Synergy

A Language and Framework for Robot Design

Submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy in Robotics

By Lalitesh K Katragadda

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

December 1997

This research was supported in part by NASA grant NAGW-1175. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of NASA, Carnegie Mellon or the U.S. Government.

Abstract

Robot design is increasingly difficult due to escalation in complexity, capability and application. A design environment can automate many design tasks and relieve the designer. Prior to robot development, designers usually compose a robot from existing or custom developed components, simulate performance, optimize configuration and parameters, and write software for the robot. Robot designers find themselves developing these facets custom to the robot using a variety of software tools from spreadsheets to C code to CAD tools. Valuable resources are expended, and very little of this expertise and development is reusable. This research considers that a language to comprehensively represent robots is lacking and the aforementioned design tasks can be automated once such a language exists. This research proposes and demonstrates the following thesis:

"A language to represent robots with a framework to generate simulation, optimize design and generate control software increases the effectiveness of design"

Synergy was prototyped and demonstrated in the context of lunar rover design, a challenging real-world problem with multiple requirements and a broad design space. Synergy was used to automatically optimize robot parameters and select parts to generate effective designs, while meeting constraints of the embedded components and sub-systems. The designs so generated are better in performance and consistency relative to designs by teams of designers using the same knowledge. Using a single representation, multiple designs are generated for four distinct lunar exploration objectives. Synergy uses the same representation to auto-generate landing simulations and simultaneously generate control software for the landing.

Synergy is composed of four software agents. A database and spreadsheet agent compiles the design and component information, generating component interconnections and ensuring consistency of types, physical units and constraints. A simulation agent generates comprehensive dynamic simulations fusing several uni-dimensional agents. An optimization agent executes rules embedded in the design, finds roots for the implicitly interconnected design and searches the parameter and component space using a genetic algorithm. A control software generator learns a generalized "neural-net" controller using the simulation for feedback and a genetic algorithm to guide the search.

Acknowledgment

Foremost thanks to the reader of this thesis; please embark knowing that the message is simple and the work - a beginning. It is an undertaking that will need the hands of many to fulfill its promise. I hope that you are one of its voices. Gratitude goes to the people who known and unknown to me sourced many ideas that this thesis brings together; ideas that have since become lore. Equally instrumental is NASA and the Space Foundation for funding Synergy.

Red Whittaker has been an inspiration, sometimes an advisor, and always a great heart in the grand adventure to put a rover on the Moon. I sense that this adventure is not over for either of us and dream that the best is to come. My deep gratitude to him for allowing me my crazy dreams for the Moon and Synergy. Mike Samuels was central with his Herculean coding effort and its current effectiveness owes a great deal to him. Many thanks to Dot for the “prime-ministership” of FRC in my stay.

I thank Eric Krotkov, Martial Hebert, Reid Simmons and Alonzo Kelly for their partnership, friendship and humor during the early days of Ratler, and the experience of making it a reality.

Thanks to Mason, Bares, Tony for agreeing to steer this thesis, for valuable guidance, and having the patience to let me find my way. Its been my added pleasure to have had long ranging discussions with Matt and worked with John on Dante II.

Suryanarayan (IIT-B), Khosla, and Cutkosky (Stanford) inspired with advice that brought me to the Robotics Institute. Thanks to them and others who have similarly been friends and advisors along the way include Sohini, Betty, Gump, Marce, Sandy, Shafer, Chen, Nechyba, Moravec, Barry, Sib, Eric, Deepak, Kevin, Dimi, Eric, Gerry, Thorpe, Voyles, Hancock, Sanjiv,... The people of the Robotics Institute and Lunar Rover Initiative are a wonderful space to work in.

John Murphy continues to match my insanity for a robotics enterprise, and time will tell of the true extent of it. His presence has added lots-o-spice to graduate life. A special thanks to him and friends everywhere for giving context to life.

I am here due to years of leadership and devotion of my mom and dad, encouragement and support from my brothers, aunt Jaya and the joint family in America’s dairyland. To them, AK Rao, Mujumdar, JC, Pierson; Master Pak, Master Kong, and untold keystones; thank you - a new journey begins.

Words do injustice to express my gratitude to Sharda for standing beside me during this thesis, and the many ways she inspires and transforms my life through her being. It is cool that i have the rest of my life to thank her...

Contents

Chapter 1

Introduction

As robot complexity grows, the tasks of comprehending systemic engineering relationships, finding suitable designs and analyzing them is becoming increasingly unwieldy. The rapid expansion of robotic capability and application calls for software to assist designers. Many design tasks are well suited for automation through a design environment. These tasks include system level analysis, simulation, optimization and generation of control software. A language to represent robot design is central to such a design environment. This thesis creates such a language and implements a framework of supporting agents to demonstrate several ways in which robot design can be facilitated.

The assumed meaning of common words used in this thesis can cause unintended interpretations. The reader is thus requested to glance at the Glossary.

Thesis Statement

Software environments for designing and developing robots are important, while current ones lack in comprehension or generality. Central to a successful design environment is a language that represents robot design. Implementation of such a language allows several ways of actively supporting design. The purpose of this research is to understand the nature and formulation of a robot design environment that will relieve designers of cumbersome tasks that demand vast knowledge, perfect memory and intensive effort. The outcome is an implemented language and framework, with demonstrations of potential and effectiveness. This research motivates, distinguishes, demonstrates and inquires into the implications of the following thesis:

“Language to represent robots with a framework to create simulation, optimize design, and generate control software increases the effectiveness of robot design"

Motivation

Robot complexity and capability is growing rapidly, fuelled by advances like the geometric growth of electronic capability. Robots do not share a common architecture due to their broad application and widely differing needs. Since robot applications and functionalities are so diverse, designs defy the patterns and recipes that facilitate development of more mature artifacts. Robot design is a difficult task due to the breadth of engineering dimensions (e.g. mechanical, electronic, thermal, ...) involved and the tight coupling amongst them in effecting robot function. This extreme level of integration coupled with diverse architecture, stretches even experienced designers and design teams during the development process. Consequently the development of robots is long drawn, inefficient and frequently results in implementation of flawed designs, requiring several prototype iterations.

Prototyping robots is usually expensive and multiple iterations are undesirable, hence it is pragmatic to carefully design the robot apriori. A design environment is a unified toolkit that could greatly facilitate robot design. It could assist by tracking constraints, performing analysis and simulations of the emerging robot design, searching the design space to find optimal designs, assisting in software development, prototype integration, and communicating the design and expertise across the design stages and across design cycles.

Language forms the basis of representing, understanding and manipulating a design. Existing languages do not support robots generally and comprehensively. Language also forms the means of transferring expertise and communication among various designers and design agents alike. A single, central representation of robot design could facilitate the interaction of agents with each other and the designers. A language to represent the robot comprehensively and provide the necessary information to automate the identified tasks is then key to such an environment.

A quantitative prediction of robot performance is needed to understand and evaluate diverse design options. Since a robot's performance is a result of its behavior of perceiving its environment and reacting to it; dynamic simulations are needed to evaluate designs and predict robot performance. Such a simulation would need to be comprehensive, spanning several dimensions due to the tight coupling of these dimensions in the functioning of a robot.

Robots are composed of tens of components with multiple choices for each, leading to a design space of millions of configurations and a vast parameter hyperspace to search in. An optimization agent can assist the designer in systematically searching for effective designs.

A robot uses control software to generate desired behavior from a perceived world and state, hence control software is integral to robot performance and part of a realistic robot simulation. However, writing such control software can be tedious in resources and time. Since the search for effective designs can span widely differing designs, it is important to auto-generate both simulations and control software, to enable evaluations of multiple designs without consuming vast resources.

These observations motivated this research and shaped it ongoingly. The identified need is for a software environment to assist in robot design, with the intention of lowering the effort and expertise required of a designer in the broad disciplines that robotics comprises. “Synergy” is a unified design environment, developed in this research to assist in robot design. Synergy is so named due to the strong interdependence of its agents on each other to be useful to design as a whole. Components of Synergy emphasized in this research are a language to describe robot design, simulation and optimization to evolve robot design, and auto-generation of control software for evaluating robot design.

Background

Existing general robot environments lack the dimensionality to represent robot designs comprehensively. There are effective design environments for mature commercial products like VLSI chips. However, the complexity and diversity of robots requires Synergy’s capabilities to go above and beyond any of the existing tools. There is no general environment in existence with claims to representing robot designs and providing vital support like simulation, optimization and software generation. Existing simulation and analysis tools are either uni-dimensional or static in nature. Current optimization software is mature though not interfaced to assist robot design effectively. Research in generating control software specific to the general robot is nascent, and an effective approach is lacking. Chapter 2 provides a background and overview of existing design environments and supporting agents.

Current languages either capture a specific design space (e.g. Cars) or capture the entire robot at a conceptual level or reflect a subset of the physical dimensions (e.g. ProE). Existing design methods and tools either analyze a complete system at a conceptual level or a single component in detail. Alternately, specialized tools either model a specific dimension of machine (e.g. electronic circuit) or synthesize artifacts with well defined structure and history (e.g. manipulators).

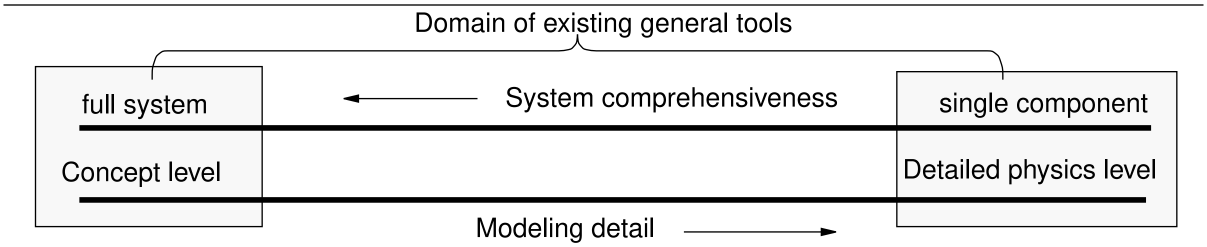

Figure 1-1: Analysis domain of existing tools

Some existing robot tools include simulations for Ambler[Hoffman94], UGV[Kelly95], Ratler[Keller92][Katragadda94], Ranger[Kuester95], and excavation[Hoffman94]. These are all dynamic mechanical simulators, with visualization and in some cases sense-control-actuation loop simulation. General electromechanical simulators such as TeleGrip and RobCad fall into the same category but fall short of capturing other dimensions like electronics and software comprehensively. Some robot simulators for manipulators are more refined to cover geometric analysis. Existing multi-dimensional design tools are custom-made for specific applications and ignore mechatronic phenomena not important for that application. These include design tools or environments for commercial aircraft, VLSI chips, PCB layout, climate control systems, and manipulators. Even these highly specialized tools often segregate the analysis of different dimensions, for example chip design tools perform thermal analysis assuming worst case static heat loads created by the electronic transistors, hence ignoring dynamic interactions with the electronic dimension. Such an approach would fail for advanced robots which push the performance envelope by depending on software for safety rather than the electromechanism itself.

Support agents for existing design environments are specialized to specific design spaces they address and are hence often restricted. For example, simulations either perform static analysis or disallow certain dimensions of behavior like change of mass. Agents that support software development impose a certain architecture on the software or do not help create critical control software, turning this aspect into a long project. Table 1-1: summarizes the properties of existing environments and Synergy against those identified for a robot design environment.

Tool Name	domain size	Representativ Comprehen-sion	Multi dimensio-nality	Dynamic modeling	Optimizing ability	Software generation
Viper	Lo	Med	Lo	Med	Med	No
ProE	Hi	Lo	Lo	Lo	Lo	No
Algor	Hi	Med	Med	Lo	Med	No
TeleGrip	Med	Lo	Lo	Med	Lo	Lo
VLSI	Lo	Med	Med	Lo	Hi	Hi
Manipul.	Lo	Hi	Hi	Hi	Med	Hi
Synergy	Hi	Hi	Hi	Med	Med	Med

Table 1-1: Comparing Synergy with representative design environments robo-centricly

So far a general robot design environment may not have existed partly due to the lack of sufficient maturity in analysis techniques, search methods, information technology and computing resources. Factors that enable the design environment are understanding of design important mechatronic issues, engineering modeling techniques, field experience of building capable mechatronic systems and high performance computing. These factors are emerging to maturity and set the stage for Synergy.

Scope

This research formulates and prototypes a primal example of synergy, and demonstrate its use in robot design. New methods are formulated and implemented as needed, while using existing techniques and software wherever possible. This includes research to develop the language so that it is comprehensive, useful and supportive of the agents for simulation, optimization and control software generation. The simulation module is developed to an extent that shows fusion of multiple dimensions and demonstrates flexibility to add more dimensions as needed. The optimization module is created to be general and effective in searching through highly non-linear (even discontinuous) problems in spaces with both real number and discrete value variables. The software generation module generates control software to enable the robot to exhibit designer defined behaviors. Other areas of “higher-level” robot software generation are not addressed, partly because there is significant research to generate software in these other areas.

The demonstrations are inside the domain of lunar exploration robots. The primary objective of the research is to demonstrate through implementation, software for general robot design while implementing components and modules with sufficient comprehension and fidelity to support design processes, with contribution and relevance to the identified application context (Lunar Rover Initiative[Katragadda94]). The application to the Lunar Rover Initiative is used to reveal uses and limitations, while ensuring relevance to many facets of a realistic robot design process. Throughout the research and development process, the ideas and software created will be tested, applied to the Lunar Rover Initiative. The scope of the design data and knowledge-base is restricted to enable this application.

The research leverages on ideas of design theory and methodology, modeling methodology, optimization, and software engineering. Specific tools for dynamic simulation, graphics and electronic simulation have been used to instantiate the framework. Four major tasks express the research scope:

- Creation and implementation of a robot design language keeping in mind the various needs of the designer and supporting software agents. The language is not complete in representing all possible robot information but has implemented features that allow new constructs to be added ongoingly and communicated to the various software agents that assist in the design process.
- Development of the modeling environment to automatically generate robot models from the design and generate a dynamic simulation. The simulation module is restricted in scope severely due to the resources needed to implement all the possible dimensions. On the other hand the simulation is created to take advantage of advances in simulation technology in mainstream dynamic simulators. The simulation models at least three dimensions - mechanical, electronic, and resource tracking, demonstrating an approach of adding more. Dimensions that are not be simulated include thermal and programmatic (cost, reliability, schedule, personnel). The general resource tracking agent is used to create budgets and act as a baseline agent for the missing dimensions.
- Development of a general optimization module to search the design space and identify optimal design(s) in the boundary of the given constraints guided by a designer specified objective function. This module interfaces with the design environment so that it can be used for both configuration selection and parameter sizing. Geometric configuration and innovation are not addressed at this point although the extensions for them are formulated.
- Develop software generation algorithms to meet a designer specified goal or behavior. A new approach is formulated and demonstrated in a rudimentary implementation.

- Discover and investigate the uses and limitations of this implementation and the Synergy's approach by creating lunar rover designs with this software.

Certain implicit assumptions restrict the research scope and implementation. Robots are approximated as interconnected rigid bodies. The implementation is focussed on electromechanical robots. Other dimensions like biological are outside the scope, while extensions like hydraulic power, though representable, are not implemented. Finally, it is assumed that the process of cognitive understanding of robot design and evaluation is cheaper than the cost of prototyping and deploying the robot, otherwise the usefulness of this thesis is marginal.

Approach

The approach of this thesis research is to prototype a robot design language and supporting framework, apply it to the identified lunar rover design problem, discover vital attributes of a robot design environment through generalization, incrementally implement them and demonstrate their use.

