

AN EVOLUTIONARY APPROACH
FOR
PROJECT ORGANIZATION DESIGN:
PRODUCING HUMAN-COMPETITIVE RESULTS
USING
GENETIC PROGRAMMING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF CIVIL AND ENVIRONMENTAL
ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Bijan KHosraviani

December 2005

UMI Number: 3197452

Copyright 2006 by
KHosraviani, Bijan

All rights reserved.

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform 3197452

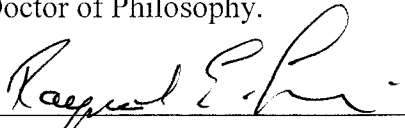
Copyright 2006 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

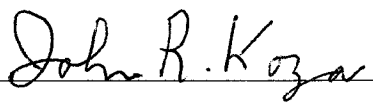
© Copyright by Bijan KHosraviani 2006
All Rights Reserved

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.



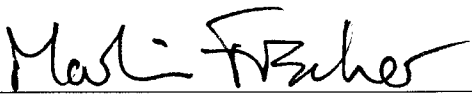
Raymond E. Levitt, Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.



John R. Koza

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.



Martin A. Fischer

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.



Yan Jin

Approved for the University Committee on Graduate Studies.

*And may we be among those who make this life
fresh!¹*

¹ Zarathushtra (Zoroaster - 1771 B.C.)

Acknowledgments

I would like to say “Thank You” to:

- My mother—Pouran—for staying up late at night from time to time, so I could finish my homework when I was in elementary school.
- My father—Bahram—for being one of my best role models in life.
- My sisters and brothers, and their children—Homa, Nahid, Behrooz, Mehrdad, Sima, Bahar, Nima, Avishan, and Abrisham—for being there for me when I needed them.
- My principal advisor—Dr. Ray Levitt—for guiding me through this journey. Ray, thanks for taking me on as one of your students, thanks for your continuous encouragement, inspiration and support, and thanks for your great revisions on my writings and your great advice both academically and otherwise.
- My co-advisor—Dr. John Koza—for his inspiration, kindness and continuous support. Thanks John for your guidance, your detailed and astute comments and feedback, and all the thoughtful discussions and meetings we had either at your company, Genetic Programming, Inc., or at your home.
- My co-advisor—Dr. Martin Fischer—for his great advice and the detailed and insightful feedback on this dissertation.
- My co-advisor—Dr. Yan Jin—for his great suggestions and for the many thoughtful discussions that we had via conference calls and/or netmeetings.
- My colleagues at school—Ashwin Mahalingam, Ryan Orr, John Chachere, Michael Murray, Tamaki Horii, and Monique Lambert—for their thought provoking discussions, remarks, and suggestions from time to time.
- Marc Ramsey, the senior research programmer of our VDT research group, for helping me greatly with the programming aspect of this research project.
- Dr. Sean Luke, the developer of ECJ (A Java-based Evolutionary Computation and Genetic Programming Research System), for providing the easy-to-use programming environment that was used for this research project.

- Stanford Center for Integrated Facility Engineering (CIFE) and National Science Foundation (NSF) for their financial support during my graduate studies at Stanford.
- Pat Gorman, VP of product development at ePM, LLC, for providing the SimVision-R software and assistance in understanding details of the implementation.

And last but not Least:

- My wife—Negin—for her love and support, and leaving her good job in Boston and coming to the Bay Area and patiently supporting me for the past three years until I finish my education. Thanks much my dear Negin!
- and our dearest first child—NeeKee—who patiently waited in Mommy's tummy, and listened to my oral defense on June 9, 2005 (Negin's due date), and finally came to this world on my graduation ceremony day to become the best graduation gift that I could ever receive!

From 4 to God²!

I remember the day ...

...when I was only 3 or 4 years old and one of my cousins asked me, “How much do you love me?” I said: “from *Chah* to God!” - *Chah* in Persian means a “Well” and in my imagination that was the longest distance I could think of at the time (i.e., from the bottom of a well to the top of the sky where God was) and that was how much I loved him and that was my equivalent word for saying “infinity”. On the other hand, “*Chahr*” (with an R at the end means 4) – So for weeks and months when I told my family members how much I love them, they thought I was saying from 4 to God, and they always wondered why 4!

I remember the day ...

...when I was about 8 years old and I dreamed of coming to America and going to one of the best schools in the world. But I was wondering why the people who do that, don't come back and make a better world?

I remember the day ...

...when I was a teenager and all Iran's borders were closed because of the 1979 revolution. For two years, I was trying every border and made many attempts to escape ... from The Caspian Sea side to the Persian Gulf side from the Azarbayjan side to the Afghanistan side.

I remember the day

...when in my one of my attempts to escape, I was captured at the Kurdistan border for a week in a cottage and I had very little hope of getting out of the area alive.

I remember the day

...when I finally managed to escape from Iran on a horse from the Turkish border through the mountains, but when my smuggler dropped me off on the road in the middle of the night, I had no idea which direction led toward Turkey and which side led back to Iran!

I remember the day

...when I saw the first Latin sign on the road in Turkey, it was the best day of my life, because then I was sure I was finally out!

I remember the days

...that I had to travel for months through Turkey, Spain, Denmark and Italy to find my way to America.

² This is the speech that I prepared to read during the 2005 graduation ceremony, but I did not get a chance. Well, who knows maybe I will next year!?

I remember the day...

...after being rejected twice for a US Visa, the US consulate told me that there is no way that you can go to the US. But I did not take “No” for an answer.

I remember the day ...

...when I finally managed to come to the US, and I was handcuffed at the airport. When they were taking me to the jail through NY city streets, I was so happy to finally be in colorful America, and I could see it with my own eyes– but disappointed that I should see it while handcuffed!?

I remember the day ...

...when I was released from the NY jail after a couple of months, I set my next goal. That was to live with an American family, so I could learn the language and American culture quickly.

I remember the day ...

...that after 4 months of disappointing and exhausting search, I was so frustrated that I just asked a very old lady at a bus stop: “I want to live with an American family?” and she screamed: “What?!” – And after I repeated my request, she said “Aha - why don’t you go to a church?” After that for the next couple of months, my daily routine was to look for crosses from a distance and go to any churches that I could find.

I remember the day ...

...that I walked to a church and the pastor of the church offered me the chance to live with them, which turned out to be the best two years of my life.

I remember the day ...

...that I could not get a job as an auto-mechanic (although I had done mechanical work in Iran), so I started working as a gas station attendant and learned how to wash toilet bowls with a scotch in my bare hands, in order to pay for my living expense and college.

I remember the day ...

...that I had to sleep in my car for a few weeks when I came first to the Bay Area and had to shower every morning in the college locker room. (I am sure some of you have done this, too).

After I received my BS in Electrical Engineering and applied to Stanford for an MS degree in Electrical Engineering,

I remember the day ...

...that I did not get accepted. I remember the lady in the graduate admission office who asked me, “Why don’t you apply to Operations Research instead?” And that was what I did. It happened that I liked this much better than Electrical Engineering.

I remember the day ...

...that it was tough to find a professor in Civil & Environmental Engineering who would accept me as his PhD advisee, but I finally managed to find one of the best advisors at Stanford.

Well, here I am finally, at the end of my PhD! With a couple of messages:

My message for the young people in the audience:

- Don't take the first "No" for an answer.
- Always try to make the Best of the Worst.
- And as one said: "know that whatever doesn't kill you will make you stronger"
- Persistence, Persistence, Persistence!

My message for you who are graduating:

Let's not forget the World and the humankind! I always asked: "Why do people who are honest and intellectual only work in their labs on their important experiments - like "counting the stars" – And those who are dishonest and perhaps anti-intellectual run our world?" (Of course, I am talking only about the other side of the world not this side!) – I think we all have a responsibility to make positive changes in this world. And as someone said: "Let's be among those who make this world fresh!"

My name is Bijan. Incidentally, there is a famous 10th century Persian poem named "*Shahnameh*"³ (The Epic of Kings). One of its main characters is Bijan, who was captured in a well (a *Chah*) by his enemies for what seemed liked forever. Finally, he was freed from that pit and given a chance to begin his life.

I feel I am out of the well now, too. But I also feel this is only the beginning and I am just at the ground level. There is a long way to the highest peak of the mountains and certainly a very, very long way to the top of the sky!

Thank You, and Love to You All –
From *Chah* to God!

³ See <http://www.Shahnameh.com/>

Abstract

In the complex and rapidly changing business environment of the early 21st century, designing an effective and optimized organization for a major project is a daunting challenge. Project managers have to rely on their experience and/or trial and error to come up with organizational designs that fit their particular projects. Painful and costly experience in a wide range of governmental and private organizations has demonstrated that projects to develop buildings, software and other products often fail, not because the design of individual components was at fault, but rather because the organization performing the complex supervision and coordination tasks required for system integration failed due to information overload.

The Virtual Design Team (VDT) simulation system, based on the information processing theories of organization science, was a successful attempt to develop an analysis tool for project organization design (Jin and Levitt, 1996). However, like the analysis tools that support many other design processes, VDT has no inherent ability to improve or optimize current designs automatically. It simply predicts performance outcomes — in terms of time, cost and several measures of process quality — for a particular project organization design alternative. A VDT user must thus experiment in “*What if?*” mode with different design alternatives in an attempt to find better solutions that can mitigate the identified risks for a given project configuration. The problem has many degrees of freedom, so the search space for better solutions is vast. Exploring this space manually is infeasible. VDT relies on the expertise of the human user, guided simply by intuition about ways to improve on prior designs, to find better solutions. So it offers no guarantee of optimality.

Our research extends the capabilities of VDT and other similar organizational analysis tools by going beyond computer support for “What-if?” analysis to automated design of project organizations. Evolutionary computing methods such as genetic programming are used to design and develop a postprocessor for VDT to help project managers find near-optimal designs for their project organizations.

This dissertation describes in detail the approach I developed to represent project organization design alternatives in a genetic programming format, so that the design can effectively evolve. In addition, it demonstrates how I was able to represent different project performance objectives and constraints in a fitness function which can successfully guide the model toward searching for better designs.

A preliminary version of my postprocessor optimizer beats the best human trial-and-error solutions developed by more than 40 teams over the past eight years. The postprocessor was awarded a Silver Medal for human-competitive results in genetic and evolutionary computation at the GECCO-2004 Conference.





I discuss why I chose the evolutionary computing approach as opposed to classical optimization methodology, and show some of the advantages and limitations of the evolutionary approach. There was no formal theory of project organization design nor any analysis tools for predicting the performance of project organizations prior to the development of VDT in the 1990s. Not surprisingly, therefore, research by C.B. Tatum in the early 1980s found that current human-developed project organization structures are the result of “natural” trial-and-error evolutionary processes. By applying the evolutionary computing approach to organization design, my model is thus actually mimicking the nature of human organization design. In addition, I demonstrate how my approach can create a powerful Human-Computer Interaction (HCI) environment that can motivate humans to think “outside the box” when designing project organizations.







Using a combination of “intellective” (theorem proving) and “emulation” (natural, empirical) experiments, I validate the postprocessor’s “near-optimal” solutions against findings of organizational contingency theory and human-derived solutions for a set of real test cases. By showing that “optimal” structure depends on the relative emphasis of time, cost and process quality outcome metrics, I extend contingency theory to develop a richer “micro-contingency theory” for project organizations.

This research represents a significant step towards closing the *relevance gap* between organization theory and organization practice by addressing the issues of organizational design prescriptively. I analyze alternatives in terms of fitness functions that evaluate specific designs for “survival” and “reproduction” in the spirit of contingency theory.

Finally, the thesis concludes with a summary of the contributions of this research in the three areas of organization science, project management, and computer science.

Table of Contents

Acknowledgments	v
From 4 to God!	vii
Abstract	x
Table of Contents	xiii
Figures	xv
Tables	xvii
 Chapter 1	1
Introduction	1
1.1 At the Outset.....	1
1.2 Motivation	2
1.3 Organizational Design Issues, Challenges, and Limitations	4
1.4 Criteria for a Good Organization Design System	5
1.5 Research Focus and Dissertation Objectives	6
1.6 Dissertation Outline.....	7
 Chapter 2	9
Computational Modeling of Organizations and Evolutionary Methods	9
2.1 Computational Modeling and Organizational Design.....	9
2.2 The Virtual Design Team (VDT) Organizational Analysis Model.....	11
2.3 Evolutionary Computation and Design	22
2.4 Genetic Algorithms - Background	25
2.5 Genetic Programming - Background.....	27
2.6 Related Research on Optimization of Project Organizations.....	30
 Chapter 3	32
Evolutionary Organization Design Approach	32
3.1 High-level view of the Evolutionary Model for Project Organization Design	32
3.2 Design Scope.....	34
3.3 Design and Implementation Challenges.....	36
3.4 Representation of Project Organization	37
3.5 Transforming Genetic Tree (TGT).....	38
3.6 Fitness Function	41
3.7 Implementation of EOD model and Integrating the System as a Whole	43
3.8 Similarities and Differences between EOD and other GP models.....	46
3.9 Advantages and Limitations of EOD	49
 Chapter 4	52
Case Studies: Producing Human-Competitive Results	52

4.1 Definition of Human-Competitiveness	52
4.2 Biotech Plant Case Study	53
4.3 ASIC Design Case Study	71
4.4 Are the Results Human Competitive?.....	75
4.5 Does it Matter Where We Start?	77
4.6 Does EOD Always Find the Optimal Solution?.....	78
4.7 Should CEOs Mop Floors?	78
 Chapter 5	80
Validation against Organizational Contingency Theory	80
5.1 Organizational Contingency Theory – Background.....	80
5.2 Organization Design.....	81
5.3 Definition of a “good” Design	83
5.4 Technology as a Contingency Factor	84
5.5 Intellective Validation of the Postprocessor.....	86
5.6 Validation using Intellective Experiment.....	88
5.7 Validation using Real-World Project Organizations.....	94
5.8 Discussion of Results	99
5.9 Conclusions	102
 Chapter 6	103
Conclusions	103
6.1 Overall Contributions of the Research	103
6.2 Contributions to Social Science	105
6.3 Contributions to Project Management	106
6.4 Contributions to Computer Science	108
6.5 Suggestions for Future Research.....	109
 References	112
 Appendix A	118
Sample of an alternative genetic tree that was not implemented	118
 Appendix B	120
Detailed description of TGT	120
 Appendix C	124
Sample of a modified section of ECJ parameter file with added constraints	124

Figures

Figure 2.1 User Interface of the VDT Simulator	14
Figure 2.2 A Sample of Gantt Chart as one of the VDT Performance Output	15
Figure 2.3A Sample of the VDT Project Quality Performance Output	17
Figure 2.4 A Sample of the VDT Position Backlog Output.....	18
Figure 2.5 VDT's Information Processing View of Knowledge Work	20
Figure 2.6 Sample of a Computer Program Represented in a Tree Format	29
Figure 3.1 Evolutionary Computational Approach for Optimizing Organization Designs	33
Figure 3.2 Sample of a Transforming Genetic Tree.....	40
Figure 3.3 Flow Chart of the Integrated EOD/VDT System.....	44
Figure 3.4 Similarities between Organizations and Electronic Circuits	48
Figure 4.1 Biotech Plant case study	54
Figure 4.2 Actors can have Different Skills and Skill Levels.....	56
Figure 4.3 Best Individual of Generation 16 in a Genetic Tree Format.....	58
Figure 4.4 Comparison of Gantt Charts Before (Top) and After (Bottom) Evolutionary Process.....	61
Figure 4.5 Comparison of Communication Quality Risks Before (Top) and After (Bottom) Evolutionary Process	62
Figure 4.6 Comparison of Project Position Backlogs Before (Left) and After (Right) Evolutionary Process.....	63
Figure 4.7 Comparison between GP and the Best Human Solution	64
Figure 4.8 An Alternative GP Solution, which is 2 days better than the Best Human Result while Matching the Human Solution in most cases	65
Figure 4.9 Improvement of Fitness Value Generations 1 Through 30.....	67
Figure 4.10 Alternative Solution Found by GP that was 3 Days Better than Best Human Solution and was able to Eliminate Two Positions	68
Figure 4.11 Best Solution Ever!.....	70
Figure 4.12 ASIC Design Project Case.....	72
Figure 4.13 EOD Suggested Solution met the August First Deadline	74
Figure 5.1 A Validation Trajectory Proposed by Thomsen et al. (1999).....	87
Figure 5.2 A Sample of an Idealized Project Organization in VDT	88
Figure 5.3 Comparing the effect of Formalization and Centralization on Time and Cost when Technology is Routine (FEP & PEP = 0.01)	90
Figure 5.4 Comparing the Effect of Formalization and Centralization on Process Quality (FRI) when Technology is Routine (FEP & PEP = 0.01)	91
Figure 5.5 Comparing the Effect of Formalization and Centralization on Time and Cost when Technology is Non-routine (FEP & PEP = 0.1)	92
Figure 5.6 Comparing the Effect of Formalization and Centralization on Process Quality (FRI) when Technology is Non-Routine (FEP & PEP = 0.1).....	93
Figure 5.7 Comparing the Effect of Formalization and Centralization on Project Duration when Technology is Routine (FEP & PEP = 0.01).....	95
Figure 5.8 Comparing the effect of Formalization and Centralization on Process Quality when Technology is Routine (FEP & PEP = 0.01).....	96

Figure 5.9 Comparing the Effect of Formalization and Centralization on Project Duration when Technology is Non-routine (FEP & PEP = 0.1)	97
Figure 5.10 Comparing the Effect of Formalization and Centralization on Process Quality when Technology is Non-Routine (FEP & PEP = 0.1).....	98
Figure 5.11 Sample of a Project Outcome Improvement and Changes to Decision Making Policies when Emphasis is set on Quality as GP Evolves a Real-world Project Organization	99

Tables

Table 4.1 Genetic Programming Setup for the Biotech Plant Case Phase I.....	57
Table 4.2 Genetic Programming Setup for the Biotech Plant Case Phase II	59
Table 4.3 Genetic Programming Setup for the Biotech Plant Case Phase III.....	69
Table 4.4 Genetic Programming Setup for the ASIC Design Case	73
Table 4.5 Best Human Results Produced for the Biotech Plant case study	76
Table 4.6 Graduate Students who tried the ASIC Design Case Study.....	76
Table 4.7 Total Number of CIFE Attendees for the Years that Case Study Was Given ..	77
Table 5.1 Summarizing our Findings on Formalization Properties Based on Technology Routineness and Activity Interdependency.....	100
Table 5.2 Summarizing our Findings on Centralization Properties Based on Technology Routineness and Activity Interdependency.....	102

➔ Chapter 1

Introduction

“In formal logic, a contradiction is the signal of defeat, but in the evolution of real knowledge it marks the first step in progress toward a victory”

-- Alfred North Whitehead

“If we knew what it was we were doing, it would not be called research, would it?”

-- Albert Einstein

1.1 At the Outset

When we launched this research project, we wondered whether it would be possible to come up with a tool that could help managers design project organizations for superior performance in terms of time, cost, and quality objectives. Specifically, we wondered whether an evolutionary computing approach, such as Genetic Programming (GP), would be a good way to optimize the design of a project organization. Could we use GP to represent and reason about the complex and relatively unstructured problem of organization design so the design could be evolved? If so, we wondered whether these results would be as good as or better than organization designs that humans now produce.

As described in this dissertation, not only did we demonstrate that GP is an effective method for optimizing organization designs, but our model, Evolutionary Organization Designer (EOD), contributes to management practice and organization theory. We show that EOD is more than just a postprocessor optimizer for organization design; it is a model that can motivate a designer to “think out of box” when designing organizations.

1.2 Motivation

Over the past 50 years, since computer-based analysis tools began to enter engineering practice, the formalization, automation and optimization of design processes for many kinds of engineering products have made remarkable strides. However, lacking analysis tools that can predict the performance outcomes of candidate solutions, the design of the work processes and organizations that engineers employ to create their optimized engineering products has not evolved beyond trial and error or, at best, attempts to adapt past organization designs based on prior experience (Tatum, 1983).

Over the past 15 years, the Virtual Design Team (VDT) research group has developed theory, methodology and analysis tools that now allow engineers to predict the performance of a given configuration of the work process and organization for carrying out multidisciplinary, concurrent engineering design. This work has been validated in multiple real world tests and was commercialized as SimVision⁴® (Kunz et al., 1998).

Like engineering analysis tools, VDT can predict performance outcomes for a given candidate solution—in this case a configuration of a work process and organization. However, an engineering manager attempting to find an optimal solution using this methodology and tool faces the same problem as an engineer attempting to use analysis tools to optimize an engineering design problem. The space of potential solutions for real world engineering and management problems is semi-infinite. The best an engineer or manager can hope to achieve using only an analysis tool is a feasible solution that meets or exceeds minimum requirements. Expert engineers and managers may find solutions that improve on minimally acceptable ones, but will rarely find optimal solutions to their problems using analysis tools when guided simply by their own intuition about ways to improve on prior designs.

Tools like VDT directly address the question of information flow and buildup between and among members of project teams and can help to design more robust organizations to

⁴ SimVision is a product of epm. For more information see: <http://www.epm.cc/>

handle the information processing demands for carrying out direct work, as well as for supervising and coordinating the work of subteams in a given project. But these modeling tools rely on managers' experience and intuition to propose and test "good" configurations among the semi-infinite space of possibly better configurations, and thus frequently fail to uncover even near-optimal configurations.

Classical (analytical or numerical) methods for finding optimal solutions to multidimensional problems have been applied to engineering computation problems for a long time. While they perform well for highly structured problems in many cases of everyday design practice, they can not do well in more complex situations. In real design problems, like in our case of project organization design, the number of design parameters is very large and the objective function can be very complicated and may involve non-linear behaviors. These objective functions usually have many local optima, so the classical methods, such as gradient methods or hill-climbing, will usually miss global optima in the favor of local ones.

There are three main problems in applying classical optimization methods to the domain of organization design. First, the simplifying assumptions that are usually made to define the objective and constraints make the model diverge from the real-world situation. Second, the traditional approach usually does not scale well for larger problems in realistic practical applications, and it faces the phenomenon called "Combinatorial Explosion" (Chan et al., 1996). Third, as the problem gets more complex, the solution often gets trapped in a local optimum. In such complex cases, stochastic optimization techniques such as genetic algorithms can sometimes find a design near the global optimum within a reasonable elapsed time and computational time (Renner and Ekart, 2003). The present research addresses these gaps in extant methods using genetic programming as a fresh approach.

Genetic Programming (GP) (Koza, 1992) is an extension to Genetic Algorithms (GA) (Holland, 1975). GP has been successfully applied for optimizing analogous multidimensional, topological, and attribute-based optimization problems like circuit

design (Koza et al., 1996 and 1999). These successes were the inspiration for this research. They suggested that similar approaches might prove fruitful for optimizing organizational designs. If they could, this research would have the potential to help managers better optimize organizations that deliver complex hardware and software products to broad sectors of our economy and society, systemically increasing their efficiency, reliability, and quality.

1.3 Organizational Design Issues, Challenges, and Limitations

Designing a “good” project organization has been a challenging task for many years. Some researchers, such as Romme (2003), argue that organizational design has been shifting from the academic side to the industry side. Romme argues that “in view of the persistent relevance gap between theory and practice, organization studies should be broadened to include design as one of its primary modes of engaging in research.” This research attempts to close this gap.

Project organizational design is a complex, multi-dimensional optimization problem involving both continuous and discrete variables. For example, an organizational designer must size functional teams, assign staff to tasks, and set communication and control policies. In addition, one must define the organization structure or topology. The degree of dimensionality grows very rapidly, almost exponentially, as the number of individuals/sub-teams and activities to be performed grows. Thus, it makes it very difficult, if not impossible, for a human designer to search for a near optimal design even with the help of organizational analysis tools such as VDT. Currently, an experienced VDT modeler can design a project organization based on her or his experience and use trial and error to diagnose performance risks for a given design, and attempt to reduce the risks iteratively by a series of trial and error interventions. However, the process is very time consuming and even after many trials, there is no guarantee of optimality.

As we will show in Chapter 2, existing computational modeling tools for organization designs are used merely for analysis and do not optimize or search for better designs. We chose to use evolutionary methods for the purpose of organization design optimization

for several reasons. First, unlike traditional optimization methodology, there is no need for oversimplification of the real-world problem. Second, an evolutionary approach is not based on any knowledge base or expert system, so it can generate new ideas and “creative” organization designs. Third, unlike other classical optimization methods, it allows a user to explore numerous creative solutions to problems instead of a single optimal solution. Fourth, it can motivate “out-of-box” thinking by generating counter-intuitive organizational designs that rational thinking could rarely produce. And finally, it replicates in many ways the survival and propagation of good organization designs in the real world.

The fact that an evolutionary approach can replicate what is happening in the real world by evolving organizations through generations is the key factor for being able to reconstruct some of the well established findings of contingency theory. This is discussed in detail in Chapter 5. In addition, because of the multi-weight and multi-objectiveness of the fitness function, we demonstrate how some of the empirically derived contingency theory propositions can be extended and refined based on the relative emphases of different outcome performance criteria, such as time, cost, and quality of the project.

Evolutionary methodologies, however, have their own challenges and limitations. For example, representation of the design problem that can cover both the attributes and the topology of the organization in a format that can be evolved through generations is a challenging task. Developing a fitness function that can represent the multi-objective fitness function and specific hard constraints of a given project can be difficult.

Evolutionary methods are resource and time consuming. Thus, framing the problem so it can produce promising results with our limited software and hardware capabilities is a significant challenge.

1.4 Criteria for a Good Organization Design System

We laid out the criteria for creating a useful organizational design system as follows:

The system should be able to:

- represent the important dimensions of a project organization (not a simplified representation of it), including both the attributes and the topology of the organization.
- handle the multi-objective performance criteria for a project organization such as shortening the project schedule while still improving quality.
- create new designs not dependent on any knowledgebase and/or expert system, which might limit possible designs to the knowledge that was put in the model in the first place.
- help and motivate the human designer in creating new designs rather than trying to replace humans.
- create a medium for conducting organizational research— evolutionary computational experiments.
- motivate “out-of-box” thinking by generating counter-intuitive organizational designs that rational thinking could never produce.
- generate multiple near-optimal alternatives—some may be more feasible than others—rather than a single optimal solution; thus, it helps the researcher/practitioner to come up with new criteria, approaches, and ideas as starting points.
- start from any point — rather than just from scratch — and improve on an existing plausible design, thus creating opportunities for fluid human-machine interaction to search for desirable solutions in the path that it is directed

1.5 Research Focus and Dissertation Objectives

The main thrust of this research is to propose an approach that uses the genetic programming methodology in a new domain — project organization design — to search for optimal or near optimal designs. In this process, the research questions that we try to investigate are:

1. How can we represent organization designs in a GP?

2. How can we represent different project objectives, and trade-offs among those objectives, in a GP fitness function?
3. How close are the “optimal” solutions found by GP to the predictions of organization theory and management best practices?
4. Can we produce results that are human competitive using GP?

Thus, the core objectives of this research were to:

- Design and implement a post processing optimizer for VDT using evolutionary computing techniques such as genetic algorithms and genetic programming to generate an optimal or near optimal project design.
- Validate the postprocessor by demonstrating the usefulness of the optimizer for project managers and by showing that the results obtained by GP are in-line with predictions of organizational “contingency” theory.
- Create an environment for organizational scientists to refine their organization design hypotheses and develop new theories.
- Provide a tool for project managers that not only helps them create a single, near-optimal design, but also suggests a set of innovative and creative designs for them to pick from.

1.6 Dissertation Outline

Beyond this introductory chapter, which discussed our motivation, objectives, research questions and the structure of this dissertation, the remaining chapters cover the following subjects:

Chapter 2 contains background on computational modeling of organizations, the Virtual Design Team (VDT) organizational analysis approach, and related research on project organization design. In addition, it introduces the evolutionary computing methodology and gives an overview of genetic algorithms and genetic programming.

Chapter 3 attempts to answer our first two research questions mentioned in the previous section. It demonstrates our design and implementation approach and shows how we

represent project organizations in genetic programming format using a Transforming Genetic Tree (TGT) so the project organization can be effectively evolved toward near-optimal designs. In addition, this chapter discusses the EOD multi-objective/multi-constraints fitness function. It concludes with a discussion of the advantages and limitations of EOD design and some of the challenges that we faced during the design and implementation phase.

Chapter 4 presents some of the promising human-competitive results that EOD has been able to generate for two real-world project organizations. It also explains some of the interesting implications and counterintuitive results that EOD can produce.

Chapter 5 discusses validation of EOD against organization contingency theory. It also explains how EOD can be used to extend project organization contingency propositions to specify optimal designs more precisely, based on the relative emphasis that a given project's sponsors place on different performance criteria, such as time, cost, and quality.

Chapter 6, the conclusion of this dissertation, examines the implications of this research in three areas: organization science, management practice, and computer science. It discusses the limitations of this research and proposes fruitful areas for future research.

⇒ Chapter 2

Computational Modeling of Organizations and Evolutionary Methods

“Creativity may have killed a few cats, but evolution certainly eliminated many more incurious ones.”

-- Guy Claxton in ‘Wise Up’

In this chapter, we first give a brief background on computational modeling of organizations, and some of the organizational analysis tools that have been developed during the past two decades. We explain the Virtual Design Team (VDT), upon which our EOD model operates, in depth. Finally, we cover the concept behind evolutionary computation and provide background on two of the most popular evolutionary computing methodologies, namely Genetic Algorithms (GA) and Genetic Programming (GP).

2.1 Computational Modeling and Organizational Design

In recent years, computational modeling and simulation have been growing in popularity as a methodological approach among organizational researchers. Simulation, unlike mathematical modeling, allows researchers to reflect the natural complexity of organization systems as givens. If other methods used in organizational science answer the questions “what happened, and how, and why?” simulation helps answer the question of “what if?” (Dooley, 2002). Computational modeling enables studies of more complex systems than traditional mathematical approaches, because it can generate predictions by “moving forward” into the future, whereas other research methodologies try to look backwards into history to resolve, for example, what happened, why, and how. In a sense, we can say that computational modeling of organization is an approach that is more formal than using words, but less formal than using mathematics.

Researchers in the field of computational organization theory use computational analysis methods to study both humans and organizations as computational entities. Human organizations can be viewed as intrinsically computational because many of their activities involve sharing and transforming information from one form to another, and because organizational activity is often information-driven (Simon, 1976; Carley and Gasser, 1999).

For this reason, many researchers have developed simulation-based analysis tools for organizations during the past two decades that are primarily based on information processing theory (Cyert and March 1963,1992; Simon, 1976; March and Simon, 1958). Starting with computational modeling tools such as OrgCon in the late 1980s, VDT in the early 90's, and OrgAhead in the mid-90's, researchers and practitioners have begun using computational models of organizations to analyze existing organizations and design better ones.

The VDT model operationalizes and extends Jay Galbraith's (1977) and March and Simon's (1958) information-processing abstraction. It treats direct work and coordination/supervision work performed by actors on a project as additive quanta of information to be processed. Unlike PERT/CPM project scheduling tools that do not model coordination and rework explicitly, VDT generates reliable predictions of backlogs, task delays, overall duration, and quality risks for "fast-track," highly concurrent projects. VDT parameters have been calibrated against data from a variety of real world projects in multiple industry sectors, and VDT has been shown to be reliable in generating accurate predictions and guiding interventions (Christiansen, 1999). Although VDT does well in predicting schedule, cost, and process quality performance, it has no ability to improve or optimize a current design without expert human intervention. In section 2.2, we explain in detail the main characteristics of VDT and how it works.

Organizational Consultant or OrgCon (Burton and Obel, 2004) is another example of a computational model that is built based on the viewpoint that an organization is an

information processing entity. OrgCon is an expert system with a multidimensional contingency approach that relates organizational size, climate, strategy, technology, environment, and leadership preferences to organizational structure and design, to identify and eliminate “misfits” and thereby to ensure an efficient, effective, and viable organization. The knowledge base of Organizational Consultant has been developed based on multiple empirical findings of contingency theory research since the 1950s. Based on the initial organizational environment description entered by the user, OrgCon compares that environment with the expert knowledge base, indicates any “misfits” and offers recommendations for corrections to be made.

Other examples are OrgAhead (Louie et al., 2003), which is an organizational learning model designed to test different forms of organizations under a common task representation, or OrgMem (Carley et al. 2000), which is a multi-agent simulation program that imitates the interpersonal communication, information processing, and decision making processes in organizations.

Although all of the above computational modeling tools have been steps toward designing better organizations, none have any automated routines that help the user find an optimal or near optimal organization design. This research makes an attempt to close this gap.

Since our evolutionary model operates on and extends the Virtual Design Team, in the next section we present more details about VDT and discuss its capabilities and limitations.

2.2 The Virtual Design Team (VDT) Organizational Analysis Model

The Virtual Design Team (VDT) is a project organization modeling and simulation tool that integrates organizational and process views of strategic, time-critical projects (Jin and Levitt, 1996). The vision behind VDT is to offer a methodology to design an organization the way an engineer designs a bridge. This means that a user can first create

and analyze the performance outcomes of many alternative configurations of a virtual model using simulation, before implementing the project organization in the real world.

A project manager (PM) can develop several “cases” or scenarios for a given project that are based on different assumptions, and run simulations to predict project schedules using VDT. The PM can also identify organizational risks related to product quality, schedule, and cost outcomes. The simulation software helps the user to set up, monitor, and troubleshoot a large project or a program of projects. By altering the VDT model components, a modeler can experiment with different solutions to determine which one meets his program quality, cost, and scheduling objectives.

Using VDT’s graphical interface, project managers design the organizational structure—its size, the number of people in the group, and its topology—who reports to whom. The project manager also graphically assigns one or more activities for each individual within the group, as well as the dependencies between activities. The user sets other organization attributes such as skill levels of each actor (individual or subteam) and decision making policies. The skill level of each actor can be set to low, medium, or high. The higher the skill level of individuals, the faster the task gets done, and the lower the rate of exceptions generated.

Decision making policies include centralization, formalization and matrix strength. Centralization reflects whether decisions are made by senior management positions or decentralized to first level supervisor or worker positions. Formalization is the relative degree to which communication among positions takes place through formal meetings and memos vs. informally. Matrix strength models the “degree of collocation” of the various specialists in an organization by setting the probability that workers will attend to informal vs. formal communications. The above decision-making policies can all be set to low, medium (nominal), or high.

Figure 2.1 below is an example of a project organization that was graphically created by a user. Positions (actors) in the group are shown in green. Each position can represent an

individual (actor) or a group (subteam) encapsulating a number of individuals with the same skills and attributes. Supervision links between actors are shown in black lines. The purple box at the upper left corner of the figure represents a meeting and the grey links connecting actors to that box show the participants in that meeting.

