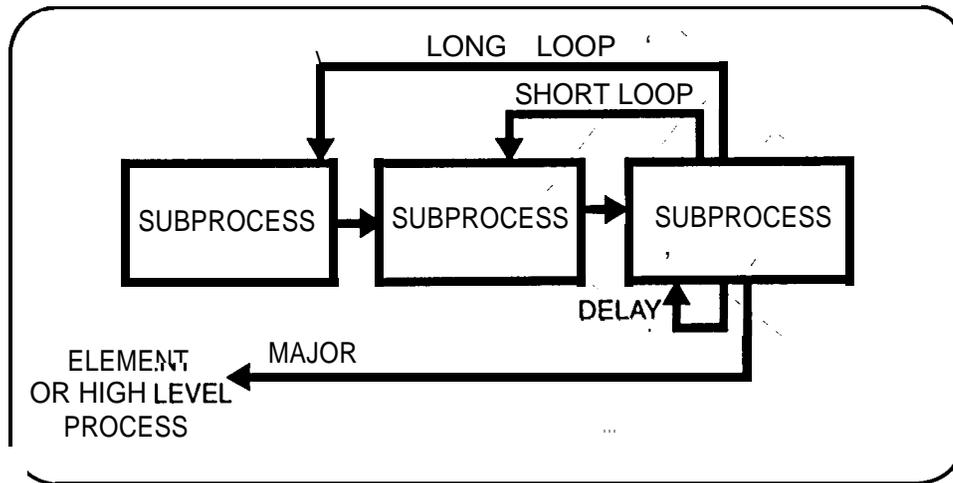


Figure 8: Problem Loops

Three queues of long loop, short loop, and delay problems were generated by the model at the beginning of the simulation. Problems were not generated at any other point in the simulation. Problems were generated randomly based on the ranges given in Table 10. Most of the problems generated by the model were delays. The remainder of generated problems were short loop and long loops, with long loops being less common than short.

Table 10 also shows the range of 'released problems. One of the DNP assumptions (see section 5.2.4) is most problems with a spacecraft design will be discovered early in the development cycle due to model based design practices. To model this assumption, the most problems were released early in the simulation. Once a problem was 'released, it had to be solved by the first subprocess encountered it. A random number of problems between 0 and the problem queue size was released. At the beginning, many problems were released because the queue was at a maximum and thus the upper limit of the range was maximum. Because the queue size decreased as problems were released, the number of problems released decreased with time.

Table 10: Random Problem Generation

Problem Type	Range of Generated Problems (put into queue at start of simulation)	Range of Released Problems (subtracted from queue at random intervals)
long loop	2 - (5 x complexity factor)*	0 - queue size at time of release
short loop	(5 x factor) - (10x factor)	0 - queue size at time of release
delay	(10 x factor) - (15x factor)	0 - queue size at time of release

* see project complexity, Section 5.2.4.1

The problems were released from the queues at random intervals, i.e. problems were not encountered at any set frequency. A subprocess never “knew” when it would encounter a problem. The released problems were placed in a “ready bucket” and were solved by subprocesses (see next section). Each time a problem was released, it was deleted from the problem queue. Therefore, the total number of problems in each queue decreased as time progressed, When a queue emptied, no more problems of type were encountered.

5.3.3.2 Problem Solving

Problems were encountered and solved by subprocesses. Problems were solved by iterating as described in section 5.2.3.1. While a subprocess was solving problems, all other requests for its use were stalled. For example, a subprocess might send out a signal to do a short loop iteration and then begin a series of delays to solve problems. While it was performing the series of delays, the short loop iteration might work its way back to the subprocess and request it to begin again. But the subprocess will wait until it has finished all its delays (and thus solves all its problems) before it takes the request.

5.4 Formulas and Assumptions

Six basic concepts were assumed in the DNP model. They were drawn from DNP methodologies and input from Lynne P. Cooper, Professor Edward F. Crawley, and engineers at JPL. These assumptions are shown in Figure 9.

The primary assumption (1) has already been discussed, The DNP team is assuming through the use of model based design, more errors in the spacecraft design will be discovered early in the development cycle. This assumption is driven by the high cost of fixing mistakes-late, in the development cycle. Model based design is a methodology will allow engineers to better understand the design early on, and thus be able to discover errors.

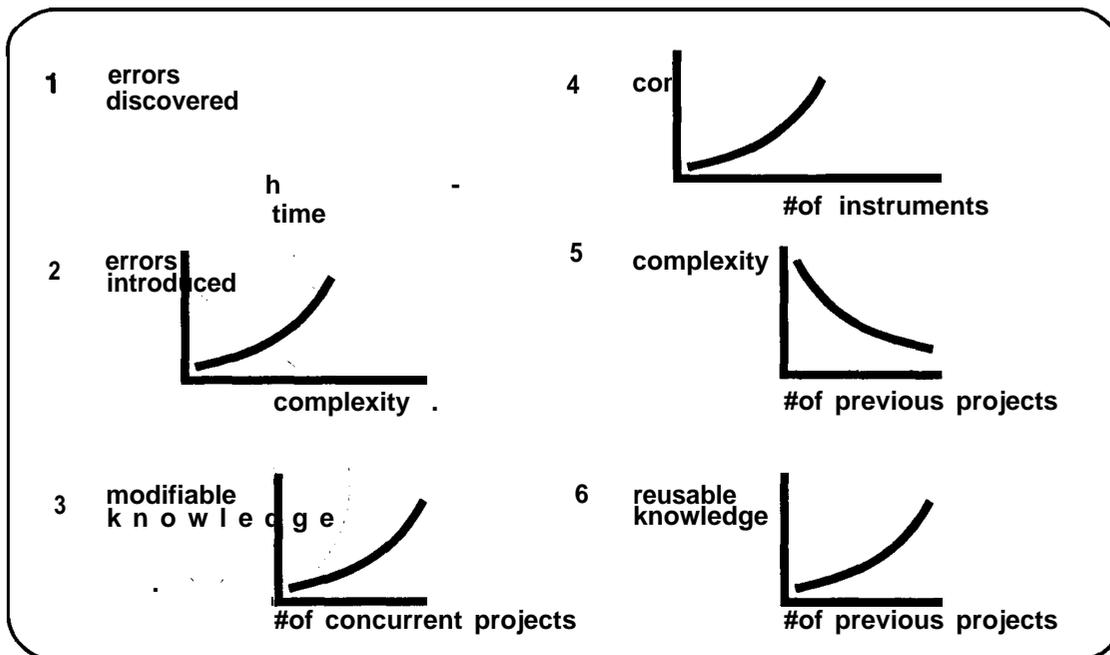


Figure 9: Assumptions in the DNP Model

Assumption 2 states as the complexity of a project increases, the number of errors in the design will increase. This assumption says complex projects are more prone to errors than simple projects.

Assumptions 4 and 5 relate project complexity to the number of scientific instruments onboard and the number of previous projects. The complexity analysis described in section 5.2.4.1 drove assumption 4, which states as the number of instruments increase, the complexity increases. Assumption 5, however, states as the number of previous projects increases, the complexity decreases. It is assumed if a similar project has been developed by JPL previously, the current project will be easier to develop. This is because engineers have a greater understanding of the requirements and design of the spacecraft based on experience with the previous project.

Assumptions 3 and 6 relate previous and concurrent projects to available knowledge. An increase in the number of concurrent projects led to an increase in the amount of available modifiable knowledge. An increase in the number of previous projects led to an increase in the amount of available reusable knowledge. Modifiable knowledge and reusable knowledge are explained in section 5.2.4.4.

5.4.1 Project Complexity

The DNP model can be scaled to a number of different projects. It can be used to study unmanned earth orbiting or interplanetary spacecraft as well as scientific instruments. The model is scaled using a complexity factor derived from a number of parameters. The variables scaled within the DNP model include process duration and the number of problems generated. As complexity increases, the duration and problems increase.