The language to represent robots is formulated as a general grammar that can represent any robot. Language to represent robot design will need several properties to assist design effectively. Robot behavior is a result of strong interaction of its components and the environment across multiple dimensions like the mechanical, electronic and control software. Hence the design language needs to be comprehensive both in representing the entire robot and in ensuring that component properties in these dimensions are captured. Flexibility and rapid reconfigurability are also needed as existing robots have no well defined structure or function-form mapping. Such an environment needs to facilitate the capture and systematization of design expertise to enable faster design cycles of increasingly complex designs which build upon established building blocks. The function of quantifying system level relationships also includes tracking component/subsystem level constraints and performance margins. To minimize the burden of communication among the design team, a common robot representation of the design would provide the necessary information to the modules that perform design optimization, rover simulation, and generation and integration of control software.

The design representation will also need to enable capture of multiple constraints of component engineering, environmental and designer imposed rules. Engineering constraints are imposed by features or limitations of components like number of motor revolutions or temperature limits. Environmental constraints are set up by the robot's mission or task, for example total distance travelled or terrain roughness. Designer imposed rules are a result of experience or condensed analysis; examples include the maximum contact pressure between robot and soil or required motor torque as a function of robot mass. Additional constraints arise from considerations of ensuring minimal system bottlenecks, safety, impedance matching, observability and controllability.

Automated simulation can be generated by merging existing uni-dimensional dynamic simulators or algorithms at each step using the robot representation as a "blackboard". This research develops an approach to unifying uni-dimensional simulations to generate coherent, comprehensive simulations. The simulation environment has been created to use existing simulation modules as agents that interact with the simulation at every time step to create a fully integrated simulation. This architecture allows direct use of highly developed state-of-the-art one-dimensional simulators like Coriolis for mechanical simulation and SPICE for electronic simulation.

There are existing approaches to optimization that match the needs of robot design that can be adapted directly to finding satisficing designs and optimizing them. One such optimization algorithm is chosen and coded incorporating robot design attributes like discrete valued variables and highly non-linear function mapping.

To generate controllers, a learning algorithm is formulated by representing a general function as a fully connected network of input/hidden/output nodes (similar to neural nets) and loosely casting the problem of generating a desired behavior as an optimal control problem. The controller is found by searching the space of network weights.

Chapter 2

Background

Rudiments of the techniques for creating robot designs, simulation, optimization and software generation have evolved into accepted, though disjoint use in robot development. The stage is set for evolving these tools and fusing them into a comprehensive software environment for robot design. However there are certain aspects lacking that would allow creation of such an environment including a language to comprehensively represent robot design, a framework that unifies design software and automated generation of control software for design evaluation.

Environments for Robot Design

A robot design environment is software that facilitates the designer to compose, evaluate, optimize, and integrate hardware and software into functional robots.

Application Specific Design Software

Current developments in robot design software exhibit some properties of the ideal. One example is Viper[Hobbs95], an automated wheeled rover configuration environment for planetary explorers. This software sizes wheels and motors given payload and soil terrain conditions. It automatically generates simulations, driving off the lander, generating power profiles for driving and steering. Configuration decisions like switching from four to six wheels are triggered by heuristics. However, Viper can only be used for wheeled planetary explorers and does not capture models of various components in power and communication technologies.

Extensive application of manipulators has motivated extensive research into design software. One such environment assists designers of reconfigurable modular manipulator from configuration to functioning prototypes. This environment generates design from a task description[Morrow95], composes task control software from an iconic description (Onika[Gertz93] and auto-configures control code[Stewart93] to shield the designer from rewriting code to adapt to changes in manipulator configuration. This environment enables a working prototype in less than a day after task specification. However, this environment can be fully used only in a narrow domain of robots, namely rigid linked, modular manipulators with predefined interfaces.

General Robot Design Software

The design environments for most robots are assembled from CAD packages, simulators, spreadsheets [Excel95][Wolfram97], schedulers [MSProject95], and compiled languages[Prata92]. Such implementations are created for sizing component parameters, tracking progress and making decisions. Extensive research is emerging to formalize and parameterize robot functionalities like navigation[Kelly95], locomotion [Apostol97], system engineering[Coulter94] and machine elements[Kannapan89]. The existing design software is unable to use this research without re-coding it; in addition design environments for robots are usually custom developed, slowing down the design team, inhibiting transfer and reuse of expertise.

Comprehensive Design Environments

Successful design environments exhibit the facility to represent the entire design in software, create simulations, assist in optimization and support integration of hardware/software. However, the specific nature of such existing environments, significant use of custom analysis and simulation, high costs ($billions), millions of man years of history and precludes adaptation of the underlying software or techniques. Example environments include those used for cars and planes[Farritor96].

A new generation of resource effective design environments is beginning to emerge. A spacecraft project design center[JPL97] allows design teams to specify problems, investigate solutions, configure, simulate and generate development programs of entire spacecraft missions in a few days. Such effectiveness is made possible by drawing on NASA's extensive aerospace database, a well defined design process and integrated software that makes use of it. However, this environment is created for use in a regulation driven development environment with extensive development resources. The following example illustrates how an effective design environment impacts a competitive and short development cycle, similar to the challenge roboticists face.

VLSI Chip Design

Eighteen months from its concept, the UltraSparc, a 166 MHZ, 3.7 million transistor was released on its second prototype iteration, where its first iteration had booted up UNIX on the first attempt[Boddu95]. These remarkable results are due to advanced design software. Using this software, VLSI chip designers verify their concepts using block[SES94] and architectural simulators[SUN94]. The chip functions are then described in a high level language - Verilog[CadLang94] or VHDL. Simulators[CadSim94] validate and optimize the functionality, while verifying constraints[ViewLogic94] for timing delay, heat, etc. A synthesizer[Synopsis94] generates a net-list which is then transformed into chip design by a layout tool and manufacturer-specific[TI94] rule databases. The resulting prototype is tested for functionality by automated test software[Sunrise94], benchmarked for performance, functionality and iterated, usually once.

VLSI chips differ from robots significantly by being mostly uni-dimensional in electronics and being composed of a small set of similar sized components. Hence, approaches like using arrays to represent the design, rule databases to generate simulations and check design validity would be less applicable to robots.

Shortfall in Robot Design Environments

Existing initiatives in robot design software are application specific and incomplete. There is no useful comprehensive general design environment for robots. Existing robot design software cannot be unified into a design environment due to its narrow focus or robot specificity. The open agenda is to create software that unifies functions vital to designers and is general enough to enable robot design.

Languages to Represent Robot Design

Language is a vocabulary and grammar of constructs so that robot design knowledge and problems are completely represented, enabling access and inference of the encoded knowledge body by the design team and software agents alike. A design represented in such a language has uses as a unified resource for expertise transfer, evolution of design and would need to represent the entire robot modularly and flexibly. Current languages used for robots enable representation of only uni-dimensional properties with rudimentary interconnections. Design theory suggests several facets of representing complex machines like robots, and methodology for using such representations to facilitate problem solving.

Current Robot Design Languages

Roboticists have traditionally made use of existing machine representation languages for designing robots. Solid model CAD (e.g. ProE[PTC97], IDEAS[SDRC97]) software is commonly used to design the mechanism. These designs are represented as an assembly of parametric models with embedded manufacturing and material information. PCB CAD layout[PADS97] tools are used to design the electronics and interconnections. Spreadsheets are used to capture parametric relationships and track the design. Simulation software extends the CAD representations by querying the designer for additional information it needs; e.g. Finite Element Analysis modules[Algor97] asks for structural boundary conditions, Telegrip[Deneb97] has facilities to incorporate geometric environments, information on sensing and control. However this information is not readily visible, accessible and usable by designers and other software. Although, many robot CAD designs exist, they do not embed robot/component functionality, rendering them ineffective for reuse in robot development.

Theory of Representing and Communicating Design

Knowledge can be encoded through natural language, mathematics, rules, semantic networks, data (databases, design examples) and algorithms. Research in design theory identifies and rationalizes several properties of an effective design language and implementation for complex design problems. Shared memory can act as group resource and communication medium to support design as a social negotiation and reconciliation process between several domains of knowledge and expertise. Since design involves several agents (human and machine), well defined interfaces are will mitigate the understanding required by one agent of another's domain. Databases are used to capture and retrieve design information effectively in areas like structural engineering and mechanical CAD. Proposed adoption of object oriented relational databases will enable properties of flexible representation and fast retrieval of multi-modal (text, image, models, algorithmic) information. These features though compelling for robot design, have not been implemented in a single language and framework.

Problem Solving Frameworks

Robot design is a problem involving multiple sources and domains of knowledge. Problem solving frameworks propose a structure for integrating the represented problem, knowledge, control mechanisms, and the designers into a coherent environment. Such frameworks include production systems, frames, structured design systems (like IDEAS, CATIA) and agent-based frameworks. Among these, agent-based frameworks, like the blackboard metaphor, provide facilities for multiple knowledge sources to interact cleanly. Newell proposed a set of experts (agents) solving a mutual problem by contributing onto a blackboard and using it as the sole feedback device for understanding the thoughts and ideas of other experts. Extensions(e.g. Craig) of this metaphor allow non-traditional behavior of agents like self-scheduling and information deletion. Examples of its use include SHARE[Petrie97] and [Baya97] which allow distributed teams of designers and software agents to co-operatively design complex systems. These frameworks have not been implemented for robot design problems and most implementations are structured for human agents to be the primary sources of knowledge and design.

Absence of Robot Design Languages

None of the existing work has provided a language that can comprehensively represent robot design with features that enhance the designer's effectiveness. This shortfall causes multiple, possibly inconsistent copies of the designs to be created for different uses. These representations cannot communicate and are incompatible, hence task the designer's memory and faculties to reconcile and merge the different parts of the robot design into a whole.

The need is for a language to represent robots comprehensively and modularly so that designs can be modified easily and expertise of the entire design or parts of it can be carried forward in time. A comprehensive language would enable representation, acquisition and retrieval of information about the physical hardware, constraints, status of software and complete dynamic state of the robot. Such a language would also interface with various software agents that assist in robot design.

Robot Simulation Software

A simulation is a computational emulation and/or evaluation of robot behavior. For the purposes of robot design, simulations need to be visual to convey perspective to the designers, dynamic to predict behavior, accurate and quantitative to enable rational comparison, selection and refinement. Robot designers usually create two simulations, a preliminary kinematic simulation to verify geometric functionality, and a subsequent custom simulation with higher fidelity to predict selected aspects of robot performance to enable refinement, integration of hardware and software.

Preliminary Robot Simulations

General mechanical modeling and simulation environments like TeleGrip and RobCad[RobCad] are often used to generate the first visual look at robot operation. They sometimes incorporate mechanical modeling and closing sense-plan-actuation loops. Capabilities to simulate physical collision, contact, and friction transitions are not present. Simulators like Vevi[Piguet95] can be used to quickly simulate robot kinematics and also enhance remote teleoperation of by providing awareness of robot and surroundings, for control of diverse tasks like underwater exploration of Antarctica. These simulators however do not completely model all physical interactions of the robot. There is void in dynamic robot simulators in many dimensions like perception and thermal. dynamic

Custom Robot Simulations

Most robot designers create custom simulations to understand behavior that is not modeled by existing simulators. These simulations are hand coded using existing libraries[Naher93] and graphics[SENSE8-93] utilities. Usually these simulations are not comprehensive, leading to consistent appearance of unanticipated behaviors in the robot prototype.

<u>Ambler</u>:[Thomas90] This simulator modeled the perceive, think, act cycle of walking behavior for the Ambler, Mars walker. The perceive module generated (noisy) laser range data, based on which the control module made walking decisions, and the kinematic module simulated walking behavior. A graphic front end enabled a look into the entire process.

<u>Ranger</u>:[Kuester95] The Space Systems Lab, Maryland is building a manipulator for an space station servicing experiment. Besides using underwater analogs, a dynamic simulator generates behavior in free space. A user controlled control module closes the actuation loop, generating visuals, and metrics of expected performance. This simulator is used for designing control strategies for desired tasks, training operators, and marketing.

Planetary robots like the unmanned ground vehicle[Hebert96], Ratler[Katragadda95] and Nomad[Murphy97] model four wheeled rovers as a single rigid body to generate metrics of expected behavior, for use in local navigation loops, evaluating suitability of the sensor configuration and optimizing parameters for the same. Dante-II, a volcano walker relied on a kinematic stick simulation[Katragadda94] of walking behavior to envision walking strategies and select between two configurations.

Autonomous excavation research used a kinematic simulator[Hoffman94] incorporating true 3D geometric models of terrain integrated with a laser scanner model, visualization and testing control software.

These simulators have proved useful and even vital in the robot development process. However significant engineering and programming expertise was required to code these simulators, which are custom and do not generalize to future robots. In addition, the small resource available to robot developments forces these simulators to be limited and incomplete in scope, leading to unmodeled behavior which is often visible as a flaw in prototypes.

Modeling and Simulation Technology

Current modeling techniques analyze a system as an interconnected collection of parts, where the behavior of each part is usually derived from physical principles or component data. These numerical analysis methods are fast replacing past approaches of closed form analytic techniques. One extreme is the finite elements approach which breaks down a continuum into a large number of small elements with well understood behavior. The other extreme is using simplified heuristics which estimate robot behavior from configuration (e.g. predict locomotion power consumption based on wheel geometry and soil parameters). A set of intermediate techniques are evolving which use models of larger aggregations and components with well known properties; examples include rigid bodies, circuit elements and actuators. These techniques are accurate and the most promising approach to creating dynamic simulations of complex devices, from configuration to detailed design. They have yet to be evolved into a comprehensive simulator for robotic devices.

Component based Dynamic Simulators

Dynamic simulators emulate and chart robot behavior as a function of time, taking into account varying environment and past state of the robot. Dynamic simulators are evolving from automated generation of equations to purely numerical forms. Popular mechanism simulators like Adams[MDI97] generate dynamics equations[Kane85], where these equations depend on the changing constraints (e.g. a different set of equations has to be generated for the same robot with three wheels on the ground vs. four wheels on the ground). Similar approaches also exist for other dimensions, but they all have limited use due to restrictions.

Purely numerical simulators treat the components, constraints and machine state as a series of linear programming problems. These include Coriolis[Baraff94] - a simulator for mechanisms of interconnected rigid bodies and complete mechanical behavior a set of rigid bodies, and SPICE[Nagel75] - a simulator for circuits of interconnected components. Similar approaches have been developed for other dimensions like thermal analysis[Thomas80]. Since all relationships are treated numerically, these approaches allow reconfiguration in real-time and are the most flexible for use in robot simulation. However, the sum of these disconnected simulators does not create a comprehensive robot simulation due to strong, transient physical interactions among these dimensions.

Finite Element Analysis

The finite element approach enables detailed, accurate analysis of component and assembly behavior. Domains of analysis include structural, fluid, thermal, acoustic, magnetic, and electrostatic fields. FEA packages are interfaced with CAD models to analyze (usually) a static set of load cases and verify the design assumptions and constraints. Current FEA software includes Algor[Algor97] which unifies structural, aerodynamic, thermal and electromagnetic dimensions, allowing analysis of highly integrated electromechanisms like motors or micromachined devices. Other software like NASTRAN[MSC97] is used for very complex objects like entire airframes and ANSYS[ANSYS97] is a medium scale package with application from acoustic materials to flow of water in plumbing fixtures. Due to finite computing, memory and numerical ill-conditioning, FEA analysis is mostly restricted to components rather than entire mechanisms and to static boundary conditions rather than dynamic simulations. FEA models also need detailed specification which is available very late in the design process. Hence, FEA analysis is not pragmatic for generating dynamic simulation of robots.