Activities are shown in yellow boxes, and the activity assignments from actors to activities are shown with blue links. Cyan boxes represent milestones. There are three types of links between the activities:

- Successor links (black solid arrows between activities) link activities and milestones either “finish-to-start” (i.e., a successor task cannot start until the predecessor task or milestone is complete), or “start-to-start” (the successor cannot start until a specified duration or lag has expired from the time that the predecessor task or milestone starts).
- Communication Links (Green dashed arrow lines between activities) link two reciprocally interdependent (Thompson 1976) activities, indicating that the positions responsible for the two tasks must communicate with each other during completion of their tasks to coordinate the interdependency.
- Rework Links (Red dashed arrow lines between activities), which link a task to a dependent task that will need rework if the driver task encounters exceptions requiring rework.

By clicking on any of the above components (i.e., positions, activities, links, meetings, etc.), a user can set the appropriate attribute/s for that component. For example, by clicking on a position, a user can set several attributes for that actor such as the name of the position (e.g., project manager), its skill/s (e.g., electrical engineering, project management), its application experience (i.e., which indicates how much a person has applied their skills to projects similar to the one to which it is assigned), role (i.e., whether it is a project manager, a subteam leader or a subteam member), number of Full Time Equivalent (FTE – see section 3.2 for full description), etc.. Or, for example, by

clicking on an activity, a user can set the name of that activity, the work volume (in FTE-minutes, hours, days, weeks, or months), and the skill required to perform that activity.

As we will discuss in section 3.9, the graphical user interface of VDT in conjunction with the suggested near optimal designs produced by our EOD model creates a suitable environment for Human Computer Interaction (HCI) and motivates Out-of-Box Thinking (OBT).

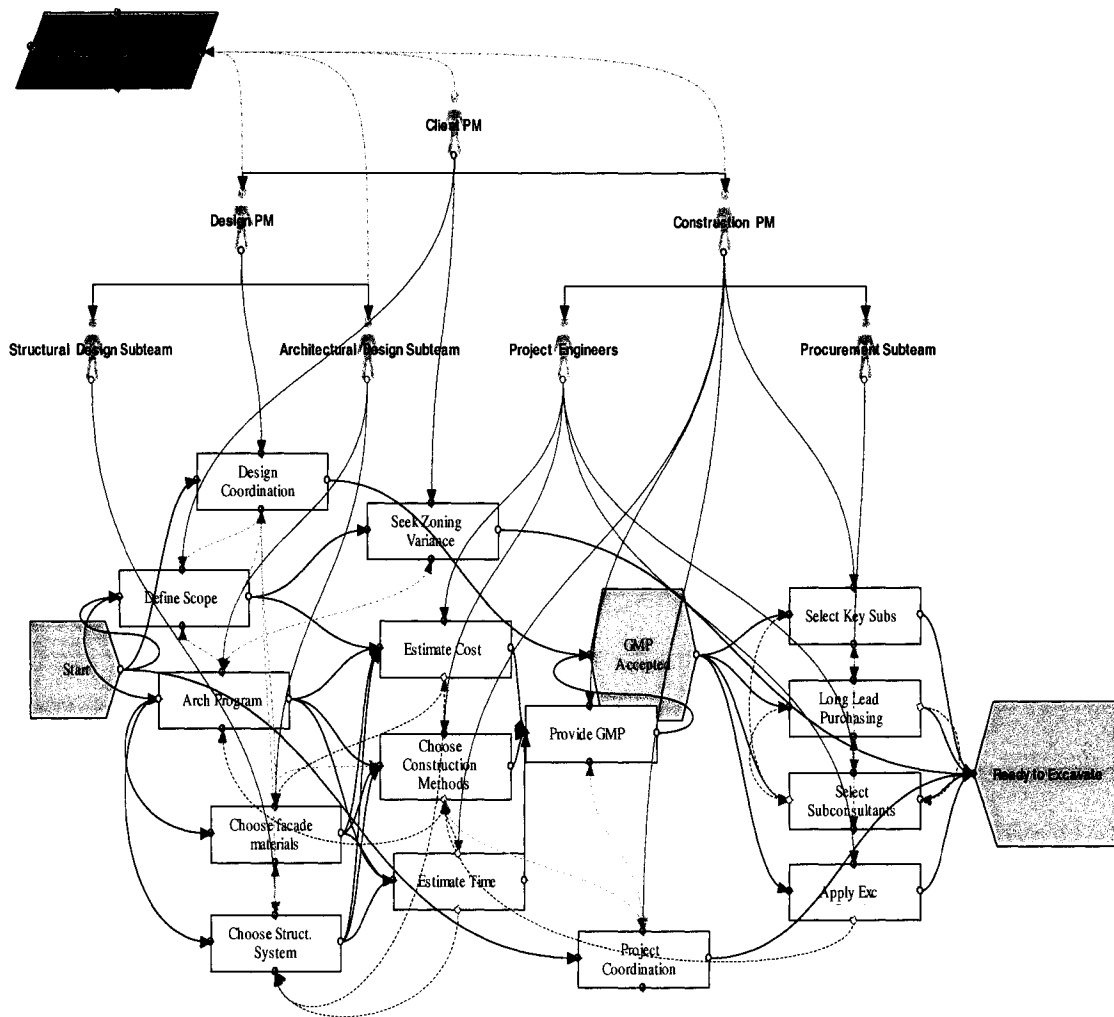


Figure 2.1 User Interface of the VDT Simulator

Each project participant fills a position in the project organizational hierarchy and works on one or more activities. The organizational structure and the interdependence links between activities define coordination requirements and rework propagation between positions.

Once the above attributes and topologies are set, the Monte Carlo discrete event simulation can be run (usually 50-100 trials is sufficient) and the model produces a set of project performance outputs. Gantt charts, quality risks, and position backlogs (shown in Figures 2.2, 2.3, and 2.4) are among a number of graphical outputs that VDT can produce. The user can then manually adjust the input parameters to obtain the organization performance output that is designed. In section 3.1, we will show how these outputs can be used in our EOD model to evolve project organizations based on the given criteria by the user. These organizational models and output charts are all produced using the SimVision® commercial implementation of VDT⁵.

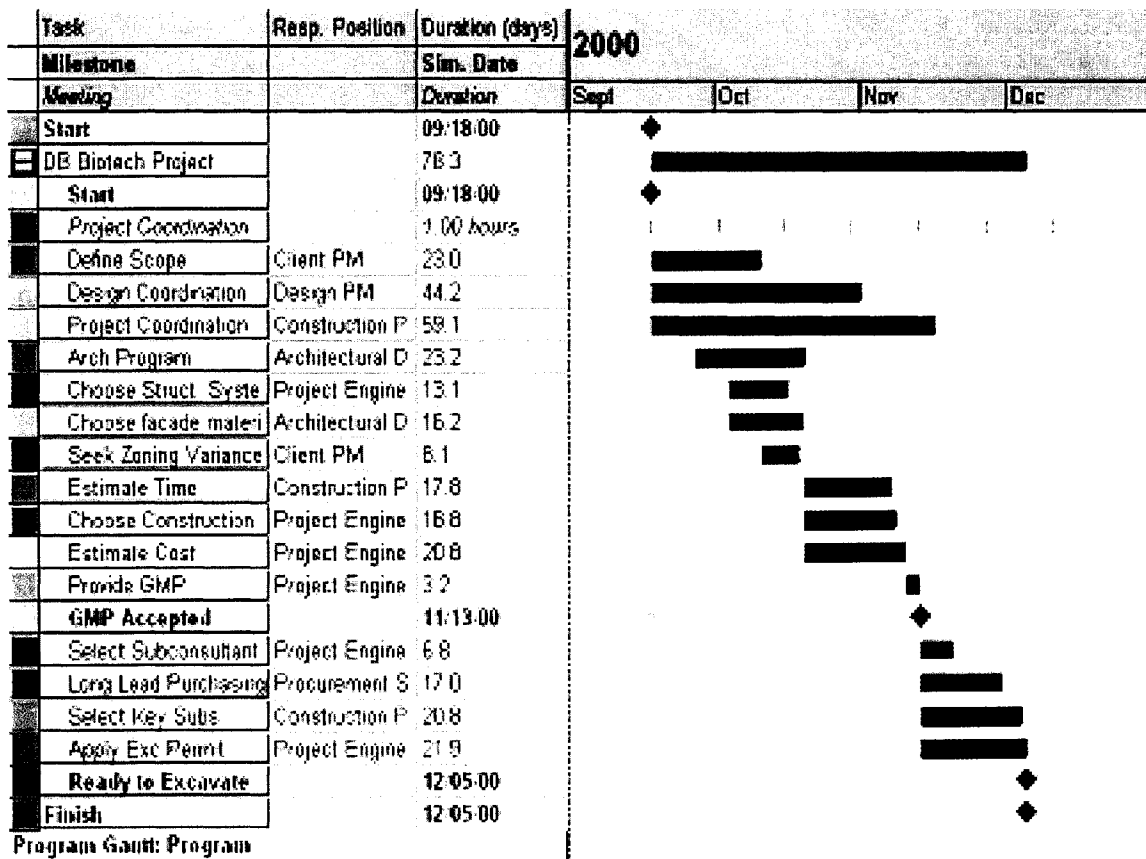


Figure 2.2 A Sample of Gantt Chart as one of the VDT Performance Output

A Gantt chart displays the simulated duration of each activity as a horizontal bar in front of the corresponding activity.

⁵ SimVision was developed by Vité Corporation, Mountain View, CA, under a license from Stanford University and is maintained and distributed by ePM LLC, of Austin Texas < <http://www.epm.cc> >

As shown in Figure 2.2 above, project simulated duration can be represented in a form of a Gantt chart. The VDT-generated Gantt chart displays the simulated duration for each activity in a horizontal bar across from the corresponding activity. If the activity is on the critical path, the bar is shown in red; otherwise, it is shown in blue. The green diamonds represent the planned milestone dates and the black diamonds represent the simulated milestone dates.

VDT also produces a variety of quality performance charts such as Function Risk Index (FRI) chart, Project Risk Index (PRI) chart, and Communication Risk Index chart. FRI, which is also known as the Component Quality Index or CQI, measures the risk to quality arising from *functional* exceptions. Functional exceptions are internal technical problems that affect only the task from which they arise. Any rework incurred applies only to that task. Rework links have no interaction with functional exceptions. In project work terms, FRI represents the likelihood that individual components produced by the tasks in this project will have internal defects based on failures of rework and exception handling.

PRI measures the risk to quality arising from uncoordinated “project exceptions” at the interfaces between tasks. Project exceptions are problems that arise in one task that may have an effect on work in another interdependent task linked to the first task by a communication and/or rework link. In the absence of rework links, project exceptions have no meaning. In project work terms, PRI represents the likelihood that the components produced by this project will not be integrated at the end of the project, or that the integration will have defects based on rework and exception handling. PRI is thus a measurement of the success of system integration.

In general:

FRI = Fraction of effort it would take to process ignored functional exceptions normalized by the total effort to rework all predicted functional exceptions.

PRI = Fraction of effort it would take to process ignored project exceptions normalized by the total effort to rework all predicted project exceptions.

The Project Communications Risk measures the risk that positions will handle communications about their tasks improperly. This process risk suggests possible product quality risk. The formula for calculating communication risk is:

$$\text{Communication Risk} = \text{Missed (ignored) Communications} / \text{Requested Communications}$$

Figure 2.3 below demonstrates a sample of a Project Communication Risk chart. As shown in this figure, any activity with a Project Communication Risk Index higher than 0.5 is shown in orange or red (i.e., implying a high risk of product quality failure).

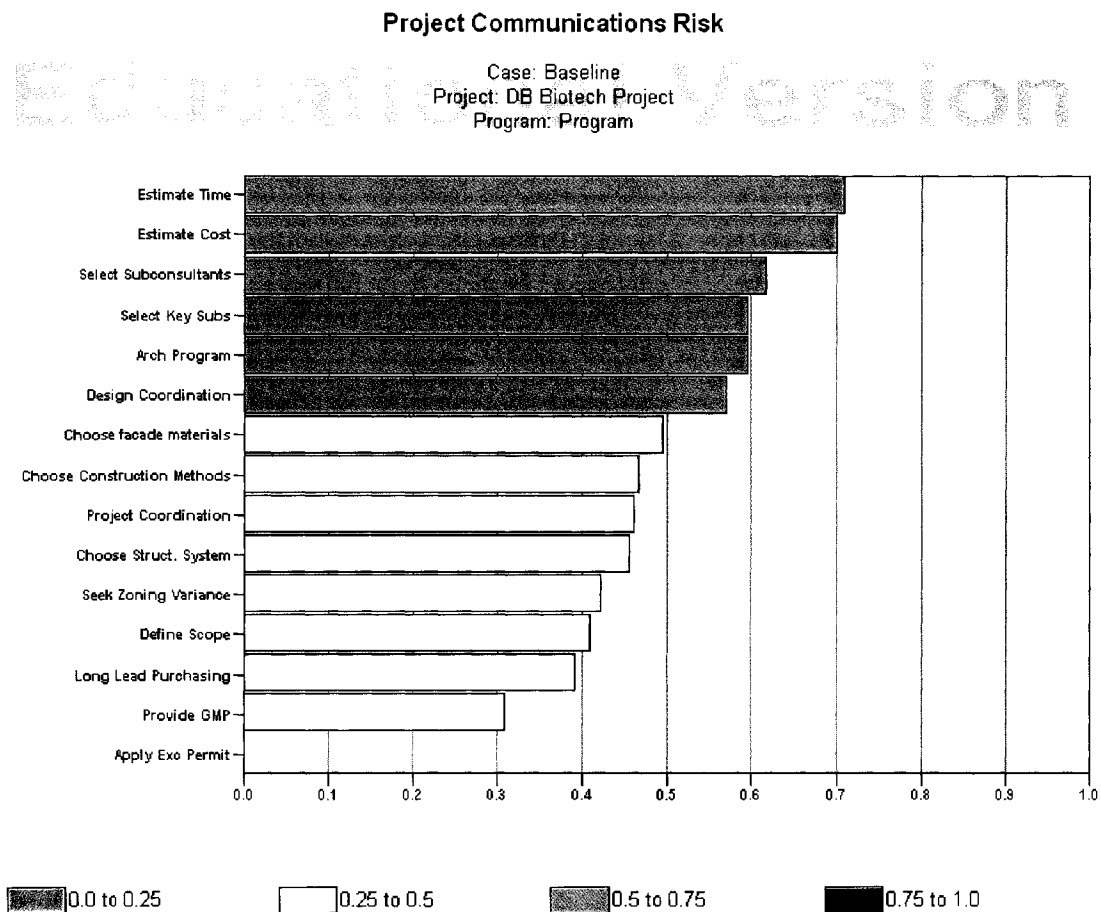


Figure 2.3A Sample of the VDT Project Quality Performance Output

Any activity with a FRI, PRI or communication risk index between 0.5 to 1.0 indicates a high risk activity. Quality risk charts highlight the tasks (and implicitly, the components produced by the at-risk tasks) that are at greatest risk of exception-handling or coordination failures.

The actor backlog output chart shows the backlog for each position in the organization, indicating the predicted pending workload for positions over time. Severe backlogs increase the likelihood that actors prioritize their own direct tasks over supervision and coordination tasks, and thus increase the risk of failures to their own, subordinates' and peers' tasks, due to the lack of required supervision and coordination. Figure 2.4 below shows a sample of a position backlog chart.

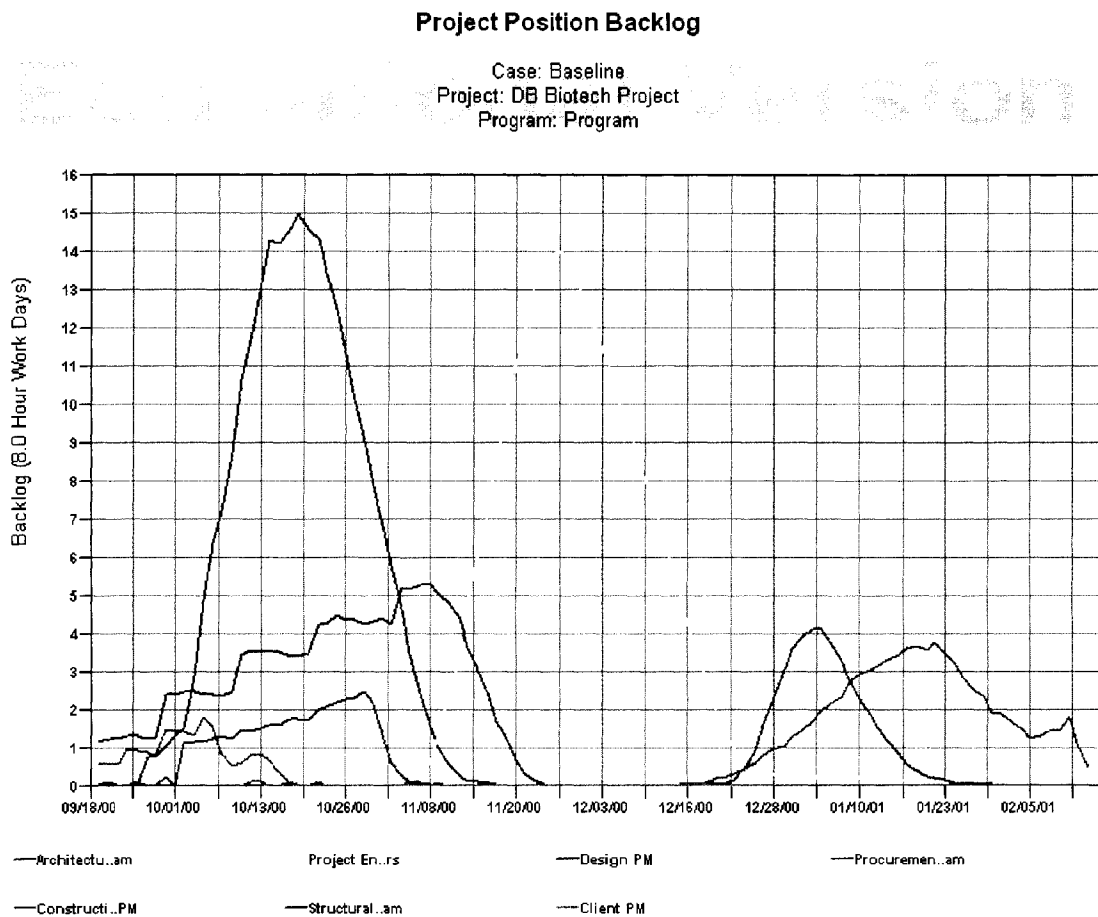


Figure 2.4 A Sample of the VDT Position Backlog Output

A Backlog chart shows the number of days of current work backlog for each position over the course of the project duration.

As we will show in section 3.6, we use some of the above performance outputs for measuring the fitness value for a generated project organization design.

The information-processing micro-behavior underlying VDT is best described in Ramsey and Levitt (2005):

“...VDT models represent work through inter-related tasks; actors communicate with one another while performing these tasks; and an organization structure defines roles and reporting relationships of the actors and constrains their behaviors. Figure 2.5 illustrates this view of tasks, actors and organization structure. As suggested by the figure, we model the organization structure as a network of reporting relations, which can capture micro-behaviors such as managerial attention, span of control, and empowerment. We represent the task structure as a separate network of tasks that can capture organizational attributes such as expected duration, complexity and required skills. Within the organization structure, we further model various roles (e.g., marketing analyst, design engineer, project manager), which can capture organizational attributes such as skills possessed, levels of experience, and task familiarity. Within the task structure, we further model various sequencing constraints, interdependencies, and quality/rework loops, which can capture considerable variety in terms of how work is organized and performed.

Each actor within the organization structure has a queue of activities to be performed (e.g., assigned work tasks, messages from other actors, meetings to attend) and a queue of information outputs (e.g., completed work products, communications to other actors, or requests for assistance). Each actor processes tasks at a rate and with an error frequency that depend upon a qualitative match between: the actor's skill types and levels vs. the skill required for a given task; the relative priority of the task; the actor's work backlog (i.e., queue length); and how many interruptions divert the actor's attention from the task at hand. Actors' collective task performance is constrained further by the number of individual sequential and parallel tasks assigned to each actor, the “work volume” of those tasks, and both scheduled (e.g., work breaks, ends of shifts, weekends and holidays) and unscheduled downtime (e.g., awaiting managerial decisions, awaiting work or information inputs from others, or performing rework).

Both primary work (e.g., planning, design, manufacturing) and coordination work (e.g., meetings, management, joint problem solving) are modeled in terms of work volume (measured in person-hours or person-days). Work volume is specified as full-time equivalent actors multiplied by time (FTE-Hours or FTE days) and represents the amount of information processing work associated with a task, a meeting, a communication, etc.

Thus, the VDT simulation engine employs both qualitative and quantitative reasoning. VDT alternates between qualitative pattern matching and numerical discrete event simulation. Results of the qualitative pattern matching adjust integer numerical variables such as numbers of missed meetings and real number variables such as error probabilities in the numerical Monte Carlo discrete event simulation part of the model.

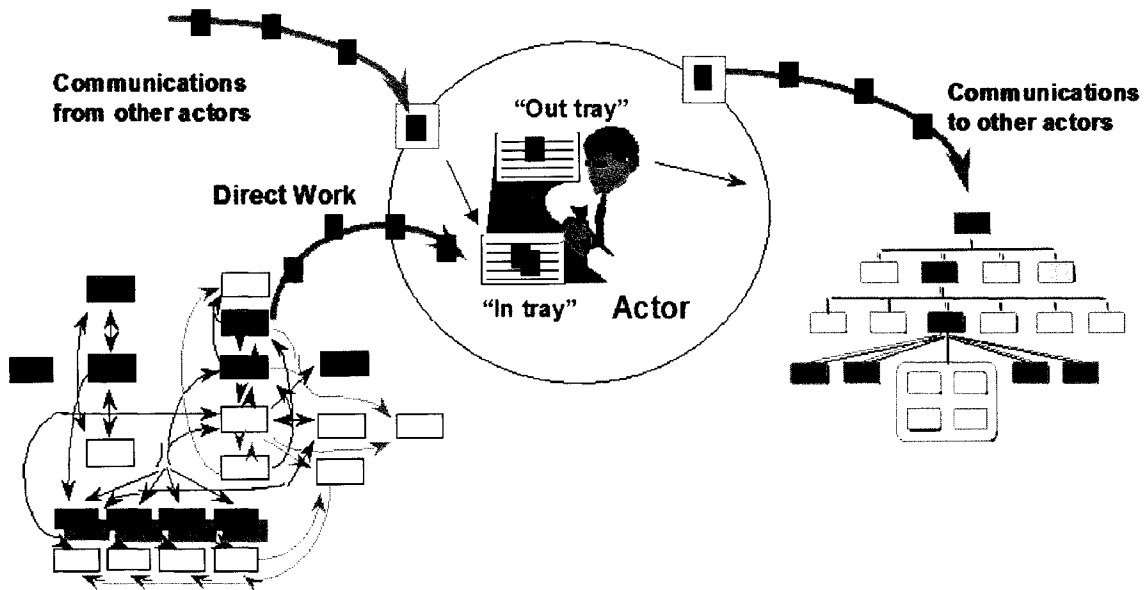


Figure 2.5 VDT's Information Processing View of Knowledge Work

VDT actors process both direct tasks and communications from other actors as “information-processing sub-tasks.” Subtasks arrive constantly during the duration of the project and accumulate in actors’ in-trays. When sub-tasks arrive faster than the actor can process them, the actor becomes backlogged, triggering delays and increased quality risks due to missed communications and meetings. Each sub-task is tagged with a specified work volume, required skill, arrival time and priority. Based on these tags, VDT actors stochastically select particular items from their in-trays to work on, experience exceptions in processing tasks, and determine which supervisory actors they will consult to help resolve exceptions that may arise. Unresolved exceptions, missed communications and missed meetings all increase the probability of exceptions in subsequent tasks.

VDT applies AI-style symbolic pattern matching—i.e., it reasons qualitatively, using pattern matching over nominal and ordinal variables, based on micro-behaviors derived from organization theory. In tandem, the discrete-event simulation engine steps a simulation clock forward in time using time steps as small as one minute to enable quantitative computation of work volume and elapsed time; and it tracks the number of missed communications, missed meetings, items of rework not completed, and other process quality metrics by task, actor and for the aggregate project team. Bridging between the qualitative and quantitative reasoning are a set of tables called “behavior files” which

represent the results of our ethnographic studies of actor micro-behavior in project teams. Each behavior file is a small matrix with about three rows and three columns containing a set of numerical values in each cell of the matrix. The qualitative inference engine in VDT reasons about nominal and ordinal variables such as actor role (one of: Subteam, Subteam Leader or Project Manager) and level of centralization (one of: Low, Medium or High) to pick a row and column in the behavior file matrix; the VDT controller takes the numerical value from this row-column intersection in the behavior matrix and passes it to VDT's quantitative, discrete simulation engine where it is used to reset a task's exception probability, adjust an actor's processing speed, etc. An advantage of this representation is that many details of the actor micro-behaviors in VDT can be calibrated over time by simply using a spreadsheet or work processor to change the values of entries in the behavior files—enabling non-programmers to develop and extend the model. Readers interested in additional details of the VDT model's implementation should see (Jin and Levitt, 1996).

Quantitative simulation places a significant burden on the modeler in terms of validating the representation of a knowledge-work process. It requires hands-on fieldwork to study an organization in action, and to formalize and calibrate the information processing micro-behaviors of its participants. Our computational modeling environment benefits from extensive fieldwork in multiple domains—e.g., power plant construction and offshore drilling (Christiansen, 1993); aerospace (Thomsen, 1998); software development, (Nogueira, 2000); healthcare (Cheng and Levitt, 2001); and other domains. VDT has been used since 1996 to teach classes on organization design at Stanford University and at more than 30 other universities worldwide. Through a process of “back-casting”—attempting to predict known performance outcomes of a past project using only information available at the beginning of a project—students in these classes have developed VDT models of real-world projects and demonstrated dozens of times that the outcomes predicted by VDT correspond well to actual performance outcomes for those projects (Kunz et al., 1998).”

VDT has several limitations that should be kept in mind.

- ❖ It models projects that are routine enough that all tasks and their relationships can be pre-specified; the organization structure and staffing are fixed; and the assignment of tasks to actors remains fixed. This implies that any expected change can be reasonably well modeled as simply adding workload to already defined tasks for pre-assigned actors.

- ❖ VDT actors have skill sets that remain fixed over the duration of the project, i.e., no improvements in skill levels occur due to learning.
- ❖ Exceptions are always referred up the hierarchy for resolution, rather than being referred to peers inside or outside the project in a community of practice.
- ❖ Inefficiencies or conflicts due to differences in actors' goals, values, beliefs or norms are not modeled.
- ❖ VDT assumes that uncertainty comes from the task, not the environment. For example, it does not model technical risk, market risk, weather-related risk, or any other aspect of the task environment. Environmental uncertainty has to be modeled by making different scenarios for good weather vs. bad weather in which the bad weather scenario, for example, might account for different task durations, additional weather protection activities, etc. in it. As we see in Chapter 5, it is because of this limitation that we chose technology instead of environment as the contingency factor for our validation purposes.
- ❖ VDT has no ability to search for near optimal designs.

Many of these shortcomings are being addressed in ongoing and future research to extend VDT (Levitt, 2005).

Since our EOD model uses the evolutionary computing approach in conjunction with VDT, in the following three sections, we provide some background on evolutionary computing methods and details on genetic algorithms and genetic programming.

2.3 Evolutionary Computation and Design

The main focus of evolutionary computation is to use mechanisms similar to natural selection to perform virtual evolution within a computer. Populations of structures are evolved in the memory of the computer based on the *survival of the fittest* as occurs in nature. These structures (called individuals) form in such a way that the genetic content of their offspring is related to the genetic content of the most successful parents.

Evolutionary computation employs a directed search that looks through the space of possible computer structures and finds solutions within hours for problems that would sometimes require random searches longer than the life of the universe (Kinnear, 1994). This evolutionary search is directed by the contents of the “genetic material” in the “fittest” segment of the population to portions of the search space that are likely to contain solutions (Holland, 1975).

While there are several types of evolutionary computation methodologies, such as Evolution Strategy (Rechenberg, 1965) and Evolutionary Programming (Fogel, Owens and Walsh, 1966), the most commonly used in recent years are Genetic Algorithms (Holland, 1975) and Genetic Programming (Koza, 1992).

But why use evolutionary algorithms for creating new designs? Peter Bentley (1999) in his book Evolutionary Design by Computers points out four main reasons:

“1. Evolution is a good, general purpose problem solver... Although it is not possible to state which of the evolutionary methods are good for which problems (Fogel, 1997), it is possible to identify methods that consistently produce improved results (compared to results produce by other techniques) for a wide variety of problems.

2. Evolutionary Algorithms have been used successfully in many types of evolutionary design... Other techniques such as simulated annealing, hill climbing, Tabu search and other methodologies have been applied in certain areas, but evolutionary algorithms such as genetic algorithms and genetic programming have been used much more often in almost all automated design systems.

3. Evolution and human design process share many similar characteristics. Some researchers claim that natural evolution and human design process are directly comparable (Fogel et al. 1966, Goldberg, 1991; French 1994). In fact, Goldberg actually attempts to formally define human design in terms of evolution by genetic algorithm (Goldberg, 1991). He compares the recombination of genetic material from parent solutions when forming a new child solution, with a human designer combining ideas from two solutions to form a new solution.

4. The most successful and remarkable designs known to mankind were created by natural evolution, the inspiration for evolutionary algorithms.

Indeed, the most complex and remarkable miracle of design ever created – the human brain – was generated by evolution in nature. Not only is it an astonishing design in its finished form, its huge complexity grew from a single cell using instructions contained in one molecule of DNA. This is perhaps the most conclusive demonstration of all that evolution-based techniques of evolutionary computation are highly suitable for design problems.”

As we will show in Chapter 4, one other reason that makes evolutionary methods, unlike conventional methodologies, a good choice for design applications is that it creates a *set of near optimal solutions* instead of a *single optimal solution*. This feature provides greater flexibility for a human designer to choose from among multiple near-optimal alternative design ideas for her application.

In addition, good designs require creativity and invention, which are different from logical deduction. While logical thinking usually plays an important role in setting the stage for an invention, it is insufficient for design and creativity (Koza, 2003). One can spend hours approaching a design problem with logic and still not be able to come up with a truly creative design. Project organization design needs creativity and ingenuity, which evolutionary methods facilitates well because they do not require or use an explicit knowledge base to guide their search.

During the last few years, evolutionary computational methods have been used to design and optimize various kinds of systems in ways that rival or exceed expert human capabilities. For example, Genetic Programming (GP) has produced design and optimization results for a wide variety of problems involving automated synthesis of electronic circuits (Koza et al., 1996 and 1999), antennas (Comisky, Jessen, and Koza, 2000; Lohn, Hornby, and Linden, 2004), optical lens systems (Koza, Al-Sakran, and Jones, 2005), and quantum computing circuits (Spector, 2004). GP has created 21 instances of evolved systems that duplicate the functionality of a previously patented 20th-century invention, seven instances where it has done the same with respect to post-

2000 patented inventions, and two instances where GP has created patentable new inventions⁶.

As we show in section 3.8, like project organization design, many of the above designs, such as synthesis of electronic circuits, involve defining both the topology and component values of different parameters. Thus, using GP methodology is a good approach for automated design of project organizations.

2.4 Genetic Algorithms - Background

A Genetic Algorithm (GA) is a parallel search methodology inspired by evolution. The GA concept was first introduced in John Holland's book (1975), Adaptation in Natural and Artificial Systems. In his book, Holland showed how the evolutionary process such as natural selection, inheritance, mutation and recombination, can be used to find solutions for a wide variety of problems using a highly parallel technique. GAs have been applied to a variety of problems including complex optimization problems (Beasley, Bull, and Martin, 1993), automated design (Renner and Ekart, 2003), machine learning (Goldberg, 1989), and scheduling (Davis, 1985).

GA transforms a population of abstract representations of candidate solutions (each called an individual) for an optimization problem into a new generation of the population. GA does this transformation using the Darwinian principle of reproduction and survival of the fittest, based on naturally occurring genetic operations as mentioned above. Individuals are combined according to their fitness value in order to generate new individuals that contain the best features of their parents.

There are four main steps for setting up a genetic algorithm problem. The user must specify:

⁶ Keane, Martin A., Koza, John R., and Streeter, Matthew J. 2005. Apparatus for Improved General-Purpose PID and Non-PID Controllers. U. S. Patent 6,847,851. Filed July 12, 2002. Issued January 25, 2005.

1. **A representation scheme**, which in conventional GA is the selection of a fixed-length character string (individual) that can represent a solution to the problem. A string (chromosome) is usually recognized with its length L and the alphabet size K. If the alphabet is binary, for example, K equals 2.
2. **A fitness function**, which assigns a fitness value to each individual in the population based on some performance criteria for the solution.
3. **GA controlling parameters** such as population size, maximum number of generations, and reproduction, mutation, and crossover rates.
4. **A way to designate the result and criteria for termination.** A commonly used method for designating the result of GA optimization for a generation is to designate the best individual obtained in any generation of the population during the run (i.e, the best so far) as the result of the run.

Once the above are specified, the following steps are executed in GA during a run until it terminates by finding an optimal or near optimal solution:

1. Create an initial population of individuals randomly.
2. Iteratively perform the following until the termination criterion has been fulfilled:
 - Each individual in the population is assigned a fitness value using the fitness function.
 - The individuals that are best fitted are probabilistically chosen to go through the following genetic operation to produce the next set of population:
 - **Reproduction** – Reproducing the exact copy of an existing individual for the next generation:



How can computers learn to solve problems without being explicitly programmed? In other words, how can computers be made to do what needs to be done, without being told exactly how to do it?