The complexity factor was derived using a 'simple linear analysis of spacecraft parameters vs. cost and development cycle duration. This analysis showed the number of instruments on a spacecraft is most directly related to its complexity. The greater the number of instruments, the most complex a project is. The linear analysis spreadsheet is shown in Figure 10.'

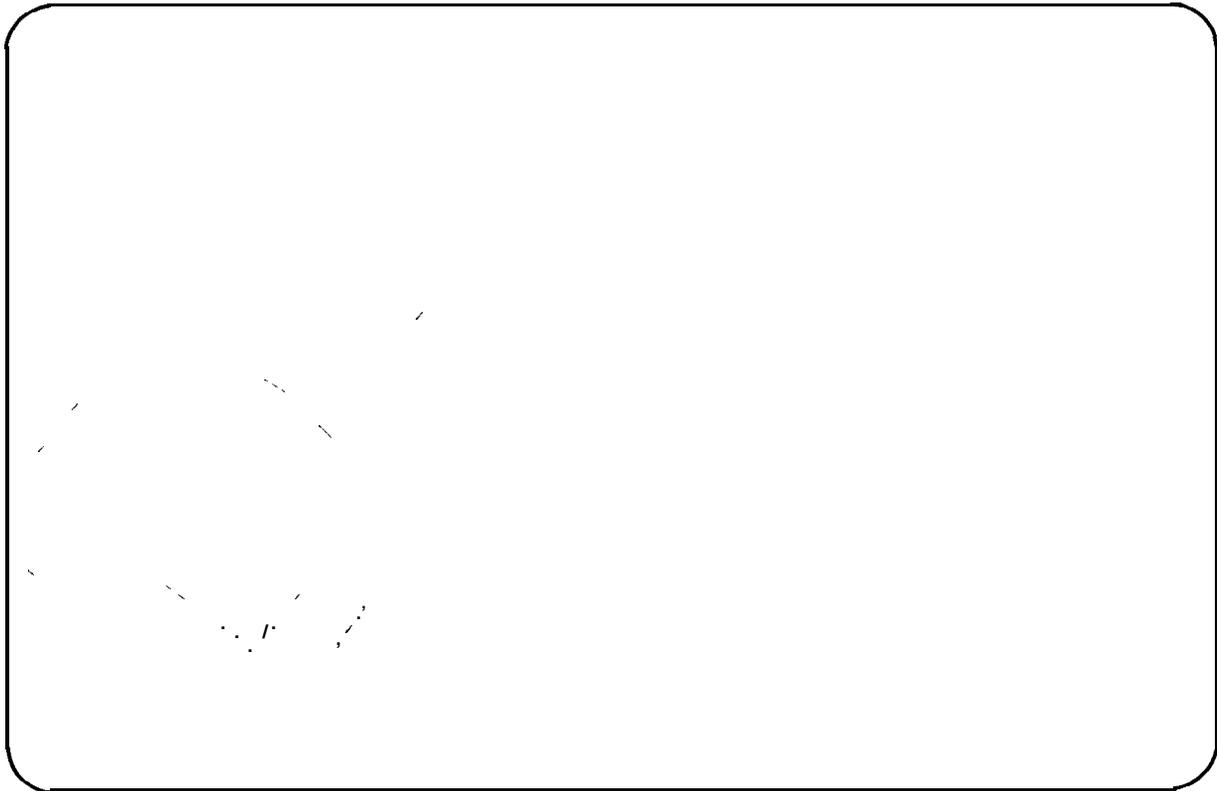


Figure 10: Complexity Parameter Analysis

Two parameters simplify a project are the number of previous projects and the percentage of in-house work. If a similar project has been developed previously, it is assumed knowledge exists will simplify development. In this model, complexity is reduced by a fraction equal to one over the number of previous projects,

If a project is developed mainly outside of JPL, it is simpler. An example of such a project is Ulysses, where the spacecraft and spacecraft operations team were provided by ESA, and only the launch of the spacecraft, radio tracking, and data management operations were provided by JPL. In this model, if a project is mainly developed outside of JPL, the complexity factor is reduced by 1.

The simplifying factors were empirically derived by calibrating the model to three different projects: Pathfinder, Cassini, and SeaWinds. The model was calibrated using development scenarios that did not employ knowledge management practices. Knowledge reuse was not practiced on these projects in the way proposed by DNP. It is the proposed practices which are the focus of this experiment.

The method of calculating the complexity factor is demonstrated in Table 11. A project is assigned a starting complexity factor based on the number of instruments it carries. If the project is developed mostly in-house, the starting factor is unchanged. If it is developed out-of-house, its complexity is reduced by 1. If 1 or zero previous similar projects have been developed; the factor is unchanged. If 2 or more previous projects have been developed, the complexity factor is reduced by a fraction equal to 1 over the number of previous projects. The final factor is rounded to an integer value. The final factor must be greater than or equal to one.

Table 11: Project Complexity Factor

Example Project	# of Instruments	Start Factor	Mainly in house?		# of Similar' Previous Projects	Final Factor
			Yes	No		
					Oorl	2 +
SeaWinds	0-4	1	+0			x 1/#
Pathfinder	5 - 8	2	+0		1	1
Ulysses	9-12	3		-1	1	2
Voyager	13 - 16	4	+0		1	4
Cassini	> 16	5	+0		1	5

- Final factor must be an integer equal or greater than 1 (1 is simplest complexity factor)

5.4.2 Development and, Production Cost

Spacecraft development and production cost was estimated to enable the model to be used for cost related simulations in the future. Cost was not a parameter studied in this experiment. Cost was estimated using the Spacecraft/Vehicle Level Cost Model (SVLCM), which is available in the NASA Parametric Cost Estimating Reference Manual. The SVLCM is shown in Figure 7.

The SVLCM is a simple, on-line cost model provides a useful method for quick, rough-order-of-magnitude cost estimating. The model can be used for estimating the development and production cost of spacecraft, launch vehicle stages, engines and scientific instruments. The SVLCM is a top-level model derived from the NASA Cost Model (NASCOM) database. The model is available through the World Wide Web at <http://www.jsc.nasa.gov/bu2/SVLCM.html>

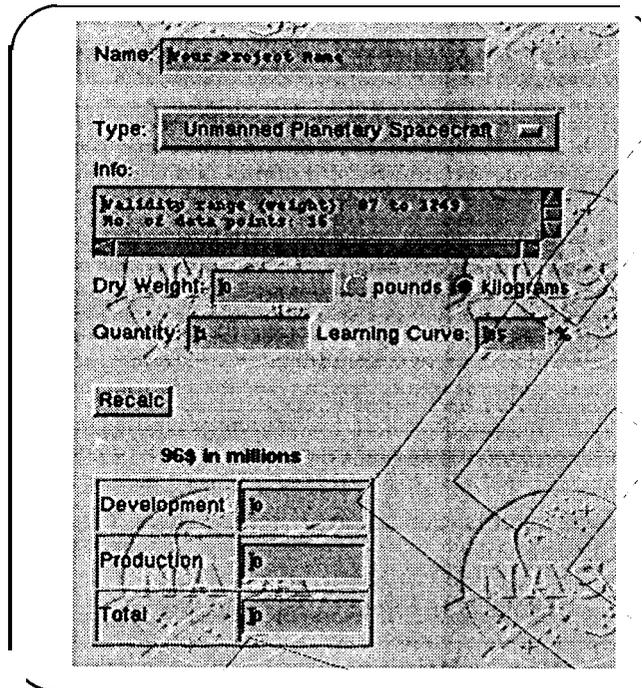


Figure 11: Spacecraft/Vehicle Level Cost Model

5.4.3 Equations for Total Cost

Equations for total cost were derived from the SVLCM. The SVLCM uses dry mass (X) to calculate total cost. Data was extracted from the SVLCM using a range of dry masses and then plotted using Matlab. A second-order polynomial curvefit was used to derive the equations from the plotted data. The total cost in this model is the sum-of development and production cost. The cost is tallied in millions of 1996 dollars.