Manufacturability Analysis

Manufacturability analysis evaluates the ease of prototyping and manufacturing a device. This enables significant cost saving in a development cycle. Some robot development cycles[Shamah96] have used CAD tools to generate CNC machining commands to produce parts, so verifying the difficulty of their manufacturability. Design-for-manufacturing environments for machine elements and sheet metal[Bourne92] parts plan and verify the manufacturing process during component design itself. Successful approaches include concurrent generation of design and manufacturing process, and generation of process from design using planning trees. These implementations have laid a foundation for future CAD systems which will be able to track process(es) concurrently[Cutkosky93] with design by querying databases and "intelligent" manufacturing machines. However, none of these tools can encode robots comprehensively or the disparate prototyping processes used in robot development, ranging from connector assembly to vacuum forming.

Robot simulations are not unified

In summary, existing simulators for robots demonstrate the value of dynamic simulation, but fall short of the needs of generality and comprehensiveness. Dynamic simulations for robots are uni-dimensional (mechanism and electronic) or incomplete and custom built for specific robot designs. There is a lack of simulators that can predict the performance of a robot in without significant custom work for a particular design. Static analysis tools enable detailed understanding of the robot components at selected performance points in dimensions like material stress, EMF fields and thermal loads. Limited simulation[SILMA89] of robots with well defined structure (like manipulators) has been automated. There is no software, static or dynamic to generate a comprehensive robot simulation from its description.

Optimization Software for Robots

Optimization is the search of a finite dimensional robot design space to find the best candidate for desired performance. Variables include design parameters (e.g. wheel dia, computing power, sensor location), component selection (e.g. harmonic vs. planetary gear) and configuration choice (e.g. wheels vs. legs). A few optimization tools for configuring specific classes of robots (e.g. manipulators) have been developed and used. Unfortunately, most robot development cycles stop at finding satisficing design, using simple approaches like generate and test, or gradient descent. Robots so developed fall short of desired behavior due to erroneous performance predictions that optimization techniques depend on.

Robot Optimization

Early robot optimization efforts (e.g. auto-configuration of manipulators[Kim92]) used prevalent quasi-newton methods. However, due to reasons described below, genetic algorithms have emerged as the predominant approach. Some applications are geometric configuration of manipulators[Murthy92] that can span a defined reach space, geometric design of quasi-static walkers[Roston94] to kinematically traverse a given 1-D terrain and geometric design of mobile manipulators[Leger97].

Some applications used evolution to sift through vast design spaces and find optimal, innovative configurations to perform tasks of locomotion. These include modular evolution of geometric configurations and gaits of mobile robots to traverse a specified terrain[Farritor96], and evolution of complex neural net controlled mechanisms[Sims94]. This software has not been used to design real world robots due to their inability to represent robots completely and restriction of the component set that can be used in the design software.

Robot optimization has so far focussed only on mechanism, mobility and planning; and has not ventured into dimensions like sensor configurations, electronics and thermal conditioning. This is due to shortfall in effective simulations in these areas. Also, the lack of a single implementation that assists in all robot design has inhibited enhancement of these algorithms for robot design. The following sections investigate optimization technology to identify viable algorithms for robot optimization.

Optimization Technology

Optimization software is well developed for a variety of search spaces like linear functions with linear constraints, differentiable non-linear functions[Luenberger73] and highly-nonlinear(stiff)/ discrete valued functions[Greenberg71]. Usually the domain space is constrained both by equality relations and by inequality boundaries. The constraints[Fletcher87] are treated by lumping them with the objective function (penalty function approach) or by mapping the constrained search space into a constraint free space of lower dimensionality. State of the art includes the Simplex method (linear), Karmarker's algorithm (linear), sequential quadratic programming based on quasi newton methods (non-linear), dynamic programming (discrete valued) and genetic algorithms.

For variables and functions that are continuous, differentiable with a reasonable initial point, a quasi-newton[Broyden67] approach is effective. This approach models the function as a parabolic surface of order two, continually updating this model as the search progresses from one solution of the parabola to the next. Popular algorithms are the BFGS[BFGS], conjugate gradient[Fletcher64] methods for unconstrained spaces and SQP[Fletcher85] for constrained spaces. Many robot design problems involve highly stiff/discontinuous functions and discrete valued variables, which these techniques are unsuitable for.

General Optimization Algorithms

General (sometimes less efficient) techniques are used for highly non-linear, discontinuous, and integer functionals. The general techniques are also used for finding satisfactory starting points for (non)-linear problems and for highly stiff problems Algorithms for such functions are dynamic programming (e.g. Monte Carlo[Neiderr95], A-Star[Korf87]) and probabilistic search methods (e.g. Genetic Algorithms [Goldberg89], Simulated Annealing). Dynamic programming involves searching a discrete space extensively using heuristics to constrain the search. This is rarely useful in robot design due to geometric growth of required computing with number of variables, and most robot design problems have large number of variables. Probabilistic search requires mapping the space using one or more initial points and generating new points based on the fitness pattern that emerges from this mapping. These approaches have been used effectively in complex design problems like optimization of national electric grid, pipeline network design and finding innovative designs for gas turbine engines.

Genetic Algorithms

Genetic algorithms use the biological evolution analogy to direct the search. The problem is defined as a function to be optimized through variation of a set of parameters. The parameters are represented as a finite set of gene bits (called chromosome), where each gene is a bit, real or a probability[Baluja95]. Given a generation of chromosomes (called population), the algorithm evaluates the value (the fitness) of the functional, assigning weighted survival probability to each chromosome. Candidate chromosomes are then statistically selected and bred (crossed-over, mutated) to generate the next generation of chromosomes. Each generation is proven to be better for non-random functionals. A variety of techniques have been developed to improve efficiency[Davis91], and generality for engineering problems. Robot optimizations continue to take advantage of these algorithms, with the drawback of reimplementation to each new application.

Innovation and Creativity Enhancement

This research expands design spaces beyond traditional parametric variations to include configurations that are generated from understanding underlying functionality. Although initial results are promising, this research assumes the existence of effective representations of the design, extensive databases to draw on or effective physical simulators to predict design behavior. This research is summarized as it will have significant impact on future robot design. It is notable that operators in genetic algorithms have demonstrated innovation in areas like writing programs[Koza92], generating music and evolving creatures[Sims94]. Roboticists rarely take advantage of these methods to find innovative designs for wide ranging problems from leg configurations to shape of wheel treads.

Knowledge acquisition

Knowledge acts as a database for future innovation[Coyne87] and design activity. Hence tools to acquire knowledge of the design domain and its manipulation assume significance. Knowledge can be acquired through human entry, querying (human) experts, translating from existing sources, learning by observing/analyzing outcomes of predictions, and observing a design or analysis process. Examples of acquisitions[Rich91] systems include MOLE (uses human input to acquire and refine), and SALT (elicits overall design constraints and local constraints/parameters from experts). Knowledge in robot design involves several dimensions and though examples exist in traditional areas like mechanical design[Brown83] and no single mechanism exists that can capture all of it.

Knowledge-based innovation

Models of cognitive processes[Coyne88] like induction, deduction, abduction, and generation (based on rules, heuristics) have been applied to generate design (spaces) from existing knowledge and guidelines that evaluate product suitability. Some techniques that successfully aid these processes include search algorithms (like A-star), analysis tools (design evaluator), production/expert systems, predicate logic reasoners, and constraint based explorers. Examples of such innovatorsController Generation For Robots are MEET[Steinberg92]/ SACON (object oriented constraint propagators), PLASHTRAN (uses a frame based data driven approach to generate finite element analyses of the design), and DOMINIC I (expert systems for V-belt design). Robot designers have rarely used expert systems due to the absence of a framework with general knowledge in robot design and the mechanisms to apply cognitive processes effectively.

Creative systems

These approaches distinguish themselves from innovators[Murthy89] by creating knowledge rather than exploring or searching its domain. Usually a manipulator of physical principles generates constructs of design primitives or structures. These constructs when associated with information on their use become knowledge. Information on uses is usually generated by physical analysis, and/or human classification. Examples of such systems[Mitchell89] include “novel combination” (creates new knowledge by combining and reinterpreting knowledge from several domains), EDISON (uses fundamental physics driven by qualitative reasoning based on planning/heuristics to generate inventions), and 1stPRINCE[Cagan87] (uses monotonicity analysis[Choy] on structural equations to search solution space and discover parametric designs). However, roboticists do not use these approaches, as a framework to construct and manipulate robot designs is lacking, and so is the simulation of underlying physics to guide the search.

Robot optimization is nascent

Optimization of the comprehensive robot is rare, and mostly restricted to finding viable designs using generate and test approaches. Designers are burdened with casting the robot design problem in a computable representation to interface with existing optimization methods. The need is for optimization codes and robot representations that robot designers can use easily for configuration selection, parameter sizing, innovation and geometric placement.

Generating Robot Control Software

Robot software refers to the code that imparts robots with desired perception, understanding and behaviors. Control software refers to robot software that commands the robot mechanism as a function of sensed world and desired behavior. Traditionally, automatic generation frameworks like adaptive controllers and neural nets have been used to solve difficult control problems. There is a well established approach to controller generation for open loop Lagrangian chains[Kelly98] like manipulators. Approaches to creating planning software[Latombe91], task switching[Brumitt97] and higher level control[Brumitt97][Kodit85] are evolving. However, there is little work in automatic generation of control software for general robots that can impart a specified behavior by mapping sensor input to actuator output.

Controller generation

Controller generation typically involves selecting a model or control law and finding parameters that optimize the behavior of the robot for a defined goal. A simulation or the robot is used to predict the behavior of the robot for a given set of controller parameter values. These robot controllers deal well with the size, models, and noise present in real world engineering problems. Successful research in this area includes adaptive controllers[Kodit85][Chen94] for manipulation, neural net[Rosenblatt62] error propagation[Rumelhart86] to drive cars[Pomerleau92][Davis95], gait generation for snakes[Dowling98], swimming behaviors for robotic tuna[Barrett97], model based regression for excavation and Kalman filters[Maybeck] for optimal tracking. These techniques are however problem specific or require significant tweaking of the parameters.

Self Evolving Controllers

Nascent research in self-evolving controller structures promises to minimize human effort in writing controllers. Recently developed approaches select a controller architecture (e.g. indexed memory[Moore95], cascade neural nets[Fahlman90], orthogonal basis functions, or splines), select an initial controller with low complexity, learn parameters that generate the best behavior and incrementally add complexity to the model until the exhibited behavior is satisfactory. However, a drawback to such approaches is the need for ongoing reinforcement[Nechyba95], feedback or a reference behavior for the controller to learn from. In other words, these controllers learn from mimicing known behavior or perform local adjustments from feedback at a given. In the problem of generating controllers for innovative designs with no history (e.g. dynamic gait for a 3 legged walker) and only a cumulative reinforcement (e.g. how far it walked in 40 sec), these techniques do not apply.

Controller generation for novel robots not addressed

There are no approaches to generating controllers that meet robot behavior goals, where there is no existing precedent to the control of such a robot. Existing adaptive controllers and learning strategies do not apply due to significant work needed to create the controller structure for the problem or requirement for output to follow.

Chapter 3 **Methodology**

Synergy supports an evolutionary design model, allowing designs to grow from concept to detail; and design evaluation from preliminary heuristics, static analysis to dynamic simulations. It enables the creation of a robot design, so that the desired functionality and objectives match the physical robot. The design is expressed as a collection of components, interlinked by parametric relationships, physical constraints and software. These relationships merge the components into robot hardware and software. Robot software is integral to design and in most cases embedded in physical components. Synergy also assists the designer in searching through configuration and parameter space to find viable and effective robot designs. Synergy supports detailed evaluation of design performance by assisting in controller generation and coupling the controller(s) with the robot for the dynamic simulation.

Synergy: A Robot Design Environment Prototype

The robot representation is central to Synergy, and manifested as a "design spreadsheet". The spreadsheet interacts with the designer and the software agents, providing interfaces to the design problem input, simulation generator, optimizer and controller-generator. Figure 3-1 overviews the interactions of the software components in Synergy. The spreadsheet integrates these software tools, providing interfaces to each tool, enabling access/modification of design information, along with facilities to generate designs from heuristics, track and satisfy constraints.

The robot design problem is an expression of design objectives, heuristics and robot component database. The simulation generator uses a set of one-dimensional dynamic simulation agents to create a comprehensive simulation, where state data from each agent flows to all other agents at each time-step through the spreadsheet. The optimizer encodes the design problem as a string of bits and real numbers, employing a genetic search in the design space, using the spreadsheet and simulator for evaluating design options. The controller generator uses a generalized neural representation, and evolves the weights using simulation feedback, incrementally adding nodes until the desired behavior is displayed by the controller. This controller can be potentially fine tuned with

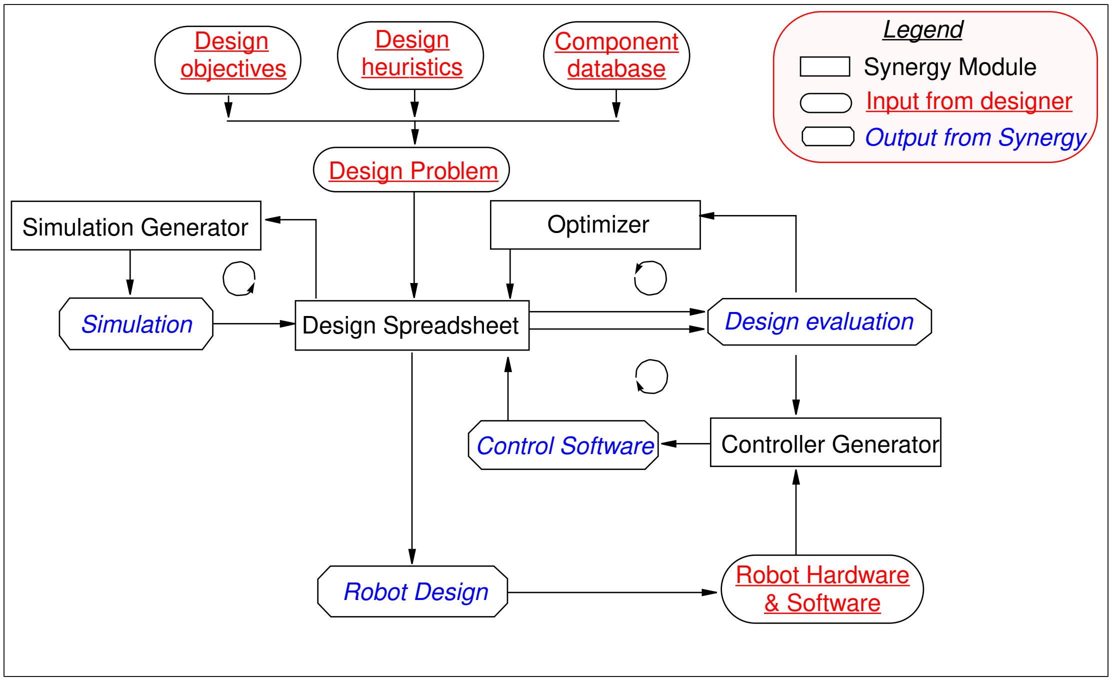

Figure 3-1: Overview of Synergy

the robot hardware to adapt to the differences between the design model and physical reality.

Robot Design Language and Spreadsheet

The robot design language is formulated to enable comprehension, extensibility, modularity and reconfigurability. This language is manifested in a design spreadsheet automating many aspects of its specification. The spreadsheet interfaces with the designer through a structured grammar, and provides ontology and data structures used by the supporting software modules and agents.

Components are the fundamental building blocks of the robot language. The robot is represented as a tree of such components. The tree of components represents a designer chosen breakdown of portable functional sub-systems like power and mobility. Each component encapsulates its engineering information and its relationships with other components. Since a component can refer to other components as children, each component can encapsulate a sub-system or an entire robot. Components inherit properties from the parents.