Genetic programming starts from a high level statement of what needs to be done and automatically creates a computer program to solve the problem. There are five major preparatory steps in applying GP to a problem. These are selecting:

- The terminal sets
- The function sets
- The fitness measure
- The parameters for controlling the run
- The method for designating a result and criterion for terminating a run

The terminals can be considered as the inputs to the undeveloped computer program. The set of terminals along with a set of functions build a computer program to solve, or approximately solve, the problem. Specifying terminal sets and function sets corresponds to the step of determining the representation scheme in GA, and the last three steps above are identical to the last three preparatory steps of GA. Like GA, GP also executes the three main steps, mentioned in section 2.4 to breed computer programs (individuals) that solve problems. Computer programs are either presented in a LISP S-expression format:

$$(* (+ 3.42 Y) (/ Z 1.21)) \quad (2.1)$$

which we would ordinarily express as

$$(3.42+Y) * (Z / 1.21) \quad (2.2)$$

or in a tree format as shown in Figure 2.6 below:

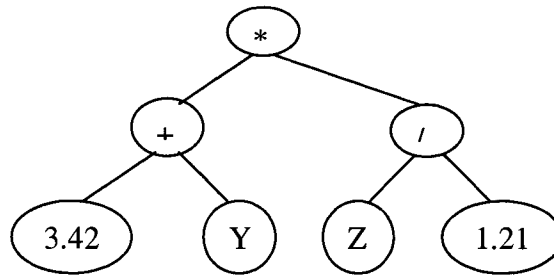


Figure 2.6 Sample of a Computer Program Represented in a Tree Format

Internal points of the tree (i.e., nodes) are defined as functions and the external points (i.e. endpoints) are defined as terminals. The crossover operation generates new offspring by exchanging sub-trees between the two parents. The mutation operation selects a mutation point (which can be an internal point or an external one) randomly. It then replaces whatever is currently at the selected point and below with a randomly generated subtree at that point. The generated offspring, although a new computer program, inherits some characteristics of its parents. For a detailed description of the genetic programming operations see section 2.3 of Genetic Programming III (Koza, 1999)⁷.

One of the main distinctions between GP and the conventional GA is that the size and the shape of the computer program in GP are variable and dynamically determined during a run, unlike GA which is a fixed-length character string. The issue with pre-specifying or restricting the size and shape of the solution to the problem is that it restricts the window by which the GP views the world and might well exclude finding the solution to the problem (Koza, 1994). Another important advantage of GP is that the structures going through adaptation in GP are active. Unlike GA, they are not passive strings that represent the solution to the problem. Instead they are active programs that are capable of being executed in their current form.

At the present time, GP is unique among methods of artificial intelligence and machine learning in terms of its duplication of numerous previously patented results, unique in its

⁷ Information about genetic programming can also be found in Koza (1990, 1992); Koza and Rice (1992); Koza (1994a, 1994b); Banzhaf, Nordin, Keller, and Francone 1998; Koza, Bennett, Andre, and Keane (1999); Koza, Bennett, Andre, Keane, and Brave 1999; Langdon and Poli (2002); Koza, Keane, Streeter, Mydlowec, Yu, and Lanza (2003a, 2003b).

generation of patentable new results, unique in the breadth and depth of problems solved, and unique in its delivery of routine high-return, human-competitive evolvable hardware (Koza et al., 2004)⁸. Although GP has been used in many different fields, it has not been used for optimizing project organization design, which is the focus of this research.

2.6 Related Research on Optimization of Project Organizations

Some attempts have been made in the past at developing a post-processor for VDT; however, those were of limited power and generality. For example, William Hewlett designed a rule-based “expert system” post processor for VDT that analyzes the outputs of a VDT simulation, and recommends small, incremental changes in the design of modeled organizations (Hewlett, 2000). Hewlett’s post-processor was tested in a design *charrette* on a group of Stanford students and showed that they were able to create better organizations when they used the post-processor than without it.

However, there were many limitations of this initial post-processor. First, the post-processor did not solve for the optimal organization; it was only a small piece of an optimization strategy. It primarily focused on team sizes and suggested reallocating personnel between teams. Thus, after running the VDT simulator, a user had to take advice suggested by the post-processor, make changes in the original design, run the simulator again, and observe whether the optimization was beneficial. Then the user had to repeat this optimizing loop until the desired output was achieved. As a result, this process was an exhaustive, never ending search. Second, although this process was shown to be beneficial for some students with less project management design experience, it provided less benefit for more experienced managers.

⁸ Recent work on genetic programming is reported in the Genetic Programming and Evolvable Hardware journal, the annual Genetic and Evolutionary Computation Conference (GECCO) [Deb, Poli, Banzhaf, Beyer, Burke, Darwen, Dasgupta, Floreano, Foster, Harman, Holland, Lanzi, Spector, Tettamanzi, Thierens, and Tyrrell 2004], the annual Euro-Genetic Programming conference [Keijzer, Tettamanzi, Collet, van Hemert, Tomassini 2005], the annual Genetic Programming Theory and Applications workshop [O’Reilly, Riolo, Yu, and Worzel 2004], the annual Asia-Pacific Workshops on Genetic Programming (Cho, Nguyen, and Shan 2003) and in numerous other conferences and journals in the field of genetic and evolutionary computation. For additional sources of information about genetic programming, including links to available software for implementing genetic programming, visit www.genetic-programming.org

Michael Murray (2004), another Ph.D. student in the VDT research group at Stanford, is conducting an ongoing research effort, which he calls the Engineering Vice-President's Problem (EVPP), to address a few selected aspects of the organization design optimization problem. The EVPP optimizer manipulates the project portfolio, project due dates, human resource investment, and design work process in an engineering organization to maximize that organization's profitability. The EVPP optimizer uses linear programming and discrete search techniques.

The focus of this thesis is complementary to Murray's research. While Murray's focus is on resource sizing and scheduling, we take these as inputs and the concentration of this research is on evolving key individual and subteam organizational attributes such as participants' skill levels; the properties of key decision-making policy variables, such as high, medium or low levels of centralization, matrix strength, and formalization; and the topology of organization structures such as the reporting hierarchy and actor-task assignments.

Prof. John H. Miller and his group at Carnegie Mellon University have done similar work in terms of evolving organizations (Miller, 2001), but for much simpler structures than the ones discussed in this thesis. In their research, they show that simple adaptive mechanisms allow for the creation of superior organizational structures. In addition, they conclude that, while they do not have proofs of optimal structures, the genetic algorithm was designed to solve difficult, nonlinear problems, and thus the structures that emerge from the algorithm should contain valuable hints about optimal form.

We introduce additional discussions of related research in the area of computational modeling and evolutionary methodology in later chapters.

⇒ Chapter 3

Evolutionary Organization Design Approach

Success is often the result of taking a misstep in the right direction.

-- Al Bernstein

This chapter will describe in detail how we designed and implemented the Evolutionary Organization Designer (EOD) model. We will explain how we used a Transforming Genetic Tree (TGT) to represent transformations to the organization structure and policies, instead of representing the organization directly, in order to evolve the initial organization design given by the user. Furthermore, we will discuss some of the challenges and obstacles that we faced during the design and implementation phases of our GP postprocessor as well as the advantages and limitations of EOD.

3.1 High-level view of the Evolutionary Model for Project Organization Design

A high-level view of how the evolutionary computing model is used for designing near optimal project organizations is as follows:

First, the model initially generates a population of random but valid organization designs as minor variants of the original user input and a given set of constraints. Each individual design is then rated by a fitness function defined by the user. The fitness function reflects the importance of different measures of organizational performance to the user. For example, in a given project, meeting a fixed completion deadline may be the highest priority and total capital cost might not be as important, while in another project quality is the most important factor and the organization has more flexibility with regard to schedule and cost. The various performance outputs that VDT produces (see section 2.2) are used in the EOD fitness function to measure a fitness value for a given design (see section 3.6).

Next, individual designs are selected probabilistically for “survival”—i.e., each design’s chances of being chosen are proportional to its fitness as defined above. The selected designs then go through the genetic operations such as reproduction, crossover, and mutation, as mentioned in section 2.4, to generate a new population of individual designs. The steps repeat until an optimal or near optimal solution is found, as shown in Figure 3.1. In this figure, only 3 individual organizations are shown as a representation of the population in that generation. However, the actual EOD model uses hundreds or sometimes thousands of individuals in each generation.

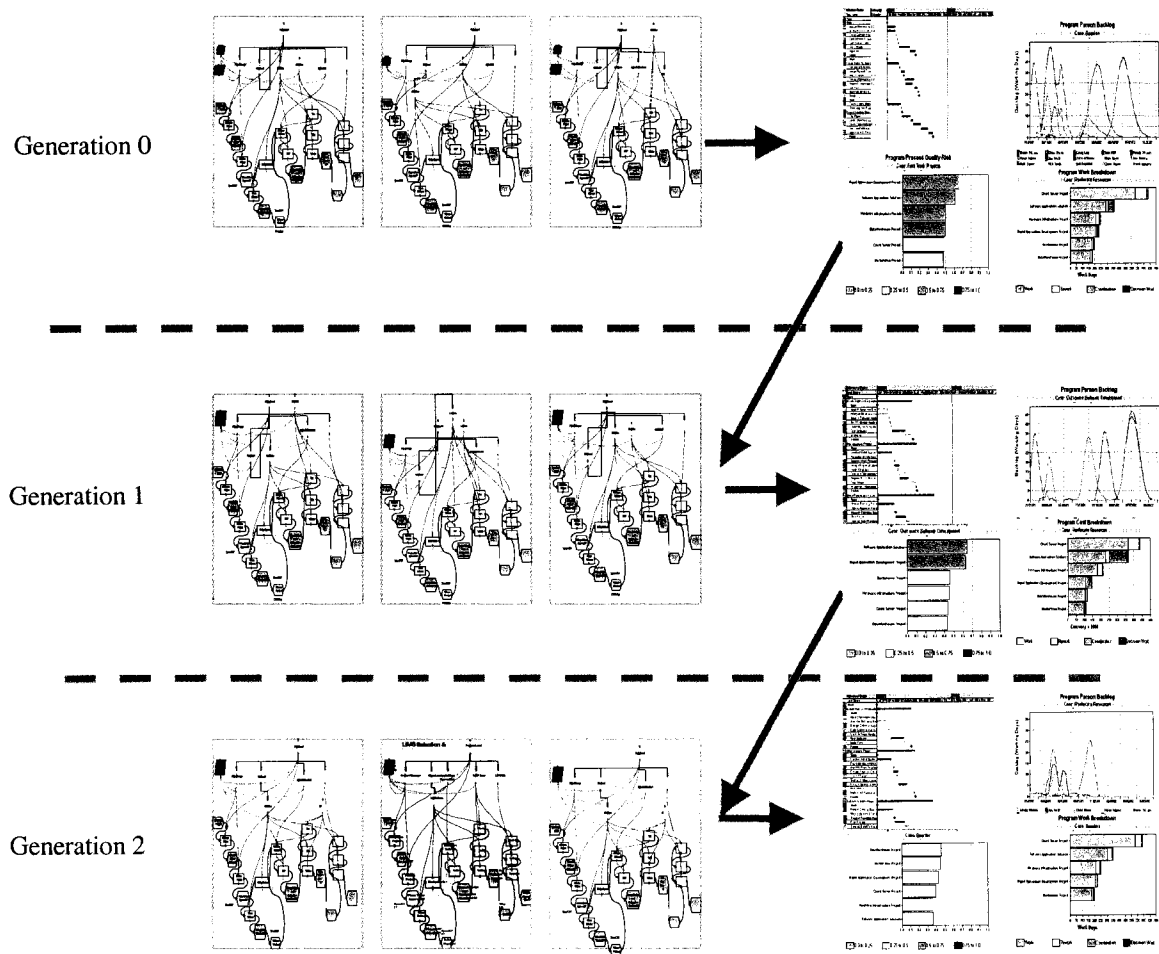


Figure 3.1 Evolutionary Computational Approach for Optimizing Organization Designs
 Individual designs are selected for fitness in terms of desired output, and go through operations of genetic mutation and genetic crossover (as in sexual reproduction) to create new generations with improved fitness, until an optimal or near optimal solution is found. Notice the shrinking durations (upper left) and backlogs (upper right), and the quality risks (lower left) going from orange to yellow over these three generations).

3.2 Design Scope

As mentioned earlier, we used the Virtual Design Team (VDT) as the primary computational modeling tool to describe and analyze the performance of a given project organization. In VDT, as described in Chapter 2, there are a number of different attributes that a manager can use for designing her/his project organization. These attributes can be grouped in four different categories:

1. Individual Positions' attributes such as:
 - Full Time Equivalent (FTE)
 - Skills and skill levels
 - Percent of attention allocation to each activity
2. Project attributes such as:
 - Decision making policy (centralization, formalization, and matrix strength)
 - Team experience
 - Who should participate in which meetings and how often
3. Organization Topology attributes such as:
 - Supervision Hierarchy
 - Which actor is assigned to each activity
4. Activities' attributes and relationships such as:
 - The name and the work volume of each activity
 - The sequencing constraints among activities
 - The reciprocal and rework dependencies between pairs of activities

In this research, we chose only a few of the design variables from each of the first three categories mentioned above. For the scope of this research we assumed the number of activities and their duration, the precedence and interdependence relationships between activities are given and fixed. As mentioned in introductory chapter, optimally scheduling activities and sizing resource pools is the focus of research by Michael Murray (2004), another PhD student in our research group.

Thus, from the above categories, we narrowed our focus to the following design variables:

- ***Skill levels of the actor, which is:***
 - level of each specific skill of each actor (position) that can be set to low, medium, or high
- ***Full Time Equivalent (FTE) of actors, which is:***
 - a measure of an actor's (position's) capacity to perform a task. For example, a position with an FTE value of 3 has the equivalent of 3 full-time employees to perform assigned tasks.
- ***Attention allocation of actors to activities, which:***
 - states what fraction of the actor's (position's) capacity is devoted to the task.
- ***Decision making policy variables, which are:***
 - ***Centralization***-The qualitative degree to which decision-making and exception-handling responsibilities are decentralized to individual responsible positions (Low) or are centralized to senior project managers (High)
 - ***Formalization***-The relative degree to which communication among positions takes place informally (Low) or through formal meetings and memos (High).
 - ***Matrix Strength***-The extent to which positions are located in skill-based functional departments and supervised directly by functional managers (Low) or co-located with other skill specialists in dedicated project teams and have project supervision from a project manager (High).
- ***Organization structure (supervision hierarchy), which determines:***
 - Who supervises whom
- ***Activities assignments to actors, which determines:***
 - Which position is responsible for each activity

We limited the scope of research to the above design parameters for two primary reasons:

- Since this was an initial proof of concept, we wanted to narrow our focus, find viable solutions to the problems of representing organizational design alternatives and project fitness functions, and show whether this approach can produce promising results, before covering all organizational design attributes.
- There were already similar problems attempted by humans using the same design attributes against which we could compare our results.

In the next section, we discuss some of the challenges and obstacles that we faced during the design and development of EOD.

3.3 Design and Implementation Challenges

There were three types of challenges that we faced when we started this research project. The first challenge was to find a representation of the organization design in a genetic programming form, so it can be evolved through generations. Although as shown in section 3.1 the general concept behind our approach was simple, the representation of a project organization in a genetic programming format so that it covers both the attributes and the topology of an organization was a challenging task.

The difficulty was increased since we had to combine individual attributes (such as individual skill levels and FTEs) with organization attributes (such as centralization, formalization and matrix strength), and with topological relationships (like supervision hierarchy and activity assignment) all into one genetic tree. At the same time, we had to ensure that whenever they are crossed over or mutated during genetic operations, the genetic tree produces a meaningful design.

Moreover, we needed to come up with a design that was complex enough to cover all features mentioned above, and simple enough so that it could produce good results in a reasonable time with our limited software and hardware availability. For example, Koza et. al. (1999) have been able to design patentable electronic circuits using 1000 Pentiums⁹

⁹ see <http://www.genetic-programming.com/machine1000.html>

running in parallel for about a month. However, in our case we wanted to come up with a design that could produce results in a matter of minutes or hours using a single a PC (i.e., we felt that it was not feasible to wait for about a month to produce a project organization – at least at this preliminary, proof of concept stage of our research).

The second challenge was to come up with a representative fitness function that could cover the multi-objectives, multi-constraints nature of the project organization problem. To come up with a precise fitness function that can represent schedule duration and quality outcomes, and at the same time satisfy constraints such as limits on the maximum numbers of FTE staffing that can be added to positions, was not an easy task.

The third challenge was to overcome some of the difficulties that we confronted in implementation of our design. These issues are discussed in section 3.7 of this chapter.

3.4 Representation of Project Organization

The biggest challenge in this research project was to represent the topology and attributes of a project organization so that it would be implemented using GP. There are many ways to represent a project organization; however, a model that can work with GP and yet fully represent the project organization was a challenging task. Appendix A, displays some of the alternative GP representation that we evaluated. These representations either did not work or they were difficult to implement with our limited hardware, programming knowledge, and time constraints due to the focus of this research project.

Our final approach uses a **Transforming Genetic Tree (TGT)**, described in detail in the next section, **to represent a set of legal transformations to the organization, instead of representing the organization itself** (i.e., instead of directly manipulating the organization attributes/topology mentioned in section 2.2, we manipulate the transformation to those objects at one abstraction level higher).

The general idea of using TGT is similar to what was used in synthesizing electronics circuits using GP (Koza, 1999). However, there is a significant difference. Koza's model

creates a circuit design from scratch by inserting, refining and manipulating electronic components to perform a certain outcome; our model uses an existing project organization and manipulates the organizational attributes and topology to achieve certain goals with the given constraints.

In section 3.9 we show how our approach (starting from an existing design instead of from scratch) can create a powerful Human-Computer Interaction environment that can motivate humans to do Out-of-Box Thinking (OBT). But first let's see how TGT works.

3.5 Transforming Genetic Tree (TGT)

TGT is a set of instructions represented in a genetic tree format that manipulates the organizational attributes and topology to produce a new organization design. Using GP techniques we evolve these TGT programs that, in turn, operate on and transform the organization. This approach represents evolution at one level of power and abstraction higher than directly evolving the organizational attributes and topology. We believe that the approach was a key contributor to the success of EOD.

The design of TGT, although similar to some of the other genetic programming trees used in a variety of application domains, has its own unique characteristics. For example, TGT uses a combination of attributes and topological functions along with encoding functions to produce a set of instructions. When applied to the current project organization, these instructions create a new design. Also, unlike some other genetic trees, it does not create a design from scratch and/or add expressions to the current design. Rather, it creates instructions that transform an existing organization design to a new form with different attributes and topological structure (see also section 3.8).

In a TGT, there are two groups of function sets and two groups of terminal sets (see section 2.5 for definitions of function and terminal sets):

- Function Sets:
 - Attributes and topological functions that can only appear as immediate parents of terminals such as: FTE, Assign, Aloc
 - Encoding functions that create instructions for setting attributes and topological changes such as: Up, Down, Same

- Terminal Sets:
 - Actor terminals – $P_1 \dots P_n$
 - Decision making policy terminal – CFM

A combination of the above functions and terminals produces an instruction for transforming the current organization design. The general idea of how TGT works is as follows (see appendix B for a more detailed description):

Terminal set P1 through P7 represents people (actors) in the group. CFM stands for Centralization, Formalization and Matrix Strength. Functions Up, Down, Same can have different meanings depending on the Terminals that they connect to and whether there are FTE, Assign or Aloc functions in between. For example, the function FTE increases or decreases the number of FTE for each actor depending on the number of Up/Down functions preceding it in the genetic Tree. The Assign function assigns an activity to an actor, and Aloc specifies percentage allocation for the responsible actor's time to the given activity.

A combination of these functions and terminal sets transforms the initial project organization suggested by a project manager through several generations into a near optimal one. Figure 3.2 shows a sample of a transforming tree produced by this genetic operation. In this configuration, for example, the skill attributes of P3 in the organization structure changes based on the type of its parent and grandparent nodes. So, in this case, P3's first skill level (e.g. Project Management) remains the same, her second skill level (e.g. Software Engineering) increases, and her third skill level (e.g. Design Coordination) decreases. In the sample tree below, CFM's parents and grandparents are Same, Up, and

Down. So, in this case, the genetic program tree suggests that centralization should remain the same, formalization should increase by one level, and matrix strength should decrease by one level to optimize the overall project outcome.

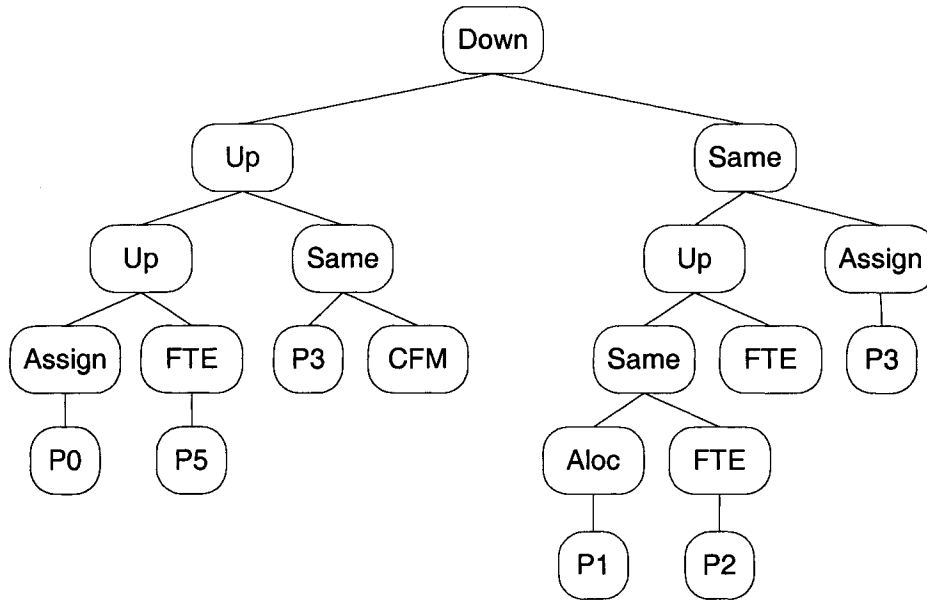


Figure 3.2 Sample of a Transforming Genetic Tree

Program trees created by genetic operations modify the structure and attributes of a project organization. The genetic tree above transforms an organization design (not shown here) proposed by a project manager to a near optimal one.

The reporting hierarchy is changed using the same branch that transforms the skill level of each actor. However, we use a switch to use the branch as either the skill level branch or the topology branch (for more details see appendix B). For reporting hierarchy changes, we added the following constraints to the code to avoid creating loops:

- A project manager can only report to a project manager or to no one.
- A subteam leader can only report to a project manager, another subteam leader, or no one.
- Subteam actors can report to a project manager, a subteam leader or another subteam actor, or no one.

These constraints would not, in fact, need to be added if VDT produced “bad” results any time there was a loop or role reversal (e.g., a PM reporting to a subteam member) in the reporting hierarchy, since the fitness function could filter out those nonsensical cases.

However, VDT does not check for reporting loops or role reversals, and was developed and validated for cases where none of these exist. Thus, we chose to inject these rules that embody “reasoning about appropriateness” into the evolutionary process to avoid such cases and save computing cycles.

3.6 Fitness Function

In Chapter 2, we described the three main outputs of VDT: Time, Cost and Quality. In order to measure the fitness of each individual in each generation we needed to come up with a fitness function, which contains the three factors. However, since they are of either different units (i.e., days or dollars) or dimensionless units (i.e., quality risk index), we needed to convert all either to the same units or dimensionless units.

Moreover, the three factors mentioned above are not all equally important in any given project. That is, in one project, meeting a tight schedule is of a great importance, and in another project quality may be the critical factor for success. Thus, it was clear to us that in our fitness function each of the above factors should have a weight associated with it. Thus a generic form of a fitness function for a project organization problem would look like this:

$$F_{(T,C,Q)} = T * W_T + C * W_C + Q * W_Q \quad (3.1)$$

where

T, C, and Q are Time, Cost and Quality and W_T , W_C , W_Q are the associated weights for Time, Cost and Quality.

In order to bring everything into comparable units, it would make sense to bring everything into dollars. That is, depending on a project, a day of delay in schedule duration beyond a certain deadline would cost the company a certain amount of dollars; similarly going below a certain quality level can be associated with a defined cost. So, by adding the Dollar Conversion Factors (DCF), our fitness function would look like this:

$$F_{(T, C, Q)} = T * W_T * DCF_T + C * W_C + Q * W_Q * DCF_Q \quad (3.2)$$

where

DCF_T , DCF_Q are the associated Dollar Conversion Factors for Time and Quality

It is important to notice that setting the right weights and DCF mentioned above is highly situation-dependent and could vary depending on the specific project criteria. In addition to the above factors, other constraints (e.g., limits on staffing levels) can be added into the fitness function to penalize the fitness value if these constraint values exceed their allowable limits.

For the two experimental case studies that we used to test our EOD model, namely the biotech plant case and ASIC design case described in Chapter 4, we added one representative additional constraint — a constraint on the maximum permissible increases in FTE staffing levels — to the fitness function.

In these cases, in each successive generation, evaluation of the fitness function is calculated based on three primary performance metrics. First, the total simulation duration, or number of days to complete the project. Second, the quality risk values including communication risk, functional risk, and project risk. Third, the total FTEs, since there was a constraint that a maximum of three additional FTEs could be allocated to the project. Formula 3.3 below shows how weighting factors are applied to these inputs to calculate the total fitness value. Note that the weighting factors are designed such that the fitness function heavily penalizes violations in the quality risk or FTE constraints.

$$SPD + TFTE * FTEW + \sum_{i=1}^M (FRI_i * FRIW_i + PRI_i * PRIW_i + CR_i * CRW_i)$$

(3.3)

Where

SPD = Simulated Project Duration

TFTE = the Total FTE added

FTEW = FTE Weight (if TFTE > maxAddedFTE equals FTEpenalty otherwise 1)

FRI_i = Functional Risk Index for activity i

FRIW_i = FRI weight for activity i (if FRI_i>FRIThreshold equals FRIpenalty otherwise 1)

PRI_i = Project Risk Index for activity i

PRIW_i = PRI weight for activity i (if PRI_i>PRIThreshold equals PRIpenalty otherwise 1)

CR_i = Communication Risk for activity i

CRW_i = CR weight for activity i (if CR_i>CRThreshold equals CRpenalty otherwise 1)

M = Number of activities

For the experimental cases mentioned above, maxAddedFTE was set to three and the FRIThreshold, PRIThreshold, and CRThreshold were set to 0.5, as they were given as the problem constraints. The FTEpenalty, FRIpenalty, PRIpenalty, and CRpenalty were set to 1000.

3.7 Implementation of EOD model and Integrating the System as a Whole

Creating the interface between our model and VDT was also a challenging task. A few steps needed to be taken so that the EOD model could work as a whole.

1. The GP program should generate the genetic trees.
2. A transforming program should decode the genetic trees to a set of instructions that modify the existing project organization.
3. The existing VDT project model needs to be converted into an XML format, so our model can make the necessary changes to the XML file according to the instruction set produced by the transforming program. This step was easy, since SimVision already produces XML-format model descriptions and outputs.

4. A new XML file should be produced based on the suggested modifications by TGT.
5. A program (SimVision) should run the VDT simulation using the newly generated XML file and write its output results into another XML file.
6. A program should read the output file and, based on the associated fitness function described in the previous section, generate a fitness value.
7. The fitness value for each solution that is a member of that generation should be fed back to the GP program, so the next generation of TGTs can be produced by selective genetic operations of reproduction, recombination and mutation.

Figure 3.3 displays a flow chart that represents how the EOD GP transforming tree optimizer was integrated with VDT.

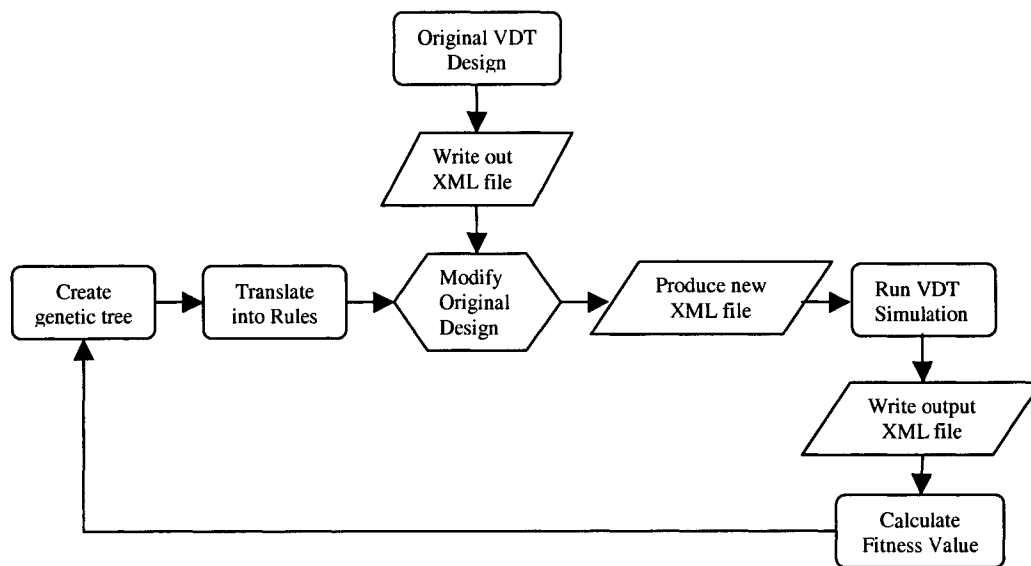


Figure 3.3 Flow Chart of the Integrated EOD/VDT System

3.7.1 Software/Hardware used

All programs necessary to implement the above steps were written in Java. We used the ECJ 10, the Evolutionary Computation and Genetic Programming System by Sean Luke, as our primary GP engine for producing the genetic trees. The runs were executed on a PC with a Pentium® 4 CPU – 1.71 GHz with the Windows 2000 operating system. The VDT simulations were performed using SimVision-R, a research version of the

SimVision commercial implementation of VDT that allows for API extensions. SimVision-R was provided to us by ePM, LLC.

3.7.2 Genetic Tree Constraints

One of the other challenges in implementing our TGT design was to set a constraint, so the attributes and topological function sets can appear only in the designated area of TGT. So, since FTE, Aloc, and Assign functions can not appear anywhere but next to the bottom of the genetic tree (i.e., Terminal sets should be the only children of the FTE, Aloc and Assign, and Up, Down, and Same can not be the children of the Attributes and topological functions mentioned above), constraints were added to the ECJ parameter files to enforce such limitations (See Appendix C for an example of the modified ECJ parameter file which include these constraints). Once the constraints were added, the genetic program produced only valid genetic trees, as shown in Figure 3.2.

3.7.3 VDT Setup and Duration of each Run

We used the following VDT setup for running the experimental case studies mentioned in the next chapter:

Number of Trials: 50 VDT is based on a stochastic Monte Carlo simulation engine, so each trial produces a slightly different result. This is the number of times the VDT simulation runs to produce an average simulated result.

Random Number Seed: 123 We used an arbitrary fixed number such as 123, so we could compare our results from each run more precisely. If seed number 0 is chosen, the internal VDT Monte Carlo simulator starts from different initial seeds, thus the outcomes can be slightly different.

Information Exchange Probability: 0.5 The information exchange probability measures the level of communication between positions that are responsible for interdependent tasks linked by communications (green) links.

Noise Probability: 0.1 Noise is a way to measure the effect of interruptions in the ordinary working day that take time away from doing the project tasks.

Functional Error Probability (FEP): 0.1 FEP is the likelihood that an activity will generate “functional” exceptions that require rework only to that activity.

Project Error Probability (PEP): 0.1 PEP is the likelihood that an activity will encounter “interface” exceptions that require rework for itself and for all failure-dependent activities (activities connected to it by rework links).

Depending on the size of the problem and the number of trials for each run, each VDT run for a suite of trials on a machine specified in section 3.7.1 could take from a few seconds to a couple of minutes. For the two real-world projects, which we discuss in the next chapter, each set of 50 simulation trials took about 0.32 second. Thus, for example, an EOD run with a population of 1000 individuals in each generation and a maximum run of 50 generations could take about $0.32 * 1000 * 50 = 4.51$ hours and about 27 minutes for a population of 100 individuals.

3.8 Similarities and Differences between EOD and other GP models

The method we used for designing the EOD is similar to what has been used in designing an improved version of Astrom and Hagglund’s PID controller (Streeter et al., 2003). Instead of redesigning a project organization from scratch¹⁰, we used an existing design

¹⁰ Note: Starting from scratch or a chosen good design as a starting point is a characteristic of GA.

done in the traditional human-generated way, and tried to adjust various attributes and relationships to improve the final outcome of the project. However, there is a difference between the two methodologies. For the case of PID controller, Streeter et. al. used GP to evolve four mathematical expressions which, when added to the four mathematical expressions devised by Astrom and Hagglund, yield improved overall performance. However, in our case, the TGT would actually transform the whole initial project organization design to a new form rather than adding something to the existing design. It is worth mentioning that, at the time that we first came up with this design, we were not aware of the Streeter et. al. work, which was done about a year earlier.

As mentioned earlier in section 3.4, the general idea of using a Transforming Genetic Tree (TGT) was obtained from Koza's several patentable inventions on synthesizing electronic circuits described in his book, Genetic Programming III (Koza, 2003). We both use a form of genetic tree to generate near optimal designs that can meet certain objectives and satisfy certain criteria. The noticeable similarities between electronic circuits and project organization designs, and the fact that GP had produced promising results in electronic circuit design, were the key factors that motivated us and convinced us that this would be the right approach. These similarities are:

- Electronic circuits have components such as capacitors, resistors, amplifiers, etc., which have attributes (microfarads, ohms, etc.) and are connected in a certain topological circuit structure in order to perform a specific function. On the other hand, individuals, groups, and activities are the components of a project organization, which have attributes (skills, application experience, etc.) and are structured and linked in certain topologies to produce a specific organizational outcome such as the design of a new product.
- Electronic circuits deal with flows of electricity or electrons, and project organizations deal with flow and exchange of information between individuals and groups within the organization.

These similarities are shown graphically in Figure 3.4 below.

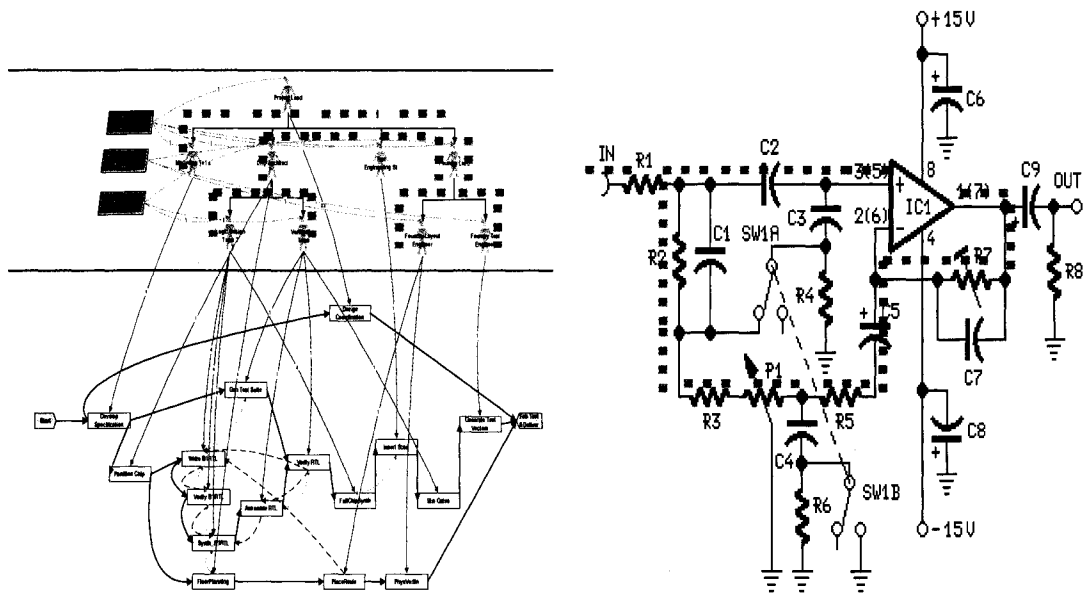


Figure 3.4 Similarities between Organizations and Electronic Circuits

Orange dots in the left graph above demonstrate the flow of information between actors in an organization. Similarly the orange dots in the right hand graph demonstrate the flow of electrons in an electronic circuit.