The cost curves differed depending on the type of project, The three equations for cost are shown below. Dry mass (in kg) is represented by the variable X in the following three equations.

For Type 3 Projects (Unmanned Interplanetary Spacecraft):

$$\text{(Total Cost (96 dollars))} = 100 \left((-8.95 \text{ E } -7)\chi^2 \right) + (5.93 \text{ E } -3)\chi + i .05$$

For Type 2 Projects (Unmanned Earth Orbiting Spacecraft):

$$\text{Total Cost (96 dollars)} = (-6.46 \text{ E } -6)\chi^2 + (1.17 \text{ E } -1)\chi + 48.4$$

For Type 1 Projects (Scientific Instruments):

$$\text{Total Cost (96 dollars)} = (-6.36 \text{ E } -5)\chi^2 + (1.39 \text{ E } -1)\chi + 6.64$$

5.4.4 Workforce Cost

Workforce cost is the total salary plus overhead paid to engineers working on a project. Although the workforce cost was calculated in this model, it was not one of the parameters studied. It was included for the benefit of future experiments.

A workforce cost formula was derived to calculate the cost of personnel based on yearly salary and overhead. The yearly salary was based on a 250 workday year. The engineer's yearly salary was divided by the number of work days in a year (270) and then multiplied by the number of days the engineer was used by the project (T). Guru engineers made \$110 K a year, generalists made \$75 K a year, and specialists made \$55 K a year. The salaries of the engineers are preliminary values pending further research, as they may not accurately reflect the distribution of salaries at JPL.

The cost per year was summed to produce the total cost of workforce over the life cycle. The burden (or overhead) factor of 2 was based on cost estimating methodology used by line managers at JPL. These managers typically multiply an engineer's salary by 2 (for burden) when creating a fiscal year budget.

A "fudge factor" of 2 was also applied. It was empirically derived from trial runs of the equation and Pathfinder cost calibration data. Workforce cost is calculated in millions of 1996 dollars.

The formula for calculating workforce cost is as follows:

$$\text{Cost} = \left(\frac{\text{\# of personnel requested} \times (\text{fudge factor}) \times (\text{burden factor})}{\text{yearly salary in millions} \times (\text{work days in year}) \times T} \right)$$

5.4.5 Knowledge Availability

Two forms of available knowledge were modeled: reusable and modifiable. Reusable knowledge is in a form can be used again with little modification. Modifiable knowledge is in a form must be altered before reuse. The availability of knowledge affects subprocess duration, such that the more knowledge available, the shorter the process duration. Individual subprocess durations were related to knowledge availability as follows:

- If reusable knowledge available: duration decreased by 3/4
- If modifiable knowledge available: duration decreased by 1/4

Another way of understanding knowledge availability is through time savings. A 75% time savings in process duration is realized by using reusable knowledge. The use of modifiable knowledge leads to a 25% time savings.

Reusable knowledge only saves 75% (rather than 100%) because it is never truly "reusable: i.e. one cannot necessarily photocopy a document and reuse it for another project. It may be considered plagiarism to reuse knowledge in such a way.' Also, despite the similarities between the projects, at the very least one is going to have to change the name from "Project A" to "Project B."

Modifiable knowledge only saves 25% because it is modeled as the "worst case" of modifiable knowledge. Some forms of "modifiable knowledge may save 70%, other 50%, but at the very least modifiable knowledge is assumed to save 25% of process time. Any knowledge saves less than 25% is not modeled and the work is considered to be done from scratch.

Knowledge is stored in a Foresight resource block. A knowledge value of 1 represents one unit of knowledge, such as a document. The resource is reduced each time knowledge is used by a subprocess. This knowledge consumption represents the fact most knowledge is specific and will only be used once during a life cycle. Knowledge is requested and consumed in multiples of 1.

The knowledge resource is increased each time knowledge is created. Reusable knowledge is created in two ways. First, it is created at the start of the life cycle if similar previous projects existed. The amount of reusable knowledge created is equal to the number of previous projects multiplied by a uniformly

distributed random number. Secondly, it is created each time a subprocess repeats itself due to an iteration in the high level process or element in which it occurs. The methods for creating reusable knowledge are summarized below.

- Reusable knowledge = (# of previous projects) x (random number between 1 - 10)
- Reusable knowledge available increased by 1.0 each time subprocess repeats itself

Modifiable knowledge is also created in two ways. First, modifiable knowledge is created at the start of the life cycle if other projects are being developed concurrently. The amount of modifiable knowledge created is equal to the number of concurrent projects multiplied by a uniformly distributed random number. Secondly, it is created each time a subprocess is executed. The methods for created modifiable knowledge are summarized below.

- Modifiable knowledge = (# of concurrent projects) x (random number between 1 - 20)
- Modifiable knowledge available increased by 0.10 each time a subprocess is executed

The methodology for creating and consuming knowledge was developed by Lynne P. Cooper and Esther S. Dutton of JPL and verified by members of the JPL technical staff.,

6.0 Simulations

Simulations were run to collect data for two different analyses: process duration vs. knowledge availability and process duration vs. knowledge scenario. Duration vs. availability data was collected using a simulation with random inputs. Duration vs. scenario “data. was collected using a simulation with six different test cases. These simulation methods were chosen because they produced data best suited for analysis.

6.1 Random Inputs

Project complexity, process duration, and development plus production cost are all based on inputs to the DNP model. In order to obtain duration vs. knowledge availability data for a variety of projects, these inputs were varied randomly. Uniformly distributed random numbers were generated for each of the input parameters. The range of values for each parameter is listed in Table 12.

Table 12: Random Inputs

Parameter	Description	Range
Projects	Number of, concurrent projects under development	0-10
Previous	Number of similar projects developed previously	0-5
Dry Mass	Estimated dry mass in kilograms	10-3000
In House	Is project developed mainly at JPL? 0 = Yes, 1 = No	0-1
Instruments	Number of instruments	1-20
Type	Type of project: 1- Instrument, 2- Earth Orbiter, 3- Planetary	1-3

6.2 Test Cases

One of the objectives of this experiment is to convince project managers through the use of knowledge reuse they can reduce development time. Duration vs. scenario data was collected to explore this claim. Six test cases were simulated using four different knowledge scenarios. The four different scenarios are listed in Table 13.

Table 13: Knowledge Scenarios

Scenario	Description
1: Default	all types of knowledge enabled; products created from reusable and modifiable knowledge and scratch
2: All Reuse	reusable knowledge is always available; products always created from reusable knowledge
3: No Reuse	reusable knowledge is never available; products created from scratch or from modifiable knowledge
4: All Scratch	reusable and modifiable knowledge are never available; products always created from scratch

The default knowledge scenario is also known as the “all types” scenario. In this scenario, projects are allowed to use reusable and modifiable knowledge, as well as create products from scratch. In the “all reuse” scenario, an artificial supply of reusable knowledge is always available to the project, so there is no reason to use modifiable knowledge or create products from scratch: -- ..

In the “no reuse” scenario, the project is not allowed to use reusable knowledge. The project can only create products using modifiable knowledge or from scratch. In the “all scratch” scenario, the project must always create products from scratch.

The “Knowledge Spectrum” shown in Figure 12 illustrates the relationship between available knowledge and subprocess duration. As the arrow moves to the right, the duration decreases. The “all from reusable” scenario decrease the duration the most. The “all from scratch” scenario does not decrease duration at all.