The spreadsheet interfaces with the designer through an object oriented, C-style syntax. Using this syntax and user commands, the designer can specify the design problem, query the robot design, setup objective functions, modify the design, perform optimization, simulation and generate control software. Synergy's modules can manipulate the spreadsheet similarly through the language syntax or using direct pointer address for efficiency.

The following descriptions profile the features, uses and rationale of the entire language and spreadsheet. Figure 3-3 gives an overview of the language and a taxonomy of the syntax.

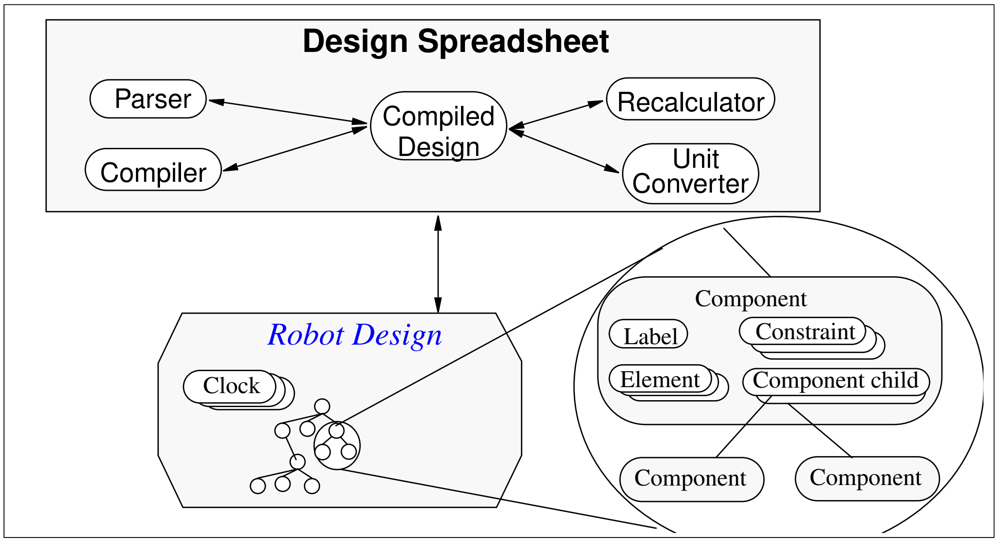

Figure 3-2: Overview of Robot representation and spreadsheet

Components

Synergy uses components as the functional and structural units of robot design. These are parameterized with respect to functionality, facilitating scaling and optimization. Each component has an identifying label which is automatically augmented with its hierarchical ancestry to give it a unique address. A component encodes multi-dimensional properties as a collection of mechatronic "elements". In addition, each component can be separately clocked, enabling nested calculation iterations. Component type(s) are keywords for set grouping. A component inherits several properties from its parent which can be overridden by specifying them in the component.

Components have a tree hierarchy, where each component has a parent and can have multiple children. A component can be moved to any other node in the tree, and such a move causes its children to move with it. A component can either be referred to by name or as a pointer reference. The use of component pointers instead of specific names allows the designer and optimizer to explore configuration changes without changing the rest of the design. The designer can also create generic components with undefined or default values that can be specified by the user at instantiation. For example this allows a wheel component to be created once and used in several places with different locations and sizes.

Each component has three possible utility states - active, frozen or dormant. "Active" means that it is fully functional. "Frozen" means that its elements are accessible, but their values cannot change; enabling rapid recalculation and extensive parametric testing of a part of the design. "Dormant" means that it is invisible to the spreadsheet; enables efficient reconfiguration, while retaining design knowledge in the spreadsheet.

The spreadsheet can incrementally recompile and propagate changes in component status, location and add/delete operations.

Example component

```
Component A {
    mass = 4;
    power = 25 * B.mass;
    Component B {};
    Component ptr D = &C;
};
```

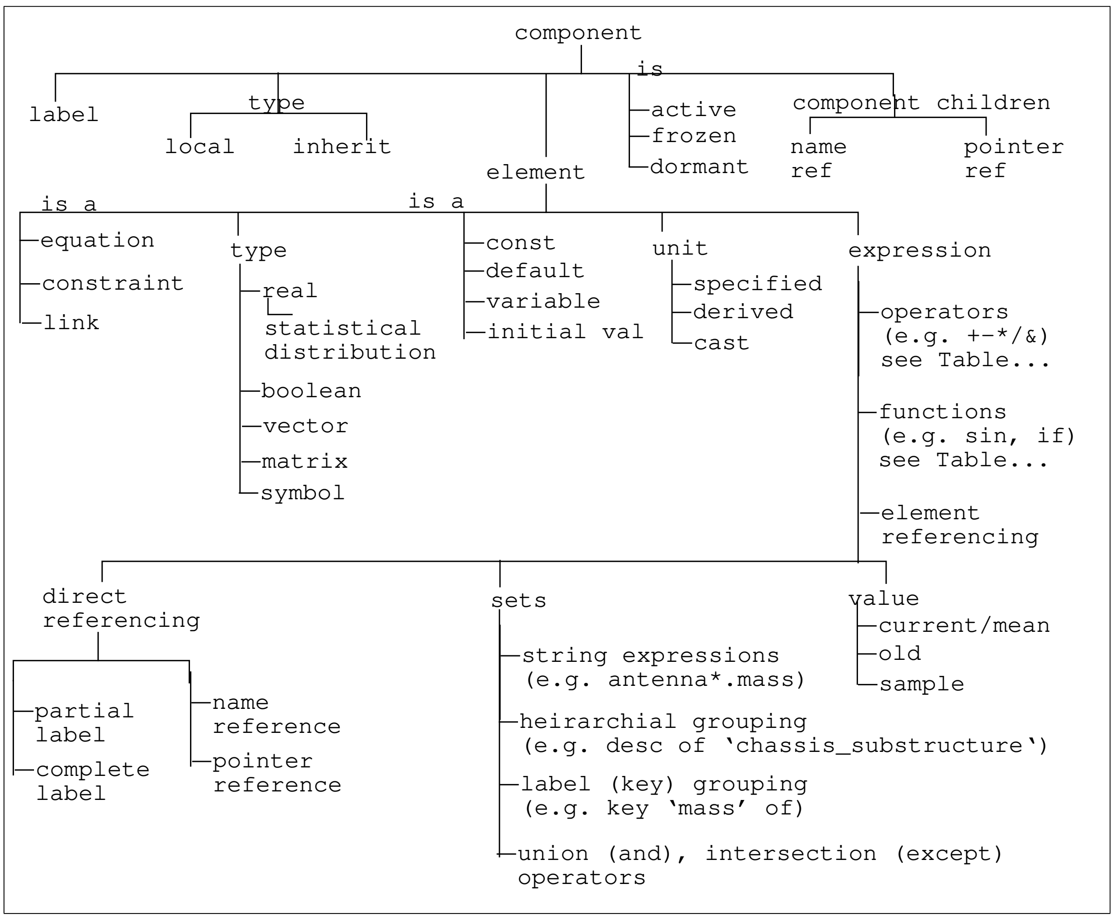

Figure 3-3: Overview of robot design language

Elements and Constraints

Elements and constraints constitute components, where each describes a particular property or relationship. Element’s express a components mechatronic properties (e.g. shape, size, power draw, heat content, V-I response, software) and relationships (e.g. mechanical connection, electric connectivity). Constraints express the limitations and conditions that other components or the robot need to satisfy for the component to function as engineered. The particular property associated with an element is defined by its label. An element's property is expressed in a particular type, units, and its value is either a constant, inherited default or a function of other elements (of the entire robot). Definition of element value in terms of other elements allows efficient expression of complex robot designs and relationships. The following bullets describe the parts of an element:

- An element has a type that categorizes its value - from real numbers to character strings to matrices.
- A special type is "statistical real" which allows a real number to be defined as a gaussian distribution or a {min,avg,max} triple, enabling noisy/chaotic variables like temperature, power draw or sensor output to be modeled.
- Each element can be a constant, a variable (defined by the corresponding equation), or be inherited as a default (for example the default mass of each component can be 0, which creates an element "mass" of value 0 if undefined in the component).
- An element can be declared "transparent", excluding it from all set operations. Functionally, such elements are convenient calculated properties but do not denote a physical quantity.
- Constraints are special boolean elements that need not have a label and are tracked separately. They are defined as expressions the result of which when false denotes that the component may not function as designed.
- Link elements provide modeling information for simulation agents, with information about other components it is connected to, the type of connection, etc.
- An element's value can be defined with a mathematical expression of other element values, with operators like +-*/. Each "type" has a defined operator and function set that can operate on values of that type. Certain functions can accept multiple types and/or convert values of one type to another. Functions like sum and product operate on sets. Table 3-3 lists the types with corresponding operators and functions.

Table 3-3: Synergy "types" and associated operators and functions

<table>
<tr><th>Type</th><th>Operators</th><th>Functions</th></tr>
<tr><td>real</td><td>= + - * / ^</td><td>sin, cos, tan, asin, acos, atan, atan2, hypot, exp, epm1, log, log1p, log10, rint, sum, product, max, min, mag, pow, sqrt, cbrt, poly</td></tr>
<tr><td>distrib</td><td>= + - * / ^</td><td>get_min, get_max, get_mean, get_stddev, sample</td></tr>
<tr><td>int</td><td>= + - % /</td><td>sum, product, max, min, mag</td></tr>
<tr><td>boolean</td><td>= ! == != < > >= <= && ||</td><td>not, if</td></tr>
<tr><td>vector</td><td>= + - * /; == !=</td><td>sum, product, mag</td></tr>
<tr><td>matrix</td><td>= + - * \\ /; == !=</td><td>sum, product, solve, mag</td></tr>
<tr><td>set</td><td>= & key desc of except</td><td>count</td></tr>
<tr><td>string</td><td>= +; == != > < >= <=</td><td>string, ““, mag</td></tr>
<tr><td>triple</td><td>= + - * /; == !=</td><td>sum, product, mag</td></tr>
<tr><td>quaternion</td><td>= + - * /; == !=</td><td>sum, product, mag</td></tr>
</table>

Spreadsheet Operations

The spreadsheet parses the robot design and designer input, passing it on to the compiler. The compiler operates incrementally, continually updating its internal data structures as components are added, deleted and modified. This allows the designer to use the spreadsheet to build up the design in pieces (even with loose pointers), test incrementally and create sub-systems that are not complete by themselves but can be used several times. The parser allows components and elements to be referenced by partial names (for e.g. a.b.c.d.mass, can be referred to as d.mass), if the partial name is non-unique, the nearest neighbor is matched. Elements and components can also be addressed as sets using several schemes like wildcard matching (CPU*), grouping (descendents of...) or local name match (key “power”). If changes to the design insert a closer element with the same partial name, or adds more members to sets, the compiler updates the design accordingly.

The compiler checks incoming design information for consistency and links it with the existing design network. The compiler stores the design as a networked set of elements, where the distinction of component is only for bookkeeping and inheritance purposes. Potentially each element is connected to every other. This complex network is searched for recursion loops and the designer is informed of the minimal set of such loops. The recursion loops can be broken up by using a previous value of any element in that loops. The expression in each element compiled into an efficient structure for re-calculation and stored in an indexed list.

The spreadsheet recalculates element values at the advance of the “iteration” clock counter. Potentially, elements can have different clock counters enabling recalculation to proceed at different rates as desired. The ability of elements to reference previous values enables the design to evolve and converge with each advance of the iteration clock. Other clocks synchronize the operation of simulation agents and the optimizer.

Physical units (like kg, meter, second, $, watt) are vital to expressing the robot design and the units of all expressions and constants in a design are rarely consistent. The spreadsheet automatically converts units to a common standard (like SI) for consistency. In addition, the units of mathematical expressions are automatically derived and checked for consistency with the equation. This is an effective check for correctness of heuristics and expressions, where errors are common and most often visible in these checks.

Complete Language Grammar

parse_block <- <component> [<parse_block>] | <element> [<parse_block>] | <global_def> [<parse_block>] | <clock> [<parse_block>]

label <- <alphabet_character>|’_’ [<label2>]

label2 <- <alphanumeric_character>|’_’ [<label2>]

component <- ‘component’ <label> ‘{‘ [[(‘inherited’ | ‘local’)] <user_type>] [<parse_block>] ‘}’

// constraints must be of type bool.

// transparent means that sets should exclude this element

element <- [‘transparent’] [<dtag> | <symbolic_type>] [[‘element’ | ’constraint’] <label> [‘(‘<args>’)’]’=’] <function>|’<nil>’[‘:’<data>][‘,’<e_clocks>]’;’

dtag <- ‘bool’ | ‘int’ | ‘matrix’ | ‘real’ | ‘triple’ | ‘vector’

args <- [<dtag>] <label> [‘=’ <function>][‘,’ <args>]

// main clock secondary clocks

e_clocks <- ‘clock’ <label>[‘,’ <label_list>]

function <- <arg> [<op_tag> <function>]

arg <- ‘(‘<function’)’ | ‘(‘<dtag>’)’<arg> | <unary_op> <arg> | <data> | <data path> | <element path> | ‘old’ <element path> | ‘self’ | ‘fcall’ | ‘iter_loop’

] fcall <- <func_tag> ‘(‘ <f_list> ‘)’ | <func_tag> ‘(‘ <set> [<exception>] ‘)’ | <label> ‘(‘ <arg_lbl_list> ‘)’

arg_lbl_lst <- [<label>’=’] <function>[‘,’<arg_lbl_list>]

f_list <- <function1> | <function1> ‘,’ <f_list>

iter_loop <- ‘for’’(‘ <label>’,’ (<set> | <label>) ‘)’’{‘ <function> ‘}’

op_tag <- (see op_tag_list)

func_tag <- (see func_tag_list)

data <- [‘bool’] ‘true’ | ’false’ | [‘int’] <int> | ‘matrix’ <matrix> | ‘&’<path> | [‘real’] <real> | ‘set’’(‘<set>’}’ | ‘shape’ <filename> [‘*’ <triple>] | ‘string’ [“]<string>[“] | “<string>” | [‘symbol’] <symbol_name> | ‘triple’ <triple> | ‘vector’ <vector>

triple <- ‘(‘<real>’,’<real>[‘,’<real>]’)’ | ‘{‘<real>’,’<real>[‘,’<real>]’}’

quaternion <- ‘(‘<real>’,’<real>’,’<real>’,’<real>’)’ | ‘{‘<real>’,’<real>’,’<real>’,’<real>’}’

vector <- <vrow> | ‘(‘ <int> ‘)’ <vrow>

matrix <- ‘(‘ <int> ‘)’ ‘<vrow> (w) { elements } | ‘(‘ <int> ‘,’ <int> ‘)’ <vrow> (h, w) { elements } | ‘{‘ <vrow_list> ‘}’

vrow <- ‘{‘ <real_list> ‘}’

vrow_list <- <vrow> [‘,’ <vrow_list>]

real_list <- <real> [‘,’ <real_list>]

// distribution requires two of (min, max, mean) or both mean & stddev

distribution <- ‘distrib’ ‘(‘ <distarg_list ‘)’

distarg_list <- (‘min’|’max’|’mean’|’avg’|’typical’|’stddev’) ‘=’ <real> [‘,’ <distarg_list>]

user_type <- ‘type’ <type_name>’;’