In spite of these similarities, when we tried to design our genetic tree, we noticed an evident difference. In electronic circuits, components are described by a single attribute and are thus generic (i.e., a 50 ohm resistor is functionally identical to any other 50 ohm resistor, and similarly for capacitors, etc.) so the same generic resistor or capacitor can be reused in several places in a circuit. In contrast, in a project organization, actors tend to have unique sets of skills, roles, experience levels and so on; similarly activities have unique combinations of work volume, skill required, complexity and uncertainty levels, and so on. Thus if actor P1 or Activity A1 shows up in one place, it cannot show up in another part of the organization structure, and hence in another part of the transforming genetic tree). This difference, unique versus generic components, made it a challenging task for us to come up with a TGT design that could produce meaningful instructions without replicating actors or activities in going through genetic operations.

3.9 Advantages and Limitations of EOD

The current design of EOD has several unique and advantageous characteristics as well as a few limitations. The fact that EOD, which at the heart of it is TGT, can represent a complex multidimensional optimization problem such as project organization design so it can be evolved by GP and produce near optimal solutions is unique and novel.

The fact that EOD can start from any point and take any given design, whether generated by human or machine, and evolve it toward optimality is a great advantage for two reasons:

1. The best solutions produced by GP can always be given back to EOD for further optimization to search for a better 'fit', if any. This, in fact, is a 'self-test' for the current EOD solution!
2. Since EOD can build upon human designs, it can motivate and support human-computer interaction. For example, when EOD generates a new design, it can create new ideas for the human designer. The designer can then modify her/his initial design or the new EOD-generated design and return it to EOD to search for better fitted designs; and this cycle can continue. Eventually, this can generate a design environment that motivates humans to think out-of-the-box for new ideas and see themselves working in cooperation with the machine rather than competing against it. If EOD were to start from scratch each time, a user might feel that her/his design would be useless since the machine would be starting from scratch anyway. This characteristic may be restricted to the domain of organization design and not, for example, to the case of circuit design. Changes to the manager's project organization design that were made by the machine can be visually and graphically seen and comprehended by humans at the granularity that we model organizations (about 50 tasks, 20 actors, maximum). The same is not necessarily true for a similarly complicated electronic circuit with 70 components.

In essence, with this design, the human is getting ideas from the machine for doing better designs; and the machine is getting ideas from the human for doing better designs. The author calls this phenomenon HCI@, which reads Human Computer Interaction Helix.

We will present a few examples that demonstrate how our evolutionary model can produce counterintuitive results that sometimes can be rather surprising for a rational human mind. For example, one would never think that it would be possible to eliminate two positions by rearranging some activity assignment, reporting structure, and other organizational attributes and yet improve project schedule and/or quality. In another example, our model also generates the unusual idea that two people can actually supervise one other; or that assigning an activity to a person that is not in her/his area of expertise may bring the best outcome because of a specific project situation. Although these ideas might not make sense at first with a logically constrained human mind, they might open up a new way of thinking in a novel direction. This is the idea of Out-of-Box Thinking (OBT) that we will explain in section 4.7. We believe that the EOD model's ability to generate counterintuitive results is one of the other greatest advantages of using evolutionary methods.

This prototype version of EOD has some key limitations. Although the transforming genetic tree in its current form was shown to produce improved results, it might not be the most efficient implementation. The tight dependencies of function set FTE, Assign, and Alloc on their preceding nodes can cause some deficiencies during the crossover operations, where only part of a branch of one individual is swapped with another.

Another limitation is that for the case of skill level and FTE branch, where the recurrence of actors can occur in TGT, the very last actor to the right of the tree was used. Thus, after many generations, for the above mentioned functions the right side of the genetic tree tends to be more active than the left side.

In spite of these limitations, the prototype version of EOD produces solutions that compare favorably with the best solutions developed by human experts for some

benchmark problems, and that conform to accepted organizational contingency theory. The results of these two kinds of validation experiments are discussed in the following two chapters.

⇒ Chapter 4

Case Studies: Producing Human-Competitive Results

Grown-ups like numbers. When you tell them about a new friend, they never ask questions about what really matters. They never ask: “What does his voice sound like?” “What games does he like best?” “Does he collect butterflies?”. They ask: “How old is he?” “How many brothers does he have?” “How much does he weigh?” “How much money does his father make?” Only then do they think they know him.

-- The Little Prince (Antoine de Saint-Exupéry)

In this chapter, we demonstrate the power and generality of our EOD model by showing some of the human-competitive results that EOD has been able to produce, and showing how it can be applied to different project organization problems. We compare the results produced by GP with those generated by humans for two real-world projects that have been used as benchmark case studies for the past eight years in Stanford project management courses and workshops. We also discuss some of the counterintuitive results that our evolutionary model provides.

Before discussing these experimental case study results, we explain what we mean by human-competitiveness.

4.1 Definition of Human-Competitiveness

An automatically created result is “human-competitive” if it satisfies at least one of the eight criteria below (Koza, 2003):

- (A) The result was patented as an invention in the past, is an improvement over a patented invention, or would qualify today as a patentable new invention.
- (B) The result is equal to or better than a result that was accepted as a new scientific result at the time when it was published in a peer-reviewed scientific journal.
- (C) The result is equal to or better than a result that was placed into a database or archive of results maintained by an internationally recognized panel of scientific experts.
- (D) The result is publishable in its own right as a new scientific result — *independent* of the fact that the result was mechanically created.

- (E) The result is equal to or better than the most recent human-created solution to a long-standing problem for which there has been a succession of increasingly better human-created solutions.
- (F) The result is equal to or better than a result that was considered an achievement in its field at the time it was first discovered.
- (G) The result solves a problem of indisputable difficulty in its field.
- (H) The result holds its own or wins a regulated competition involving human contestants (in the form of either live human players or human-written computer programs).

In the following sections we demonstrate how our model can produce human-competitive results by meeting three of the above eight criteria, namely criteria E, G and H.

4.2 Biotech Plant Case Study

Once we designed and implemented the preliminary version of our postprocessor optimizer, we applied it to a case study that has been used for several years in a project management course taught at Stanford. The results produced by the GP were then compared against the best solution discovered by student groups and senior project manager groups over the last five years. In this case study, student and project manager groups are given the project organization for design and preconstruction planning of a new biotech plant and asked to modify some of the individual/subteam attributes, task assignments, and organizational policy attributes in order to reduce the project schedule duration as much as possible, while maintaining acceptable levels of quality risk. The case study that was given to the students had two main components. A narrative description of the problem, reproduced below, and a VDT model, depicted in figure 4.1.

Your group has been brought in as a consultant to PharmaCorp, the owner of a fast-track Biotech plant project. This plant will produce a blockbuster new arthritis drug that just cleared its Phase III FDA approval. Competitors are rushing similar drugs to market and the remaining time on Pharma's patent is fast expiring, so time is of the essence. To meet Pharma's aggressive marketing goals, the plant must break ground by the first week in December. The SimVision model shows the pre-construction activities for this plant, culminating in the milestone: "Ready to Excavate".

Your Group's Goal is to find a way to complete the Design Build Biotech Project pre-construction activities, and be "ready to excavate" by the end of the first week in December. Members of the group should run simulations in teams of two or three in advance of your group meeting.

The group should then consider the solutions generated by each of the subteams, and come up with a consensus solution for the case. You may employ any of the “acceptable interventions” below. While shortening the simulation duration, you need to insure that you also maintain acceptable quality risk, i.e., no activity may have a communication, project or functional risk index above 0.5.

Acceptable interventions:

- 1 Add a total of up to 3 FTE’s in increments of not less than 0.5 FTE to any combination of actors
- 2 Increase the skill level (from low to medium, or medium to high) for any one skill for any one actor
- 3 Change levels of centralization, formalization, or matrix strength
- 4 Reassign tasks to different actors or change the assignment % that an actor is allocated to an activity

Unacceptable interventions:

- 1 Shortening activity duration
- 2 Delete an activity or any rework or coordination links
- 3 Adding a new skill to an actor
- 4 Changing error rates
- 5 Changing activity precedence

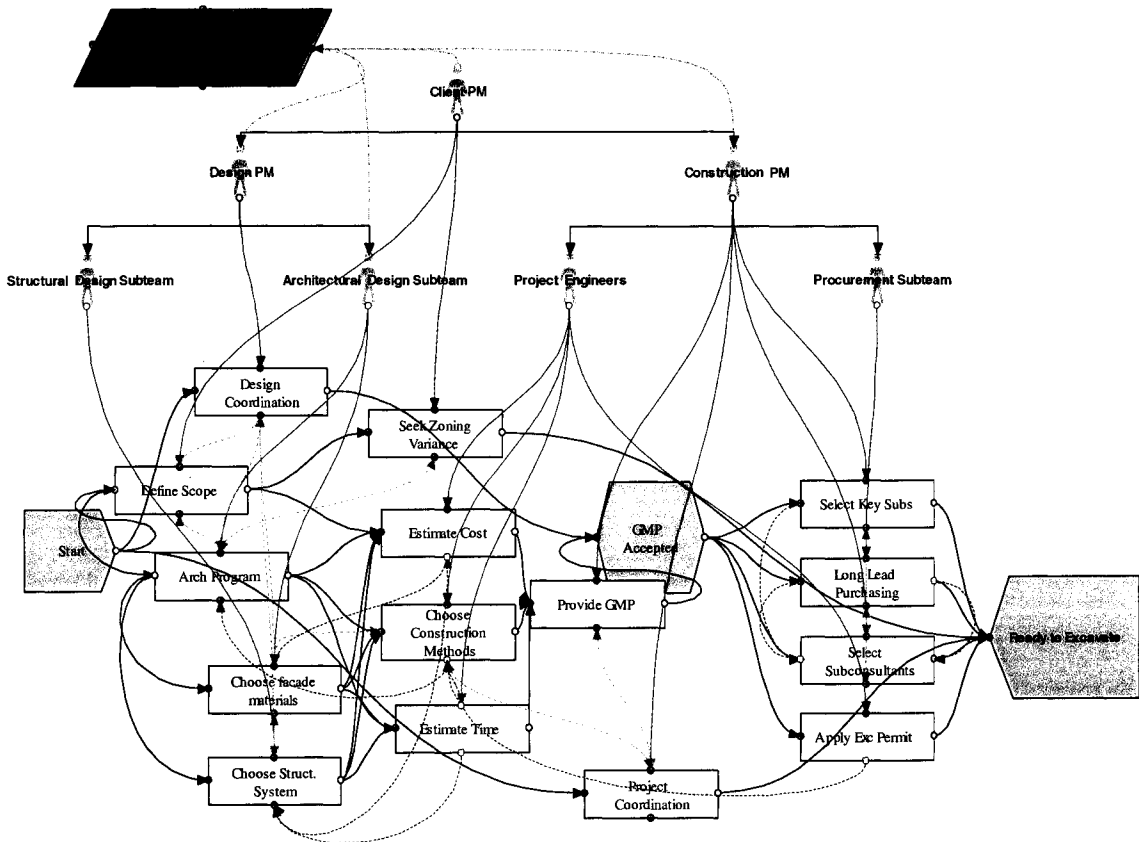


Figure 4.1 Biotech Plant case study

We used the biotech plant case study to test our EOD model by developing a three stage experimental approach. In Phase I, we defined a simplified GP. In this process, we varied only the levels of the actors' skills. Then, we compared the results found by the GP with a known "optimal" solution.

In Phase II, we kept skill levels constant and varied the number of Full Time Equivalent FTEs (i.e., human resources) added to different positions, attention allocation to activities, and reassignment of activities to different positions. We also varied organizational policy attributes such as the levels of centralization, formalization and matrix strength using GP. We then compared the GP results against the best solution found by previous student and manager groups. In phase III, we let EOD modify the organization reporting hierarchy, in addition to the parameters mentioned in phase II above. In the next three sections, we discuss the findings of this experiment.

4.2.1 Varying Actors' Skill Level

In order to check the face validity of our model, we created a simple case, so we would know what the optimal solution would be. That is, we assumed that in a project organization where cost is not an issue, if we hire the most skilled people (i.e., setting all skill levels to high) for the project, we would obtain the best results in terms of quality and project duration.

Problem Statement

In the case of the biotech plant, as shown in figure 4.2 below, there are seven positions (actors) in this project organization, and each one of these positions has two to eight different skills. The skills range from biotechnology to design coordination to mechanical/electrical, etc. There are a total of 29 skills for all seven positions. Each one of these skills can be set to three levels of low, medium, and high. Therefore, the total number of combinations that one could try to find an optimal solution exhaustively is $3^{29} = 6.8 * 10^{13}$. Thus, the sample space is vast and an exhaustive search is infeasible.

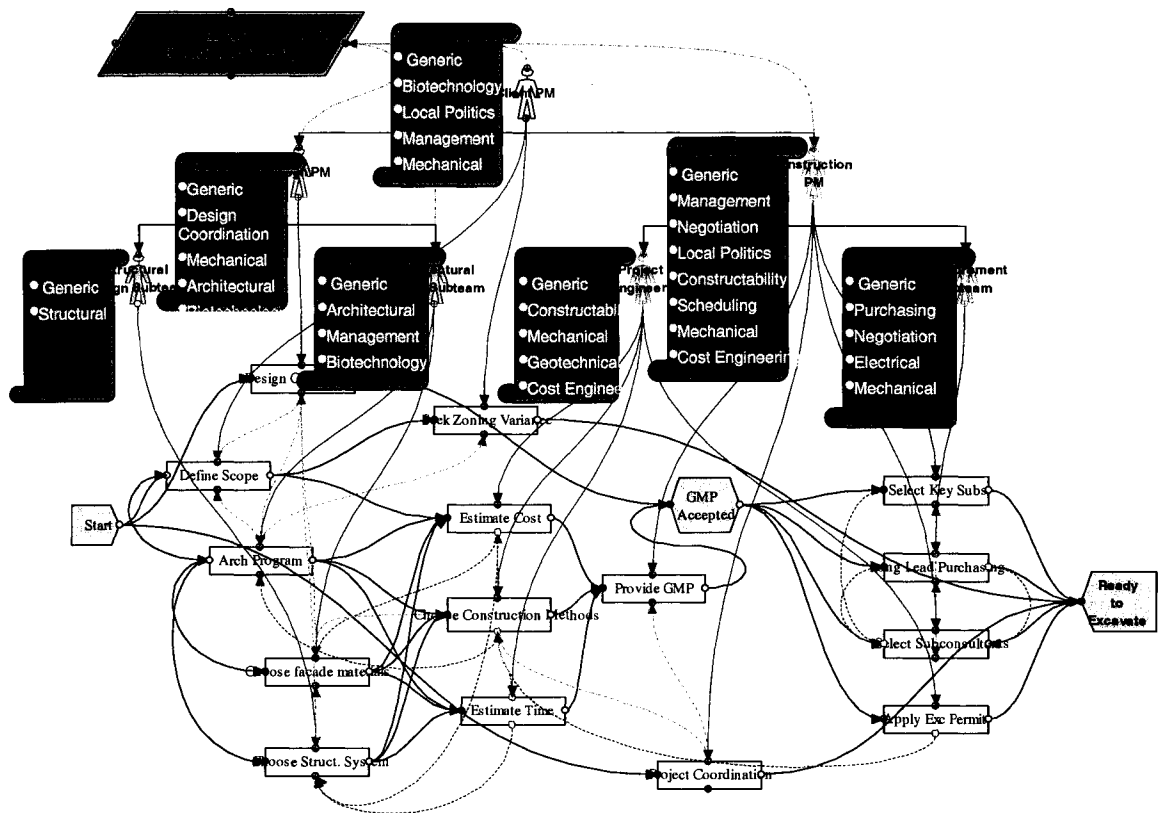


Figure 4.2 Actors can have Different Skills and Skill Levels

In the biotech plant case study, there are total of 29 skills among 7 Actors each skill can be set to 3 levels of low, medium, high. So, the total number of combinations is equal to $3^{29} = 6.8 * 10^{13}$

It should be obvious that the more skilled the actors, the faster the tasks get done, and the fewer the exceptions (i.e., situations when an actor requires additional information or a decision to complete part of a task, or the actor generates an error that may need correcting). In this case where we are not concerned about cost, the optimal solution would be when the skill levels of all skills for all actors are set to high. Knowing the above fact, in one scenario we set skill levels of all actors to “high”, ran the VDT simulation and compared the results with the base results where we had the skill levels of all actors set to “medium”.

Experimental Setup

We used the following genetic tree parameters to obtain the optimal solution described in results section next.

Table 4.1 Genetic Programming Setup for the Biotech Plant Case Phase I

Objective:	Reduce project simulation duration while maintaining quality by varying only skill levels of each position
Terminal Set	P1, P2, P3, P4, P5, P6, P7
Function Set	Up, Down, Same
Raw Fitness	$SPD + \bullet (FRI_i * FRIW_i + PRI_i * PRIW_i + CR_i * CRW_i)$ (see section 3.6 Fitness Function Evaluation)
Parameters	Population size $M = 100$ Maximum number of generations, $G = 50$ Crossover = 90% Mutation = 3% Reproduction = 7%
Success Predict	None – search for the shortest simulation duration with the given quality constraints

Results

At the base level, when we set all skill levels to medium, we found that the simulated schedule end was March 28, 2001, and when we set all skill levels to “high”, the project duration was reduced by 76 days and the simulation showed that the project schedule end would be January 10, 2001. Then, we ran the simulation again using the suggested solution by GP; the completion date was identical to the result when all skill levels were set at high. The best individual of generation 16, which found the optimal solution, is shown in Lisp-type format below and in tree-type format in Figure 4.3:

```
(Up P4 (Same (Same P1 P2) (Up (Up P5 (Same (Same (Up (Same P3 P1)
(Up P6 (Up P4 (Up (Down P3 P2) P0)))) (Up (Down P3 P2) P0)) (Same
(Up P0 (Up P0 (Down P3 P2)))) (Up (Down P5 P2) P0)))) P0)))
```

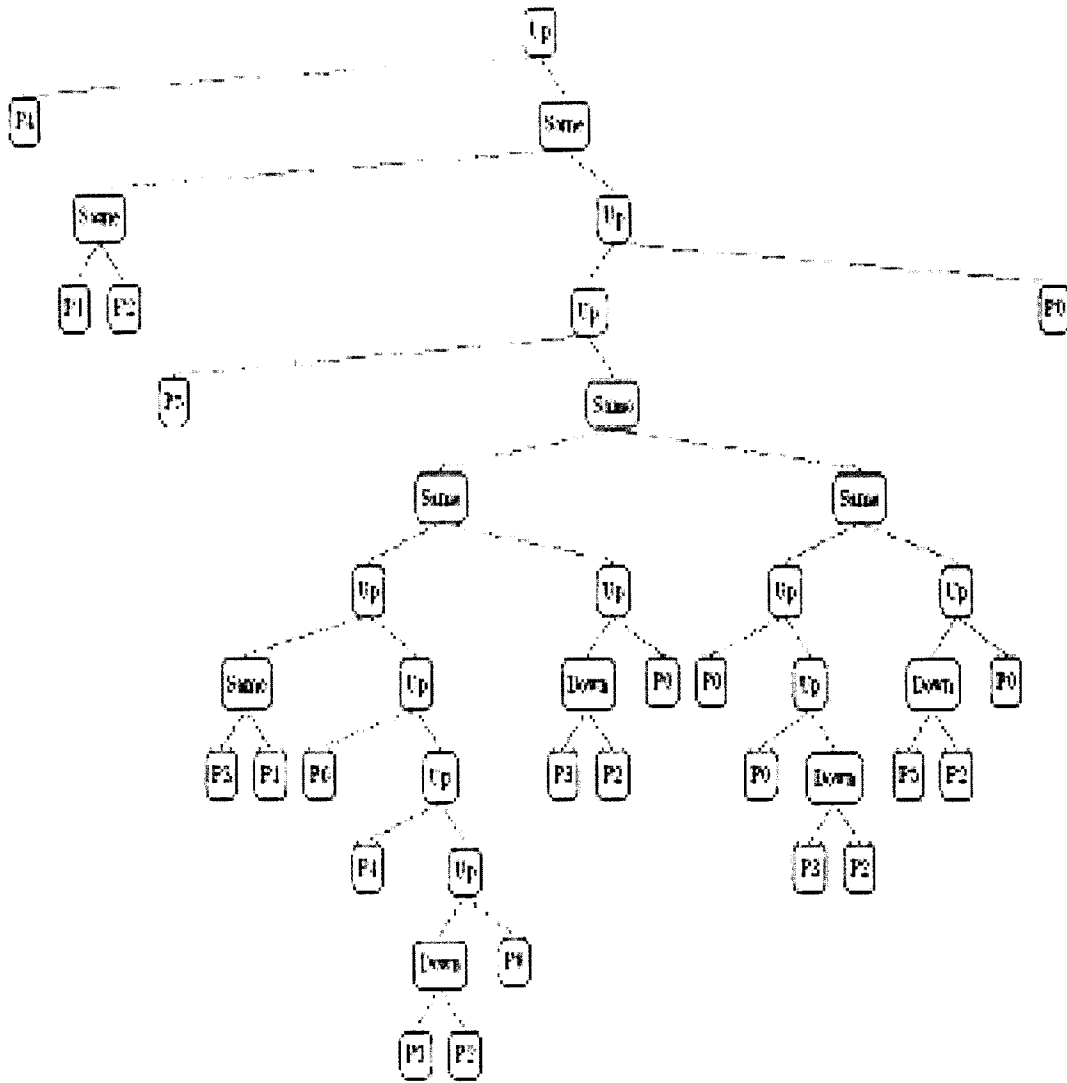


Figure 4.3 Best Individual of Generation 16 in a Genetic Tree Format

The solution matches the theoretical optimal solution when skill levels are the only parameters to be set.

Discussion

As mentioned above, the performance results found by GP were identical with the optimal case. However, interestingly, the suggested solution found by GP was not identical to the optimal solution (i.e., there were multiple solutions that yielded the same optimal outcome). Unlike the optimal case scenario, GP did not have to set all skill levels to “high”. In fact, there were situations where the levels of some actors’ skills were reduced from “medium” to “low”, and still the outcome matched the optimal solution. For example, the “General” skill of the “Structural Design Subteam” was reduced to

“low”, and the “Mechanical” skill of the “Construction PM” was kept at “medium” (see Figure 4.3 above). It turns out that these skills were not required for any of these actors’ assigned tasks in this case, so the outcome was insensitive to these skill levels. This showed that GP could generate different solutions to a problem, so that a project manager can better decide which solutions to pick. For example, different solutions suggested by GP might advise that we could reduce the skill levels of certain actors and increase the skill levels of others. So the project manager has the choice between different alternatives to pick a solution that better fits her/his specific project and available resources.

4.2.2 Varying Actors’ FTE, Activities Assignment, Attention Allocation and Organization’s Decision Making Policy

In Phase II, we kept the skill levels constant and allowed the GP to vary the assignment of activities to actors, percentage allocation for each activity, the Full Time Equivalent’s (FTE) of each actor in 0.5 FTE increments, and organizational policy properties such as levels of centralization, formalization and matrix strength using GP.

Problem Statement

In this case, the problem was exactly as stated in section 4.2, as given to student teams. The only difference is that at this stage we did not allow any reporting hierarchy changes.

Experimental Setup

We used the following GP setup shown in Table 4.2:

Table 4.2 Genetic Programming Setup for the Biotech Plant Case Phase II

Objective:	Reduce project simulation duration while maintaining quality by varying Actors’ FTE, Activities Assignment, Attention Allocation, and organization’s decision making policy
Terminal Set	P0, P1, P2, P3, P4, P5, P6, Assign, Alloc
Function Set	Up, Down, Same, CFM
Raw Fitness	$SPD + TFTE * FTEW \sum (FRI_i * FRIW_i + PRI_i * PRIW_i + CRI_i * CRIW_i)$ (see section 3.6 Fitness Function Evaluation)
Parameters	Population size M = 3000 Maximum number of generations, G = 100 Crossover = 90% Mutation = 3% Reproduction = 7%
Success Predict	None – search for the shortest simulation duration with the given quality constraints

Results

The best individual found by GP in generation 26 beats the best human-discovered solution by 6 days. The best human solution reduced project completion from Feb 16, 2001 to Dec 7, 2000 whereas the GP-suggested solution reduced the project end date to Dec 1, 2000 while satisfying all quality criteria.

The best solution of generation 26 is shown in lisp-type format below:

```
(Up (Same P1 (Same (Same (Assign P1) (FTE P1)) (Down (Same CFM
P5) (Assign P3)))) (Same (Same (Up (Same (Up P5 P4) (Same (FTE
P1) (Assign P4))) (Up (Assign P4) (Down P2 (Same
(FTE P1) (Same (Same (Down (Up (Up (Assign P0) (Assign P0)) (Down
(FTE P5) (Up (FTE P1) (Same P6 CFM)))) (Same (Assign P0) (FTE
P2))) (Same (Up (Same (Same P4 CFM) (Same P5 CFM)) P3) (Assign
P4)) (FTE P1)))))) (Same (Up (Same (Assign P0) (Assign P0)) P3)
(Up P0 P6))) (Up (Same P3 P4) (FTE P4))))
```

Figures 4.4 and 4.5 below compares the project schedule and communication quality risk before and after the intervention by EOD. In addition, although backlogs of different positions were not one of the measuring factors in the fitness function, as a side effect, they were improved in some cases by more than 50% (see Figure 4.6 below).

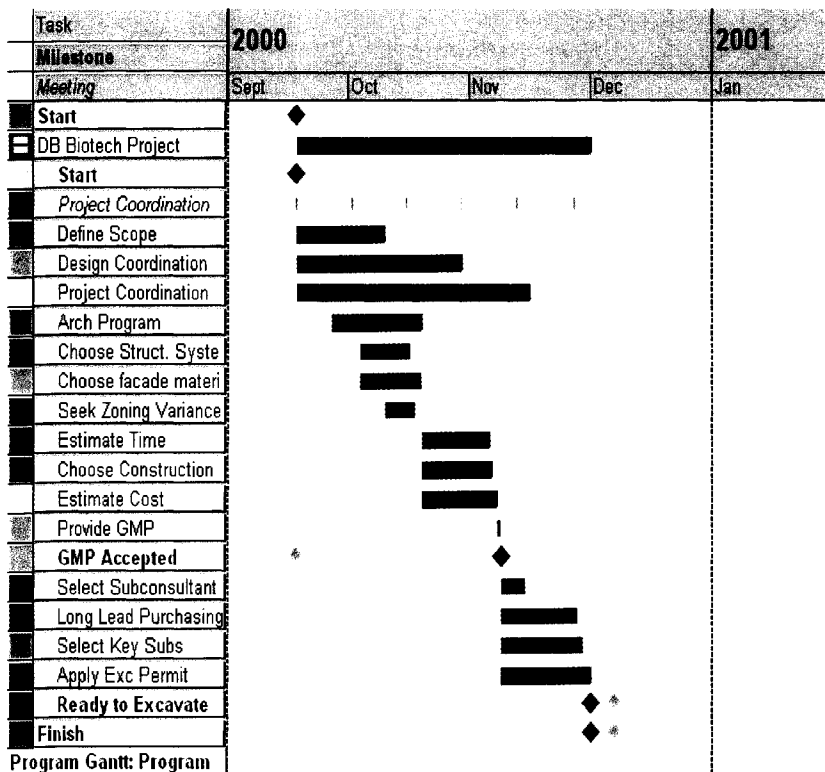
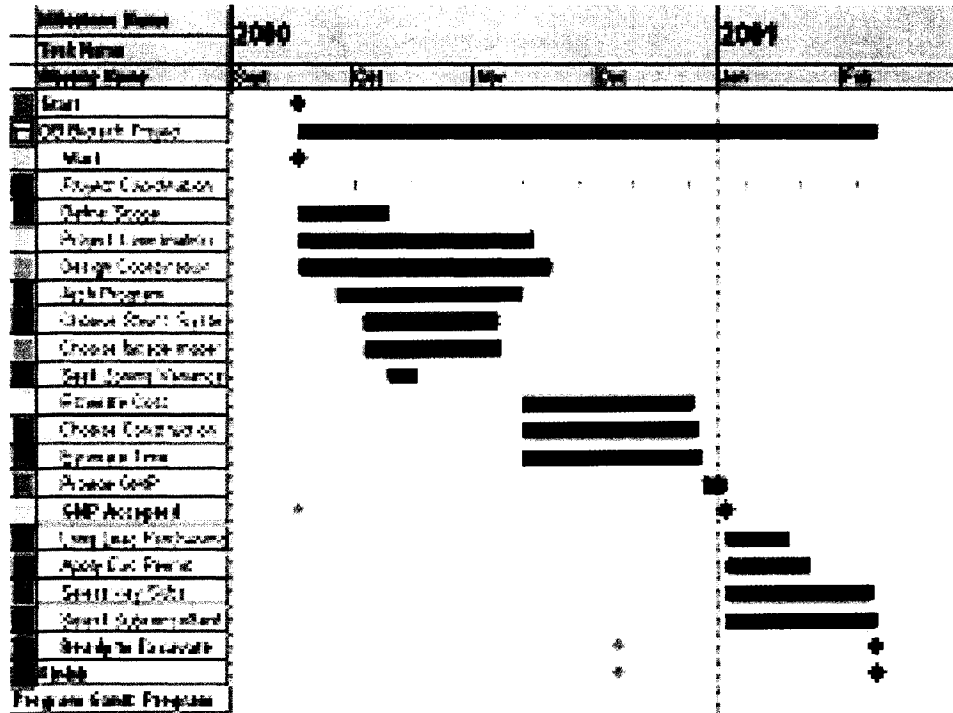


Figure 4.4 Comparison of Gantt Charts Before (Top) and After (Bottom) Evolutionary Process

GP reduced the end date from Feb 16, 2001 to Dec 1, 2000. This GP solution is 6 days better than the best solution (Dec 7) found by graduate student and manager teams for this problem over a five year period.

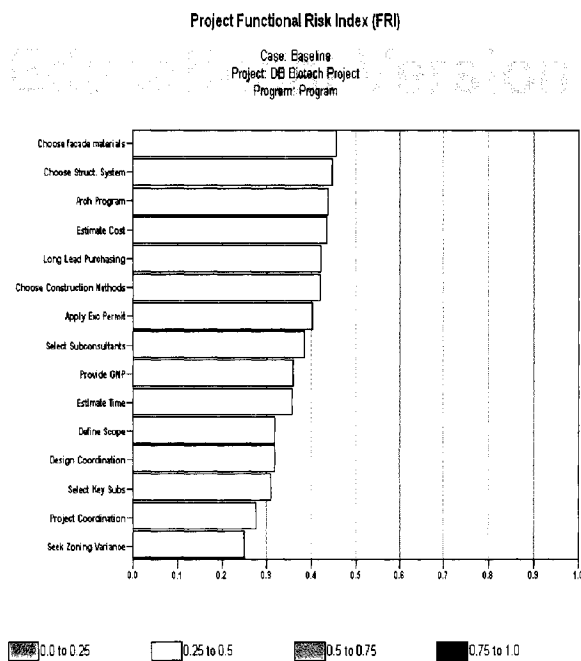
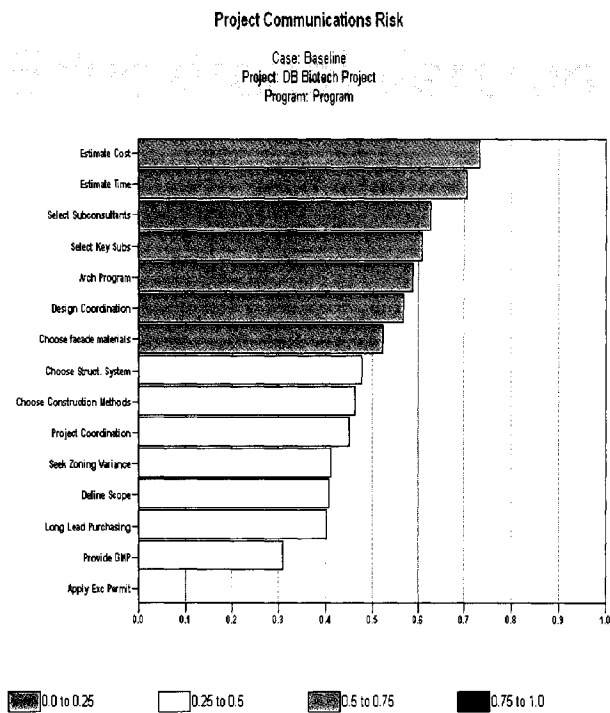


Figure 4.5 Comparison of Communication Quality Risks Before (Top) and After (Bottom) Evolutionary Process

Originally 7 out of 14 activities had quality risks higher than the acceptable 0.5 threshold (orange bars). With the suggested organizational changes, quality risks for all activities improved sufficiently to meet this constraint.