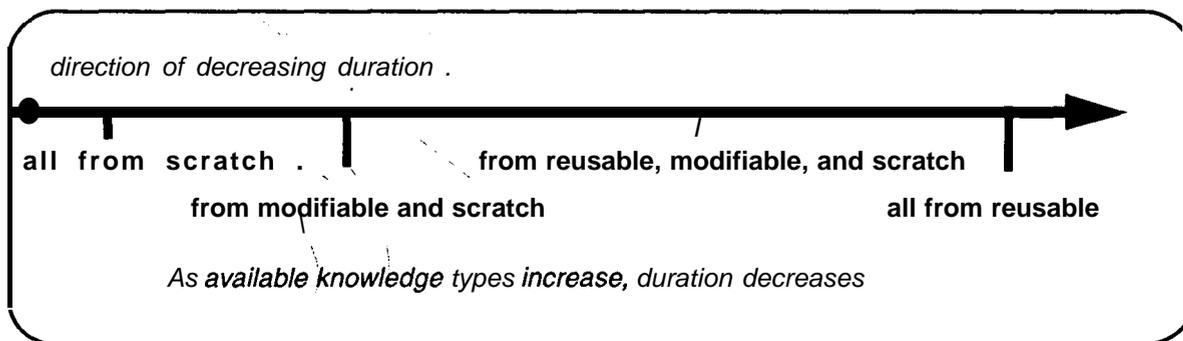


Figure 12: Knowledge Spectrum

The six test cases used in the second simulation are actual JPL flight projects that are either planned, in development, or in operation. Data to support the input parameters list in Table 14 was gathered from JPL documents and by querying engineers familiar with the projects. The test cases represent

all three types of projects (earth, planet, and instrument) as well as varying levels of complexity.

Table 14: Test Cases

Spacecraft	Type	Concurrent Projects	Previous Projects	instruments	Dry Mass	In House?
Cassini	planet	8	1	18	2670	yes
Pathfinder	planet	6	1	4	570	yes
Mars Global Surveyor	planet	6	1	6	674	no
Microspacecraft	planet	4	0	4	12	yes
SeaWinds	earth	1	4	1	200	yes
State of Health Monitor	instru	2	2	,"1'	1	yes
(Project Name)			(s i m i l a r)		(kg)	

7.0 Data Confidence

Great care was taken to ensure the validity of the data gathered in this experiment. Although the Pathfinder project was used to calibrate the model, a number of other projects were used in preliminary simulations to determine the range of the model. Parameters were gathered from JPL experts and rechecked exhaustively. The final results underwent three reviews, two at JPL and one at MIT.

The model relied heavily on the generation random numbers, which is one of the weaker features of the Foresight tool. In order to ensure the numbers were truly random, the model was re-seeded each time it was run. A seed is used by Foresight to generate uniformly distributed random numbers within a given range. The re-seeding, however, did not eliminate the occurrence of identical inputs. Rather than double-plot identical sets of data, these sets were grouped together and plotted only once.

The model was unable to accurately model complex projects. The problem solving iterations performed by the model grew unstable at high complexities. A stop function was incorporated into the model to halt excessively long runs; This was done so that the hundreds of Monte Carlo simulations could be run within a limited time. If any one simulation iterated through a high level process for longer than ten years, it was stopped and the data marked invalid. This happened most notably in the modeling of the Cassini Project. The Cassini simulation would iterate continuously in the Systems Engineering process and never move on to subsequent processes.

The stalling in the Cassini Project can be attributed to the occurrence of major problems. For each year that went by in the simulation, a major problem was encountered that would cause the Systems Engineering process to iterate back to the beginning of the process. Because the Systems Engineering process was lengthy for complex projects such as Cassini, it would accumulate a large queue of major problems by the time it was ready to move onto Subsystem Design. This large queue of problems had to be solved before the life cycle could progress. Therefore, the complex projects were often stalled in Systems Engineering while they attempted to solve an endless queue of major problems. A few highly complex projects were modeled. These are discussed further in Section 9.1

Because of the instability of the model for highly complex projects, the data collected for such projects is not valid. Further development of the model will allow for the modeling of complex projects.

The first simulation was run in Monte- Carlo fashion, and the resulting data showed a uniform distribution. The second simulation was run on a case by case basis. The results collected from the two simulations were checked against subjective knowledge of the JPL flight projects. For the most part, all the results were verified and are discussed further in Section 9.0.

8.0 Data Collection

Test Matrix A, shown in Table 15, outlines the data collected in the duration vs. availability simulations. 1000 simulations were run using randomly generated inputs. The resulting data was grouped into five categories based on complexity factor.

Table 15: Test Matrix A

Factor	Previous Projects	Concurrent Projects	Knowledge' Scenario'	Duration
1	0-5	0-10	1 (default, see Section 5.4.2.1)	?
2	0-5	0 - 10	1	?
3	0-5	0 - 10		?
4	0-5	0-10	1	?
5	0-5	0-10	1	?
(complexity)	(range of values)	(range of values)	(all knowledge avail.)	(days)

Test Matrix B, shown in Table 16, outlines the data collected in the duration vs. scenario simulations. 1000 simulations were run for each of the six test cases.

Table 16: Test Matrix B

Test Case	Knowledge Scenario			
	1	2	3	4
Cassini	(duration?)	(duration?)	(duration?)	(duration?)
Pathfinder	(duration?)	(duration?)	(duration?)	(duration?)
Mars Global Surveyor	(duration?)	(duration?)	(duration?)	(duration?)
Microspacecraft	(duration?)	(duration?)	(duration?)	(duration?)
SeaWinds	(duration?)	(duration?)	(duration?)	(duration?)
State of Health Monitor,	(duration?)	(duration?)	(duration?)	(duration?)

9.0 Results and Discussion

Data was collected from Foresight in the form of tab separated value files and analyzed using Matlab. Curvefits and graphs were generated by Matlab. Results from the two simulations are presented in the following sections.

9.1 Process Duration vs. Knowledge Availability

The first simulation was run to explore how process duration was affected by different types and amounts of available knowledge. A random complexity factor and a random number of concurrent and previous projects were input to each simulation. Concurrent projects represent the availability of modifiable knowledge. Previous projects represent the availability of reusable knowledge.

Both knowledge types produced similar data sets. The results of the first simulation are presented here in two forms in order to illustrate two conclusions. The first simulation, used the default knowledge scenario in which all types of knowledge were available for use.

The first graph (Figure 13) shows total process duration as a function of concurrent projects for four complexity factors. Note the total number of work days in a calendar year is 270. This graph illustrates how the model grows unstable for high complexities. As the factor increases, the data shows greater diversity. Insufficient data was collected for factor 5 projects, so it was not plotted in Figure 12. The data shown represents Monte Carlo simulations run with random inputs.

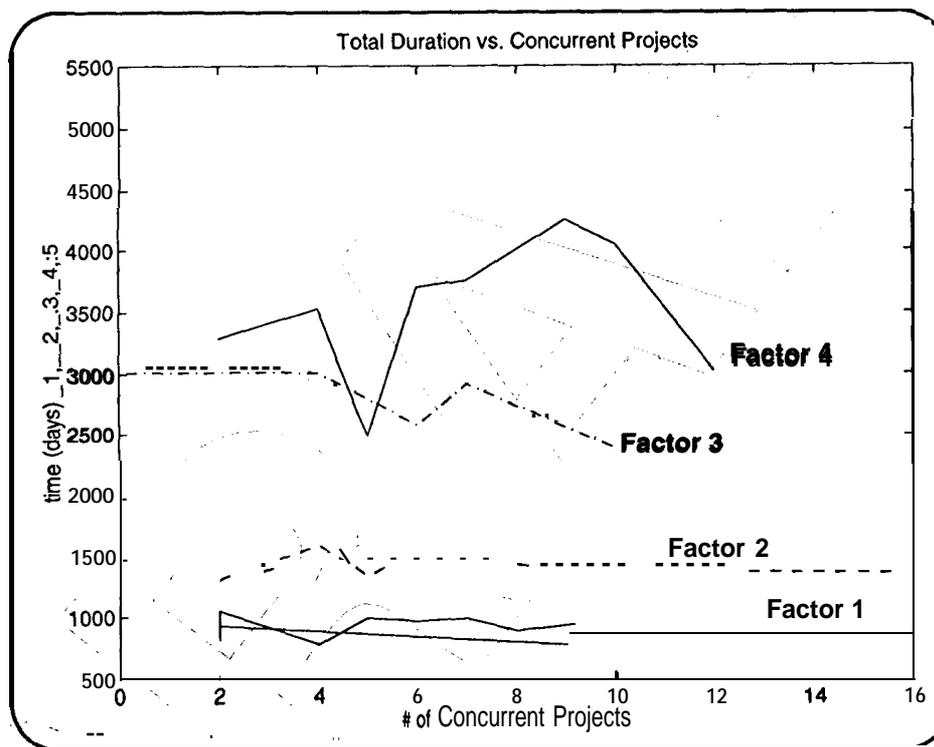


Figure 13: Process' Duration vs. Concurrent Projects (Modifiable Knowledge)