// an elements main clock is specified in the element def. if an element is

// specified in a clock’s set it is a secondary clock for that element

clock <- ‘clock’ <label> [‘:’ <set>]’;’

set <- <set_list> [‘except’ <fset_list>]

fset_list <- <finite_set> [‘&’ <fset_list>]

finite_set <- ‘key’ <pattern> [‘of’|’,’ <set_specs>] | ‘type’ <type_name> [‘of’|’,’ <set_specs>] | ‘desc’ [‘of’] <label> [‘of’|’,’ <set_specs>] | ‘set’ ‘(‘ <label_list> ‘)’ | ‘self’

set_specs <- ((‘desc’ [‘of’])|’of’)<label> [‘of’|’,’ <set_specs>] | ‘key’ [‘of’] <pattern> [‘of’|’,’ <set_specs>] | ‘type’ <type_name> [‘of’|’,’ <set_specs>]

global_def <- <default_def> | <type_def> | <enum_def> | <symbol_def> | <link_def>

default_def <- ‘default’ [<dtag>] <label> ‘=’ <data>’;’

type_def <- ‘define’ <user_type> [‘implies’ <type_list>]’;’

type_list <- <type_name> [‘,’ <type_list>]’;’

enum_def <- ‘enum’ <symbolic_type_label> ‘{‘ <label_list> ‘}”;’

symbol_def <- ‘define’ ‘symbol’ <label>’;’

link_def <- ‘link’ <label> [‘->’|’,’ <label>][‘:’<link_specs>]’;’

link_specs <- attachment

// src_pt tgt_pt distance | ‘dist’ <triple>’,’ <triple>’,’ <real>, // src_pt1 src_pt2 tgt_pt1 tgt_pt2 | ‘axial’ <triple>’,’<triple>’, ’<triple>’,’<triple> // force src_pt tgt_pt | ‘force’ <function>’,’<triple>’,’<triple>

label_list <- <label> [‘,’ <label>]

Generating Simulation

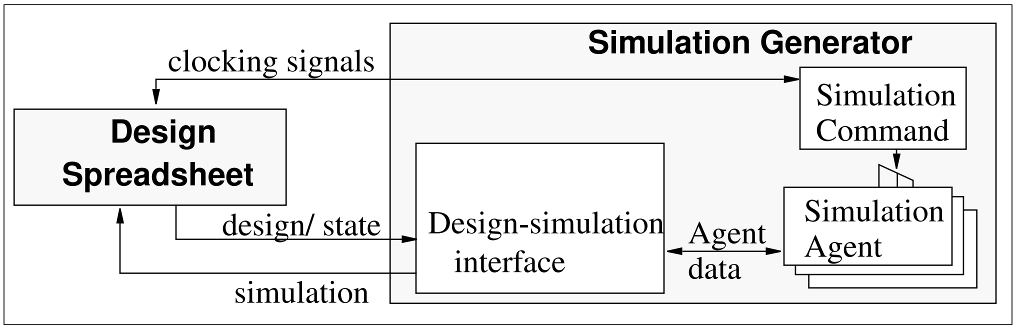

Figure 3-4: Overview of Simulation Generator

Simulation is generated by merging output from one-dimensional agents into the design spreadsheet. Agents include modelers, resource evaluators or constraint tracker. The spreadsheet injects numerical model and state information to the agents. The agents integrate the models and generate state information for the next time step. These results are fed back to the spreadsheet which fuses the latest state information from all the agents into a comprehensive simulation.

Each agent is controlled by a clock with potentially different time steps to enable uniform fidelity and computational effectiveness. The agent clocks are concurrent to enable a valid multi-dimensional simulation.

Each agent queries the spreadsheet for the model and state data relevant to its dimension. This is accomplished by the agent which knows the vocabulary of element labels and links specific to its dimension. The spreadsheet creates a graph of agent specific links and passes on the values of all elements with the requested label to the agent. The agent casts the model and state information into a numerical problem and solves it to produce simulated state information for the next time step. The spreadsheet ongoingly replaces the old state information with the new, to create the comprehensive simulation. The simulation controls clocks to ensures that all simulations are in sync to within the largest time step, and also schedules spreadsheet updates to ensure that the simulation data at any given time is consistent.

One-Dimensional Simulation Agents

The ideal simulation agent is a state-of-the-art uni-dimensional simulator interfaced to Synergy. However, this approach was only possible for mechanical and electrical simulations. The other simulation agents were coded for minimum functionality, where current packages did not have facilities to communicate model information and simulation output at every time step (e.g. thermal agent), or did not address the particular agent need (e.g. constraint agent).

Mechanism

Dynamic simulation of the mechanism uses Coriolis, a numerical modeling and simulation package for rigid-body worlds. It has the advantage of creating models on the fly by treating the entire problem numerically. Coriolis models the world as a set of mechanically interconnected rigid bodies, allowing connections like clamps, hinges, pins, springs and strings to be defined among rigid bodies. Gravity and other fields can be specified as global influences. Friction is modeled accurately using metrics static and dynamic friction associated with each body. Other forces like motor interactions and viscous drag can be modeled as externally applied forces. The objects have virtual methods allowing many of these functionalities to be augmented smoothly.

Coriolis generates dynamic simulation by integrating the motion of each body as a free rigid body, treating the constraints locally as a set of linear equations. These equations are dependent on the rigid body configuration and change with each iteration. As collision or bouncing of bodies is detected, the constraint equations are reconfigured. Although Coriolis uses the best available automated dynamic simulation methods, interactions of rigid bodies with soft substrates like soil or rubber are poorly modeled as rigid interaction.

Coriolis simulates the Newtonian behavior of the robot, subject to forces from gravity, friction, and actuators. The dynamics agent attempts to compute properties of motion, and inter-component forces by modeling the robot as a collection of rigid bodies with specified interrelationships (e.g. pin joints, axis joints). In this approach, each rigid body behaves independently subject to Newton's laws, considering external forces induced by a field (gravity), actuators, and/or constraints of motion, and friction. The primary constraint types impose a rigid body connection, a simple rotation with respect to another body, a non-penetration condition, or Coulomb friction with a non-penetration condition. Viscous friction is modeled as an externally applied force proportional to the relative velocity. All constraints, and governing conditions are generated and satisfied numerically. Hence no equations of motion are ever derived. Although this property is less desirable from an abstract analysis standpoint, numerical constraint generation allows realistic simulation of wheel contact, collision, and friction behavior. Such a facility is essential for mobility characterization of vehicles with suspensions, skid-steering, or legs.

Modeling Assumptions and Limitations

The robot consists of a set of rigid bodies, representable by convex polyhedra. Frictional contact between surfaces is reduced to point contacts. Terrain contact includes effects of static, and dynamic friction, but currently uses a idealized model of sinkage, and ignores changes in terrain geometry due to the rover's actions. Aerodynamic lift and drag are reduced to direct functions of dynamic pressure, using user specified coefficients. Forces due to electromagnetic fields are not considered at this time.

Model Structure

The robot is treated as a list of independent bodies as explained above. Each component can consistent of only one rigid body. Two bodies can be related by a connection (rigid, pivot, common axis, spring), and/or an actuator. Hence all rigid body constraints are binary. In addition, the agent detects collisions, and extracts appropriate constraints of contact and friction.

Model Equations

C: constraint vector, X: state vector of all bodies, M: mass & inertia matrix, F:applied force vector, F_e : Externally applied forces; F_c : Forces needed to satisfy constraints

The constraint vector C is assembled from governing physical relations of Coulomb friction and imposed holonomic motion constraints. The external forces F_e are direct applications of the gravity field, actuator torques/forces, aerodynamic lift/drag, and forces generated by spring constraints. The resulting problem is a Quadratic Programming (QP) problem.

$$M = \begin{bmatrix} [m_1] & 0 & 0 & \dots \\ 0 & [I_1] & 0 & \dots \\ 0 & 0 & [m_2] & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix}$$

$$M\ddot{X} = F; W = M^{-1}; \ddot{X} = WF$$

$$F = F_e + F_c; b = \left(\frac{\partial C}{\partial X}\right); A = b^T W b \qquad \text{Eqn. [3-1]}$$

Implementation

The resulting QP is solved by iterating through each contact force and frictional contact to satisfy the corresponding constraints sequentially (keeping previous constraints satisfied). The technique converges in O(n) operations.

Electronics

Dynamic simulation of the electronic sub-system used Spice, an industry standard for simulating analog. spice models as either a numerical model or a network of primitive components like RLC circuits, transistors and logic. Overview of Spice and approach to simulation.

This agent determines power usage of the electrical elements, including voltage & current properties, given the set of electrical elements in the robot, with corresponding interconnections. Common circuit elements like resistors, battery, capacitors, transistors, and amplifiers are modeled. The simulation generates transient analysis of the circuit.

Model

The model is a network of nodes and elements. Each node is a junction of three of more elements or a point of interest for voltage and current flow. A sparse set of linear equations constrain the system to follow Kirchoff's laws of current, and voltage. A set of (non)-linear equations specify the V-I behavior of each element. Each circuit element defines its governing law along with connectivity to other circuit elements. This information allows the formulation of the needed topological equations necessary to analyze the circuit. In general, a circuit element's V-I relation is non-linear, and time variable. This facility enables inclusion of changes in electrical properties due to environmental changes in properties like strain, temperature, and light.

Equations

The V-I characteristics of the network is determined by satisfying Kirchoff's topological laws (KCL, KVL), and the circuit element relations as in Eqn. [3-2].

$$
\begin{aligned}
&\text{Kirchoffs Current Law: } \sum_{i \in \text{node}} I_i = 0, \forall \text{nodes} \\
&\text{Kirchoffs Voltage Law: } \sum_{i \in \text{loop}} \Delta V_i = 0, \forall \text{ClosedLoops} \\
&\text{Circuit Branch Element Relations } I = f(\Delta V); \Delta V = f(I)
\end{aligned}
\qquad \text{Eqn. [3-2]}
$$

e.g., for a resistor, $\Delta V = IR$, where R is a known function of temperature

e.g., for an ideal capacitor, $Q = C\Delta V$, $I = \dot{Q}$, where C is constant

Implementation

The network equations are numerically formulated using the nodal analysis technique, where the state vector consists of the node voltages. The element voltage drops are linear combinations of the node voltages. Linearizing all non-linear elements around a operating point, element currents are similarly linear combinations of the node voltages. This yields a set of linear equations to be solved to obtain the node voltages (and hence element currents) at a time instant. Significant difference between the assumed operating point, and the resulting V-I relation results in repeating the iteration.

$$
\text{branch connectivity matrix } A_{kj} = \begin{bmatrix} 1 \text{ if current in element k leaves node j} \\ -1 \text{ if current in element k enters node j} \\ 0 \text{ if element k and node j are not connected} \end{bmatrix}
\qquad \text{Eqn. [3-3]}
$$

For each element, do $dH = dI/dV; dT = -Ig + Vg/Rs$;Ig & Vg - sources, Rs int resist

Update admittance matrix Y += (jj,kk: dH, jk,kj: -dH); j, k are element nodes

Update generalised current vector Is += (j: -dT, k: dT)

Solve node voltages Vn in, Y.Vn = Is, Iterate Until |Vn-Vnold| < max(.001Vnold,1E-5)

For each iteration, recompute Y,Is; This process repeats at each time step

If !(converged) in ~10 iterations, time step is halved until step <= .5/f; max exciting freq

Branch element voltage Vb = Transpose(A).Vn

Thermal

Heat is generated, absorbed by the robot due to various interactions like friction, insolation, terrain contact, power component inefficiency. This heat is distributed within the robot by processes of conduction, convection, and radiation. This agent models and simulates heat distribution. Basic elements include linear/cylindrical conductors, grey body radiators, and free/mass convection.

Assumptions and scope

Intra-surface temperature gradients and reflective phenomenon are ignored. Conduction is restricted to one dimensional transfer through either uniform or cylindrical cross-sections.

Model

The thermal interactions are modeled as a non-linear, capacitive electrical network. Table 3-1 below shows the analogy for various thermal components and constraints.

Quantity	Electrical Analog	Symbol
Heat capacity	capacitance	C
Heat flow	current	$\dot{Q}$
Temperature	potential	T
Conductance	1/R	G
Heat transfer	Ohm's law	$\dot{Q} = GT$

Table 3-1: Analogy between thermal and electrical systems

The network elements join at nodes. Each node is either a capacitive node (heat capacity), empty node (no capacity), or a source node (infinite capacity, constant temperature). Each capacitive node is grounded as the heat quantity is 0 for absolute 0. The units of temperature are hence Kelvin.

Equations

$$
\begin{gathered}
\text{Conductance, Convection } \dot{Q} = G(T_2 - T_1) \\
\text{Linear conductor } G = k\frac{A}{L} \\
\text{Cylindrical conductor } \frac{A}{L} = \theta\frac{d}{\ln(R_o/R_i)} \\
\text{Nodal capacitance, heat reservoir } C = MC_p \\
\text{Convection, } G = h_c A \text{ ;A: surface area} \\
h_c = f(T_{surface}, T_{fluid}) \\
\text{Mass convection } G = \dot{M}C_P \\
\text{Raditor } \dot{Q} = G(T_2^4 - T_1^4);(G = \sigma\varepsilon_{eff}F_{21}A) \quad \text{A:area, F21: black body view factor 2 to 1} \\
\sigma = 5.6677^{-8} W/m^2K^4; \text{Emittance } \varepsilon_{eff} = 1/(1/\varepsilon_1 + 1/\varepsilon_2 - 1)
\end{gathered}
\tag{3-4}
$$

Implementation

The generated electrical analog model is solved using the techniques, algorithms, and code developed for analyzing electrical circuits.

Resource

The resource agent is used to keep track of budgets for quantities like mass, power, communication, computing, cost and reliability. Primitives for set definitions and operations like “sum”, “max” are used to generate these metrics. The resources are dynamically recalculated at each time step allowing tracking of dynamic resources like fuel and heat.

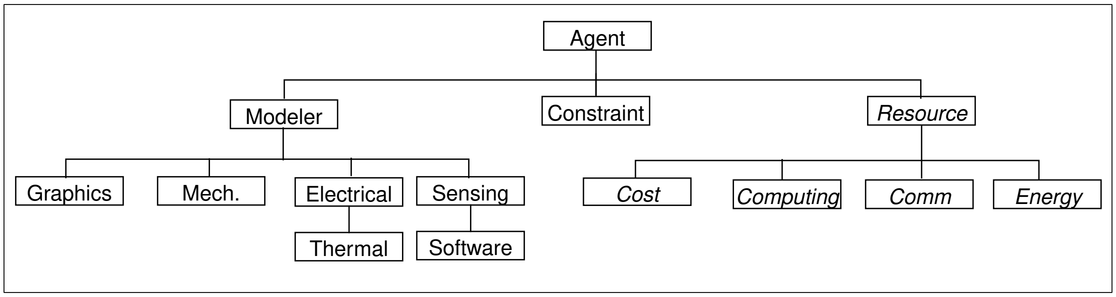

Figure 3-5: Simulation agents used in Synergy

Optimization

Synergy implements three kinds of optimization; rule based, root finding and genetic search. These three methods have increasing computing cost and correspondingly increasing general applicability. Hence rule based and root finding are used wherever applicable, and genetic search is used only when needed.

Rule based switching

Rule based optimization is devised as a switching behavior triggered by the spreadsheet during recalculation or propagating of parameters. Rules are used to significantly prune the design space when the choice is clear based on experience or analysis. These rules represent precompiled knowledge and expertise. However, these rules should be used carefully within their domains of applicability.

Root finding

Root finding is implemented using an iterative algorithm that computes the LHS (left hand side) of equality constraints from its RHS (right hand side). The RHS for each iteration is computed from a previous LHS or initial values. Root finder finds viable and optimized designs by finding parameters that just satisfy resource needs. This method is applicable when the cost of using a resource increases monotonically with increase in its quantity.

These implicit equality constraints which comprise a non-linear root finding problem are generated from expressions of generalized impedance matching of resources like power, energy, data rates and computing. The problem of design consistency is one of finding a set of component parameters so that all parametric relationships are satisfied. This problem is also modeled as a root finding problem.