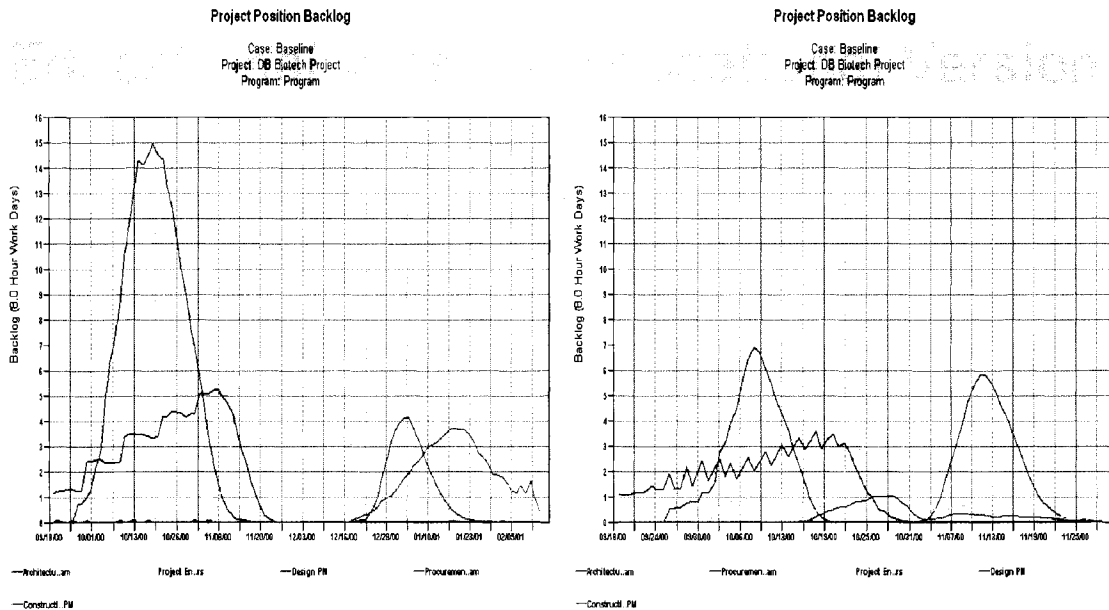


Figure 4.6 Comparison of Project Position Backlogs Before (Left) and After (Right) Evolutionary Process

As a side effect of improved project organization, backlogs for some positions were reduced by more than 50%.

It was also interesting to see that the details of an alternative solution found by EOD, which was two days better than the best human solution (i.e., schedule end date of Dec 5, 2000), was quite similar to the best human solution. This is discussed in more details in the discussion section below. This solution was found in generation 21 and it is shown in a Lisp-type format below:

```
(Up (Down (Same (Same P5 P4) (Down (Down P1 P5) (Up (FTE P0) (Up
(Down (Up (FTE P0) (Down P5 P5)) (Up (FTE P1) (Up (FTE P0) (Same
P3 P6)))) (FTE P5)))) (Up (Same (Same (Down (Up (Up (Assign P0)
(FTE P1)) (Same (Up (Same (Down (FTE P4) (FTE P0)) (Down (FTE P2)
(Up (Up P6 (Up (Up P0 (FTE P1)) (FTE P4))) (FTE P1)))) (Up (FTE
P4) (Assign P4))) (Up (Up (Up (FTE P5) (FTE P5)) (FTE P4)) (Up
(FTE P0) (Up (Assign P0) (Same P5 P4)))))) (Up (FTE P5) (Aloc
P0)) P2) (FTE P0)) (Same (Same (Down (Up (Up (Assign P0) (Same
P5 P4)) (Same (Up (Same (Up (Assign P0) (Up (Assign P1) (Assign
P0))) (Aloc P1)) (Up (FTE P4) (Assign P4))) (Up (Up (Up (FTE P5)
(FTE P5)) (FTE P4)) (Up (FTE P0) (Up (Assign P0) (Same P5
P4)))))) (Up (FTE P5) (Aloc P0))) P2) (FTE P0))) (FTE P4))
```

Discussion

In the second phase, where GP was allowed to vary the number of FTEs added to actors, the assignment of activities to actors, percentage allocation for each activity, and the organization policy attributes the solution found by EOD surpasses the best Student/Manager (S/M) solution ever found. GP was able to do this by adding 3 FTEs to different positions, reassigning 4 activities to different actors, and changing formalization and matrix strength to high. The amount and location of FTEs added by S/M team and GP are shown in orange and green boxes respectively; reassigned activity links are shown in brown lines in Figure 4.7 below. Note that this creative design suggested by GP not only improved the schedule and met the quality constraints but also was able to eliminate one position in the organization structure. This is graphically shown in the figure below.

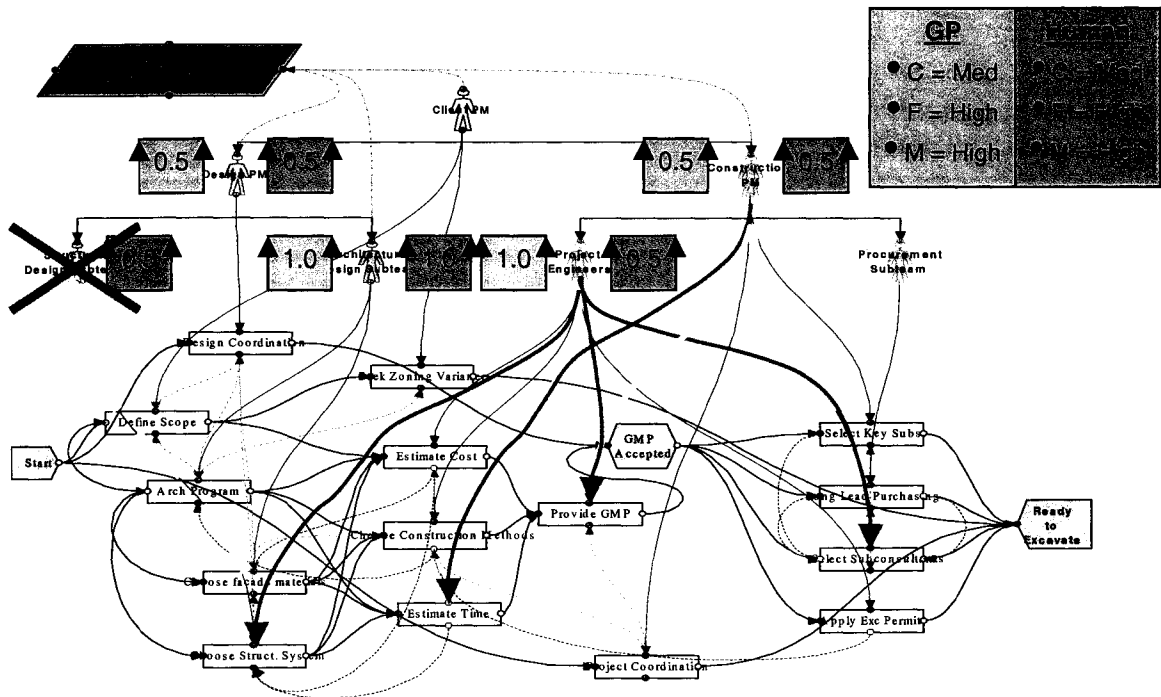


Figure 4.7 Comparison between GP and the Best Human Solution

GP found a solution that was 6 days better than the best human solution and yet was able to eliminate one position. The best Individual was found after 26 generations. The human solution simulated end date was Dec 7, 2000 whereas the GP solution simulated end date was Dec 1, 2000. The orange and green boxes represent the amount and location of FTEs added by human and GP respectively. The four brown arrows from actors to activities shows the reassigned activity links suggested by GP. The position on the left with no task assignments (blue arrows) and no subordinates to supervise can be eliminated without affecting project performance.

In an alternative solution found by EOD, which beat the best human solution by 2 days (i.e., Dec 5, 2000 schedule end date), it was also interesting to see that GP found almost the exact same solution as the best human-discovered solution with a couple of additional changes. In this case, strikingly, the FTEs were added in the *exact* same location and the *exact* same amount as the best human team previously suggested (see Figure 4.8).

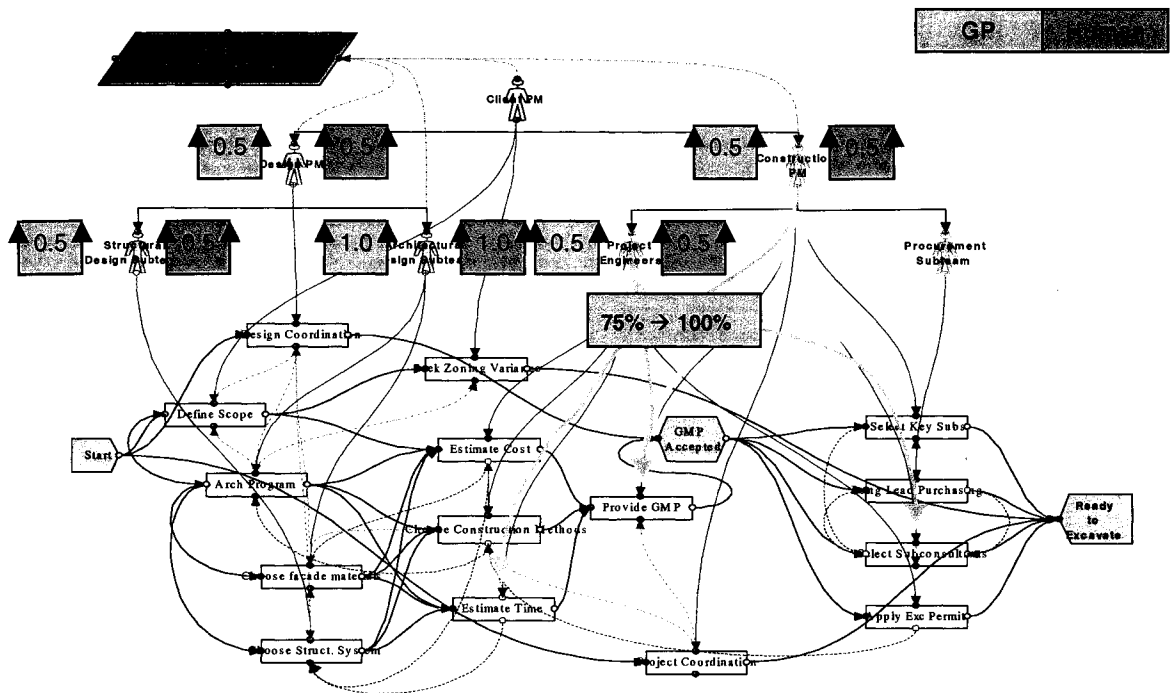


Figure 4.8 An Alternative GP Solution, which is 2 days better than the Best Human Result while Matching the Human Solution in most cases

The GP suggested solution added an additional reassignment of an activity and changed an attention allocation from 75% to 100%. Green links from Actors to activities indicate reassignments suggested by GP. The best Individual was found after just 21 generations.

In addition, the human solution had suggested swapping two activities between two actors. Namely, “Select Subconsultants” activity was reassigned from Construction PM to Project Engineers position, and “Estimate Time” activity from Project Engineers position to the Construction PM. In addition to the swaps suggested by the S/M solution, the GP solution suggested an additional reassignment of an activity and changes in the percentage allocation of a position to an activity. GP suggested reassigning the “Provide

GMP” activity from the Construction PM to the Project Engineers team. The reassigned links from actors to activities are shown by green lines in Figure 4.8 above. In addition, the attention allocation of the Project Engineers position to the “Provide GMP” task was changed from 75% to 100%.

Alternative Solutions

As mentioned in Chapter 2, one of the greatest advantages of genetic programming, unlike many operations research optimization methodologies, is that it can find a set of optimal or near optimal solutions instead of a single point solution. Below we demonstrate how this can offer significant practical benefits in the area of project organization design.

For one of our GP runs, we plotted the trend of improvements through different generations. Figure 4.8 below shows that the greatest improvement of the best individual fitness value was made between generations 1 and 6 (solid pink line). This figure also displays the improvement in mean fitness through different generations (yellow dotted line). Although the fitness value of the best individual has not improved after generation 21, the mean fitness value continues to improve. This means that GP has produced more individuals within a generation that nearly match the solution found by the best individual. This provides the opportunity for a project manager to select from a set of multiple, near-optimal solutions the one that best matches her/his specific project needs and constraints, such as the availability of additional persons with specific skill levels.

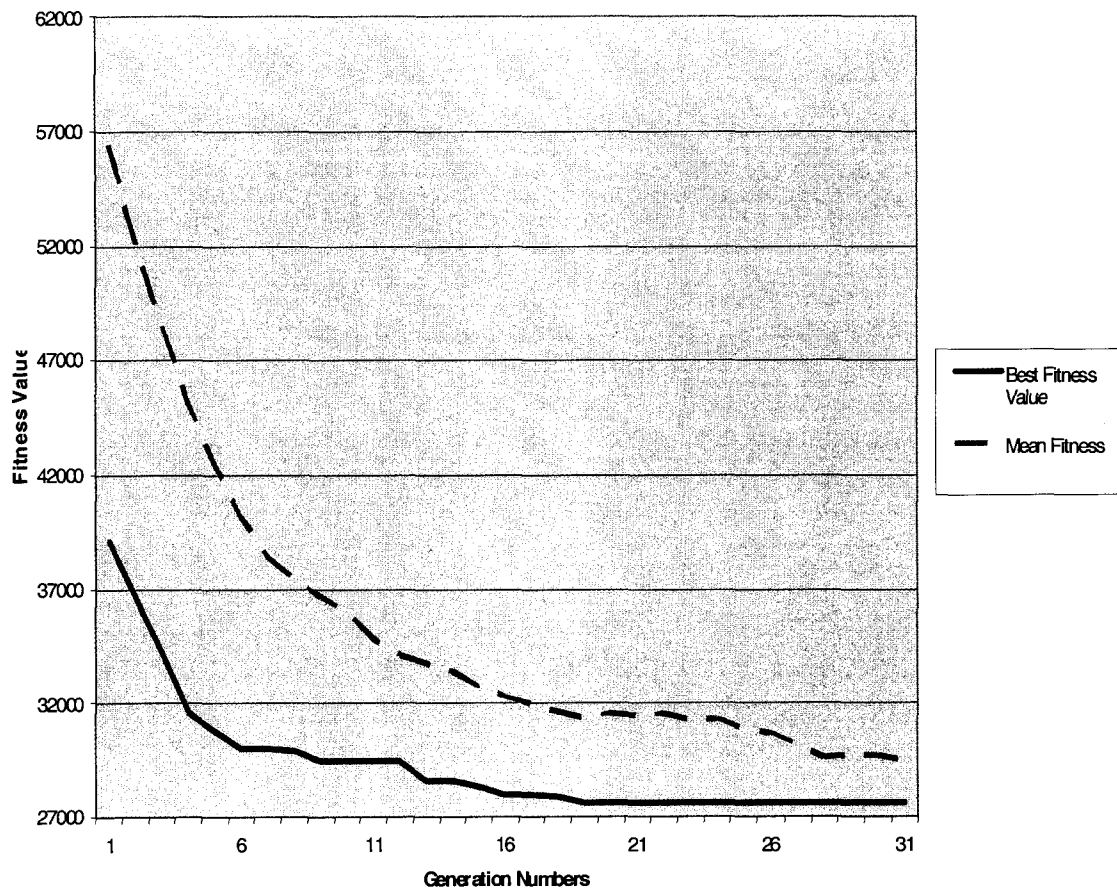


Figure 4.9 Improvement of Fitness Value Generations 1 Through 30

Although the best fitness value (solid line) does not improve after generation 21 (best solution found), the mean fitness value continues to improve. This means that the GP continues to improve the overall fitness of the set of solutions in each generation, providing more near-optimal alternative solutions from which a manager can pick the best solution for a given context.

For example, one of the alternative solutions we found was 3 days better than the best human solution, but was able to eliminate 2 positions from the organization structure. This is shown in Figure 4.10 below. That means, it gives flexibility to a project manager to pick the first solution above, if time is the critical factor or the second solution if it is okay to finish the project three days later with the advantage of having two less positions for the project.

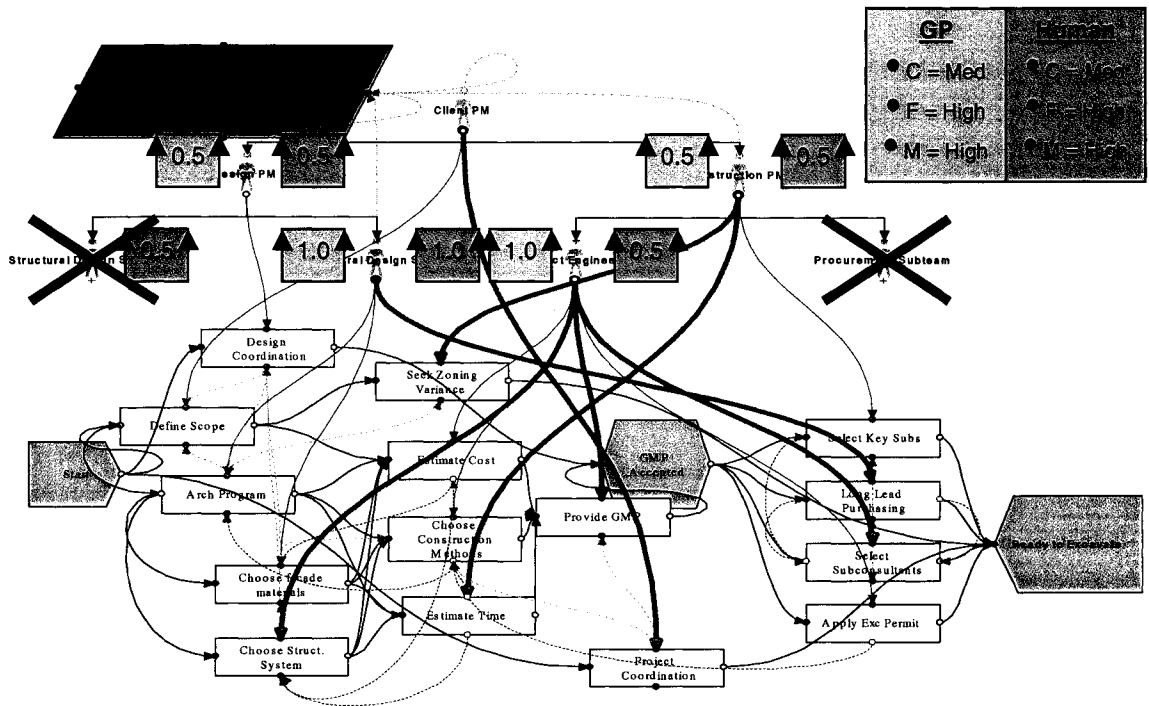


Figure 4.10 Alternative Solution Found by GP that was 3 Days Better than Best Human Solution and was able to Eliminate Two Positions

Brown link from Actors to activities indicates reassignments suggested by GP. The positions with no task assignments (blue arrows) and no subordinates to supervise can be eliminated without affecting project performance.

4.2.3 Varying the Organization's Reporting Hierarchy

In the last phase, we let GP make changes in the organization reporting hierarchy, in addition to all the acceptable changes mentioned in the phase II. However, the best result was not improved. Then we tried a different strategy. We took the best results obtained from phase II and had EOD try to optimize just the organization structure and the decision making policy of the organization. With only 100 individuals and in less than 6 minutes GP was able to find a different supervision hierarchy that actually could improve the schedule end date by one day while maintaining acceptable quality levels.

Problem Statement

The problem was exactly as described in section 4.2.

Experimental Setup

We used the following GP setup shown in Table 4.3:

Table 4.3 Genetic Programming Setup for the Biotech Plant Case Phase III

Objective:	Reduce project simulation duration while maintaining quality by varying organization reporting hierarchy and organization's decision making policy
Terminal Set	P0, P1, P2, P3, P4, P5, P6
Function Set	Up, Down, Same, CFM
Raw Fitness	$SPD + \bullet (FRI_i * FRIW_i + PRI_i * PRIW_i + CR_i * CRW_i)$ (see section 3.6 Fitness Function Evaluation)
Parameters	Population size $M = 100$ Maximum number of generations, $G = 50$ Crossover = 90% Mutation = 3% Reproduction = 7%
Success Predict	None – search for the shortest simulation duration with the given quality constraints

Results

The best solution found in generation six produced a result which was seven days better than the best human solution (i.e., schedule end date Nov 30, 2000). The best individual of generation six is shown in a Lisp-Type format below:

```
(Up P0 (Same (Same (Same P2 (Up P0 (Same (Up P1 CFM) CFM))) CFM) CFM) )
```

The instructions produced by the Transforming Genetic Tree changed the supervision hierarchy and eliminated one position as shown in Figure 4.11 below.

Discussion

Unlike what we originally anticipated that the greatest improvement would come with changing the reporting hierarchy, we noticed, at least in this case, just a one-day improvement in schedule. Nevertheless, as mentioned before, this is the best solution that has ever been found for this problem to date by either human or machine.

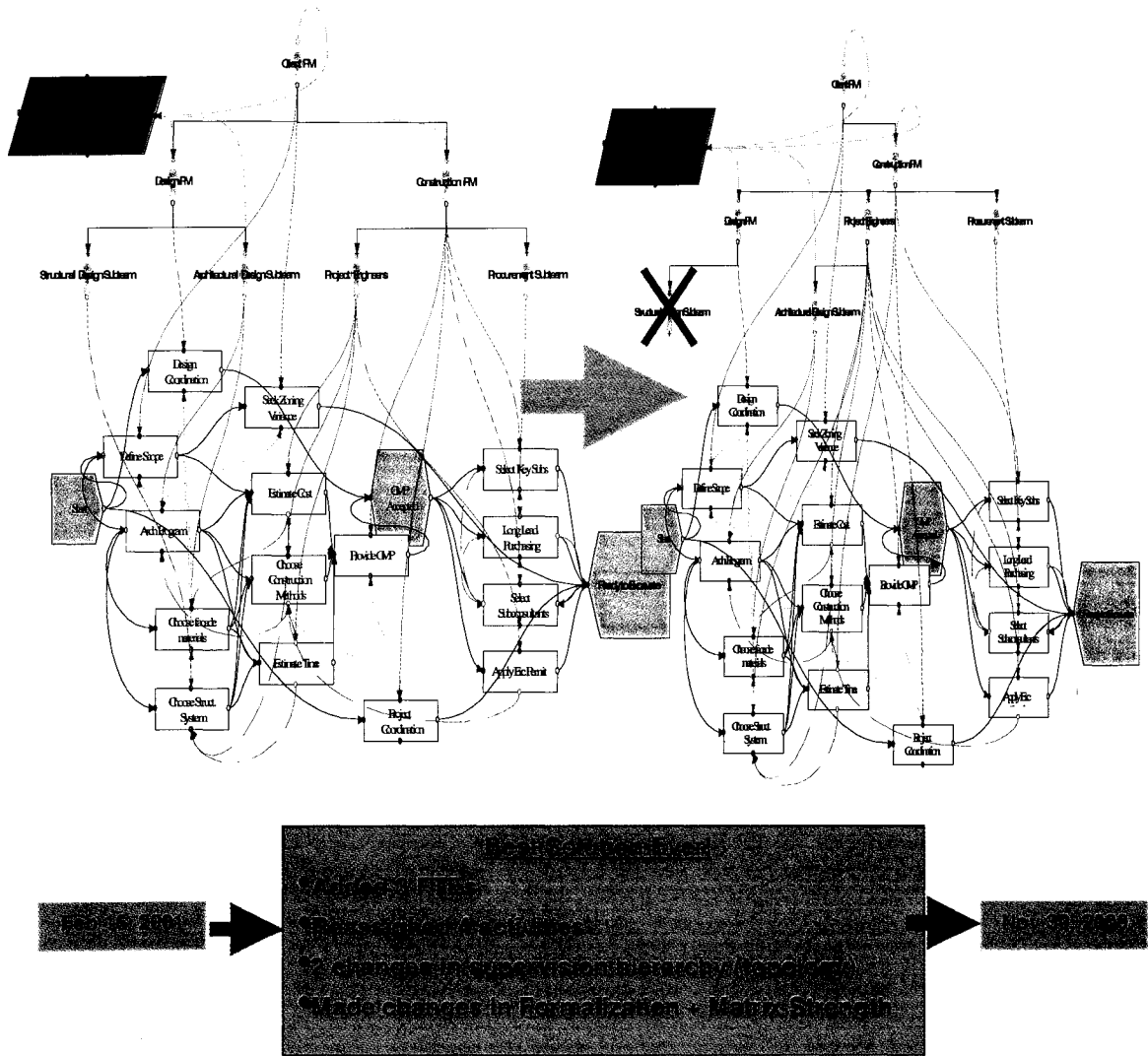


Figure 4.11 Best Solution Ever!

In addition to individual attributes and organization decision making policy, the topology (both reporting hierarchy and activity assignments) of the organization changes to produce the best solution ever found to date by human or machine.

4.3 ASIC Design Case Study

In order to make sure that our model was not over-fitted to one problem, we chose a second real-world project, the ASIC design case, which has also been tried by both Stanford graduate students and professional project managers.

Problem Statement

The problem, which was titled “Reduction in Time to Market for Application-Specific Integrated Circuit (ASIC)” is described as follows:

In early January of 1998, a leading application specific integrated circuit (ASIC) designer was asked design and fabricate a chip set for a new personal digital assistant (PDA) product that a mobile telephone manufacturer was trying to develop on an unusually tight schedule. The ASIC developer was very experienced with this type of project, but needed to dramatically accelerate its normal 11 month design and production process since the product containing this chip set was to be unveiled at a trade show in August of 1998 – Just 8 months away! The design phase was further complicated because the fabrication of the chip was to be outsourced to a foundry, so the project involved managing both its own design tasks and the foundry’s layout, testing and manufacturing tasks, while maintaining control over verification. The objective is to:

- 1 Fast track the design and construction of the chip in order to meet the August deadline
- 2 Maintain product quality and allow for proper verification of the chip

Acceptable interventions were the same as those described for the biotech plant. That is, to add a total of up to 3 FTE’s in increments of not less than 0.5 FTE to any combination of actors, to change levels of centralization, formalization, or matrix strength, to reassign tasks to different actors or change the percentage attention allocation that an actor is allocated to an activity, and to change reporting hierarchy as needed.

Differences between Biotech Plant and ASIC Design Case

The modification that needed to be made to EOD, so the model can be run with this new case was minimal. The ECJ parameter file was modified to cover nine positions instead

of seven positions (see Appendix C). No other changes were necessary since the objectives and constraints of both problems were the same.

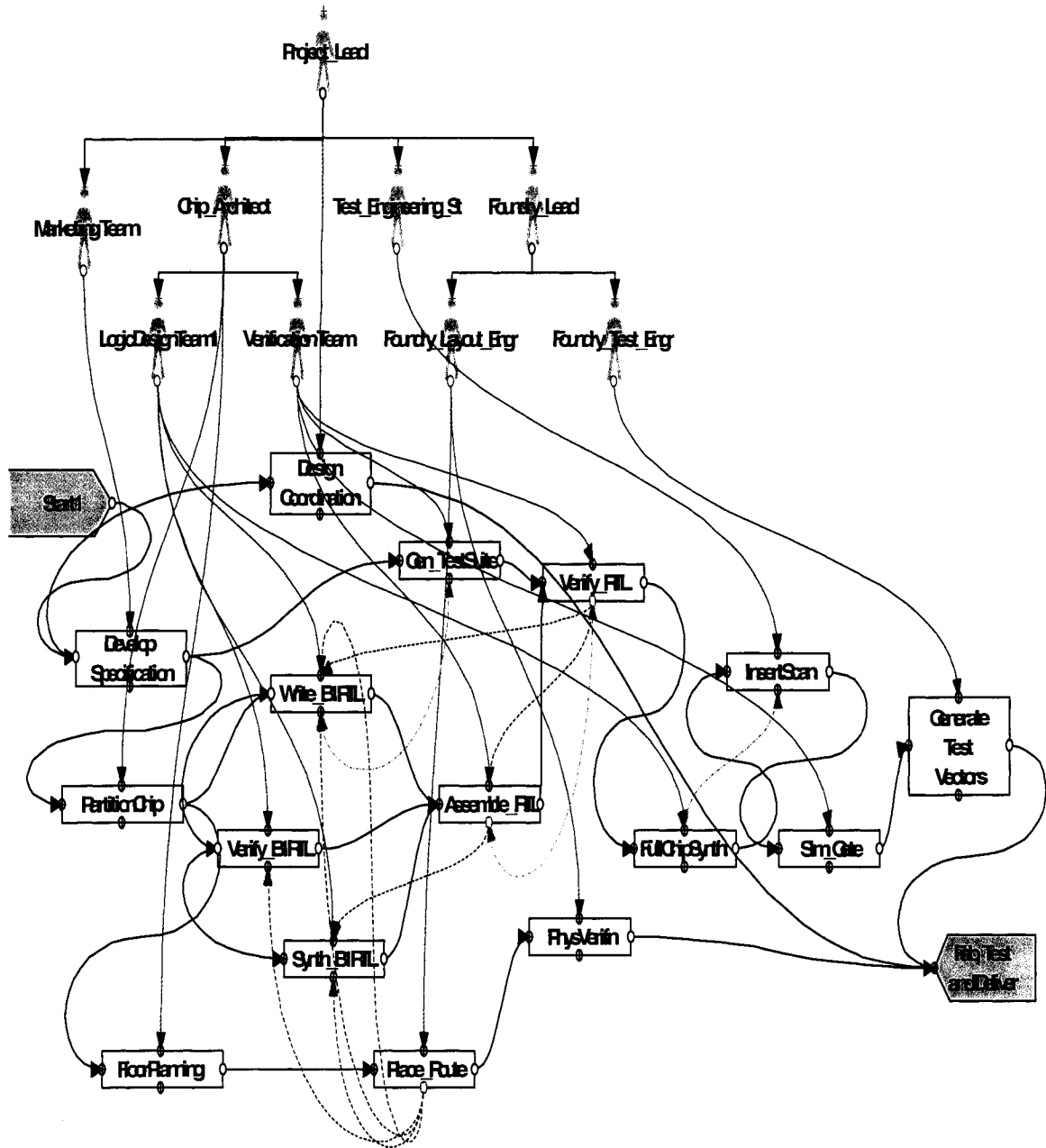


Figure 4.12 ASIC Design Project Case

The Project Start date was January 1, 1988. The VDT simulated finish date for the baseline plan was November 9, 1988. The project needed to be done in 7 months to meet the August 1st trade show deadline.

Experimental Setup

We used the following GP setup shown in Table 4.4:

Table 4.4 Genetic Programming Setup for the ASIC Design Case

Objective:	Reduce project simulation duration while maintaining quality by varying Actors' FTE, Activities Assignment, Attention Allocation, and organization's decision making policy
Terminal Set	P0, P1, P2, P3, P4, P5, P6, P7, P8, Assign, Alloc
Function Set	Up, Down, Same, CFM
Raw Fitness	$SPD + TFTE * FTEW \sum (FRI_i * FRIW_i + PRI_i * PRIW_i + CRI_i * CRIW_i)$ (see section 3.6 Fitness Function Evaluation)
Parameters	Population size $M = 3000$ Maximum number of generations, $G = 100$ Crossover = 90% Mutation = 3% Reproduction = 7%
Success Predict	None – search for the shortest simulation duration with the given quality constraints

Results

The best individual found by EOD in generation 46 beats the August 1st deadline by far. The EOD solution shortened the simulated duration from the original Nov 9, 1988 to April 27, 1998 while maintaining the quality levels within the required limits. The EOD accomplished this by making the following modifications (see also Figure 4.12):

- Adding a total of three FTEs to different positions
- Reassigning five activities to different actors
- Changing Matrix Strength from medium to high
- Making five modifications in the supervision hierarchy

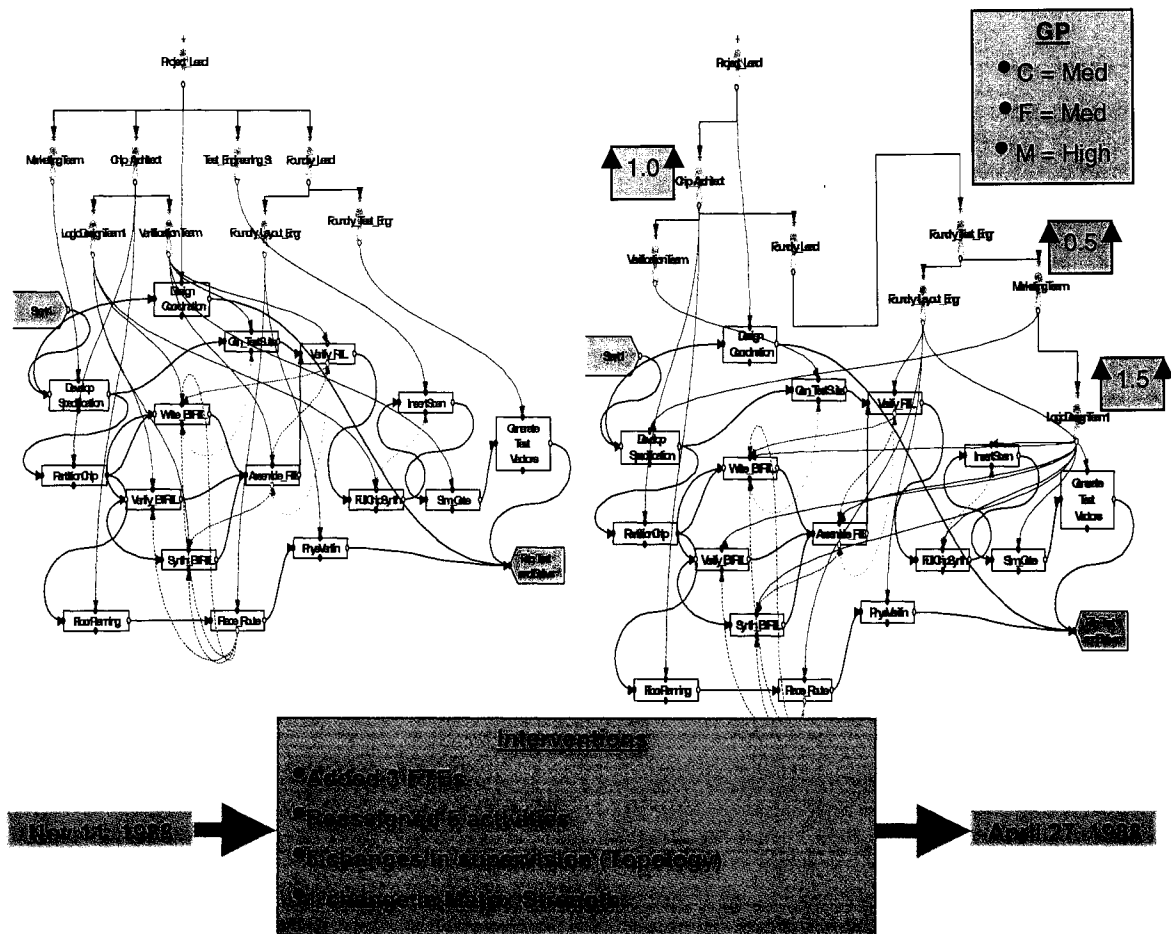


Figure 4.13 EOD Suggested Solution met the August First Deadline

The left figure above shows the ASIC design project organization before the GP intervention, and the right figure shows the project organization after the suggested interventions by GP. The result was obtained by adding 3 FTEs, reassigning activities, making changes in the reporting hierarchy, and making changes in the decision making policy.