The next graph (Figure 14) shows total process duration as a function of previous projects. The data is similar to the previous graph and again shows instability at the higher complexities. In this graph, however, linear slopes are roughly fitted to the data to show sensitivity. The lower complexities (1 and 2) show little sensitivity to increasing amounts of reusable knowledge (their slopes are nearly zero). The higher complexities, however, have more prominent slopes, and thus, more sensitivity to available knowledge.

The points seen on this graph represent sets of data. Although Foresight is a powerful modeling tool, it lacks a good random number generator. Because of this, many runs produced identical input parameters and hence, identical sets of data. Thus, rather than graph multiple identical data points, the data was grouped into sets and then plotted.

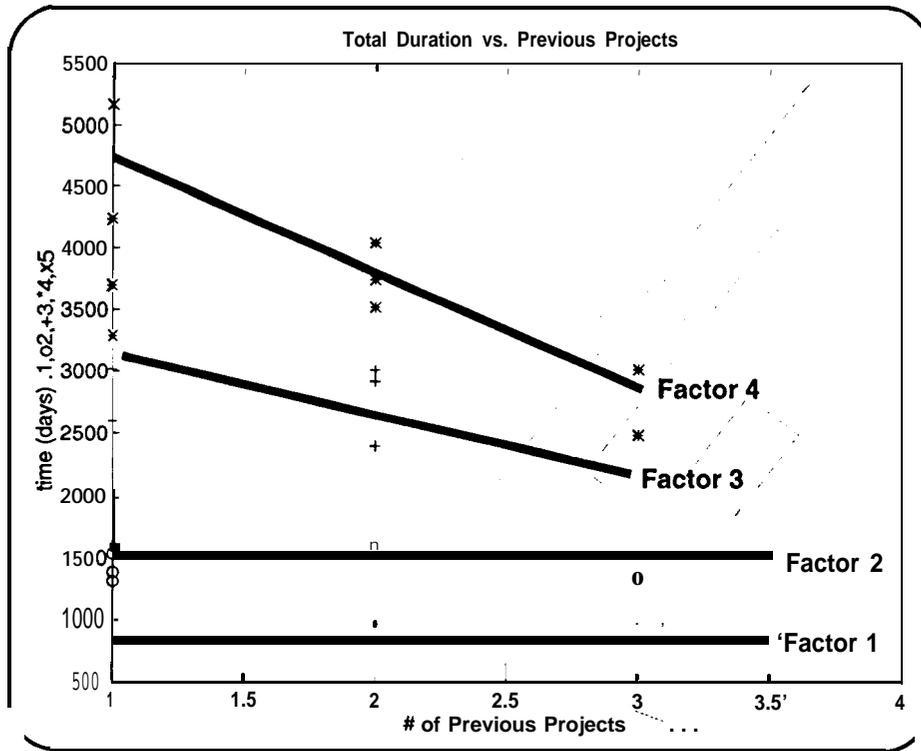


Figure 14: Process Duration vs. Previous Projects (Reusable Knowledge)

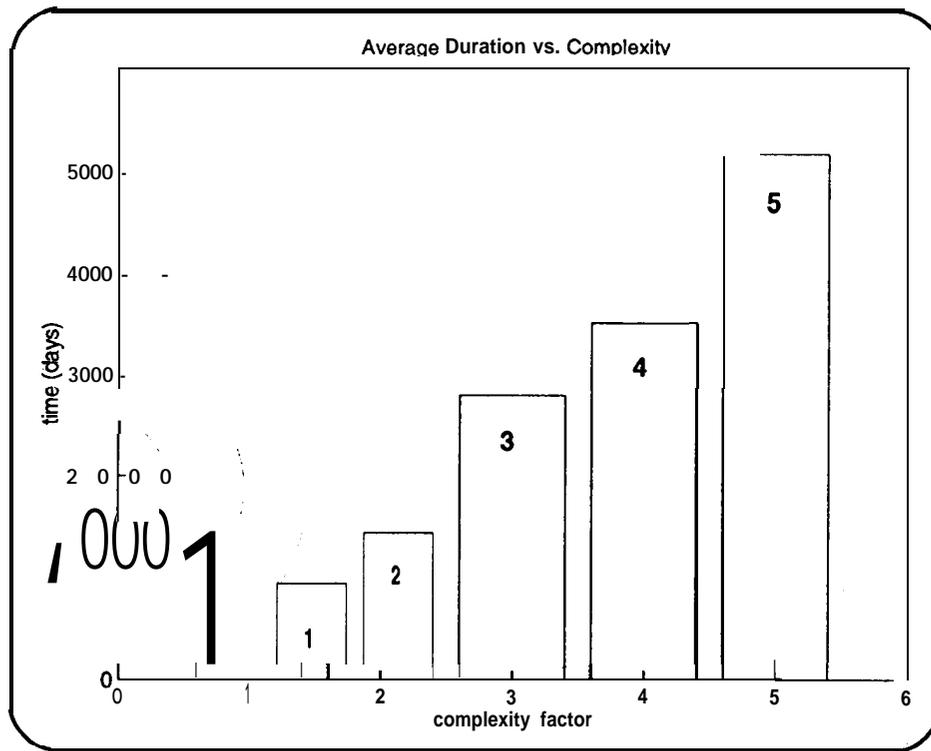


Figure 15: Average Process Duration vs. Complexity Factor

The final graph (Figure 15) shows the average value of the Monte Carlo simulations. You will notice the high value for factor 5 complexities. They average time was over 15 years for the completion of a project. Likewise, the average value for factor 1 projects was also high, at over 3 years: JPL expects factor 1 projects to take no longer than 18 months and factor 5 to take no longer than 5 years. So the model did not produce promising data in this area.

However, this graph does show the complexity factor is not a linear relationship with process duration. The results appear to show an almost exponential rise in duration **with complexity**. This exponential rise could be attributed to the susceptibility of complex projects to major problems. Because a major problem is designed to take place every year, the longer a project drags on, the longer it will have to deal with setbacks. The compounding of these setbacks causes complex projects to take even longer than would be linearly projected.

It must be noted here the factor 5 results were generated **manually**, by trial and error of parameters. Left to run automatically, the random inputs failed to produce a factor 5 project could complete systems engineering process (the first step) in less than 10 years, Exhaustive manual manipulation of the input parameters finally produced twelve data points for factor 5 projects. So whereas the other four complexities represent data averaged over hundreds of data points, the factor 5 projects were only averaged over 12.

9.2 Process Duration vs. Knowledge Scenario

The second simulation was run to **test** a number of projects under different knowledge scenarios. The inputs were based on five cases, each representing a real or planned flight Project. Each case was subjected to four different knowledge scenarios. **Although these simulations were run in a Monte Carlo fashion, the data returned showed vary little variation.** Again, this could be due to Foresight's inability to produce truly random values.