Genetic search

Genetic search is used when the design has more parameters than constraints, rules are insufficient or when monotonicity assumptions do not apply. The genetic optimizer finds optimal designs using either spreadsheet formulae or simulations to evaluate the designs.

The genetic search algorithm uses operators for both real valued and bit based representation of the variables. Though real-valued representation is more efficient, bit representation is needed to encode discrete variables like configuration choices. Hence, both representations were used depending on the design problem. In these representation each design is encoded as a string (chromosome) of reals and/or bits. The

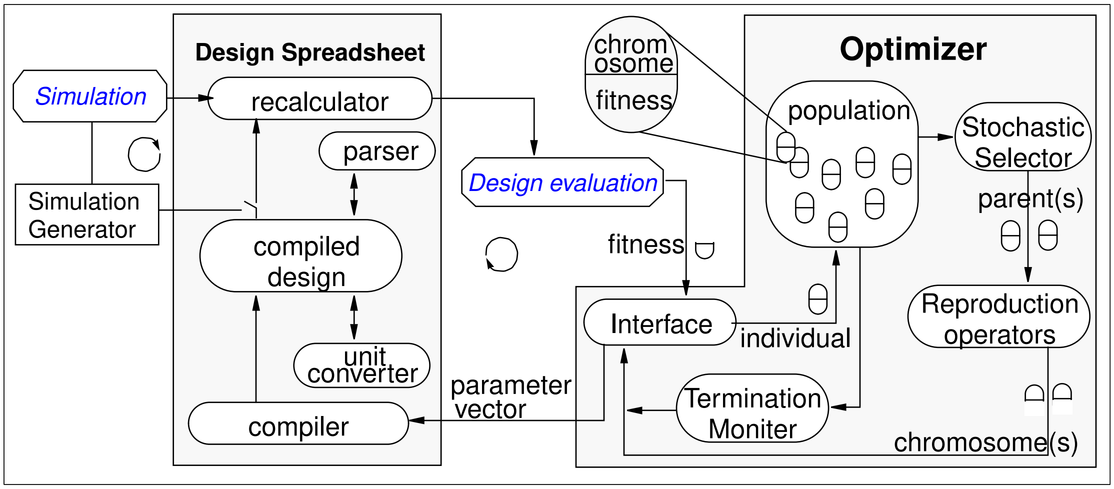

Figure 3-6: Overview of Optimizer using a genetic algorithm

variables that make up the chromosomes are designer defined, who controls the parameter space. The rest of the design is a function of these chromosomes. New chromosomes are recompiled by the spreadsheet and a designer defined expression of fitness is evaluated to compare the design with others.

The genetic algorithm operates with a population of such chromosomes, usually 100 to 1000. New members are generated from existing ones, where the probability of an existing member being selected for this "reproduction" operations is proportional to its fitness relative to others. The reproduction operators used are the uniform crossover and mutation operators. The Uniform crossover refers to exchanging random parts of the chromosomes of both parents, while mutation refers to replacing a part of the parent with a randomly generated value. The operators are triggered at designer defined probabilities, typical values being 0.5 for crossover, 0.3 (0 to 0.15 probability of mutation/bit) for mutation and 0.2 for other miscellaneous operators.

The algorithm terminates if the design fitness of any member exceeds a specified value, or if the fitness stagnates by being constant for a number of generations, or if the allocated computational time is exceeded.

Creating Control Software

Control software is created by finding a function to map sensed input to actuator command, so that the robot exhibits the desired behavior. The desired behavior is expressed as the integral of a defined metric over a defined period in a test problem. For a given controller, a simulation is generated with the given initial values and time period to evaluate its performance. A genetic algorithm is setup where the variables are the parameters of the controller function. The controller function is a neural model, where each node is connected to all input, output and hidden nodes. Each node has persistence, enabled by feeding back its signal through a linear filter of variable time delay.

The genetic algorithm is run until satisfactory behavior is exhibited or if the behavior saturates. If the result of the run is not suitable, the models complexity is increased by adding one more fully connected node, and the process is repeated until a good controller is found or adding nodes is not impacting robot performance.

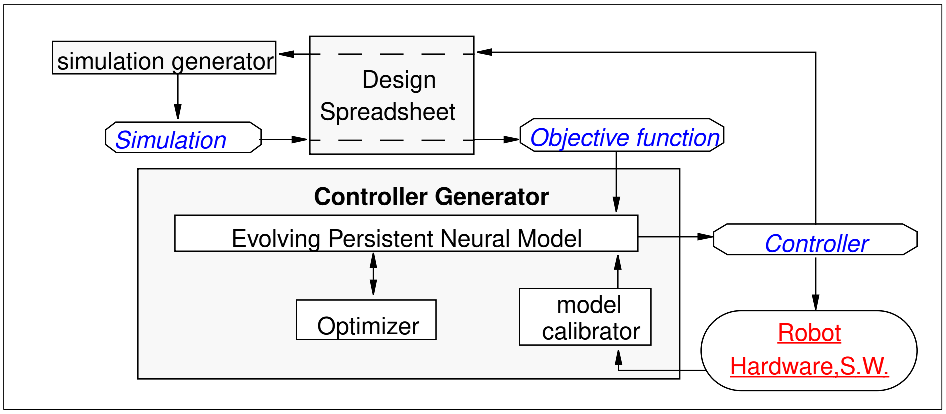

Figure 3-7: Overview of Controller Generation

The number of simulations involved in generating a controller can be large. If the controller is to be tested on the physical robot prototype, the genetic algorithm approach is ineffective. However, the controller model generated in simulation is expected to model the physical robot's behavior with discrepancies in values of the model and environment. A controller for the robot can hence be generated efficiently by recalibrating the existing controller model. Since the robot's behavior is not deterministic with each test run, a statistical perturbation and descent algorithm (SPSA) is used to adjust the parameters of the controller.

Chapter 4

Lunar Rover Design

Synergy is applied to several aspects of lunar rover design. Lunar rover design exercises a wide range of technologies, robot complexity and diverse requirements for several robot tasks. Design objectives, technologies and knowledge are encoded in the Synergy's language and used to generate several designs. In this process, several uses of Synergy are demonstrated. This chapter profiles the objectives and technologies of lunar rover design in a form translatable to Synergy's grammer. When using Synergy, the designer is called to quantify and parameterize the various objectives and technologies involved. Synergy interconnects these disparate knowledge, generates a unified design and proceeds to generate simulations, optimize the design and generate control software.

Although Synergy can be applied to any robot design problem, the Lunar Rover Design (LRD) was chosen to illustrate Synergy's use and power. LRD aspires to formulate flight rovers with real-world considerations of viability, cost and time. In not having a predefined or an existing design, it is distinct from well evolved designs like manipulators or communication satellites. The generated designs need to deliver unprecedented capability using current technology and existing hardware components within a reasonable cost. The range of tasks and missions in LRD allowed Synergy to exhibit its flexibility. The following sections outline the various design objectives addressed by Synergy and the technologies that were brought to bear.

Lunar Rover Design Objectives

LRD challenges include surviving the lunar environment for over two-years or venturing into permanently dark areas, and driving in dusty vacuum for 2000km. Commercial design objectives cause a dramatic departure from traditional space article design, demanding ambitious, reliable capability, while meeting tight cost and time budgets. These challenges include navigating unknown terrain, maintaining continual high-bandwidth communication, high reliability for human support/ commercial needs, and generating high quality immersive video.

Ice Discovery

The objective is to confirm and map the extent and composition of volatiles including water at the lunar South Pole as hypothesized by data from the Clementine mission. The needed duration is at least 14 days, 4x1 km long sorties into the permanently shadowed areas (cold traps), 30 km of exploration in the lighted areas and 30 (10 cold trap, and 20 lighted area) regolith assays. Each regolith assay involves drilling to a depth of up to 1m, draw 10 1gm samples, with at least two samples spaced at 1cm.

Edutainment

One objective is a commercially motivated pair of teleoperated, unmanned rovers visiting the historic sites of Apollo 11, Ranger, Surveyor, Apollo 17 and Lunokhod 2. This scenario would generate excitement in the lay public for lunar exploration through immersive imagery and participation.

Commercial needs call for low cost (< $200 Million) and four year programmatics. Return on investment calls for 24 hr/day, two year mission and a 2000 km trek returning more than a peta-byte (10^{15} bytes) of high quality imagery of the lunar terrain. This scenario calls for a rover spacecraft, the like of which has never flown.

The needs are high-quality, immersive, color video returned in real-time to Earth and viewed in theme parks, movies, television broadcasts, CD-ROMS and photographic stills. Rover motion-history and science data is needed to generate a realistic telepresence experience accessible to millions of explorers worldwide.

While exploring at a slow 0.2 m/s, the rover is teleoperated from Earth (the round trip time-delay is about 5 seconds) by amateur drivers. During this time, the rover must safeguard itself from unforeseen events, human error and deliberate operator mischief.

Mining (ISRU)

The Moon is a premiere venue for utilization of space resources and continuous human presence beyond Earth. The economics of in-situ resource utilization (ISRU) call for lunar manufacturing bases for propellants, power systems and structural materials. NASA’s long-term vision includes lunar circumnavigation, prototype ISRU, power generation for terrestrial use, habitat construction and lunar telescope deployment. In all these scenarios, robots are the precursors and tools of choice for minimizing both cost and risk, while maximizing capability.

The first robotic ISRU devices would farm the lunar top soil and/or transport ISRU materials and products across the landscape. One task is to repeatedly transport ISRU products across small distances on benign terrain. The objective is to maximize the total volume of operations with a given robot in tonnage and distance.

Earthrise 2001

This objective is a short commercial lunar mission (less than a lunar day), landing a few days prior to Dec 31, 2001 and sending video of Moon and Earth at the dawn of the new millennium. The imagery needs are broadcast quality television. Video of orbit and landing are preferable and not necessary.

Lunar Rover Design Environment and Requirements

The Moon is highly suitable for robotic exploration. Its proximity to Earth enables remote teleoperation and real-time video transmission, while constant visibility of the near side allows continuous line-of-sight communication. The Moon's barren terrain is favorable to robotic perception and navigation algorithms. Without an atmosphere, a steady supply of rich solar power is available. However, the extremes of lunar environment and significant difference vs. terrestrial conditions present a stiff challenge to rover design.

Temperature and Light

The surface temperature during lunar day approaches 130˚C (superheated steam), plummeting to -180˚C (liquid nitrogen) at lunar night. The temperatures at the poles are cooler [-50C lighted, -180C cold trap]. In daytime, the rover is exposed to insolation of 1385 Watts/m^2 and infrared surface radiation of up to 1146W/m^2. During the night, the rovers are submerged in cold vacuum of -270C.

Radiation

Expected radiation dose is 3 kRad/yr ambient and up to 15 kRad/yr in solar flares. A margin of safety of up to 4 is desirable. High energy radiation, including ultraviolet, damages optical surfaces, requiring special materials and coatings for paints, radiators, glass and solar arrays. For short missions flare activity (10 kRad worst case) is incorporated. The onset of a flare can be observed with a lead time of minutes.

Electrostatic Dust

Lunar dust is comprised of very fine, highly abrasive particles which are levitated by the terminator (day-night transition) at 300 gm/m^2/yr. Additional dust is created by 1 km/s micro-meteorite impact at 10^{-1}gm/m^2/yr. The dust particles about 1 micro-meter in size carry an electrostatic charge, causing cling and creep into all dielectric surfaces. The rovers will need to inhibit the accumulation of lunar dust on optics and thermal radiators. The effect of dust on radiative efficiency remains unknown as the previous lunar rovers (LRV-Apollo and Lunokhod) had differing experiences.

Terrain

Known lunar terrain indicates that rovers of 1-3m size will need to encounter 25 degree slopes and 0.5m diameter craters or obstacles 15-20 cm in height. The craters have dimensions of 0.2D depression and 0.04D height. The extremely soft soil, restricts ground pressure to 6kPa or less.

Other Factors

The moon has a diameter of 3476 km and is about 400,000 km from Earth with a day-night period of 27.32 earth days. The rotation axis is inclined to earth at 6.6 deg and to the sun at 2 deg, with a total precession period of 27.32 days and a cycle of 18 years. The gravity on the moon is 1/6^{th} that on Earth. Robots on the moon will encounter deep vacuum of 10^{-10} Torr, far lower than orbital space and highly difficult to reproduce on Earth. In this vacuum, most plastics and composites outgass and deteriorate.

Lunar Rover Technologies

LRD draws upon a broad set of technologies and component options to generate viable, effective rover designs. Synergy automatically generates and quantifies robot interrelationships, creating a bottom-up derivation of system parameters like mass and cost. Such a derivation early in an evolutionary design process (like LRD) is difficult without Synergy and seldom found in design of spacecraft or robots. This system view allows rapid analysis of various design modifications.

Lunar rover designs are encoded as a set of interconnected components. It was discovered early in the design process that components can be sized or alternates used to improve the overall design. However, the optimal choice is related to the design objectives and the other components used. Hence for Synergy to generate consistent designs and optimize them, it is important to encode the components and underlying technology in a modular, parametric form. The technology modules include power, communication, locomotion, thermal control and imaging. These modules are related usually by aggregate parameters like mass and power, and by engineering constraints.

The following sub-sections distill existing information and knowledge relevant to technologies needed for lunar rover design. Majority of the knowledge used has been validated by other research through analysis, experimentation and independent review.

Launch

In assessing launchers, factors such as payload, volume, orbit energy, economy, availability, and lander compatibility are evaluated. Table 4-2: compares several launchers relevant to a lunar mission.

Table 4-2: Overview of launch vehicles relevant to lunar insertion of 1-500 kg rover target

Launcher	GTO,ton Payload	TLI, ton payld	Volume h m x dia m; taper h x d	Cost $M	Availability
Proton	20	6.5	5.5 x 3.8; 2.7 x 1.5	80	3 yr lead
HII-A, 2 ton	10.5	3.00+	3.7 x 4.6	75	3 yr lead
HII-A, 3ton	-	5.00+	3.7 x 4.6	105	in development
Soyuz II+ Fregat	7	1.7	-	25+20	3-4 yr lead
Delta-II, 7925	-	1.315	1.8 x 2.4; 3.0 x 0.0	55	3 yr lead
Ariane 5, mini-Sylda	-	.85-1.2	1.8 x 3.6	34-45	tentative, 4 yr lead
Taurus	-	425	2.7 x 1.4; 3 x 0	26	3 yr lead
Pegasus XL	-	120	2.7 x 1.4; 3 x 0	16	3 yr lead
Ariane 5, aux	.3	0	1.5 x 1.5	6-9	3 yr, low priority
Ariane 5, mini-aux	.1	0	.6 x .6 x .8	1	3 yr, low priority

Descent and Landing

There are two existing landers with lunar relevance, a Phobos lander derivative and the 1998 Mars lander. The Mars lander utilizes promising modular technology, and although its current form may/may-not be usable for lunar landing, its potential for adaptability needs examination.

When fully fueled, the Phobos derivative weighs 6700 kg with a delivered surface payload of 600 kg. It can land with a CEP of 4.3km of the designated coordinates with a maximum velocity of 2m/s vertical and 1.2m/s horizontal. The cryogenic fourth stage carries the lander and rovers into an equatorial lunar orbit of 100 km. The fourth stage disconnects after orbital corrections which shift the orbit to intersect the landing site.