The best solution of generation 46 is shown in lisp-type format below:

```
(Same (Same (Up (Up (Assign P6) (Assign P2)) (FTE P2)) (Up (Same (Assign P1) (Same (Same (Down (Same P0 P0) (FTE P4)) (Same (Down P3 P1) (Assign P5))) (Same (Up (Same (Up (Same (Same (Assign P5) (Down (Up (Same (Same (Down P3 P1) CFM) (Assign P6)) (Assign P1)) (Assign P8))) (Assign P2)) (Same (Same (Assign P1) (Assign P2)) (Same (Same (FTE P1) CFM) (Assign P6)))) (Up P5 P1)) (Up (FTE P4) (Up P4 (Same (Assign P1) (Same (Assign P1) (Assign P6)))))) (Same (Same (FTE P1) (FTE P5)) (Assign P6)))) (Assign P6)) (Same (Up (Up P3 P6) (Up (FTE P2) (Up (Same (Same (Down P3 (Up P5 P1)) CFM) (FTE P5)) (Same (Assign P1) (Same (Assign P1) (Assign P6)))))) (Same (Same (FTE P5) (Same (Assign P1) (Assign P6))) (Assign P6))) (Assign P6)))
```

Discussion

Comparing the results with the biotech plant case, we can see that there is a much greater improvement from the GP interventions. This can be at least due two factors. The initial duration of the ASIC design project is longer than the Biotech plant (i.e., eleven months versus five months). So, there is more room to work with. Second, there are nine actors and 16 activities involved in this case and seven actors and 15 activities in case of Biotech plant. So, once again GP has more variables to work with.

In addition, unlike the Biotech plant case, we could do the supervision hierarchy changes at the same time that other changes were happening so we did not require a complementary run.

4.4 Are the Results Human Competitive?

This section compares the GP results against the historical Student/Project manager results with the objective of demonstrating the human competitiveness of the EOD model. As mentioned before, every year since 1997, multiple student teams and practitioner project manager teams at Stanford have been given challenging real-world assignments to redesign the organization of biotech plant case and ASIC design case mentioned in sections 4.2 and 4.3 of this chapter. Unfortunately, for the early years and the CIFE¹¹ summer workshops—a management workshop taught at Stanford —, no outcome data was recorded for these cases. However, for the past five years the following data for the number of individuals and groups who have tried the biotech plant case is known to be accurate.

Table 4.5 below show the number of students in each class, the number of groups formed, and the best results that they were able to get for that year for the biotech plant case study. Students in each group first were asked to try to find the best solution by themselves. Then the individuals in each of these groups put their solutions together to

¹¹see Center for Integrated Facility Engineering (CIFE) website: <http://www.stanford.edu/group/CIFE/> for more information.

try to find an optimal solution. As shown below, a total of 182 individuals and 29 groups have tried the biotech plant case study and the best results to date by humans is Dec 7, 2000.

Table 4.5 Best Human Results Produced for the Biotech Plant case study

Course / Academic Year	Number of Students	Number of Groups	Best Group Solution
CEE-242			
2000-2001	20	3	December 12 th
2001-2002	39	5	December 8 th
2002-2003	28	4	December 10 th
2003-2004	37	3	December 8 th
2004-2005	35	4	December 7 th
CEE-047Q			
2000-2001	5	3	December 9 th
2001-2002	6	3	December 7 th
2002-2003	8	4	December 8 th
Total	182	29	

For the years 1997 through 1999, the ASIC Design case was used for the CEE-242 (Organization Design for Projects and Companies) course. However, the best results were not documented. What is known is the number of students and student teams in each class (see table 4.6 below), and the fact that neither the actual project team at the ASIC design firm which provided this case nor the graduate student teams were able to meet the deadline mentioned in the ASIC Design case study using VDT as an analysis tool in trial and error mode.

Table 4.6 Graduate Students who tried the ASIC Design Case Study

Course / Academic Year	Number of Students	Number of Groups	Best Group Solution
CEE-242			
1997-1998	28	3	unknown
1998-1999	28	3	unknown
1999-2000	32	4	unknown
Total	88	10	

In addition to data mentioned above, practitioners who attended the CIFE summer program tried the case studies. Unfortunately, no accurate data was recorded as to how many people actually participated, how many groups were formed and/or the best results obtained in the related sessions. However, what we know, according to the CIFE data, is the total number of the CIFE attendees for the year that case studies were given, and the fact that the project managers' solutions did not beat the best student team results.

Table 4.7 Total Number of CIFE Attendees for the Years that Case Study Was Given

Year	CIFE Summer Workshop Attendees
1998	46
1999	36
2000	86
2001	74

As mentioned in section 4.2.3 the best result produced by our EOD model for biotech plant case study is November 30, 2000. This result is 9.1% better than the best human results produced, as demonstrated previously in table 4.5. So, our GP-produced result is competitive with — and surpasses — a challenging human-produced result over an extended period, and satisfies the following three of the eight best human results criteria already mentioned in section:

- (E) The result is equal to or better than the most recent human-created solution to a long-standing problem for which there has been a succession of increasingly better human-created solutions.
- (G) The result solves a problem of indisputable difficulty in its field.
- (H) The result holds its own or wins a regulated competition involving human contestants (in the form of either live human players or human-written computer programs).

4.5 Does it Matter Where We Start?

In order to verify how much the EOD's final solution depends on the initial project manager designs, we modified some of the attributes of the initial design of biotech plant manually, so the schedule end date was intentionally worsened from February 16, 2001 to December 26, 2001. We ran EOD on this project organization, which was producing a result about 11 months worse than before and with quality risk levels that were much

worse than the initial case. GP still was able to reduce the schedule to Dec 6, 2000, one day better than the best human solution. This provides some evidence that the initial starting point has little or no effect on whether GP finds a near-optimal solution.

4.6 Does EOD Always Find the Optimal Solution?

The answer to this question is that it really depends on how complex the problem is. For example, for the phase I case, mentioned in section 4.2.1, where only skill levels of actors could be changed, GP always found the optimal solution. However, for the biotech plant case, for example, when we tried 17 runs with different starting points and without using the skill level and reporting hierarchy modifications, GP was able to find the best solution (i.e., 6 days better than the best human solution) only 24% of the time. However, 41% of the time GP found a solution which was one day better than the best human solution and the remainder of the time found a solution that was worse than the best human solution by only 1 to 6 days.

4.7 Should CEOs Mop Floors?

Since EOD is an evolutionary approach and is not governed by “knowledge-based” rules like an expert system, it can create designs that are surprising and non-traditional or atypical. For example, in one of our preliminary runs of the ASIC design case, EOD created a design with supervision loops — where a subteam leader was reporting to a project manager and this project manager was also reporting to the same subteam leader. In a real-world situation this would be unusual, so a user who uses VDT would not normally create such a design. However, since VDT was allowing this condition, EOD in one case was able to find a solution, which used supervision loops, and yet produced good results.

As described in section 3.5, we added constraints to our EOD model to prevent such cases in the interest of efficiency, although this violates the open-ended recombination and mutation spirit of pure evolutionary computing. However, this made us think, “What if our model suggests solutions that would not make sense for human designers with a rational mind? And what if suggested interventions do not make sense?”

Some of the possibilities are as follows:

- 1 Maybe some of the original model creation assumptions are inaccurate
 - o then we should revise the organizational model.
- 2 Maybe there is a (not-found-to-date) bug in VDT
 - o then we should fix the bug.
- 3 Maybe the fitness function is not set accurately
 - o then we should define a better fitness function.
- 4 Maybe the CEO should, in fact, mop the floor!
 - o then we should let Out-of-Box Thinking (OBT) happen if it is for the benefit of all!

This is one of the primary advantages of using evolutionary design approaches for practical applications. Since it does not use human-provided logical rules to generate its solutions (like many knowledge-based expert systems do), a GP can sometimes produce interesting results that make us stop and think! As Bob Sutton (2002) mentions in his book: Weird Ideas that Work, the best way to learn from counterintuitive ideas is to keep asking yourself: *What if these ideas are true?*

⇒ Chapter 5

Validation against Organizational Contingency Theory

“The truth of a theory is in your mind, not in your eyes.”

-- Albert Einstein

In this chapter, we discuss the validation of the GP postprocessor using propositions from organizational contingency theory some of which have been around for more than 50 years. We use a set of propositions suggested in Burton and Obel’s (2004) book – *Strategic Organizational Diagnosis and Design: The Dynamics of Fit* – and show that in some cases the results found by GP are in accordance with those of the suggested propositions. In other cases, we demonstrate that these propositions can be extended and/or more precisely defined based on the relative emphases of the projects’ sponsors on different performance outcomes such as time, cost, or quality of the project. Before turning to our methodology and results section, we introduce organizational contingency theory, organization design, and the criteria used by Burton and Obel to select among possible designs.

5.1 Organizational Contingency Theory – Background

Organizational Contingency Theory was first defined by Lawrence and Lorsch (1967) who argue that the amount of uncertainty and rate of change in an environment impacts the development of internal features in organizations. They conducted empirical studies of organizations with different levels of environmental uncertainties ranging from low to high in order to demonstrate these relationships.

Jay Galbraith (1973) states that there are two underlying assumptions in contingency theory. The first assumption is that “there is no one best way to organize.” The second is that “any way of organizing is not equally effective.” The first assumption challenges those theorists who argue that general principles can be developed to fit all organizations in all times and in all places. Burton and Obel (2004) argue that this kind of view

overlooks both the diversity of organizations as well as the diversity of the tasks undertaken by these organizations. The second assumption challenges the view of those early economists who argue that organizational structure is immaterial to organizational performance.

Scott (1981) adds a third assumption to represent the position of the contingency theories: “The best way to organize depends on the nature of the environment to which the organization relates.” In other words, the organization whose internal features best fit the demands of their environments will attain the best results.

Environment is only one of the contingency factors that an organization can face. Throughout the past few decades contingency theory has been very much extended. Researcher has found many more factors that the design of an organization can be dependent or contingent on, for example, size, technology, leadership, geography, management style, climate, ownership, and strategy (Penning, 1987, Lawrence, 1993, Burton and Obel, 2004).

As we will show in the later sections of this chapter, for our validation purposes we choose propositions that relates to technology as a contingency factor since the complexity of technology can be easily simulated in VDT, the organization analysis tool incorporated in our GP optimizer. Other factors such as environment, for example, can not be directly represented in VDT. As mentioned in Chapter 2, environmental uncertainty can be modeled in VDT by creating different scenarios of a project such as adding, subtracting or modifying positions and/or tasks and changing task durations (see section 2.2).

5.2 Organization Design

Burton and Obel (2004) describe an approach for diagnosing and designing organizations built on a knowledge base of organizational theory. The theoretical foundation for the decision model offered is contingency theory, with organizations modeled as “information processing systems.” The authors incorporate literally hundreds of studies

from more than five decades of contingency theory research on organizations to synthesize some 450 key propositions in the form of “if...then...” statements. The book presents a useful aid to research if one has an interest in testing and extending the applications of contingency theory. It is widely recognized as an authoritative compilation of contemporary management theory and practice applied to designing organizations. The focus of their research is on the relationship between uncertainty and the need for more information processing capacity. For example, they state, “The basic design problem is to create an organizational design that matches the demand for information processing with the information processing capacity.”

Unlike VDT, which is an analysis tool based on a bottom-up and micro view of the information processing approach to organization design for a specific organization, OrgCon (the computational diagnosis model developed by Burton and Obel) takes more of a top-down, macro view of the information processing approach to organizational design. Their model integrates multiple, sometimes conflicting, bits and pieces of the contingency theory literature and incorporates a range of contingency factors including:

- Management Style
- Organizational Climate
- Size/Ownership
- Environment
- Technology
- Strategy

The “if... then...” statement propositions mentioned above generate recommendations depending on the contingencies present. The recommendations relate elements of the organizations context to specific structural properties and overall structural configurations of the organization and include:

- Structural configuration, e.g., simple, functional, divisional, etc.
- Complexity/Differentiation
- Formalization
- Centralization
- Span of Control
- Rules/Procedures
- Professionalization

- Meetings
- Reports
- Communications
- Media Richness
- Incentives

As we mentioned in the previous section, we chose technology as the contingency factor to test and its relationship to formalization and centralization properties of an organization. This is because the technology factor can be well represented in VDT by varying task attributes.

Before we show some comparison results between what our GP postprocessor finds and some of the propositions mentioned in Burton and Obel's book, we need to have a clear definition of organization theory and organization design and understand the distinction between the two. In addition we need to be clear about what we mean by a "good" design and we need to describe technology as a contingency factor.

5.3 Definition of a "good" Design

Organizational theory is a positive science that states our understanding about organizations and how they function, and contrasts that understanding with a view of how the organization could potentially function. Organizational design, on the other hand, is a normative science with the goal of prescribing how an organization should be structured in order to function effectively and efficiently (Burton and Obel, 2004) in a given context. Burton and Obel use the three criteria of effectiveness, efficiency, and viability as measures for selecting the appropriate organizational configuration and organizational properties. Here is how they define these three criteria:

- **Effectiveness:** An organization is effective if it realizes its purposes and accomplishes its goals.
- **Efficiency:** An organization is efficient if it utilizes the least amount of resources necessary to obtain its products and services.
- **Viability:** An organization is viable if it exists over a long period of time.

They continue that:

“Effectiveness is contrasted with efficiency. Effectiveness is doing the right thing; efficiency is doing it right. Usually, effectiveness does not incorporate efficiency: that is, an organization can accomplish its goals but be quite inefficient in its use of resources. An efficient organization uses few resources but may not accomplish its goal well. We want to design organizations that are both effective and efficient, as both are likely to be important for viability or long-term survivability. However, an organization can survive for many years and not be known to be particularly effective and efficient. The US government is relatively long-lived but is not known for effectiveness in many of its activities and certainly is not known for its efficiency.”

Since VDT simulates project organization design rather than organization design in general, some of the above criteria cannot be used directly for the purpose of comparison. The three primary organizational performance outputs of VDT are time, cost, and quality. So, the viability criteria, for example, cannot be measured when we search for the near optimal organization design using VDT as our analysis tool. However, the effectiveness and efficiency of a project organization can be measured in terms of the initial objective(s) of the project. For example, effectiveness can be measured by whether or not a project was able to finish by a certain time and/or produced a certain quality of outcome. Or efficiency can be measured in terms of minimizing cost and/or using the minimum amount of labor resources, which in VDT terms is equivalent to minimizing total Full Time Equivalent (FTE)-days of labor used in a project.

5.4 Technology as a Contingency Factor

There is wide variety of definitions for technology. However the following general definition of technology by Robbins (1990) is widely accepted: Technology is the information, equipment, techniques and processes required to transform inputs into outputs (Robbins, 1990, p. 176). There is not one settled concept and measure for technology. Burton and Obel find that technology can be measured along the following four dimensions, which have been discussed most extensively in the literature:

- Manufacturing, service, retail, and wholesale

- Unit, mass, and process
- Routine or nonroutine, and
- High or low divisibility

The various dimensions of technology have different effects on organizational design. For the purpose of our validation, we turn our focus to the “routine/nonroutine” dimension of technology. Routineness is a central concept in technology that has been used by many researchers (Robbins, 1990; Miller et al., 1991). Burton and Obel use Perrow’s (1967) original concept of technology and define routineness as follows:

- *Routine Technology*: Contains easy-to analyze problems and few exceptions
- *Non-routine technology*: Contains difficult-to-solve problems and many exceptions

The above definitions of routine and non-routine technologies can be operationalized well within our VDT application. By changing two variables in VDT, we can easily define a set of activities to represent easy-to analyze problems that only produce few exceptions; or we can define a set of activities to represent difficult-to-solve problems that create many exceptions.

For the remainder of this chapter we focus on the following three propositions suggested by Burton and Obel about technology routineness and its relationship to organization centralization and formalization properties:

- Proposition 7.1. If technology routineness is low, then formalization should be low.
- Proposition 7.2. If technology routineness is high, then formalization should be high.
- Proposition 7.7. If technology is high and the size of organization is small, then centralization should be high.

Although the definition of formalization that they use is not exactly the same as the way in which formalization is modeled in VDT, for practical purposes, they are essentially referring to the same thing. More formalization means more rules and more formalized communication between individuals within the organization. Increasing formalization in VDT causes participants to focus on communication to coordinate interdependencies in formal meetings versus ad hoc communications between individuals. In terms of centralization, the definitions used are identical. Centralization reflects where decisions are made in the hierarchy. High centralization means that top managers are involved with making most decisions and low centralization means that responsible positions tend to make their own decisions.

The notion of size of the organization, which is mentioned in the proposition 7.7 above, is not of concern for our validation experiment since we are designing only project organizations, which are usually small size organizations.

In the next section, we will describe how we set up our intellectual experiment to try to replicate some of the propositions mentioned above.

5.5 Intellectual Validation of the Postprocessor

Thomson et al. (1999) suggest a validation trajectory consisting of three levels to validate a computational emulation model. The three stages of the evaluation trajectory are reasoning, representation, and usefulness (see Figure 5.1). The first stage attempts to validate the reasoning assumptions of a model with toy problems to evaluate whether the micro behaviors have been correctly encoded and intellectual experiments to test whether the micro behaviors generate reasonable macro behaviors in terms of extant theory. The second stage attempts to demonstrate how well the simulation system can capture and simulate the important features being modeled in terms that are relevant to managers, and to show that the method that captures the data is reliable, reproducible, and generalizable. The third stage attempts to show the usefulness of the model by using it to analyze and suggest interventions to improve real project organization designs.

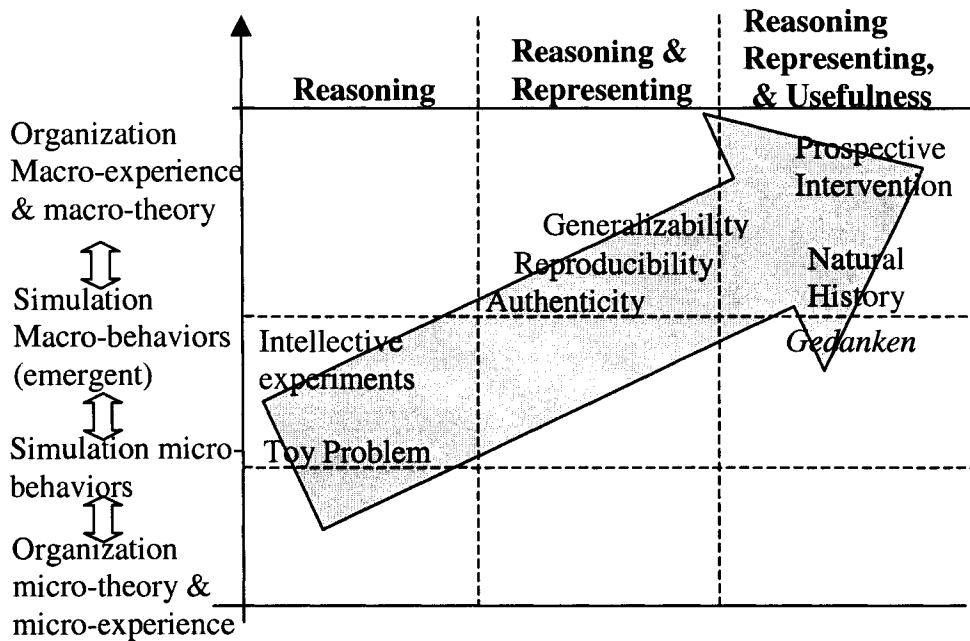


Figure 5.1 A Validation Trajectory Proposed by Thomsen et al. (1999)

This figure describes the interactions among the micro, meso, and macro level analysis of organizations. Simulation is a tool to bridge the gap between micro and macro theory and experience. Thomsen et al. suggest a three-stage validation process through reasoning, representation, and usefulness to validate organizational computational models.

As shown in the next section, we created an intellective experiment in order to validate the reasoning assumptions of our evolutionary model at the first stage. Then we use a couple of real project organization cases to verify that our model can be generalized to other real-world problems, as well. Finally, at stage three of the above framework, we demonstrated in Chapter 4 that our GP postprocessor has already produced human-competitive results, thus confirming the potential usefulness of our model. Stronger evidence of usefulness in the future requires that managers begin to make interventions in their real projects based on the results of the EOD.

In the following two sections, we show how we test the validity of our GP optimizer against the three organization contingency propositions mentioned in section 5.4. In this process, we examine the effect of technology routineness on formalization and centralization in VDT and compare our results with those found by our GP postprocessor.

5.6 Validation using Intellectual Experiment

For our intellectual experiment, we created a relatively simple ideal case of a project organization. This project included seven positions and seven activities (see Figure 5.2). The activities were each 100 days in duration and there were no interdependencies between the activities—i.e., no (green) communication links and no (red) rework links. The salary for the project manager was set at \$150/hour, the subteam leader at \$100/hour, and the subteams at \$65/hour. The project manager and the subteam leaders met every other week for one hour. Subteam leaders and their group met once a week for one hour. The level of application experience of the project manager and subteam leaders was set to “high” and remaining positions were left at the default “medium” level.

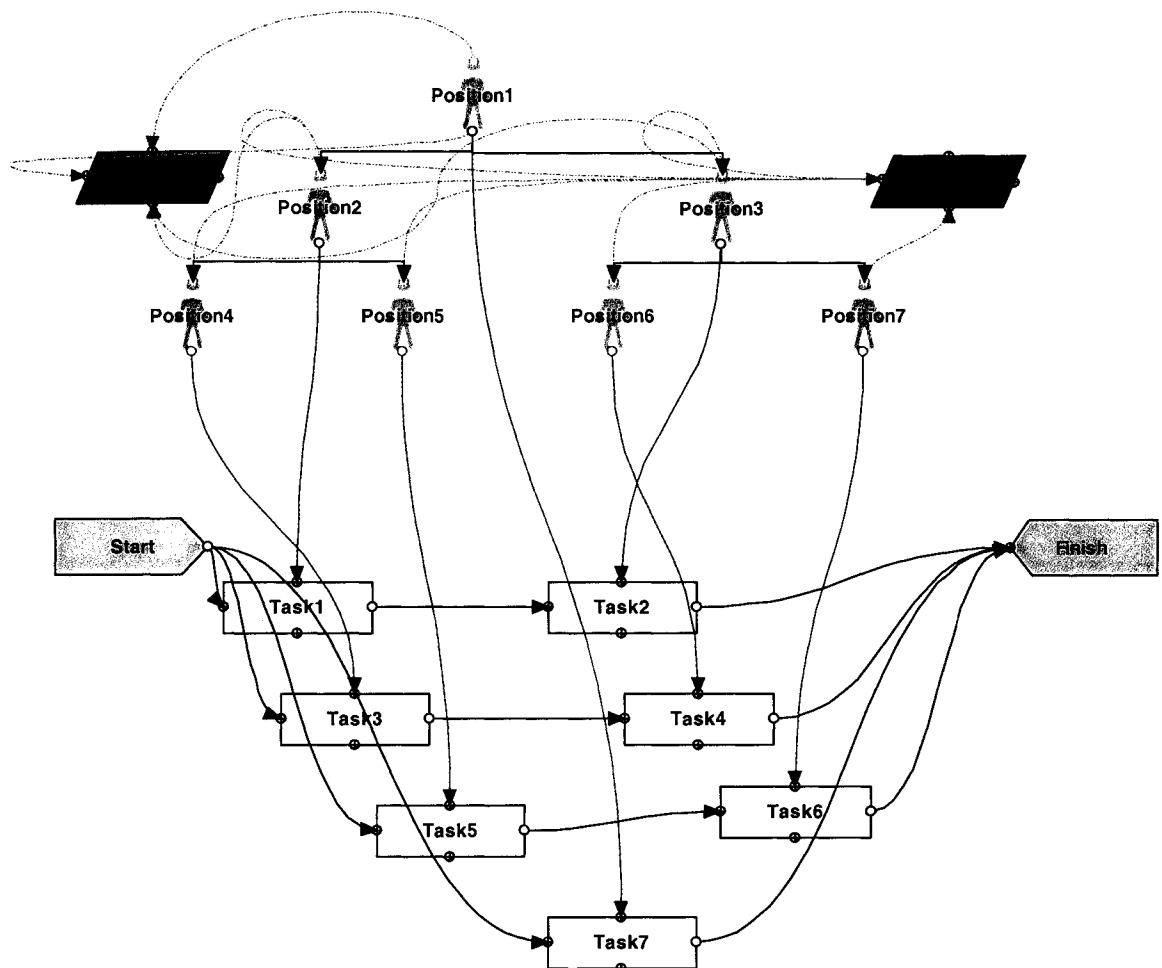


Figure 5.2 A Sample of an Idealized Project Organization in VDT

An idealized project work process and organization were designed to test the validity of our GP postprocessor against some propositions from contingency theory.

The general overview of our approach for intellectual experiments is as follows:

1. Contingency theory makes predictions about the effect of technology on decision making policy of organizations as presented in section 5.4.
2. We run VDT for a range of centralization and formalization values in each of the idealized cases to explore what its prediction are, and thus to be able to examine the model predictions and compare them to the theoretical predictions.
3. We run EOD to verify whether the solution to the idealized cases converges to the theoretically predicted values.
4. Using the idealized and real-world project cases, we interpret the results and show that we can refine and extend organization contingency theory.

In order to represent the routineness of the technology, we used variables FEP and PEP, which stand for Functional Error Probability and Project Error Probability respectively in VDT. FEP represents the probability that a subtask will fail and require rework. PEP is the likelihood that a task will fail and generate rework for itself and for all failure-dependent tasks (tasks connected to it by rework links). As mentioned earlier, Burton and Obel defined routine technology as a project that contains easy-to-analyze problems with few exceptions and “non-routine” technology as a project that contains difficult-to-solve problems with many exceptions. We used FEP and PEP of 0.01 when activities were very routine, so they generate only few exceptions, and FEP and PEP of 0.1 when activities were non-routine and many more exceptions would occur¹².

For each case of routine or non-routine technology, we experimented with three values of high, medium, and low for centralization and formalization, and compared the outcomes in terms of schedule, cost and quality. We used high centralization and high formalization as a baseline and we showed a percentage of improvement or decline with respect to the

¹² Please note that variation in PEP does not affect project outcomes for the ideal cases, where there are no rework dependencies between activities, , For consistency and comparison with real-world cases, we make changes both in FEP and PEP.

baseline for the other cases. A positive percentage number shows that the project duration is longer or the cost is higher than the baseline, and a negative number (shown in red and in parentheses on vertical axis in the figures below) shows percentage improvement compared to the baseline—i.e., shorter duration and cost.

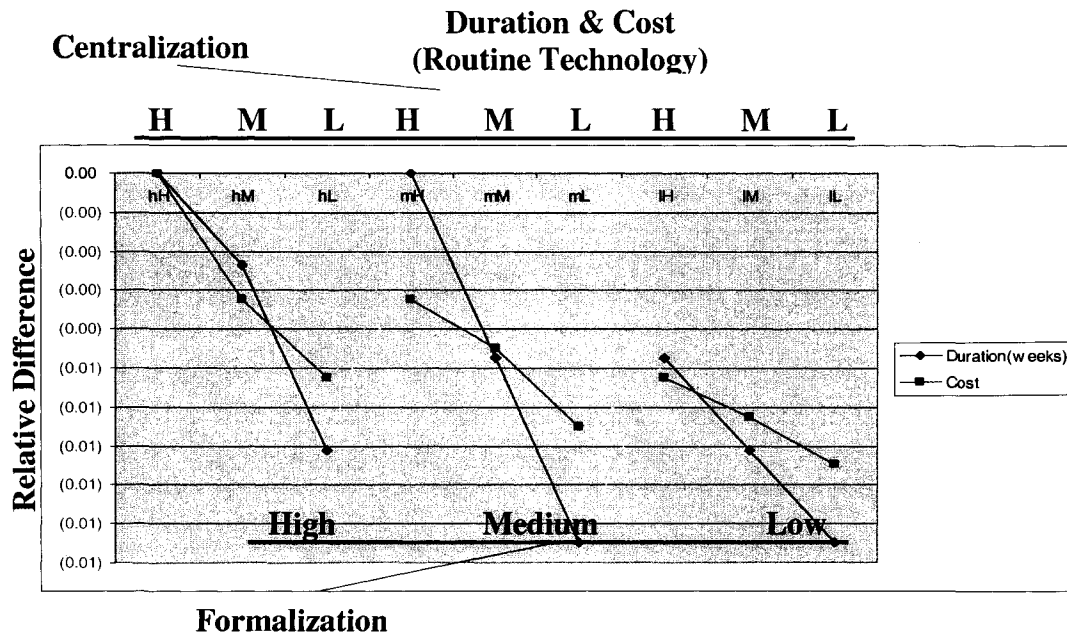


Figure 5.3 Comparing the effect of Formalization and Centralization on Time and Cost when Technology is Routine (FEP & PEP = 0.01)

Centralization and formalization have minimal effect on time and cost (less than 1%¹³) when technology is routine. The baseline case is high centralization and high formalization in the upper left of this diagram.

As shown in figure 5.3 above, centralization and formalization do not affect project simulation duration and cost significantly (less than 1% change). However, as shown in figure 5.4 below centralization's effect on process quality is significant (more than 50%) whereas formalization's effect on quality is minimal (less than 5%).

¹³ Note: The vertical axis on Figures 5.3 through 5.10 represents the percentage numbers as a fraction number between zero and one (i.e., one represents a 100% difference). Also negative numbers are shown in red and in parentheses.

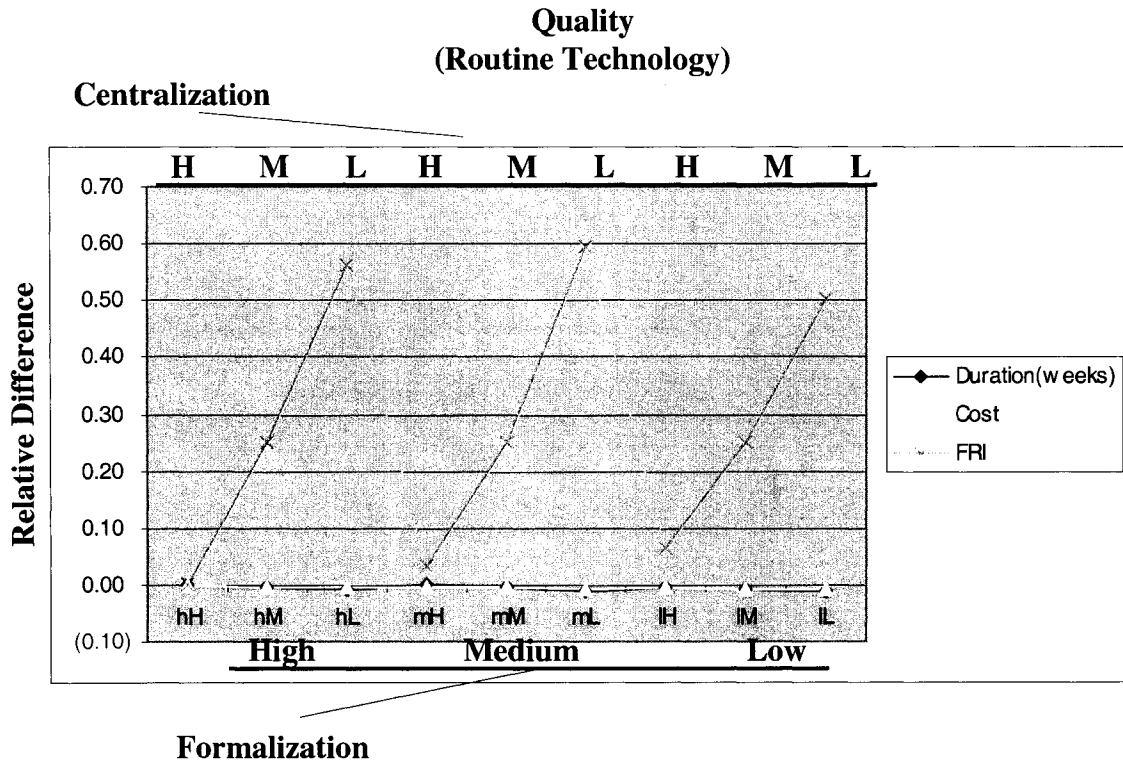


Figure 5.4 Comparing the Effect of Formalization and Centralization on Process Quality (FRI) when Technology is Routine (FEP & PEP = 0.01)

Although formalization does not affect quality significantly, centralization has a significant effect on quality (more than 50%) when technology is routine. Note: percentage relative differences on duration and cost are so small (less than 1%) that they overlap in this graph and are shown as a yellow line.

From figures 5.3 and 5.4 it is obvious that when technology is routine, centralization should be high since in that case the process quality is best and there is not much effect on cost and duration. This conclusion matches proposition 7.7, mentioned in section 5.4. However, as shown in the same figures above, formalization has minimal effects on all of the measured outputs, so it really does not matter what formalization is.

When technology is non-routine, the situation is somewhat different. Formalization has a moderate effect on cost and duration (about 6%) as shown in Figure 5.5 below. The lower the formalization, the better the cost and duration. And since the effect of formalization on quality is still insignificant (see Figure 5.6), we conclude that the lower the

formalization the better when technology is non-routine. This matches proposition 7.1 mentioned in section 5.4.

A conclusion on centralization cannot be made as easily as formalization. Like formalization, centralization has a moderate affect on cost and duration - the lower centralization, the better cost and duration (see Figure 5.5). However, as we lower centralization, we lose considerably in terms of quality (about 70%). This fact is shown in figure 5.6 below. Thus, there is a trade off between cost/duration versus quality.