A sixth test case, for the Cassini Spacecraft, was also run. "Because of the complexity of the project, however, the model never produced a final duration for Cassini. During simulation, Cassini would stall, iterating in systems engineering for over ten years, never progressing to subsystem design. The model was automatically cut off in such cases.

Each of the following **five graphs** shows the **results** from the second simulation. Process duration is shown for the four different scenarios. The 'actual duration the real project took is also shown. Duration values are clearly quantified to the **right** of the graph.

Figure 16 shows the data for the Pathfinder Project, which was the calibration case. Pathfinder was chosen as **the calibration** project because it was the first project considered to fall into the "faster, better, cheaper" category **by JPL**. Although Pathfinder was completed in 3 years for less than 171 million, it required heroic efforts **by the** engineers to meet those time and cost constraints. One of the inherent goals for DNP is to reengineer the development process so it requires no more heroic engineering efforts.

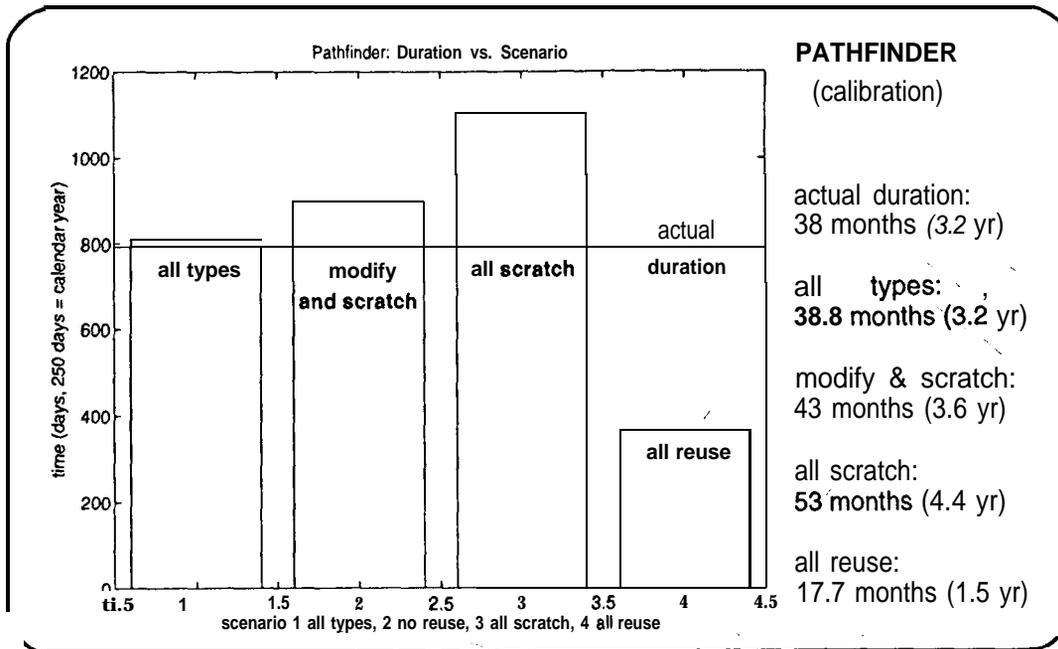


Figure 16: Case One: Pathfinder Project (Calibration)

The Pathfinder graph shows a close match between the duration for the “all types” knowledge scenario and the actual duration. This, however, was expected. The model was calibrated by changing internal duration parameters until those two values matched.

The Pathfinder graph shows the duration is longest for the “all scratch” scenario which is expected. If a project has to do all its engineering work from scratch, it is likely to take longer than if it had knowledge available for reuse. The fastest duration is seen during the “all reuse” scenario. This scenario’s duration is the same for many of the test cases. It represents the minimum duration the model produces for any flight project - 18 months.

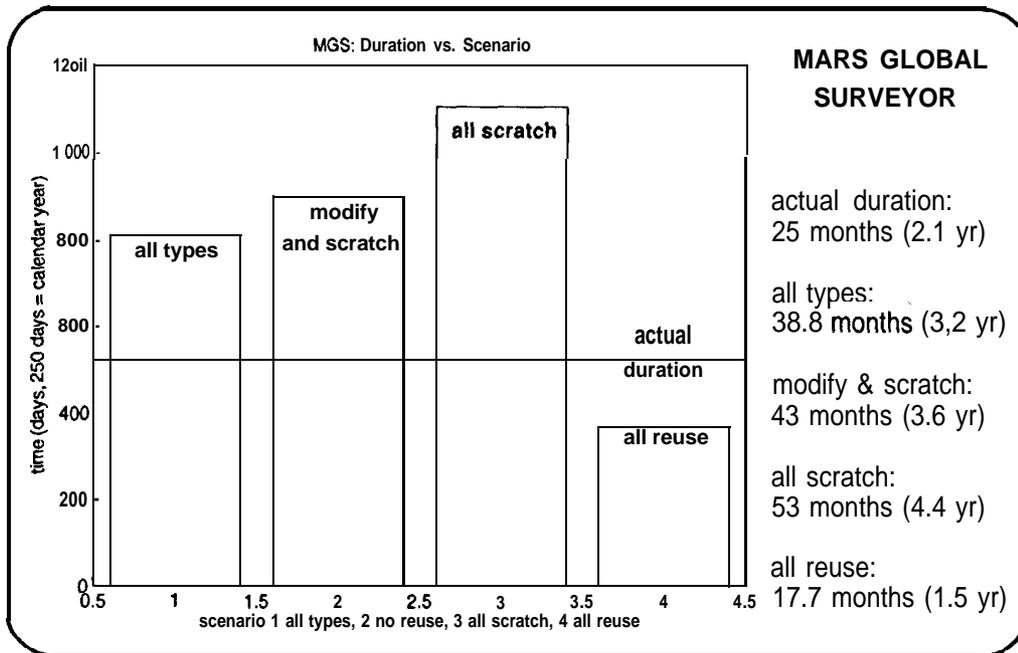


Figure 17: Case Two: Mars Global Surveyor Project

The data for the Mars Global Surveyor is shown in Figure 17. The data is nearly identical to the Pathfinder Project because of similar inputs. Both projects were developed at the same time with many of the same resources and thus had similar parameters: The difference between the two, of course, is the actual duration of the projects.

The graph shows the actual duration maps most closely to the “all reuse” scenario. This is also expected. Mars Global Surveyor was a replacement for the Mars Observer spacecraft and used largely the same design. So it was designed using mostly reusable knowledge.

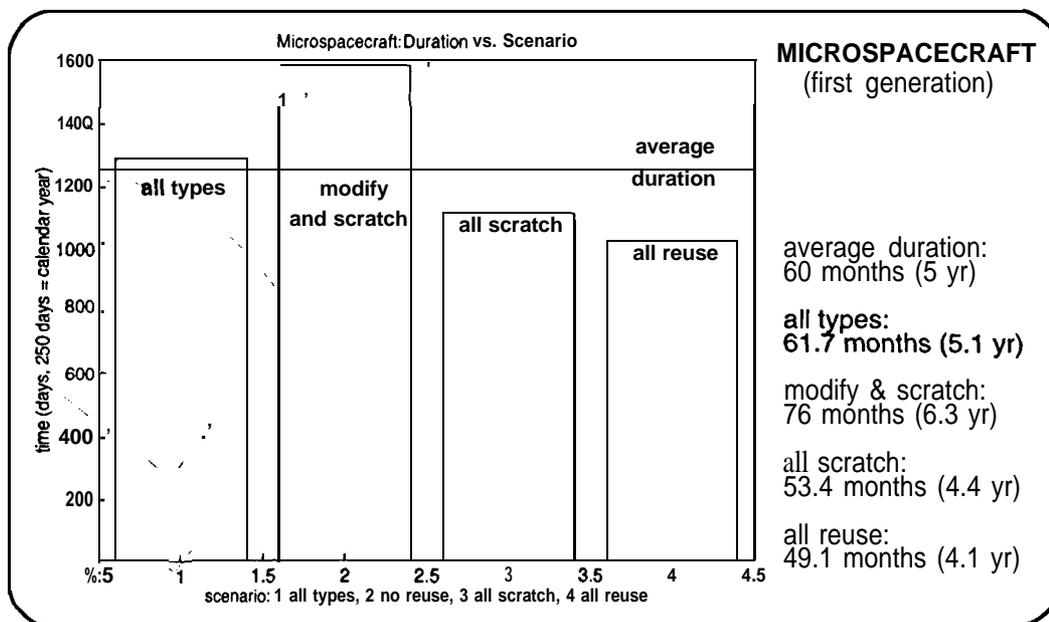


Figure 18: Case Three: First Generation Microspacecraft

A planned program for **Microspacecraft** is shown in Figure 18. Because no actual duration is available yet (the program has not yet begun) an average duration is shown. The nature of this data is unusual. It shows the duration is shorter for an “all scratch” scenario than for a “modify and scratch” scenario. This was unexpected, but can be attributed to the type of project and the mechanics of the simulation.