Table 4-4: Rocket engines of planetary relevance

Rocket	thrust N	Isp lbf/lb/s	(dry) mass	size	Life	Fuel
Thiokol, Star 31	80k	290	1393 wet	.76x1.29	46s	solid
Thiokol, Star 37FM	3048k/63.7	290	83+1067	.93x1.68	63.7	solid
Thiokol, Star 37XFP	2097k/66.5	289.9	73+884	.93x1.51	66.5s	solid
Aerojet, attitude	445	309	1.86	-	-	bi-prop
TRW, omv var.	57.8-578	280-308	6.8	-	2800s, 28hr total	MMH
Aerojet, attitude	62	287	1.13	-	-	bi-prop
TRW, mre-15/omv	44.5-89	225	1.13	-	.02s*100k	hydraz.
Aerojet, attitude	21.35	285	0.57	-	-	bi-prop
Aerojet, attitude	2	265	0.27	-	-	b-prop
Huges, XIPS	0.0178	2585	68	-	-	Xe, 304 W

A solid rocket decelerates Mars 98 lander to 50 km above the surface with a nominal delta V of 2600 m/s. The lander ignites 7.5 km above the surface at a velocity of less than 50 m/s, touching down at less than 1.5 m/s. Sensing feedback is provided by IMU and vision tracking. Additional fuel may be added to enable hover for visual confirmation of landing site.

Since the lander is modular, Eqn. [4-1] can be used to calculate propellant mass for a given fuel (Isp), delta V and final mass (Mf). Mass values of the deceleration stage and the terminal descent stage are derived applying this equation with astrodynamics models. A few relevant rocket engines are summarized in Table 4-4:. Tank mass is 1/3rd fuel mass for cold gas tanks and hydrazine tanks available in 35 liters for Mars 98. Costs of solid motors average $250/lb of rocket mass.

$$\mu = e^{\left(\frac{\mathrm{deltaV}}{\mathrm{Vjet}}\right)};M_{\mathrm{deltaV}} = \left(\frac{\mu - 1}{\mu}\right)M_f;V_{jet} = I_{sp}\cdot 9.81 \qquad \text{Eqn. [4-1]}$$

Eqn. [4-2] can be similarly used to find approximate propellant mass to hover for a given time T.

$$M_{hover} = \frac{M_f}{\left(\frac{I_{sp}\cdot 9.81}{(g\cdot T)} - 1.0\right)} \qquad \text{Eqn. [4-2]}$$

Imagery

The chosen cameras are a function imaging needs of field of view (FOV), resolution, framerate, color and dynamic range. Table 4-6: can be used to select a camera based on requirements. Most spacecraft and rovers resort to image compression to lower bandwidth needs. Table 4-7: can be used to select the compression technique as a function of available bandwidth and computing resources.

Table 4-6: A cross section of available camera technology relevant to lunar conditions

Camera	FOV	res	fps	depth	mass	power	Cost
panosphere	360	1000x1000	10	12 bit	4 kg	15 W	0.5M
digital video	45	350x240	30	24 bit	3 kg	15 W	0.25M
hires multispectral	40	512x512	1	8 bit	1.5kg	8 W	0.5M
NIR	40	256x256	1	8	2kg	5 W	0.5M

Table 4-7: Compression ratios for good visual results @ 512 x 512 x 3 byte images, 4 Hz

Technique	Ratio	Int Computing Mips	Availability
JPEG	20:1	445	mature code
MPEG 2	80:1	445	mature chips
Wavelet still	60:1	178	software
Wavelet motion	200:1	178	6 mo lead
Fractal	90:1	267	software
Fractal	600:1	178	6 mo lead

Communication

Downlink communication from the rover to earth is achieved using a phased array, parabolic or an omnidirectional fixed antenna operating in X band. Eqn. [4-3] relates number of antenna elements, diameter, mass and power to the bandwidth, required bit-rate, maximum steer angle and gain. The downlink transmission is modulated using Nyquist QPSK to band-limit the signal within the allocated frequency range. A rate 2/3 turbo-code provides error correction to a 10^{-6}BER. The gain margin is conventionally evaluated using sum of worst case gain. However, for non-critical tasks like video transmission, it can be evaluated using a 99% probability estimate of losses at any given time.

Table 4-9: Gain values of elements of a communication link, X-Band

Element	Gain (dB)	Element	Gain (dB)
Atmosphere	-0.3	Ionosphere	-0.1
Rain	-1, -0.2 average	Space	-20.log(4*Pi*L*f/c)
Temperature Noise	228.6 - 10*log(Tmoon)	Data Rate	-10*log(R)
Eb/No for QPSK, Turbo	-2	Engineering Margin	-5
Phased Array	10*log(N*cos(tilt)) + 3.0	Phased Array efficiency	-10*log(0.7)

Table 4-9: Gain values of elements of a communication link, X-Band

Element	Gain (dB)	Element	Gain (dB)
Transmit Power	10*log(Ptransmit)	Line Loss	-3.0
Parabolic Antenna	20*log(Pi*D*f/c) + 3.0	Parabloic Ant efficiency	-10*log(0.55)
Annular Antenna	-2	Co-ax cable	-0.2*length
low flex cable	-0.5*length	hi-flex cable	-2*length
DSN wave distortion	-0.2	DSN modulation	-0.2
DSN Polarization	-0.2	DSN pointing	-0.2
DSN circuit	-0.5	DSN phase	-0.2
DSN Feed	-0.1	DSN size	Parabolic(22m,34m)
effective link - SUM > 0, L = 400,000 km; f = 8.9 GHz; c = 3E8 m/s; Tmoon = 130C equator; -50C pole			

Table 4-9: lists the gains of various elements of an interplanetary communications link. The specific gain values are obtained from JPL and literature. The objective is to have the sum of relevant gains greater than 0. Eqn. [4-3] relates size and transmitted power to mass, input power and pointing needs.

$$\text{Phased Array:transmit Power } Pt_{max} = 0.025N; \text{Input Power } P = 8Pt + 6; \text{Mass} = 2 + 0.0187N$$

$$\text{Phased array: } D = 2\sqrt{N \cdot 2 \cdot dx \cdot dy}; F = \frac{1}{1 + \sin(tilt + 1.5 \cdot HPBW)}; dx = F \cdot \lambda; dy = dx\sqrt{3}$$

Half power beam width HPBW = 58.5*f/c/D; D: Antenna Diameter;

$$\text{Parabloic dish: } Mass = D^2/0.25 + 4kg; P = 0.375Pt; Pt_{max} = 15W$$

$$\text{Pointing Mechanism if tilt > HPBW } (Mass, P) = (3kg, 20W) \cdot Mass_{dish}/0.5m$$

$$\text{Tilt: deviation of antenna from line joining Earth and rover } = slopemax + \angle(gravity, \text{Earth-Rover})$$

Eqn. [4-3]

Computing

Conventional space qualified processing proves expensive for a lunar mission due to high computing needs due to results shown later. Table 4-11: lists processors, capability and power, with relevance to flight (the chip designs known to be latchup immune and radiation tolerant). In addition to the processors, memory (about 5 W), bus control (1W), digital/analog I/O (5W) needs to be added to create a board. These boards can be formatted for VME or any standard form factor and operating system. The processor(s) and quantity selected is a function of total computing need calculated from software requirements.

Table 4-11: Computer processors overview

Processor	Type	MIPS int/ fp	Watts	Radiation Effects Prediction
M 68060	CPU	60/45	2	68020 rad qual, tough
Pentium Pro	CPU	292/248	23	CISC, extremely hard
R3000	CPU	28/36	4	inherently rad tolerant, flown
R10000	CPU	428/760	6	possibly rad tolerant, $1M
PowerPC 750	CPU	496/336	5.7	possibly rad tolerant, $.5M
TI C50	DSP	40/40	3	available in 100 kRad

Table 4-11: Computer processors overview

Processor	Type	MIPS int/ fp	Watts	Radiation Effects Prediction
Sharc	DSP	40/40	4	rad-qual available
TI C80, MVP	DSP	2200	11	$1M to rad qual
TI C60	DSP	1600	11	$1M to rad qual

Typical board mass is 1.5 kg including conduction cooling plate and the housing is 4-8 kg depending on the form factor. Each board has one CPU and 1-2 DSPs. Eqn. [4-4] is used to determine number of boards given processing power (total and CPU intensive) required by software and n - number of DSPs/board.

$$(n \cdot N_{DSP} \cdot Mips_{DSP} + N_{CPU} \cdot Mips_{CPU} \geq Computing_{need}) \cdot ;(N_{CPU} \cdot Mips_{CPU} \geq CPU_{need})$$
$$N_{boards} = \text{ceiling}(\text{Minimize}\|N_{CPU}, N_{DSP}\|_{\infty}) \qquad \text{Eqn. [4-4]}$$

Sensing and Safeguarding

Radar technology is selected to meet lunar environmental conditions of bright light, darkness, temperature extremes and dust. A typical sensor has a field of view of 30^o to 160^o. The 30^o version is used in an array configuration for terrain understanding and the 160^o version for broad collision warning. The sensors generate data at up to 20 Hz. Eqn. [4-6] and Eqn. [4-6] govern the configuration of the sensor array and performance for

Eqn. [4-5]

$$\text{no of array sensors } N = Fos \cdot fov/fov_{sensor}; \text{Fos - factor of safety, 2.0; FOV = 120deg}$$
$$\text{min lookahead distance, } l_1 \geq (speed^2/(2g\mu_r) + t_{perception} + t_{plan} + t_{dynReaction}) \cdot speed \cdot 0.9$$
$$\text{sensor height; } h_s \geq l_1$$
$$\text{lookdown angle, 90 is horizontal, } \theta_x = \text{atan}(l_x/h_s)\text{, hs is sensor height}$$
$$l_2 \geq l_1 + 4 \cdot speed/f_{sensor}/Res; \text{ fsensor - sesnor freq - 20 Hz}$$
$$\text{second lookdown angle, } \theta_2 \leq \theta_1 + fov_{sensor}$$

terrain understanding and safeguarding. These equations are derived from conditions of minimal observability, controllability and throughput matching for computing.

Eqn. [4-6]

$$\text{Map resolution limit, } Res_{limit} = speed/freq_{sensor} \cdot 4 \leq Res$$
$$\text{min map res, } Res = 2 \cdot \min(d/2, w/2, O/2)\text{; d,w - wheel dia \& wid, O: traversible obstacle}$$
$$\text{max speed for complete perception, } speed_{max} = freq_{sensor} \cdot (l_2 - l_1)/2$$
$$\text{No of map cells within l1 - l2 } N_{cells} = l_1 \cdot (l_2 - l_1) \cdot \tan(fov)/Res^2/2$$
$$\text{Computing need for map perception } = 10 \cdot N_{cells} \cdot FOS \cdot freq_{sensor}$$
$$(\text{Computing for planning } = 50 \cdot Narcs \cdot (l_2 - l_1)/Res \cdot speed/Res)$$
$$(\text{Latencies } T = computing \cdot CPU_{overhead}/CPU_{mips})$$
$$t_{dynReaction} = speed/Res/freq_{sensor}^2 + delay_{actuator}$$

To determine vehicle position with respect to the low-resolution global lunar map, a stellar compass is used. These readings are fused with inertial measurements and wheel encoders to generate accurate terrain maps and maintain precise dead- reckoning. Further sensors such as thermometers, ammeters, etc. are utilized to monitor the status of the rover itself.

Some Science Instruments

Science instruments are treated as mass, volume, computing and power payload. Table 4-14: lists conventional science instruments applicable to lunar exploration. Computing needs for cameras are derived from framerate and compression algorithm needs. Computing needs for other sensors are specified by the scientist and highly mission dependent and range from 0.1-10 Mips.

Table 4-14: Science instruments identified for lunar exploration

Instrument	Mass	Power	Usage	Volume	Cost, $M
Cryogenic drill (SATM)	4 kg	15 W	5 hrs/ m	1.5m x 0.2m dia	2.0
REGA	4 kg	20 W	15-90 min/assay		0.5
IR Laser	2 kg	10 W	20% duty		0.2
Strobe	1 kg	10 W	40% duty		0.5
Pan-Tilt	3 kg	10	30% duty		1.0

Locomotion and Structure

The key components of the locomotion system are the suspension and the drive. Previous studies and prototypes indicate that locomotion and structure accounts for 35% of vehicle mass. Eqn. [4-7] relates overall dimensions to wheel dimensions, fairing constraints and static stability on slopes.

Eqn. [4-7]

$$\left[\begin{array}{c}
\text{Chassis length, } L \geq 1.1 \cdot N/2 \cdot d\text{, where N is no of wheels, d is wheel diameter} \\
\text{chassis width, } W \geq 4 \cdot b\text{, where b is wheel width, L/W are overall lenght/width} \\
\sqrt{L^2 + W^2} \leq F\text{, where F could be the launch fairing diameter} \\
\text{static slope, } \Theta_1 = \operatorname{atan}\langle\langle L/2 - d/2 - l_{cg}\rangle/h\rangle\text{, where h is CG height above ground} \\
\text{static side slope, } \Theta_2 = \operatorname{atan}((W/2 - w_{cg})/h)\text{, where lcg,wcg are cg offset from center} \\
\text{specified terrain slope, } \Theta \leq \max(\Theta_1, \Theta_2) \\
d \geq (K \cdot O, w)\text{; K: 0.25 - 4 wheels; 0.35 - 6 wheels; O: size of Obstacle/ditch to overcome} \\
\langle w, d | \text{pressure} \leq 6\text{kPa}\rangle
\end{array}\right]$$

Number of Wheels and Suspension

The number of wheels and suspension determine terrainability in terms of traversable slopes, obstacles and soil. Common choices are four wheeled (rigid/ double axle/ rocker arm suspension) and six wheeled (rocker bogie) machines. Steering is always preferred (two wheel or four wheeled) due to lower power, lower attrition and enhanced maneuverability on slopes and tough situations. Appropriate tread for the wheel needs to be evolved experimentally.

$$\text{wheelload, load} = \text{weight}/N \cdot \cos\theta, \text{ where N is no of wheels}$$
$$\text{specified obstacle height, } h_o \le 0.25 \cdot d, \text{ If } N == 4$$
$$\text{specified obstacle height, } h_o \le 0.3 \cdot d, \text{ If } N == 6 \qquad \text{Eqn. [4-8]}$$

Wheel

Results from the Lunokhod exploration indicate that a ground pressure less than 6kPa is needed to avoid becoming mired in soft soil. For simplicity and accuracy of controlling the rover, the wheels are actively steered. The motors will be brushless DC with a harmonic drive. Due to vacuum, dry lubricants and O-ring seals will be used. Eqn. [4-9]-Eqn. [4-11] relate wheel dimensions and load to sinkage, rover's "drawbar pull", required output torque and ground pressure for given soil parameters.

$$\text{sinkage, } z = \left(\frac{3 \cdot \text{load}}{(3-n)\cdot(k_c + b \cdot k_\varphi)\cdot\sqrt{d}}\right)^{\frac{2}{2n+1}}$$
$$\text{Drawbar Pull, } DP = H - R_{all}$$
$$\text{soil thrust, } H = \langle c \cdot A + \text{load} \cdot \tan\varphi\rangle \cdot \langle 1 - 0\rangle$$
$$\text{Motion Resistance, } R_{all} = R_c + R_b + R_r + R_g + R_o + R_a$$
$$\text{torque} = R_{all} \cdot (d/2 - \text{deflection}), \text{ deflection} = 0 \text{ for rigid wheels} \qquad \text{Eqn. [4-9]}$$

$$\text{compaction resistance, } R_c = \frac{\langle 3 \cdot \text{load}/\sqrt{d}\rangle^{\frac{2n+2}{2n+1}}}{\langle 3-n\rangle^{\frac{2n+2}{2n+1}} \cdot (n+1) \cdot \langle k_c + b \cdot k_\varphi\rangle^{\frac{1}{2n+1}}}$$
$$\text{bulldozing resistance, } R_b = 0.5\alpha b z^2 \tan(45 + \varphi/2)^2 + 2cbz\tan(45 + \varphi/2)$$
$$\text{rollingresistance, } R_r = f_r \cdot \text{load}$$
$$\text{gravity resistance, } R_g = \text{weight}/N_{wheels} \cdot \sin\theta$$
$$\text{Obstacle climbing resistance, } R_o = 0$$
$$\text{acceleration resistance, } R_a = \text{mass}/n \cdot a \qquad \text{Eqn. [4-10]}$$

$$\text{wheel contact area, } A = l_c \cdot b$$
$$\text{wheel contact length, } l_c = \sqrt{d^2/4 - (d/2 - z)^2}$$
$$\text{wheel contact pressure, } p_{max} = \text{weight}/(2 \cdot A)$$
$$p_{max} \le p_{soil}, \text{ max pressure on soil before digging in, 3-6 kPa for moon} \qquad \text{Eqn. [4-11]}$$

Drive

The drive is responsible for moving the rover for the required distance and duration. This task includes large number of cycles in 140C/-180C ambient, interaction with fine, abrasive, electrostatic dust and survival of cryogenic night temperatures. Eqn. [4-12]

derives required drive velocity, cycles and power using wheel analysis and efficiencies of drive train components.

$$\begin{aligned}&\text{motor electric power, } P = \mathrm{torque}\cdot\omega/\eta\\&\text{overall drive effeceincy, } \eta = \eta_{\mathrm{amplifier}}\eta_{\mathrm{reductions}}\eta_{\mathrm{bearing}}\eta_{\mathrm{axle}}\\&\text{drive angular velocity, } \omega = \mathrm{speed}/\langle d/2-\mathrm{deflection}\rangle\\&\text{drive train cycles, } N_{\mathrm{cycles}} \le \mathrm{duration}\cdot \mathrm{speed}_{\mathrm{avg}}/(\pi(d-2\cdot \mathrm{deflection}_{\mathrm{avg}}))\end{aligned} \qquad \text{Eqn. [4-12]}$$

Structure

The base plate of the vehicle is a plate of aluminum honeycomb with vertical (1/32") aluminum rib plates creating the structural skeleton. To provide further stability, aluminum gussets are used to brace the ribs. The outer framework of the rover which creates the body shape is composed of 3 kapton sheets covered in kevlar. Curving the surface, along with attachment to the structure, is done through the use of epoxy. The rib structures are perforated to reduce weight without degrading strength characteristics. Such a light structure is expected to weigh 15-20% if the mass it carries.