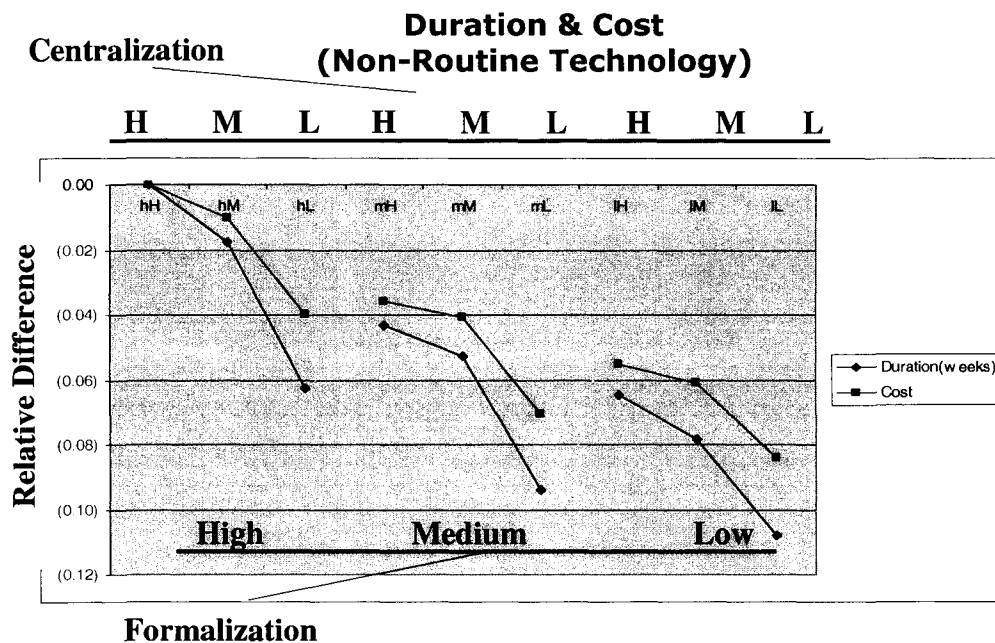


Figure 5.5 Comparing the Effect of Formalization and Centralization on Time and Cost when Technology is Non-routine (FEP & PEP = 0.1)

Centralization and formalization have moderate effect on time and cost (about 6%) when technology is non-routine.

For example, in this case, we found out that when there is less than 5 times as much emphasis on cost/duration as on quality, centralization should be high; otherwise centralization should be low.

We were able to verify this by setting up a fitness function with the following weight factors for our GP postprocessor:

$$f_{(t, c, q)} = 4t + 4c + q \quad (5.1)$$

where

t, c, and q stand for time, cost and quality respectively, and when time (t), cost (c), and quality (q) values were normalized. The normalization was done by dividing each value by the highest value obtained in all the combined cases of centralization and formalization.

We observed that whenever weights for time and cost were greater than or equal to five, our optimizer found that the centralization should be low, and otherwise it should be high.

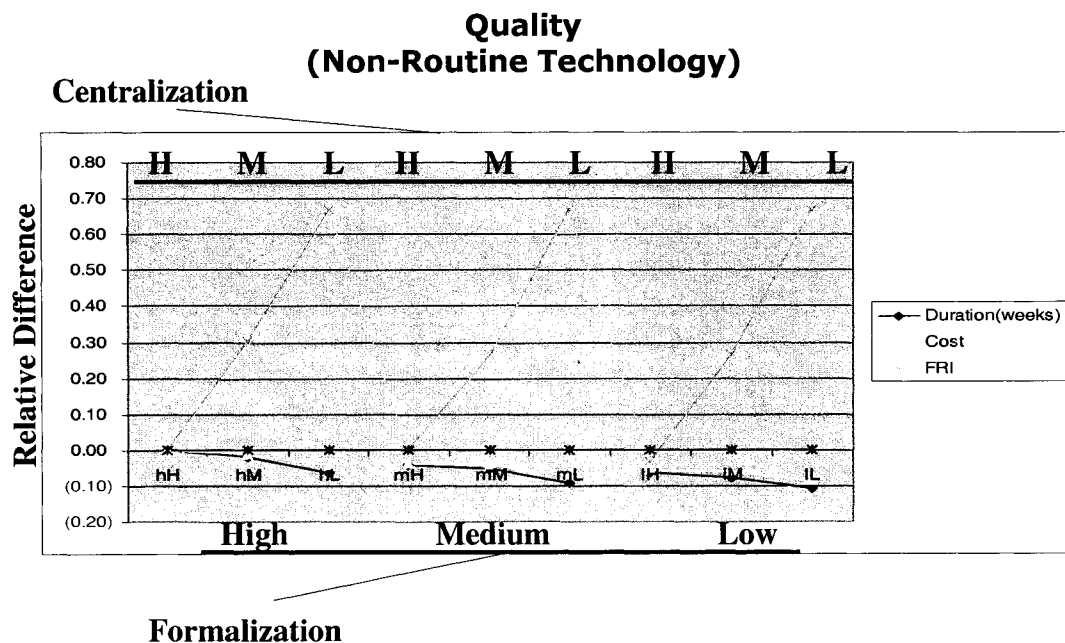


Figure 5.6 Comparing the Effect of Formalization and Centralization on Process Quality (FRI) when Technology is Non-Routine (FEP & PEP = 0.1)

Although formalization does not affect quality significantly, centralization has a significant effect on quality (more than 50%) when technology is non-routine.

Burton and Obel have no proposition for the above case (i.e., the effect of centralization when technology is non-routine). In this case using the GP postprocessor, we propose

that when technology is non-routine, the desirable level of centralization is a function of the relative weights on duration, cost and quality outcomes of a project organization.

Up to now we have only compared our results against theory using an idealized project organization. In the following section, we use a couple of real-world organizations and compare the results against our intellectual experiment.

5.7 Validation using Real-World Project Organizations

In order to verify whether or not our model can be generalized and whether it can adequately represent real-world organizations, we conducted similar experiments using two real project organizations mentioned in Chapter 4, namely the Biotech and ASIC Design projects, and compared our results with the idealized case discussed above.

In order to make the comparison easier, we made the following simplifications in the graphs:

- We used only project duration for comparison purposes since we found there was a very high correlation between duration and cost (see Figures 5.3 and 5.5), and we concluded that cost should follow the same pattern.
- We varied formalization and centralization only at two levels of high and low instead of high, medium, low.

The Biotech Plant project consisted of the same number of positions as the ideal case with 15 activities. However, there were six green communication links and eight red rework links between activities (see Figure 4.1). The ASIC Design case consisted of nine positions with 16 activities, three green communication links, and six red rework links between activities (see Figure 4.12).

5.7.1 Centralization and Formalization effects when Technology is Routine

Figure 5.7 demonstrates how project duration improves or declines based on variation in formalization and centralization for the three cases: Idealized, Biotech plant, and ASIC design. As shown below, when the technology is routine (FEP & PEP = 0.01), centralization has very little effect on project duration.

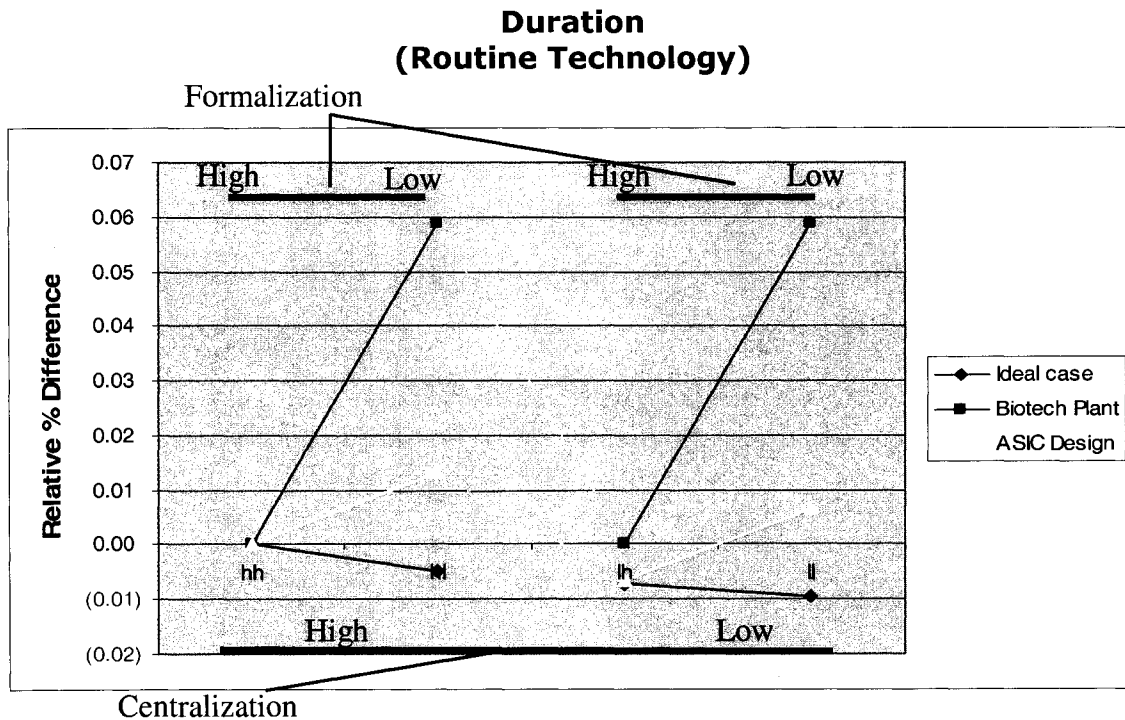


Figure 5.7 Comparing the Effect of Formalization and Centralization on Project Duration when Technology is Routine (FEP & PEP = 0.01)

Centralization has little effect on project duration when technology is routine. Formalization, however, can have a moderate effect depending on the number of communication links between activities.

However, formalization has a moderate effect depending on the type of projects.

Formalization has the largest effect on schedule for the Biotech Plant project, which has the most communication links between the activities, and the least effect on the idealized case, which has no interdependencies between activities.

Although centralization does not affect schedule, it has a major effect on quality. As shown in Figure 5.9, the higher the centralization, the better the quality. For example, for the ideal case quality improved by 55% and for the real project cases quality improved

between 20% to 30%. Because centralization has no effect on schedule and a major effect on quality, we concluded that centralization should be high when technology is routine.

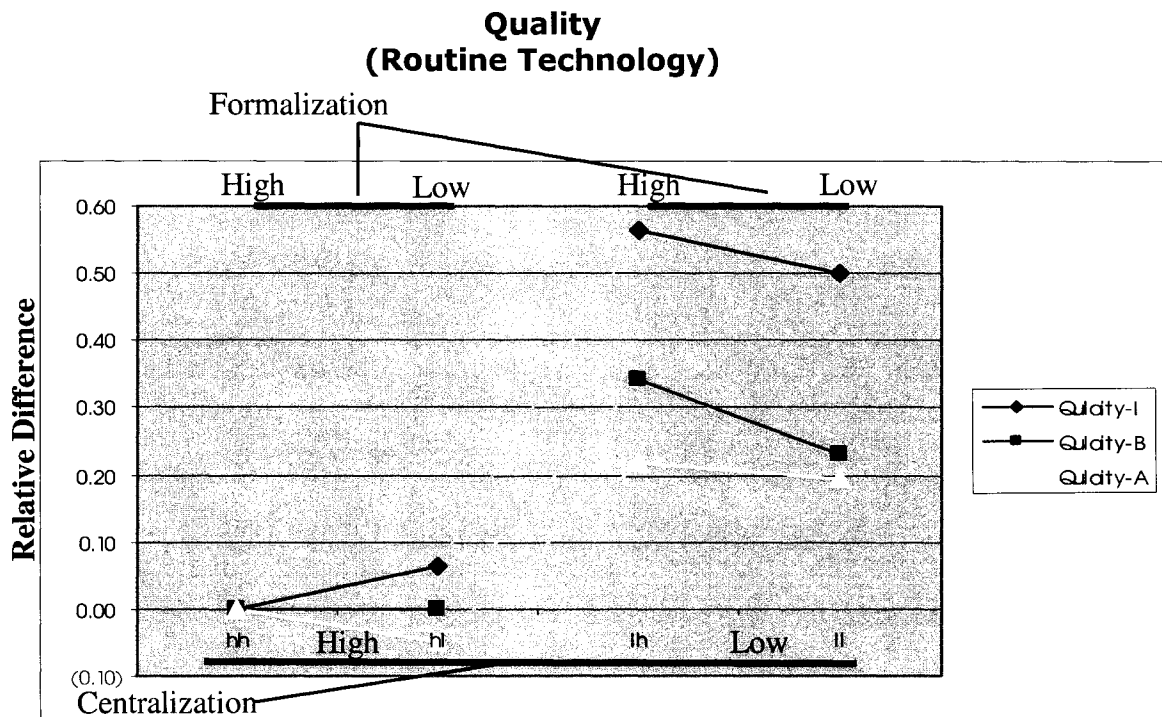


Figure 5.8 Comparing the effect of Formalization and Centralization on Process Quality when Technology is Routine (FEP & PEP = 0.01)

Although formalization does not affect quality significantly, centralization has a significant effect on quality (more than 50%) when technology is routine.

This conclusion was confirmed by our GP postprocessor. For different scenarios of the ideal, Biotech plant, and ASIC Design cases, where FEP and PEP were set to 0.01, the optimal solution found by our postprocessor always suggested that centralization should be set to high. This result confirms proposition 7.7 suggested by Burton and Obel. This demonstrates that the optimal organization evolved by our EOD model is consistent with contingency theory that has been empirically validated over many years.

Formalization has a moderate effect on project duration (up to 6%) when there are interdependencies between activities, and almost no effect (less than 1%) when there are no interdependencies between activities (the ideal case). After establishing the fact that centralization should be high, as mentioned in the previous paragraph, we could focus on

the left hand side of Figures 5.7 and 5.8, where centralization is high, in order to see the effect of formalization. We can see that in both graphs schedule and quality improves in most cases when formalization changes from low to high. Thus, when technology is routine, we conclude that high centralization and high formalization is the optimal design, since quality outcomes for that case are the best.

5.7.2 Centralization and Formalization Effects when Technology is Non-Routine

When we set the FEP and PEP to 0.1 (non-routine technology), centralization had a significant (about 70%) impact on quality like routine technology (see Figure 5.10). The higher the centralization, the better the quality. However, unlike routine technology, the effect of variation in centralization on schedule was noticeable (between 5 to 10%). Thus, the lower the centralization, the shorter the duration of the project.

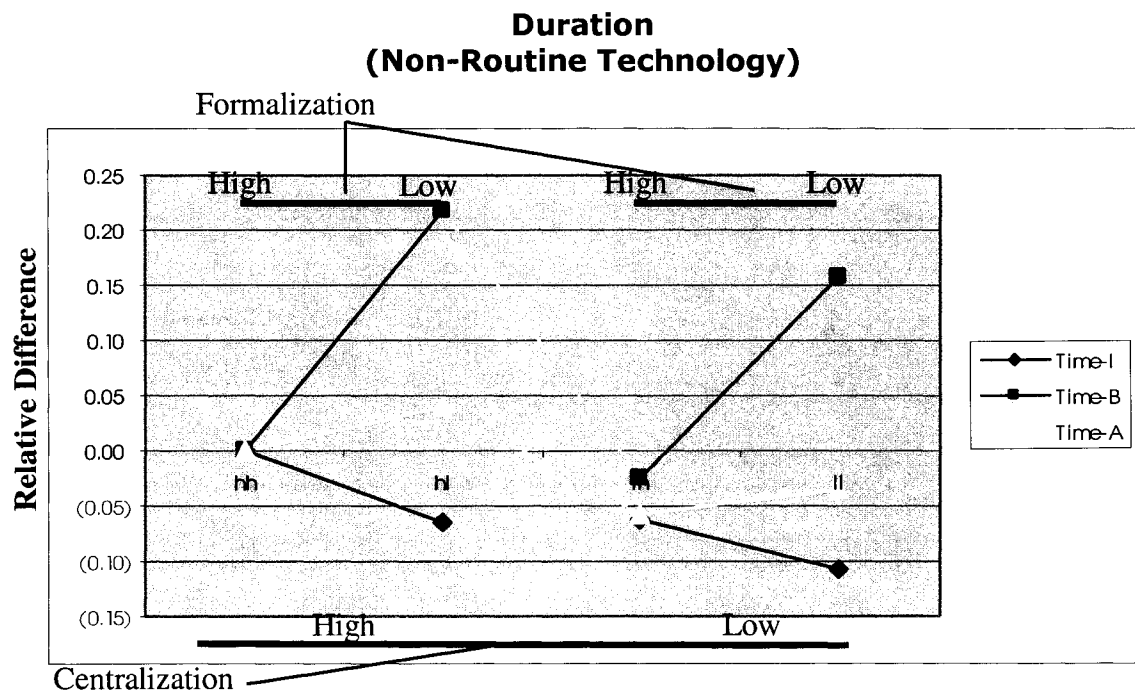


Figure 5.9 Comparing the Effect of Formalization and Centralization on Project Duration when Technology is Non-routine (FEP & PEP = 0.1)

Both centralization and formalization have significant effects on the duration of a project. As the number of interdependencies between activities increases, formalization should move from low to high.

The lower the centralization, the shorter the duration, but the worse the quality; and the higher the centralization, the longer the duration, but the better the quality (see Figures 5.9 and 5.10). This means that there is a trade off between schedule and quality. The best solution in this case depends on the specific project objectives and on whether we place greater emphasis on schedule vs. quality. In genetic programming terms this means how we set the weights in our fitness function affects our final results, as we discussed in section 5.3.

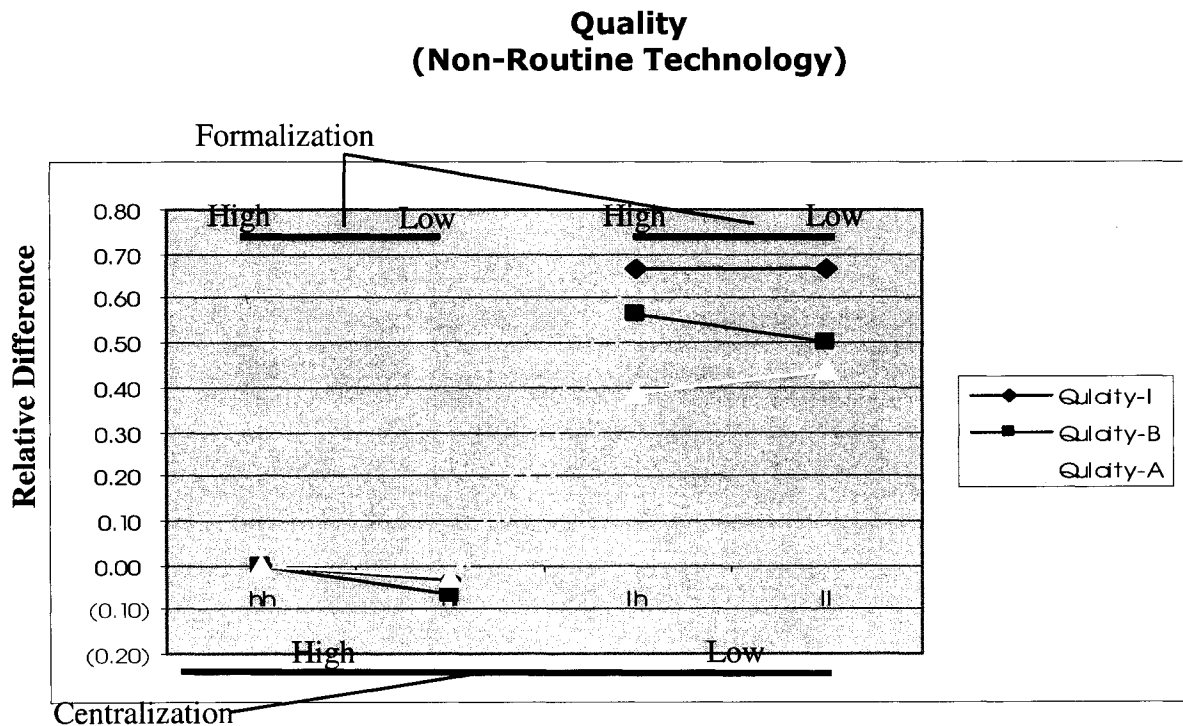


Figure 5.10 Comparing the Effect of Formalization and Centralization on Process Quality when Technology is Non-Routine (FEP & PEP = 0.1)

Although formalization does not affect quality significantly, centralization has a significant effect on quality (between 40% to 60%) when technology is non-routine.

As we showed before in section 5.6 and in Figures 5.9 and 5.10 above, when there are no reciprocal interdependencies between activities (green lines), formalization should be low. However, as the number of communication links between activities increases, as in the case of the Biotech and ASIC design projects (outcomes represented by the data points with pink squares and yellow triangles in the graphs), formalization should be high.

Figure 5.11 below shows an example of how our GP postprocessor improves the organization design to maximize the likelihood of a good outcome of a project, and makes changes to the decision making policy as it goes through each generation when the emphasis is set on quality for a real-world project.

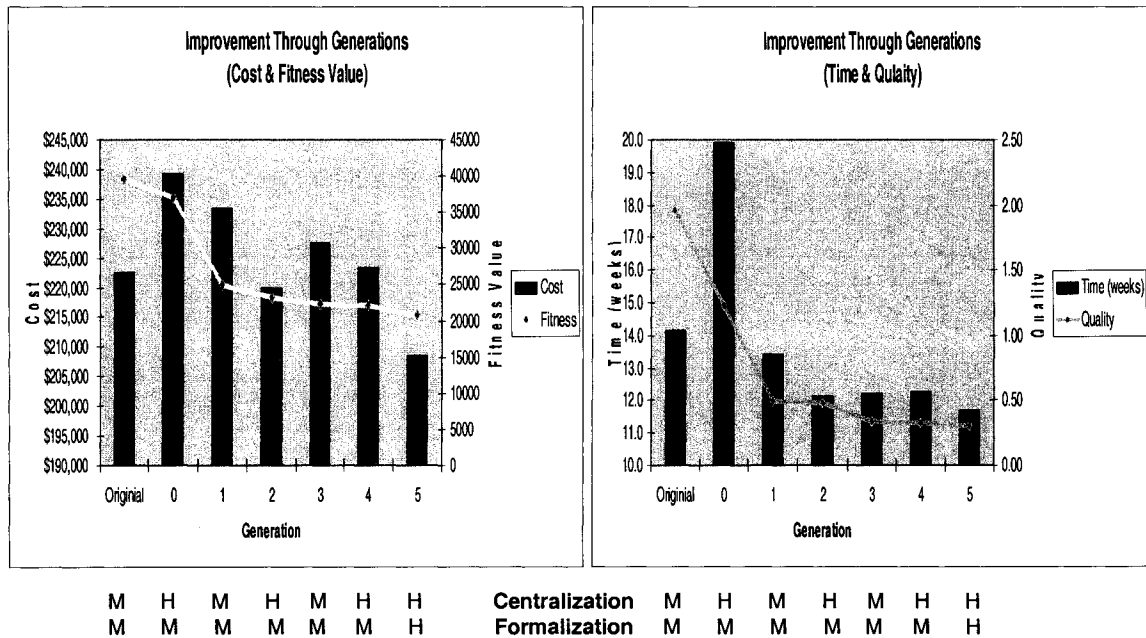


Figure 5.11 Sample of a Project Outcome Improvement and Changes to Decision Making Policies when Emphasis is set on Quality as GP Evolves a Real-world Project Organization

Although project outcomes can vary as the postprocessor goes through its generations, the overall fitness is improving consistently. Note: quality is shown as the average of the sum of FRI and communication risk index in the above figure.

In the next section, we summarize all of these findings. In those cases where we claimed to be extending contingency theory propositions, we make an attempt to explain why this might be the case for real-world project organizations.

5.8 Discussion of Results

In this section, we summarize our findings of our validation experiments, in which we simulated the effects of formalization and centralization on organization performance as technology routineness and number of interdependencies between activities changes.

These findings are summarized in Tables 5.1 and 5.2 in the following sections.

5.8.1 Effect of Formalization on Organization Performance

As mentioned in section 5.6, if there are few interdependencies between activities and technology is routine (lower left quadrant of Table 5.1), the level of formalization is immaterial. In this case, we can say that activities are so simple and trivial that everyone knows what to do. If there is a formal process and communication, people are going to follow it; and if there is none, they are still going to do the same thing.

If there are few interdependencies between activities and technology is non-routine (upper left quadrant of Table 5.1), formalization should be low. In this case, activities are complex; but, if professionals are flexible, they can come up with creative solutions to overcome that complexity. So it is good to be less formal, and to allow professionals to pursue their own solutions to their specific tasks.

If there are high interdependencies between activities and technology is routine (lower right quadrant of Table 5.1), formalization should be high. Coordination failures will be costly in this case, so it is advantageous to formalize communication and to set rules and programs to regulate and coordinate work.

Table 5.1 Summarizing our Findings on Formalization Properties Based on Technology Routineness and Activity Interdependency

The optimal formalization level is a function of routineness of technology and the level of interdependency between activities.

Formalization Level

Non-Routine Technology	Low (matches proposition 7.1)	High
	Does not matter!	High (matches proposition 7.2)
Routine Technology		
	Low Interdependency	High Interdependency

If there is high interdependency between activities and technology is non-routine, formalization should be high. There are so many unknowns. Professionals can come up with many conflicting ideas. Since the activities are all interdependent, in this case professionals can not find solutions that are effective overall, unless there is a structure that forces them to coordinate with others in order to get things done. For example, formal rules need to be set to hear all the options available before actions can be pursued.

In summary, we found that there is another contingency factor that could affect the desired level of formalization when designing a project organization, namely the level of interdependency (measured as the number of reciprocal interdependencies) between activities. Considering both the external contingency factor (technology routineness) and the internal contingency factor (activity interdependency), the Burton and Obel propositions 7.1 and 7.2 cover only two out of the four possible combinations mentioned above. In both of these cases, our findings agree with their propositions. We offer new propositions for the two combinations that they do not address.

5.8.1 Effect of Centralization on Organization Performance

As discussed in section 5.7.1 unlike formalization, interdependencies between activities do not affect the optimal level of centralization. If there is low interdependency between activities and technology is routine, centralization should be high. And if there is high interdependency between activities and technology is routine, centralization should still also be high (see Table 5.2). In this case, we can explain that a manager can relatively easily stay informed about overall operations, and therefore can handle the required information easily. Thus, there is little or no middle and/or upper management information-processing backlog; as a result, project duration is not affected. However, high centralization can improve quality significantly. This conclusion matches proposition 7.7 suggested by Burton and Obel. In case technology routineness is low, however, we conclude that centralization should be low if emphasis is on schedule and cost outcomes; and it should be high if emphasis is on quality outcomes.

Table 5.2 Summarizing our Findings on Centralization Properties Based on Technology Routineness and Activity Interdependency

The optimal formalization level is a factor of routineness of technology and the number of interdependencies between activities.

		Centralization Level	
Non-Routine Technology	Low if emphasis is on schedule High if emphasis is on quality	Low if emphasis is on schedule High if emphasis is on quality	
	High (matches proposition 7.7)	High (matches proposition 7.7)	
Low Interdependency		High Interdependency	

Again, we confirm two propositions from Burton and Obel, and provide two new propositions for combinations of contingency factors that they do not address, and refine the notion of contingency to suggest that what is a good organization design for a given case may depend on the relative emphasis placed on different performance outcomes — in particular, the emphasis placed on process quality, vs. time/cost outcomes.

5.9 Conclusions

In this chapter, we validated the findings of our evolutionary postprocessor against some propositions from well established organizational contingency theories. We used technology routineness as the main contingency factor and we demonstrated that:

1. Results found by our EOD model are in line with some technology-related contingency propositions suggested by Burton and Obel.
2. In addition, we were able to extend contingency theory by adding another dimension of internal contingency, i.e., activity’s interdependency.
3. We showed that, in some cases, the “optimal” structure depends on the relative emphasis of “time” and “cost” vs. “process quality” outcome metrics.

⇒ Chapter 6

Conclusions

“A conclusion is the place where you get tired of thinking. “

- Arthur Bloch

The overall goal of this thesis was to demonstrate how a novel approach based on evolutionary computing methods can be used to optimize project organization designs, and how our GP model was used to extend contingency theory to develop a richer “micro-contingency theory” for project organizations. Genetic Programming (GP) has been applied to a wide variety of problems in different fields, but never for designing project organizations. In addition, to the best of our knowledge, no automated optimization method has previously been used for any project organization design simulation model.

6.1 Overall Contributions of the Research

We have successfully taken the first steps toward the optimization of project organization design using evolutionary methods. We have designed and developed a postprocessor, based on GP techniques, that optimizes the output of Virtual Design Team (VDT) simulation models by evolving organizational attributes such as: (1) decision making policies, (2) individual/subteam properties, (3) activity assignments, (4) actors’ attention allocation, and (5) topology of the organization’s reporting hierarchy.

We have demonstrated that our Evolutionary Organization Designer (EOD) can evolve designs for a project organization whose performance surpasses the best outcomes achieved by individual human managers, and teams of managers, in a relatively short amount of time by varying the attributes listed above and evaluating the resulting fitness of alternative designs. The preliminary version of our GP postprocessor for VDT varied just the first four parameters and was able to beat the best human trial-and-error

performance achieved over the past eight years by more than 40 teams for a realistic organizational design problem. The final version also varies reporting relationships to produce even better performance outcomes.

This dissertation described the detailed design of our GP postprocessor and demonstrated how we explored the general concept of evolutionary methods and genetic programming to come up with a unique way of applying GP to project organization design.

In addition, we demonstrated that our model can extend organizational contingency theory to develop a richer “micro-contingency theory of project organizations,” by showing that the “optimal” organization structure for a given context depends on the relative importance of time, cost and process quality outcome goals in each case.

In June 2004, we presented the preliminary results of our research at the North American Association for Computational Social and Organizational Science (NAACSOS-2004) conference in Pittsburgh, Pennsylvania and at the Genetic and Evolutionary Computation Conference (GECCO-2004) in Seattle, Washington. At the latter conference, our paper won a Silver Medal Award for Human-Competitive Results.

The introduction of computational analysis tools for designing organizations in the 1990s was an important stepping stone in bridging between organization science and management practice. It did this by creating computer simulation programs that could emulate real organizations with some fidelity, and could then validate and extend contingency theories of organizational design (Horii, 2004).

The results of our research help to further strengthen this bridge between organization theory and management practice. This dissertation shows how organizational scientists can use the analysis results and the “near-optimal organization designs” produced by our postprocessor to test and refine their organization design hypotheses and develop new theories. At the same time, practitioners can benefit from the practical implications of

organizational theory embedded in modeling tools, to analyze and improve the performance of their organizations.

This research contributes to the state of knowledge in three disciplines: organization science; project management; and computer science. The following section elaborates our contributions to knowledge in these three areas.

6.2 Contributions to Social Science

Our Evolutionary Organizational Designer (EOD) affords a new modality for conducting organizational research—evolutionary computational experiments. Organizational contingency theory has developed through many decades of empirical studies on organizations that evolved and survived preferentially over the years in their market, social and political “ecosystems”. Our model replicates, in some sense, what actually happens to real organizations over the course of time. Those designs that are fittest evolve through generations to create better ones; and those that are less well adapted simply fade away. It could take an organizational scientist twenty years or more to observe whether organizations with certain structural characteristics would survive in a given natural situations. With our computational model this can be simulated for historical or projected future changes in the environments and objectives of organizations in a matter of minutes or hours.

Our EOD model can validate and extend extant contingency theory via a novel research modality. In Chapter 5 we demonstrated how we could validate our model against some aspects of contingency theory using “intellective” and “emulation” project experiments (Levitt, 2005). We chose technology as a primary contingency factor and showed that our GP postprocessor confirms contingency theory propositions suggested by Burton and Obel (2004). In the same way, it can be used as a means to generate additional organizational contingency hypotheses that can then be tested empirically.

Our model can also be used as a sensitivity analysis tool for organizational designers to discover which attributes have the biggest effects on project organizational outcomes in

different contexts, and how the attributes interact with one another. For example, we showed in Chapter 5 how the effect of formalization on organization performance varies as the technology routineness and number of interdependencies between activities changes.

In this process, we discovered that specific design attributes (e.g., interdependencies between activities) and the relative emphasis of time, cost and process quality outcome interact with the original contingency factors. Thus our EOD model can extend contingency theory in two ways:

1. It can help to develop a richer “micro-contingency theory” for project organizations by showing that “optimal” structure depends on the relative emphasis on schedule, cost and quality outcomes; and
2. It can help to explore the interaction effects between individual elements of organizational context and structural design on organization performance for a variety of plausible fitness functions.

6.3 Contributions to Project Management

The concept of organizational design may be a factor in resolving the fundamental weakness of organization and management theory — the so-called *relevance gap* between theory and practice (Romme, 2003). That is, organization and management theory is inclined to be neither clear nor pertinent to practitioners (e.g., Miner, 1997, Priem and Rosenstein, 2000). The study of organization currently utilizes qualitative and quantitative findings from descriptive social science research, but it also needs to engage this research in a prescriptive mode. This research has taken a step toward closing the *relevance gap* by addressing the issues of organizational design prescriptively in terms of fitness functions that describe the fitness of specific designs for “survival” and “reproduction” in the spirit of contingency theory.

The developers of computational modeling tools such as VDT and OrgCon (described in Chapter 2) have tried over the last fifteen years to close this gap by creating tools for

modeling and simulation that are rigorously based on organizational contingency theory, yet can be applied directly by practitioners in the field. With the introduction of optimizing tools such as our EOD model, the usability of such tools will be even more appealing to project managers and those practitioners in the field, who say, “We already knew what the outcome of this project would be, but how can we improve on it?”

One of the immediate benefits of our evolutionary model is that it can help project managers make better decisions and create better organization designs. We demonstrated (in Chapter 4) with two real project examples that, lacking analysis tools, project managers could not always create project organizations and schedules to meet their objectives. In later years, graduate student and project manager teams were able to produce better results using organizational analysis tools like VDT. And now project managers can create near-optimal design using our evolutionary optimizer in tandem with existing analysis tools like VDT.

Our GP postprocessor does not use logical design rules—in fact, it does not use any explicit knowledge-based system. If it did, the solutions produced would be constrained by the knowledge and logic that was built in. Clearly, logical thinking is useful for many purposes and usually plays an essential role in setting the stage for finding optimal solutions. However, logic alone is insufficient for generating radically innovative designs. Because of this unique characteristic of evolutionary methods — specifically genetic programming — our model can produce results that are sometimes counter-intuitive, and rather surprising. This can motivate Out-of-Box Thinking (OBT) mentioned in section 4.7. By generating counter-intuitive organizational designs that rational thinking could rarely produce, our model can help practitioners think beyond rational boundaries of logical thinking and come up with new ideas themselves.

In Chapter 4 we demonstrated that, unlike many other artificial intelligence and/or operations research techniques that produce only a single good solution, GP can generate a whole set of near optimal solutions. In just twenty to fifty generations of evolution, our postprocessor generates multiple near-optimal alternatives for project managers to

consider, of which some may be more feasible to implement than others. Thus it helps managers to come up with a selection of potential new approaches and ideas as starting points for their organization design problems.