In the “all types” and “modify and scratch” scenarios, each process “waits” for reusable or modifiable knowledge before it starts to do things from scratch. This waiting period is based on how people search for information. When doing a WWW search for example, people will generally search for 15 to 20 minutes before giving up. The same search theory applies for engineering. Engineers will search for a limited amount of time before giving up and doing it themselves. Another basis for the waiting periods is the limited utility of reusable knowledge. An engineer maybe presented with a number of pre-programmed models, but may find them hard to understand or full of logical errors.. Because of this, engineers may choose to reprogram models entirely after wasting a short amount of time reviewing the ones provided.

Microspacecraft as shown in this case are first generation, in they are the first of their type to be manufactured. Because of this, no modifiable or reusable knowledge will be available for their use.

And so in the “all types” and “modify and scratch” scenarios for the **Microspacecraft** case the processes “wait” for modifiable and reusable knowledge is never available. , In the “all scratch” scenario the processes never wait, but rather immediately start doing things from scratch. The accumulation of the wait times in the “all types” and “modify and scratch” scenarios causes them to be longer than the “all scratch” scenario in the **Microspacecraft** case.

The “all reuse” scenario, however, is shorter than the “all scratch” scenario. This is because the “all reuse” scenario is programmed to provide an artificial constant supply of reusable knowledge. And so it is always has the shortest duration.

Another interesting result from the **Microspacecraft** case was the long durations. The **Microspacecraft** concept is to produce smaller, faster, cheaper spacecraft with development durations much less than predicted in this model. But **Microspacecraft** rely on a number of new technologies have not yet been developed. The the graph shown in. Figure 18 is a first generation **Microspacecraft** will suffer from the development time delays of new technologies, manufacturing processes, and launch vehicle developments. Second generation **Microspacecraft** which can reap the benefits of qualified and stockpiled new technologies will likely have shorter development times.

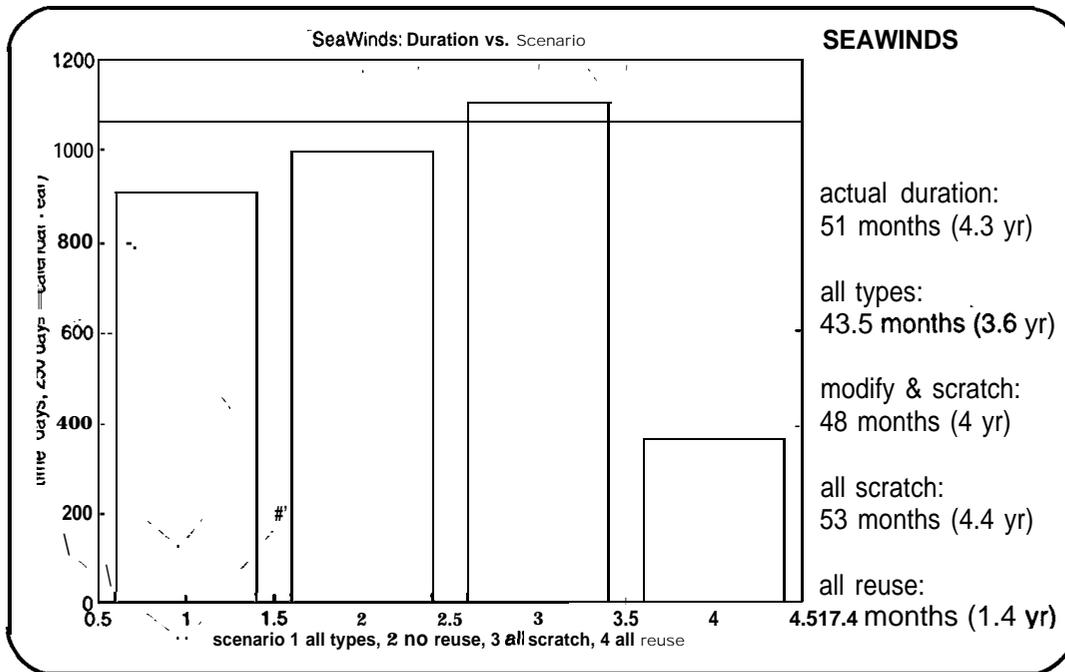


Figure 19: Case Four: Seawinds (Earth Orbiting Instrument)

Figure 19 shows the results of the SeaWinds case. SeaWinds is unlike the first three cases in it is an earth-orbiting instrument and not an interplanetary spacecraft. The actual duration of the SeaWinds project is shown and closely matches “all scratch” scenario. Although SeaWinds had the NSCAT instrument and other weather instruments to provide it with modifiable and reusable knowledge, a large part of the engineering was done from scratch anyway. Thus, the graph shows how much time might have been saved if the program had taken full advantage of available resources.

The State of Health Monitor instrument shown in Figure 20 is again unlike the previous cases in it is a microinstrument designed to be carried on interplanetary probes and landers. The actual duration for the project most closely matches the “all reusable” scenario. In reality, the instrument was indeed designed and manufactured using reusable knowledge. It was produced by JPL’s Microdevices Laboratory, which essentially stockpiles the technologies it creates so they can be quickly and easily assembled into a flight instrument.

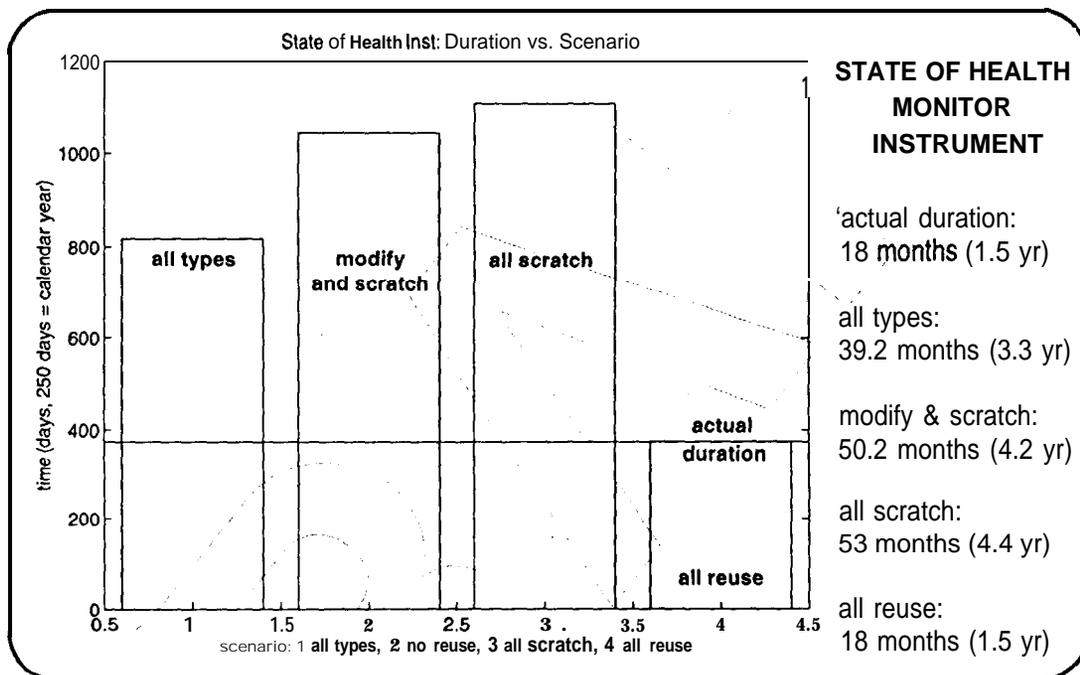


Figure 20: Case Five: State of Health Monitor (Micro Devices Laboratory Instrument)

10.0 Conclusions

The results detailed above support the hypothesis that the availability of the appropriate knowledge type will decrease process duration. It data did not support a quantified percentage of decreased duration, but it did show a minimum value of 18 months for simple projects. Therefore, 18 months was concluded to be the quickest possible cycle time for spacecraft development from concept to launch.

The results from the first simulation suggest that complex projects will benefit the most from knowledge reuse. Simpler projects may not realize significant time savings by reusing knowledge. The results from the second simulation suggest that on a project by project basis, knowledge reuse produced dramatic time savings. In some cases, such as the Pathfinder Project, the duration was decreased by almost 50% by reusing knowledge.

The experiment as a whole supports the implementation of knowledge management initiatives at JPL. The most important element of the initiative will be the knowledge infrastructure. The use of knowledge scenarios in this experiment show that although knowledge maybe available, it is of no use if it is not accessible by the project.

The experiment also showed a surprise result in that it maybe useful as a preliminary schedule

estimator. The model was able to accurately "(predict" the development durations of many of the test cases using only simple parameters and a rough idea of available knowledge as inputs.

11.0 Suggestions for Further Research

Although simple formulas for estimating cost were included in this model, it was not a parameter analyzed in this experiment. The effect of decreasing design margins during development is likely to impact the cost and duration of the DNP Process. As margin decreases, the cost to fix a problem increases. Also, there is an increase in the time and effort necessary to solve a problem without exceeding margin. The effect of knowledge availability on risk would be the next best utilization of this model. First, methods of relating risk to knowledge must be developed. JPL has recently begun a study of risk management.

12.0 Acknowledgments

- Professor Edward F. Crawley, my thesis advisor
- Lynne P. Cooper, my JPL advisor
- Dr. Robert Shishko, for expert advice and insight
- Eric Hopkins and NuThena, for invaluable technical support
- Neil Yarnell and Cecilia Guiar, for guidance
- John Baker, for hands on experience with reengineering
- Randy Parlier, for assistance with writing scripts
- Professors Jack Kerrebrock and Daniel Hastings for the confidence to seek my goals
- Sharonleah Brown, for SERC support
- Luc Besson - for The Big Blue

13.0 References

1. Sailor, J. D., "Systems Engineering Management Guide," Lockheed Missiles and Space Company, Defense System Management College, Fort Belvoir, VA, May 1983.
2. Develop New Products Reengineering Team, <http://vision.jpl.nasa.gov/JPL/redesign/dnp/dnp.html>
3. Spacecraft/Vehicle Level Cost Model, <http://www.jsc.nasa.gov/bu2/SVLCM.html>
4. Hammer, M., Stanton, S., *The Reengineering Revolution: a handbook*, Harper/Ballinger, New York, 1994.
5. Hammer, M., Champy, J., *Reengineering the Corporation: a manifesto for business revolution*, Harper/Ballinger, New York, 1993.
6. Hammer, M., *Beyond Reengineering: how the process-centered organization is changing our work and our lives*, Harper/Ballinger, New York, 1996.
7. Shishko, R., "NASA Systems Engineering Handbook; Jet Propulsion Laboratory, Pasadena,

CA, SP-61 05, June 1995.

8. *Foresight User's Guide*, Rev. 4.1, NuThena Systems, McLean, VA, 1996

9. "NASA Systems Engineering Process for Programs and Projects: Ver. 1.0, Lyndon B. Johnson Space Center, Houston, TX, JSC 49040, Oct. 1994

10. Statement by Administrator Daniel S. Goldin March 19, 1996

11. May 15-21 1995 issue of Space News

14.0 Appendixes

Appendix A: Foresight Modeling Tool

Foresight is a tightly integrated suite of tools support the full **range of** activities of systems development -- the development of requirements specifications and rough drawings, the construction of an executable system model, the execution of the model and data collection. **Foresight's toolset** extends to a broad range of life cycle needs, including building an operational prototype **with** a GUI interface, plotting simulation data, documenting the model using work processing tools, and extracting the model to an ASCII format.

Foresight includes graphical and textual model editors for construction of executable system models using hierarchical flow diagrams, state diagrams, procedural descriptions, and libraries. Includes a collection of 100+ **pre-built** library elements (such as queues, filters, resources, etc.) as well as the capability for the user to create their own reusable library elements.

Foresight contains a highly interactive, simulation engine for executing system models and performing a **wide variety of analysis: functional verification, performance analysis, behavioral correctness**. Includes a host of debugging, data monitoring/collection, and execution options. A data visualizer is available for displaying simulation results in a variety of ways, including x-y plots, strip charts, **timelines**, and histograms.

Foresight **Altia-Lite** is a human interface prototype, **to mock up a realistic user interface for your** system, and while linked with your Foresight executable system model, interact with the system under design. Also included in the Foresight toolset is **FS/DOC**, an automatic generation of documents from your system model.

Appendix B: NASA Program/Project Life Cycle Process Flow

The **NASA Systems Engineering Process for Programs and Projects** establishes a common set of suggested top-level technical processes for developing NASA missions. Developed by a NASA-wide team, it **consists** of a structured set of program/project technical activities and milestones. These are designed to effect a structured evolution of activities and products so objectives are met effectively and efficiently. The purpose of this document is to provide guidance, criteria, approach, procedure, and product and terminology **standards** for the successful completion of these activities. Especially important are the progressive, structured, traceable steps of **system** baselining and configuration control. This document is subordinate to and supports NASA Management Instruction (NMI) 7120.4, Management of Major System Programs and Projects, and the associated NASA Handbook (NHB) 7120.5.

Appendix C: DNP Process Map

Appendix D: Index of Terms and Acronyms

DNP: Develop New Products
CAD: Computer Aided Design
CDR: Critical Design Review
CEM: Concurrent Engineering Methodology
COS: Community of Science
DBAT: Design, Build, Test and Assemble
D&IM: Data and Information Management
DNP: Develop New Products
ECAD: Electronic Computer Aided Design
EDMS: Electronic Data Management System
EIS: Enterprise Information System
Element:
Generalist:
Guru:
High level process:
ICD: Interface Control Document (or Drawing)
Knowledge scenario:
KM: Knowledge Management
KSD: Knowledge Space Directory
MCAD: Mechanical Computer Aided Design
MCR:
MDR:
MDS: Mission Data System (provides the environment for hardware and software development of flight assets)
Model:
Modifiable knowledge:
MPF: Mars Pathfinder
MRR:
MSD: Mission and System Design
NMP: New Millennium Project
PDC: Project Design Center
PDMS: Product Data Management System
PDR:
PM: Project Management
PPIC: Project Planning, Implementing, Closing
Process: A process consists of a set of interrelated activities and supporting resources transform inputs into outputs.
PRR:
Resource:
Reusable knowledge:
RFI: Request for Information
Scratch:
S D R :
SE: System Engineering
Simulation:
Specialist:
SRR:
Subprocess:
TRL: Technology Readiness Level
VIVO: Validate, Integrate, Verify, Operate