Every project is unique with its own specific characteristics. It is true that everyone wants to finish a project in the minimum amount of time and cost, and with the best quality, but this is usually infeasible. As David (2000) reports NASA tried to make its projects faster, better, and cheaper in the 90's; they concluded that it has cost them too many failures in return! The balance between schedule, required resources, and quality is always a trade off. Experience shows that, for a given project, at least one of these factors typically has a greater degree of importance than the others. For example, meeting a fixed deadline might be the key criterion for success in one project, whereas staying within the budget or producing a high quality product might have the greatest importance in another. Our model allows product managers to come up with an adapted organization structure based on a specific project's relative degree of emphasis on schedule, cost, or quality. This can be done simply by changing the weighting factors for these outcome metrics in the fitness function of the GP.

6.4 Contributions to Computer Science

This research contributes to computer science by finding a way to formally represent both the design attributes and performance goals of previously ill-structured human organizations clearly enough that a GP can work together with an organizational analysis tool like VDT to evolve their designs against fitness functions and generate new and innovative designs.

This research thus opens the way to generalize a relatively new computer science methodology — genetic programming — to a new domain: organization and management science. Genetic programming has been applied in a wide variety of different fields, as listed in Chapter 2, but never to project organization design. Our new methodology makes the initial connections between evolutionary computing methods and the science and practice of organization design.

We showed in section 3.9 how our approach (starting from an existing design instead of from scratch) can create a powerful Human-Computer Interaction (HCI@) environment that can motivate human to do Out-of-Box Thinking (OBT). Our postprocessor model can start from any point in the design process to improve on an existing, plausible design. The advantage of this approach is that it creates opportunities for a continuous loop of human-machine interactions to search for a desirable solution. For example, after running the initial project design through the optimizer and observing the new design, the project manager might come up with new ideas to change some configuration attributes or modify some criteria manually. Then s/he can restart the optimizer from this new starting point. So, the interaction between the modeler and GP optimizer can be very close and very flexible. In a sense, we can say that both human and machine can learn from one another, to come up with a better and better organization design to satisfy the ultimate goals of the project.

This capability of both our GP optimizer and a human designer to iterate from any intermediate point in a design process might not be applicable to applications of GP in all areas. For example, a circuit designer might not be able to come up with new ideas just by looking at a complex evolved electronic circuit as easily as a project manager can do this by looking at an evolved project organization design. In the organization design domain, the number of objects and attributes is relatively small and the number of relationships between them is limited, so it is feasible to represent solutions and their performance outcomes graphically in a way that managers can easily understand. The advantage of applying this type of new interactive human-GP methodology may thus be unique to the domain of organization design and similar domains where the graphical representation and some of its implications can easily be understood by the human problem-solver. This assertion can be tested in future research.

6.5 Suggestions for Future Research

As mentioned earlier, this is only the initial stepping stone on a new path that can help future researchers make advances in the area of organization design.

The GP postprocessor described in this dissertation is a prototype that offers a proof of concept that evolutionary methods can be used effectively to optimize project organization design. We are confident that its current design can be much improved or completely overhauled. For example, unlike our approach, one might decide to start from scratch to search for new designs, which remains a challenging new area for future work. We claim to have made contributions to the field of computer science (as mentioned in section 6.4), but this was not meant to be a computer science research project. The emphasis of this dissertation was not on the computer programming aspects; rather it was on demonstrating an effective methodology to reach our initial objectives mentioned in Chapter 1. Thus, there is potential for a computer science student and/or a professional programmer to improve its current design and implementation significantly.

The current GP postprocessor is a standalone application that communicates externally with the VDT/SimVision organizational analysis software. Eventually, it needs to be integrated as part of VDT and become more user-friendly before it can be commercialized and made available to organizational consultants and managers.

The GP postprocessor only addresses the design of project organizations doing relatively routine tasks. This is a limitation of the VDT analysis tool with which our GP was paired in this set of experiments. The GP could relatively easily be adapted to work with analysis tools like OrgCon (Burton and Obel, 2004) that analyze the design of enterprises, and with other current and future organizational analysis tools to address a wider range of organizational types.

In addition, considerable work needs to be done in terms of optimization and efficiency of the current design. There are many parameters in genetic programming that can be set and adjusted for specific problems such as crossover rate, mutation rate, population size, etc. For our purposes, we chose a set of generally accepted genetic parameters which produced acceptably good results. Certainly, these values can be tested and optimized for different situations and problems.

As mentioned in sections 2.2 and 5.1, the current EOD model does not consider project environment changes since VDT does not directly represent environmental uncertainty. Because of this limitation, a great number of contingency theories, which are related to the project organization environment cannot be examined and/or extended. Therefore, there is a great need for research and expansion of VDT and EOD model, so organizational environment uncertainties can be directly represented, and thus create a new domain for doing research in the area of organization environment related contingency theory.

Moreover, the current VDT/EOD models assume that individual/organizational attributes stay constant during a project. For example, skill levels of individuals do not change during a project. Creating an environment where attributes can change dynamically and designing an evolutionary model that can represent these active changes is a great challenge that opens up a whole new area of required research.

Finally, we believe that one of the major contributions that our model can make is to provide a medium for developing new theories of organization design. For example, one can explore and test hypotheses about the circumstances in which, for example, hiring more people vs. training existing people in an organization would be more advantageous. Different scenarios can easily be modeled by setting different parameters within VDT and then tested by tuning the fitness functions for different cases.

References

- Banzhaf, W., Nordin, P., Keller, R.E., and Francone, F.D. (1998) *Genetic Programming - An Introduction*. San Francisco, CA: Morgan Kaufmann and Heidelberg: dpunkt.
- Beasley D., Bull D.R., Martin R.R. (1993) An overview of genetic algorithms: part 1, fundamentals. *University Computing* **14** 58-69
- Bentley, P. (1999) *Evolutionary Design by Computers*, San Francisco, CA: Morgan Kaufmann
- Burton, R., Obel, B. (2004) *Strategic Organizational Diagnosis and Design: The Dynamics of Fit*, 3rd edition, Boston, MA: Academic Publishers.
- Carley, K. M. and Gasser, L. (1999) *Computational Organization Theory* (Weiss, Gerhard, ed.) Distributed Artificial Intelligence, Ch. 7 Cambridge, MA: MIT Press.
- Carley, K. M., Ren, Y., and Krackhardt, D. (2000) Measuring and Modeling Change in C3I Architecture *In Proceedings of the 2000 Command and Control Research and Technology Symposium. Conference held in Naval Postgraduate School, Monterey, CA, June 2000. Evidence Based Research*, Vienna, VA.
- Chan, W., Chua, D. K., and Kannan, G. (1996). Construction Resource Scheduling with Genetic Algorithms, *Journal of Construction Engineering and Management*, June 1996
- Cheng, C.H. and Levitt, R.E. (2001) Contextually changing behavior in medical organizations, *Proceedings of the 2001 Annual Symposium of the American Medical Informatics Association*, Washington, DC.
- Christiansen, T.R. (1993) *Modeling Efficiency and Effectiveness of Coordination in Engineering Design Teams*, Stanford, CA: Stanford University Ph.D. Dissertation.
- Christiansen, T.R., Christensen, L., Jin, Y., Kunz, J.C., and Levitt, R.E. (1999) Modeling and Simulating Coordination in Projects. *IEEE Journal of Organizational Computing*, **9** pp.33-56.

- Cyert, R.M., and March, J. G. (1963 and 1992) *A Behavioral Theory of the Firm*, 2nd Edition, Cambridge, MA : Blackwell Business.
- David, L. (2000) NASA Report: Too Many Failures with Faster, Better, Cheaper, Space, 13 March
http://www.space.com/business/technology/business/spear_report_000313.html
- Davis, L. (1985) Job Shop Scheduling with Genetic Algorithms, *Proceedings of the 1st International Conference on Genetic Algorithms*, pp.136-140
- Dooley, K. (2002) *Simulation Research Methods: Companion to Organizations*, Joel Baum (ed.), pp. 829-848. London: Blackwell,
- Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966) *Artificial Intelligence through Simulated Evolution*. New York, NY: John Wiley
- French, M.J. (1994) *Invention and Evolution: Design in Nature and Engineering*, 2nd Edition. Cambridge University Press
- Galbraith, J.R. (1973) *Designing Complex Organizations*, Reading, MA: Addison-Wesley.
- Galbraith, J.R. (1977) *Organizational Design*, Reading, MA: Addison-Wesley
- Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley
- Goldberg, D.E. (1991) Genetic Algorithms as a Computational Theory of Conceptual Design. *In Proc. Of Applications of Artificial Intelligence in Engineering* 6, 3-16
- Hewlett, W.R. (2000) *Design and Experimental Results of a Post-Processor of VitéProject*, B.S. Honors Thesis, Symbolic Systems Program, Stanford University
- Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*. Ann arbor, MI: The University of Michigan Press.
- Horii, T., Jin, Y., and Levitt, R.E. (2005) Modeling and Analyzing Cultural Influences on Project Team Performance," *Computational and Mathematical Organization Theory*, 10, 305-321.

- Jin, Y. and Levitt, R.E. (1996) The Virtual Design Team: A computational Model of Project Organizations. *Computational and Mathematical Organization Theory*; **2** 171-196
- Kinnear, Jr. K.E. (1994) *Advance In Genetic Programming*. Cambridge, MA: The MIT Press.
- Koza, J.R. (1990) Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems. Stanford University, *Computer Science Department technical report STAN-CS-90-1314*. June 1990
- Koza, J.R. (1992) *Genetic Programming, On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: The MIT Press
- Koza, J.R. (1994a) *Genetic Programming II: Automatic Discovery of Reusable Programs*, Cambridge, MA: MIT Press.
- Koza, J.R. (1994b) *Genetic Programming II Videotape: The Next Generation*, Cambridge, MA: MIT Press.
- Koza, J.R., Bennett III, F.H., Andre, D., and Keane, M.A. (1996) Automated WYWIWYG design of both the topology and component values of analog electrical circuits using genetic programming. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*, pp 123-131. Cambridge, MA: The MIT Press.
- Koza, J.R., Bennett III, F.H., Andre, D., and Keane, M.A. (1999) *Genetic Programming III: Darwinian Invention and Problem Solving*. San Francisco, CA: Morgan Kaufmann.
- Koza, J.R., Bennett III, F.H, Andre, D., Keane, M.A., and Brave S. (1999) *Genetic Programming III Videotape: Human-Competitive Machine Intelligence*. San Francisco, CA: Morgan Kaufmann.
- Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G. (2003), *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*, Norwell, MA: Kluwer Academic Publishers
- Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G. (2003), *Genetic Programming IV Video: Routine Human-Competitive Machine Intelligence*, Norwell, MA: Kluwer Academic Publishers

- Koza, J.R., Keane, M.A., and Streeter, M.J. (2004) Routine high-return human-competitive evolvable hardware. In Zebulum, Ricardo S., Gwaltney, David, Hornby, Gregory, Keymeulen, Didier Lohn, Jason, and Stoica, Adrian (editors). *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware*, pp. 3–17. Los Alamitos, CA: IEEE Computer Society Press.
- Koza, J.R., Al-Sakran, S.H., and Jones, L.W. (2005) Automated re-invention of six patented optical lens systems using genetic programming. In Beyer, H.-G.; O'Reilly, U.-M.; Arnold, D.V.; Banzhaf, W.; Blum, C.; Bonabeau, E.W.; Cantu-Paz, E.; Dasgupta, D.; Deb, K.; Foster, J.A.; de Jong, E.D.; Lipson, H.; Llorca, X.; Mancoridis, S.; Pelikan, M.; Raidl, G.R.; Soule, T.; Tyrrell, A.; Watson, J.-P.; Zitzler, E. (editors). *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2005*. New York, NY: ACM Press.
- Kunz, J.C., Christiansen, T.R., Cohen, G.P., Jin, Y., and Levitt, R.E. (1998) The Virtual Design Team: A Computational Simulation Model of Project Organizations. *Communications of the Association for Computing Machinery (CACM)* **41**, 84-91
- Langdon, W.B. and Poli, R. (2002) *Foundations of Genetic Programming*, Springer-Verlag.
- Lawrence, P.R. and Lorsch J.W. (1967) *Organization and Environment*. Boston: Harvard Business Press.
- Lawrence, P.R. (1993) The Contingency approach of Organization Designn *In Handbook of Organizational Behavior*, edited by Robert T. Golembiewski, New York: Marcel Dekker.
- Levitt, R.E., Thomsen, J., Christiansen, T.R., Kunz, J.C., Jin Y., and Nass, C. (1999) Simulating Project Work Processes and Organizations: Toward a Micro-Contingency Theory of Organizational Design. *Management Science*, **45**, 1479-1495
- Levitt, R.E. (2005) Organizational Design as 'Virtual Adaptation': Designing Project Organizations Based on Micro-Contingency Analysis, *CRGP The Collaboratory for Research on Global Projects, Working Paper #004*, Available at: <http://crgp.stanford.edu/publications/working.html>
- Louie, M., Carley, K. M., Haghshenass, L., Kunz, J.C., and Levitt, R.E. (2003) Model Comparisons: Docking OrgAhead and SimVision. *In proceedings of 2003 North American Computational Social and Organization Science (NAACSOS) NAACSOS Conference*, Pittsburgh, PA.
- March, J.G. and Simon, H.A. (1958) *Organizations*, New York: John Wiley and Sons

- Miller, C., Glick, W.H., Wang, Y., and Hubber, G.P. (1991) Understanding Technology-Structure Relationships: Theory Development and Meta-Analytic Theory Testing, *Academy of Management Journal*, **34** 370-399
- Miller, J.H. (2001) Evolving Information Processing Organizations, In Alessandro Lomi and Erik R. Larsen (eds.), *Dynamics of Organizations: Computational Modeling and Organization Theories*, pp. 307-327. Cambridge, MA: The MIT Press.
- Miner, J. B. (1997) Participating in profound change. *Academy of Management Journal*. **40** 1420-1428
- Murray, M.M. (2004) Solving the Engineering Vice-President's Problem, *Technical Report #003, CRGP The Collaboratory for Research on Global Projects*, Available at: <http://crgp.stanford.edu/publications/technical.html>
- Nogueira, J.C. (2000) *A Formal Model for Risk Assessment in Software Projects*, doctoral dissertation, Department of Computer Science, Naval Postgraduate School.
- Pennings, J.M. (1987) Structural Contingency Theory: A Multivariate Test, *Organizational Studies*, **8** 223-240
- Perrow, C. (1967) A Framework for the Comparative Analysis of Organization, *American Sociology Review*, **32** 144-208
- Priem, R. L., Rosenstein, J. (2000) Is Organization Theory Obvious to Practitioners? A Test of One Established Theory. *Organization Science* **11** 509-524
- Ramsey, M.S. and Levitt, R.E. (2005) A Computational Framework for Experimentation with Edge Organizations. *Proceedings of the 10th International Command and Control Research and Technology Symposium*, McLean, Virginia.
- Rechenberg, I. (1965) Cybernetic Solution Path of an Experimental Problem. Royal Aircraft Establishment Translation No. 1122, Farnborough Hants: Mistry of Aviation, Royal Aircraft Establishment
- Renner, G. and Ekart, A. (2003) Genetic Algorithms in Computer Aided Design, *Computer Aided Design*, **35** 707-708

- Robins, S.P. (1990) *Organization Theory: Structure, Design and Application*. Englewood Cliffs, NJ: Prentice-Hall
- Romme, A. G. L. (2003). Organizing Education by Drawing on Organization Studies. *Organization Studies*, **24** 697-720
- Scott, W.R. (1981) *Organizations: Rational, Natural and Open Systems*. Englewood Cliffs, NJ: Prentice Hall.
- Simon, H. A. (1976) *Administrative Behavior: A Study of Decision-Making Processes in Administrative Organization*, New York: Free Press.
- Spector, L. (2004) *Automatic Quantum Computer Programming: A Genetic Programming Approach*. Boston: Kluwer Academic Publishers.
- Streeter, M.J., Keane, Martin A., and Koza, J.R. (2003) Automatic synthesis using genetic programming of improved PID tuning rules. In Ruano, A. E. (editor). *Preprints of the 2003 Intelligent Control Systems and Signal Processing Conference*, pp. 494 – 499.
- Sutton, R.I. (2000) *Weird Ideas That Work: 11½ Practices for Promoting, Managing and Sustaining Innovation*, New York, NY: The Free Press.
- Tatum C.B. (1983). *Decision Making in Structuring Construction Project Organizations*. Stanford, CA: Stanford University, Ph.D. Dissertation.
- Thomsen, J. (1998) *The Virtual Team Alliance (VTA): Modeling the Effects of Goal Incongruency in Semi-Routine, Fast-Paced Project Organizations*, Stanford, CA: Stanford University, Ph.D. Dissertation.
- Thomsen, J., Levitt, R. E., Kunz, J., Nass, C., and Fridsma, D. (1999) A Trajectory for Validating Computational Emulation Models of Organizations, *Computational and Mathematical Organization Theory*, **5** 385-401
- Thompson, James D., (1976). *Organizations in Action*, New York, NY: McGraw-Hill.

Appendix A

Sample of an alternative genetic tree that was not implemented

Position / Activity Attributes

Each position node has 4 children allocated for setting the position/activity attributes and two for reporting hierarchies, as shown below:

$F_{\text{Position}} = \{\text{FTE, Apl-Exp, Assign, Skill, P1, \dots, Pn, End}\}$

$T_{\text{FTE}} = \{-3.0, -2.5, \dots, 0.0, \dots, 2.5, 3.0\}$

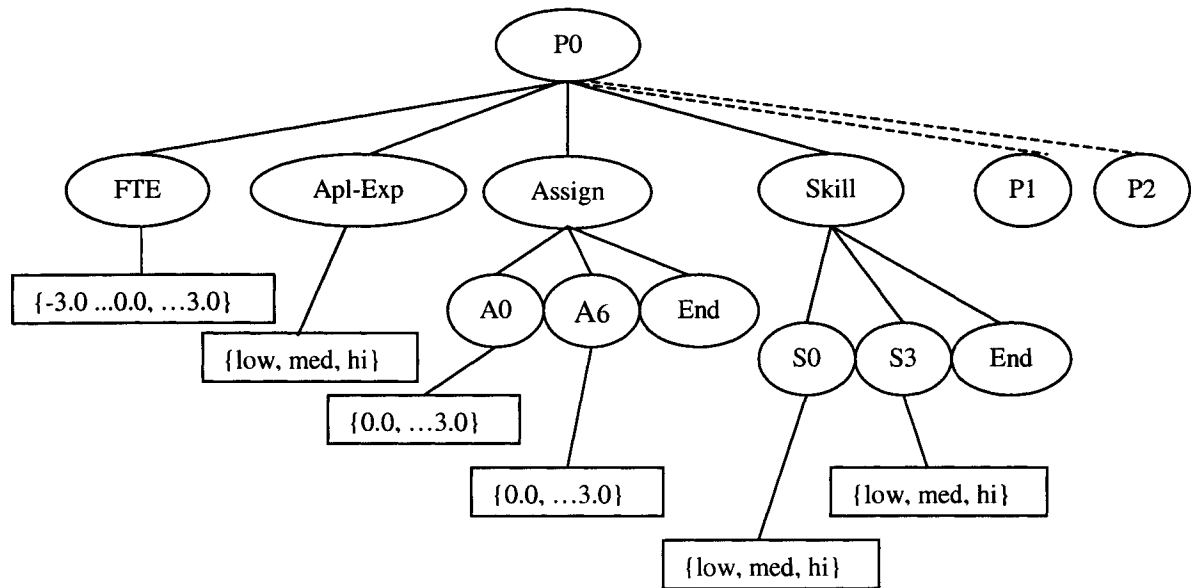
$T_{\text{Apl-Exp}} = \{\text{low, med, hi}\}$

$F_{\text{Assign}} = \{\text{A0, \dots, A6, End}\}$

$T_{\text{Aloc}} = \{0.0, 0.2, \dots, 3.0\}$

$F_{\text{Skill}} = \{\text{S0, \dots, Sn, End}\}$

$T_{\text{Sn}} = \{\text{low, med, hi}\}$

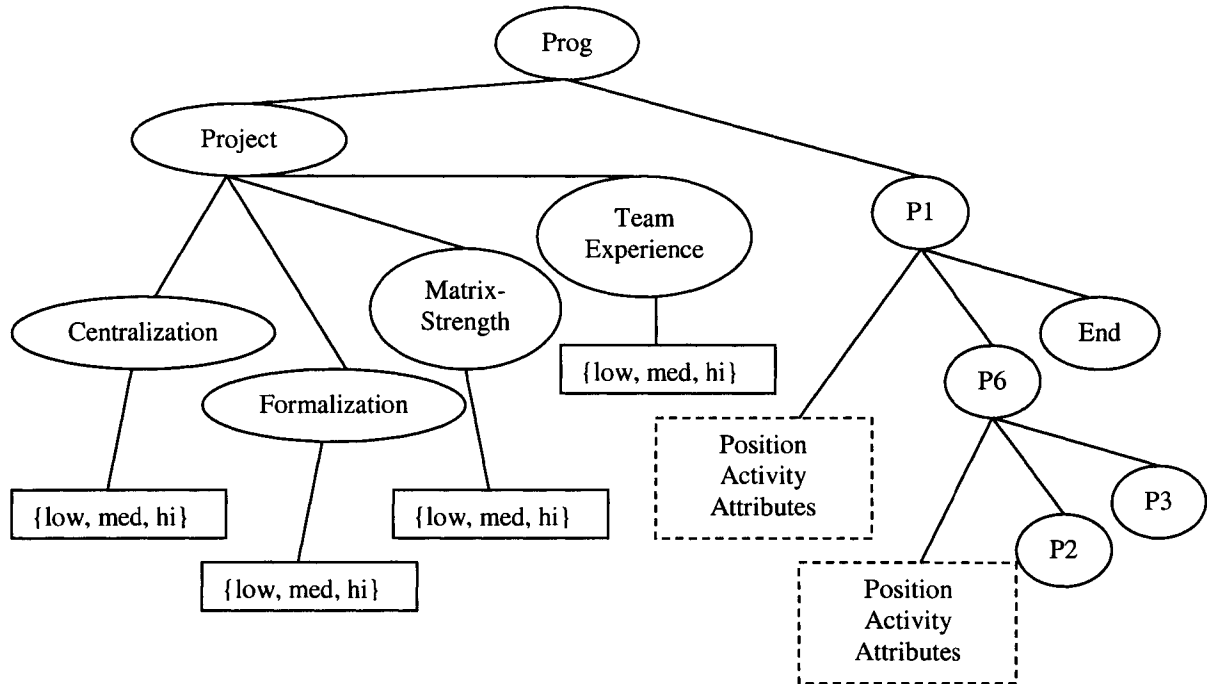


The complete genetic tree

$F_{\text{prog}} = \{\text{Project}, P1, \dots Pn\}$

$F_{\text{project}} = \{\text{Centralization}, \text{Formalization}, \text{Matrix-Strength}, \text{Team Experience}\}$

$T_{\text{project}} = \{\text{low}, \text{med}, \text{hi}\}$



➔ Appendix B

Detailed description of TGT

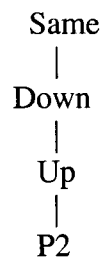
Transforming Genetic Tree (TGT) is a set of instructions represented in a genetic tree format that manipulates the organizational attributes and topology to produce a new project organization design. In TGT, there are two groups of function sets and two groups of terminal sets:

- Function Sets:
 - Attributes and topological functions that can only appear as immediate parents of terminals such as: FTE, Assign, Aloc
 - Encoding functions that create instructions for setting attributes and topological changes such as: Up, Down, Same
- Terminal Sets:
 - Actor terminals – $P_1 \dots P_n$
 - Decision making policy terminal – CFM

Each branch in TGT is dedicated as: Skill branch (for changing skill levels) or FTE branch (for changing FTEs) or CFM branch (for changing the decision making policy), Assign branch (for reassigning activities to actors), Aloc branch (for changing attention allocations), Supervision hierarchy branch (for changing the reporting hierarchy). A new instruction is generated based on the type of the branch and the functions and terminal sets appear in that branch as described below:

Skill Branch

- Functions **Up, Down** increase or decrease the actor skill level by one level correspondingly. For example, from medium to high or from medium to low.
- Function **Same** does not make any changes in actors skill level.
- Depending on where the above three functions are located in the hierarchy, they affect the 1st, 2nd, or 3rd, ... skill of an actor. For example:



Increases the skill set 1, decrease the skill set 2, and does not change the skill set 3 of actor P2 accordingly.

FTE Branch

- Function **FTE** (Full-Time Equivalent) increases or decreases the actor FTE by 0.5 per each Up or Down preceding the FTE. For example:



Adds $2 \times .05 = 1$ FTE to actor P5 current FTE.

CFM Branch

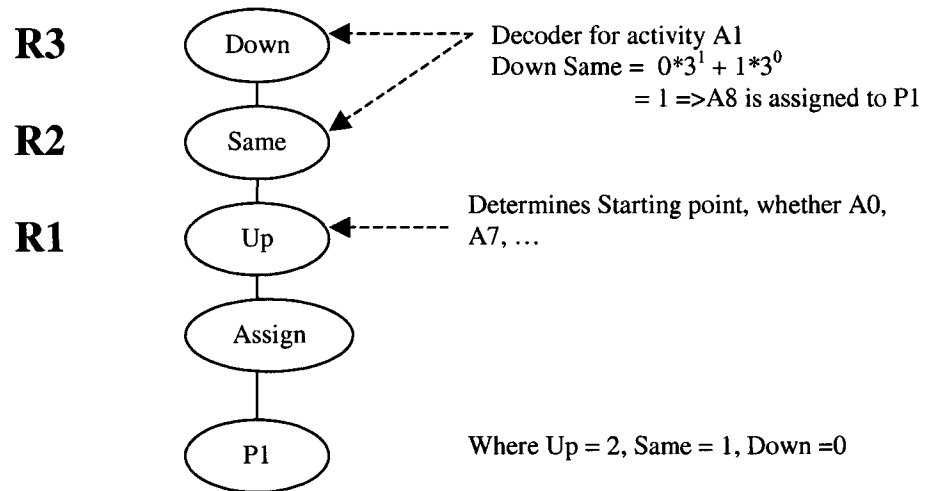
- Terminal **CFM** stands for Centralization, Formalization and Matrix Strength. Functions Up, Down, Same will increase, decrease, or does not make any change to the centralization, formalization and matrix strength accordingly. For example:



Decrease the centralization by one level, increase the formalization by one level, and does not change the matrix strength level.

Assign Branch

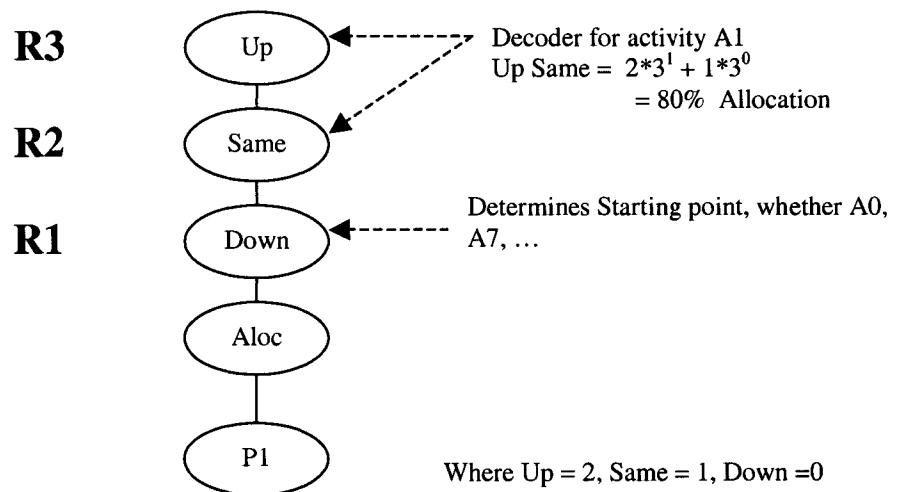
Assign branch determines who should be assigned to what activity. In an Assign branch terminal P0 through Pn represent activities when they are the children of Assign. That is, P0 stands for activity A0, P1 for activity A1 and so on. In case that there are more activities than actors, we use the immediate parent of Assign as a selector to cover all cases. That is, for example, if immediate parent of Aloc is Down, P0 thru P6 stands for A0 thru A6, and if it is Up P0 though P6 stands for A7 thru A13, and so on. Then the grandparents of Assign determine who the corresponding activity should be assigned to. For example, with the following branch:



Activity A8 is assigned to actor P1.

Aloc Branch

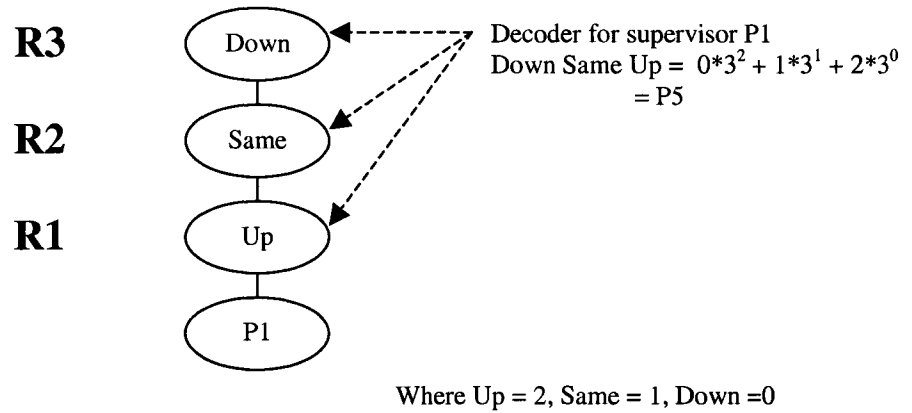
Aloc branch determines what percentage of an actor's time should be dedicated to an activity. In an Aloc branch terminal P0 through Pn represent activities when they are the children of Aloc. That is, P0 stands for activity A0, P1 for activity A1 and so on. In case that there are more activities than actors, we use the immediate parent of Aloc as a selector to cover all cases. That is, for example, if immediate parent of Aloc is Down, P0 thru P6 stands for A0 thru A6, and if it is up P0 thru P6 stands for A7 thru A13, and so on. Then the grandparents of Assign determine the percentage time allocation. For example:



the attention allocation of activity A1 will be set to 80%

Supervision Branch

A key flag is used in the main program to set a branch to be used as a supervision branch or a skill branch. Then the supervision branch determines who (a subordinate) should report to whom (a supervisor). In a supervision branch the terminals represent the supervisors and the subordinates are calculated based on the functions generated in that branch. For example:



Supervisor P1 will supervise subordinate P5.

➔ Appendix C

Sample of a modified section of ECJ parameter file with added constraints

```
.....
.....
# Make 5 atomic types
gp.type.a.size = 5
gp.type.a.0.name = pnt
gp.type.a.1.name = cfmt
gp.type.a.2.name = topr
gp.type.a.3.name = fter
gp.type.a.4.name = AssignAloc
gp.type.s.size = 1
gp.type.s.0.name = alls
gp.type.s.0.size = 5
gp.type.s.0.member.0 = pnt
gp.type.s.0.member.1 = cfmt
gp.type.s.0.member.2 = topr
gp.type.s.0.member.3 = fter
gp.type.s.0.member.4 = AssignAloc

# The return type of my tree will be topr
gp.tc.0.returns = topr
# Make some node constraints with these arguments -- I presume
# all the arguments of a given node will use the same type
gp.nc.size = 6

gp.nc.0 = ec.gp.GPNodeConstraints
gp.nc.0.name = cfmt0
gp.nc.0.returns = cfmt
gp.nc.0.size = 0

gp.nc.1 = ec.gp.GPNodeConstraints
gp.nc.1.name = pn0
gp.nc.1.returns = pnt
gp.nc.1.size = 0

gp.nc.2 = ec.gp.GPNodeConstraints
gp.nc.2.name = topr2
gp.nc.2.returns = topr
gp.nc.2.size = 2
gp.nc.2.child.0 = alls
gp.nc.2.child.1 = alls
#gp.nc.2.child.2 = alls

gp.nc.3 = ec.gp.GPNodeConstraints
gp.nc.3.name = fter2
gp.nc.3.returns = fter
gp.nc.3.size = 1
gp.nc.3.child.0 = pnt
gp.nc.3.child.1 = pnt

gp.nc.4 = ec.gp.GPNodeConstraints
```

```

gp.nc.4.name = AssignAlloc2
gp.nc.4.returns = AssignAlloc
gp.nc.4.size = 1
gp.nc.4.child.0 = pnt
gp.nc.4.child.1 = pnt

gp.nc.5 = ec.gp.GPNodeConstraints
gp.nc.5.name = pn2
gp.nc.5.returns = pnt
gp.nc.5.size = 2
gp.nc.5.child.0 = pnt
gp.nc.5.child.1 = pnt

# We have n functions in the function set.  They are:
gp.fs.0.size = 11
gp.fs.0.func.0 = updown.CFM
gp.fs.0.func.0.nc = cfmt0
gp.fs.0.func.1 = updown.Same
gp.fs.0.func.1.nc = topr2
gp.fs.0.func.2 = updown.Up
gp.fs.0.func.2.nc = topr2
gp.fs.0.func.3 = updown.Down
gp.fs.0.func.3.nc = topr2
gp.fs.0.func.4 = updown.FTE
gp.fs.0.func.4.nc = fter2
gp.fs.0.func.5 = updown.Assign
gp.fs.0.func.5.nc = AssignAlloc2
gp.fs.0.func.6 = updown.Aloc
gp.fs.0.func.6.nc = AssignAlloc2
gp.fs.0.func.7 = updown.PeoplePosition
gp.fs.0.func.7.nc = pn0
gp.fs.0.func.7.nn = P0
gp.fs.0.func.8 = updown.PeoplePosition
gp.fs.0.func.8.nc = pn0
gp.fs.0.func.8.nn = P1
gp.fs.0.func.9 = updown.PeoplePosition
gp.fs.0.func.9.nc = pn0
gp.fs.0.func.9.nn = P2
gp.fs.0.func.10 = updown.PeoplePosition
gp.fs.0.func.10.nc = pn0
gp.fs.0.func.10.nn = P3
gp.fs.0.func.11 = updown.PeoplePosition
gp.fs.0.func.11.nc = pn0
gp.fs.0.func.11.nn = P4
gp.fs.0.func.12 = updown.PeoplePosition
gp.fs.0.func.12.nc = pn0
gp.fs.0.func.12.nn = P5
gp.fs.0.func.13 = updown.PeoplePosition
gp.fs.0.func.13.nc = pn0
gp.fs.0.func.13.nn = P6
gp.fs.0.func.14 = updown.PeoplePosition
gp.fs.0.func.14.nc = pn0
gp.fs.0.func.14.nn = P7
gp.fs.0.func.14 = updown.PeoplePosition
gp.fs.0.func.14.nc = pn0
gp.fs.0.func.14.nn = P8
.....

```