Thermal Regulation

Spacecraft use mechanisms of conduction and radiation to distribute and transport internal heat. Dissipation of excess heat is through radiation, while generation of needed heat is through batteries or isotope sources (RHUs). Temperature extremes are defined by electronics operation [-20C,80C], survival [-80C,125C], and batteries [-20C,50C].

During the lunar day, the rover needs to remain cool. The rover is coated with a reflective white epoxy to reduce the heating effects of direct sunlight. The radiators which disperse the waste heat of the electronics enclosure during the day must be blocked during the night to minimize leakage. However, relying on hermetic sealing is undesirable as interference due to the lunar dust is possible. The radiators are connected to the thermal enclosures via high-conductance heat pipes. The outlying electronics, such as cameras and the phased array, have heat pipe connections to the main electronics compartment, and are maintained at the same temperatures.

Heat Transfer

Eqn. [4-13] quantifies the steady state heat balance between generated (LHS) and

$$\begin{aligned}&Q + H = P_r + P_i;\text{Q: enclosure heat, H: RHU heat}\\&\text{Pr: Radiator dissipation, Pi: Insulation loss}\end{aligned} \qquad \text{Eqn. [4-13]}$$

dissipated heat (RHS).

Insulation

Temperatures in a space environment are extreme for conventional electronic components which are usually sealed and enveloped by insulators like MLI (Multi-Layer Insulation) or aerogels. For example, a 10 layer MLI blanket has an emissivity of

0.01 and weighs $0.5kg/m^2$. Eqn. [4-14] gives heat loss. Significant (factor of 2) losses

$$P_i = \langle 5.67 \times 10^{-8} \cdot A_i \cdot Ei_{IR}(T_i^4 - T_{ie}^4)\rangle - \langle 1358 \cdot a_S \cdot Ei_{solar}\rangle \qquad \text{Eqn. [4-14]}$$

Ai: insul. area; Ei: emissivity; ir/solar: emission spectra
As: enclosure area incident to sun; Ti/ie: int/ext Temp,K

occur due to breaks in insulation enclosures. These are created by heat pipes, cabling, and rivets, and sealed using epoxy.

Radiators and Louvres

Eqn. [4-15] expresses radiator heat emission (in watts. Emissivity in IR and solar spectra

$$P_r = \langle 5.67 \times 10^{-8} \cdot 1 \cdot A_r \cdot Er_{IR}(T_r^4 - T_{re}^4)\rangle - \langle 1358 \cdot A_r \cdot Er_{solar} \cdot \cos(I)\rangle \qquad \text{Eqn. [4-15]}$$

Tr/re: radiat./space temp, K; I: solar incidence 1: louvre position (.1-.9)

can be different, enabling net heat outflow even under direct sunlight. Blackbody radiators have emissivity of 0.8 or better, while polished radiators like silver teflon have IR emissivity of 0.75-0.85 and solar emissivity of 0.2-0.3. The exact values are dependent on the coating and dust effects. Former lunar missions have experienced overheating due to dust effects. Louvres can be passively triggered by a mechanical thermocouple and reduce effective area of the radiator by a factor of 10 at low temperatures.

Heat Pipes

Heat pipes use convective wicking to achieve a constant temperature rise for a given

$$T_r = T_i + P_r / dh \qquad \text{Eqn. [4-16]}$$

dh: diode temp gradient, e.g.100 W/K if Ti > Tr; else 10 W/K

heat flow, irrespective of pipe length as described inEqn. [4-16].

Radioactive Heating Units (RHU)

These heat sources ranging from D-cell size, 2 W units to brick sized 250 W (1.5 kg) are used to heat spacecraft exposed to extended cold. Such sources are usually made of Pu238 (half life 78 years). Although, medical risks have been shown to be minimal, political and public unease follows its use.

Power

Power Management

Power flow is managed by a circuit that regulates the power source and switches power to the robot’s active components as needed. These devices include electronics, sensors, motor amplifiers and batteries. Typically such a circuit operates at 95-97% efficiency and.

Power Conversion

Power is converted from the main bus voltage to component needs using DC-DC converters that typically operate at 85% efficiency. Usually bus voltage is matched to high consumption components like motor amplifiers to minimize this loss.

Energy Storage

Energy is stored by primary or secondary cells. Primary cells have high power densities (e.g. AgZn 220 Whr/kg), while secondary cells have lower densities which drop with number of charge cycles (e.g. AgZn 170 Wh/kg @ 100 cycles, NiCd 60 Wh/kg @ 1000 cycles). For batteries used at different depths of discharge, Eqn. [4-17] estimates fraction of battery life will be used up (needs to be less than 1.0 for proper functioning). The L

$$\text{Life} = \text{FOS} \cdot \sum_{xi} \frac{\text{cycles}_{xi}}{L(xi/\text{capacity})} \qquad \text{Eqn. [4-17]}$$

xi: discharged energy; cycles: # of xi discharges; L(): battery life for givendepth of discharge

curves for each battery type are different, falling off exponentially with depth (%) of discharge. Batteries charge with efficiency of around 80% and impose severe temperature constraints due to their poor survivability, where the best technology is limited to [-40C,50C] survival range. Other technologies like regenerative fuel cells (higher density) and high density capacitors (high power density, temperature range) are viable options.

Solar Power

Photovoltaic cells made of Silicon (Si) or Gallium Arsenide (GaAs) are normally used to convert solar power to electric. Each cell is 4cmx4cm with a 0.7V output, requiring strings of 40 cells (2m long at 80% packing fraction) for a 28V output.

Table 4-18:

Cell	Efficiency	Cost, $/W	Drop off angle
Si	14	800	45
GaAs	18	1200	45

Either Gallium Arsenide or amorphous Silicon cells are possible candidates for the solar arrays. The base efficiency of these cells is 18%, but with degradation due to the launch vibrations, dust accumulation, and excessive temperature, as well as losses stemming from radiation degradation, the expected end of mission efficiency for the arrays would be 12%. Eqn. [4-18] determines the power output of a set of flat solar panels.

$$\text{Power} = \sum_{i \in \text{solarPanels}} A_i \cdot \text{Insolation} \cdot \eta \cdot \sin(\text{alpha}) \qquad \text{Eqn. [4-18]}$$

alpha: incidence angle of sun, power = 0, if alpha > drop off

Radiothermal Thermo-electric Converters

Standard space techniques for thermoelectric conversion operate at 7% efficiency. However, the more advanced AMTEC converters reach 20% (40 W/kg) system efficiency. AMTEC cells are integrated with isotope bricks (250W, 1.5kg GPHS) to generate the desired power. The cells need junction temperatures of 750 K and 550 K with > +/- 50 K margin. The excess heat generated by the RTG can warm electronics or be dissipated radiatively. AMTEC RTGs have low mass and high reliability. Existing prototypes have 55% thermal mass (of AMTEC and GPHS mass) to distribute, dissipate heat and isolate from other spacecraft components. Normal structural margins for mounting apply. For example a 450 W output RTG would weigh 60 kg.

Costs, Programmatics and Margins

The design incorporates preliminary costing models to enable rational optimization among technically viable options. Most costs are from component vendors or NASA databases. However, where such data is not available, cost of flight ready components is estimated at 10x commercial cost (S-Qual is 40x-100x). However, these costs assume traditional low cost small sat approaches and ignore possible benefits of low cost launches and cheap flight testing. NASA margins (Table 4-20:) are applied mass, power and costs are incorporated in the design. In addition overall management margins for power (20%), mass (20%), cost (20%) and cost contingency (15%) are applied.

Table 4-20: NASA mass and power margins

Readiness	Margin
Preliminary	30%
Analog Component	15%
Prototype	8%
Existing Component	5%

Chapter 5

Results

Synergy is a general robot design environment and can be applied to a range of robot problems. However, for the purposes of this research, a narrower focus of lunar rovers was chosen. Synergy is used to generate and understand designs for the four lunar exploration problems considered. During this process Synergy revealed many issues and insights formerly not evident to lunar designers. This chapter overviews these results, showing different ways of using this design environment.

Some insights into lunar rover design with Synergy

The following observations are made possible by Synergy. Although such performance is possible by human designers alone, it becomes increasingly improbable with increase in design complexity and the number of alternatives tracked:

- All four designs are generated by a single knowledge base, with changes in parameter values and configuration options effected by single line changes. Approximately 40 distinct design points are examined in a span of two days as the time to reconfigure and analyses system wide effects and "what if" trade-off queries is reduced from days to minutes. Reliability also increased due to minimization of human error. The results are designs which have higher consistency and rationality than before. Comparison with previous designs in Table 5-2: shows an improvement of 13-20% using the same information and methodology.

Table 5-2: Comparison of former preliminary designs with Synergy generated designs

	Ice Discovery	Edutainment	Earthrise
Synergy Design	258 kg	163 kg	96 kg
Former Design	329 kg	205 kg	110 kg
Improvement	71 kg, 21%	42 kg, 20%	14 kg, 13%

- A design flaw in sizing the ice discovery rover was automatically trapped by Synergy, while missed both by its designers and reviewers alike. This oversight occurred due to the assumption of a constant launch mass of the Delta II, while in reality, the allowable launch mass varies with the spacecraft center of gravity.
- The optimizer reveals that using a solar array is superior to a RTG (radio-thermal generator) for the lunar mining robot, contrary to all previous lunar designs. Although nuclear reactors have superior watts/kg performance, their minimum size (~ 10+ kW) is currently far above small explorers and work machines. Next generation space stirling engines may fill this gap.
- Analysis shows a moderate but distinct advantage in mining operations near the pole rather than the equator due to one axis pointing of both solar and communication antenna. Synergy increases the confidence level of such a comparison as the same information and knowledge base is used to generate both alternatives.
- The simulator and controller generator show that the Mars 98 lander is coincidentally suited for a lunar terminal descent as well. This simulation was automatically generated from the design representation with no lunar-rover-specific coding needed.
- Synergy is effective in debugging designs. Of every 10 problems uncovered, Synergy flagged more than 6, mostly as errors in unit inconsistency(4/10) and type mismatch(2/10). Of these, about half were due to forgetfulness (For example the Ice Discovery design team accidentally used lbs in place of lb-force causing erroneous data in the locomotion analyses) and the other half conceptual in nature. The other errors were discovered in the course of the trade studies as constraints were violated or through examination of relational dependencies generated.

Language and Spreadsheet

This section describes some of the lunar designs created using Synergy. Designs and knowledge of lunar rovers was comprehensively expressed in Synergy. These designs encode all mission requirement and technology components outlined in the previous Chapter. The components are parameterized and interconnected automatically to respond to changes and alternatives in requirements and technology. Parameters like mass, power, volume and cost are related to requirements or other components, all of which eventually relate back to the requirements. The spreadsheet automatically generates the interconnections, propagates the parametric relationships and generates budgets. These interconnections and dependency relationships were formerly hand derived and simplified. Synergy however allows and maintains the intricacy to yield more effective designs. Examples of the flow so generated are sketched in Figure 5-1 and Figure 5-2. Details of the lunar rover design exercise with Synergy follow. These details use trade studies that were performed with the optimization agent:.

Lunar Ice Discovery

The design of a lunar ice discovery rover began with the preliminary design from previous studies. The existing design fails on scrutiny using the lunar and planetary knowledge base given to Synergy. Several requirement and design relaxations are

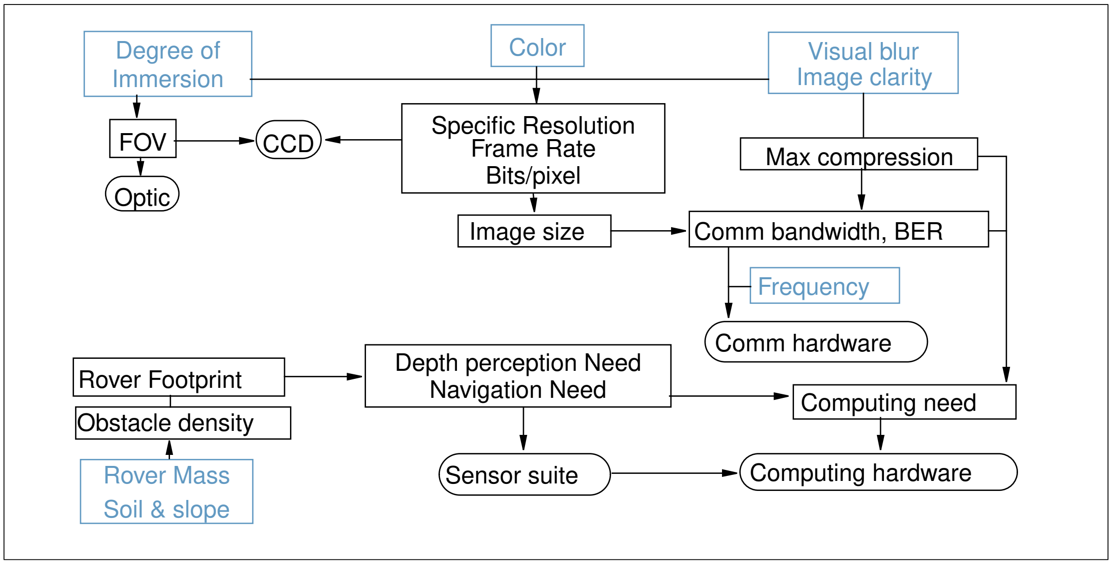

Figure 5-1: Approximate payload causal dependency structure generated in Synergy

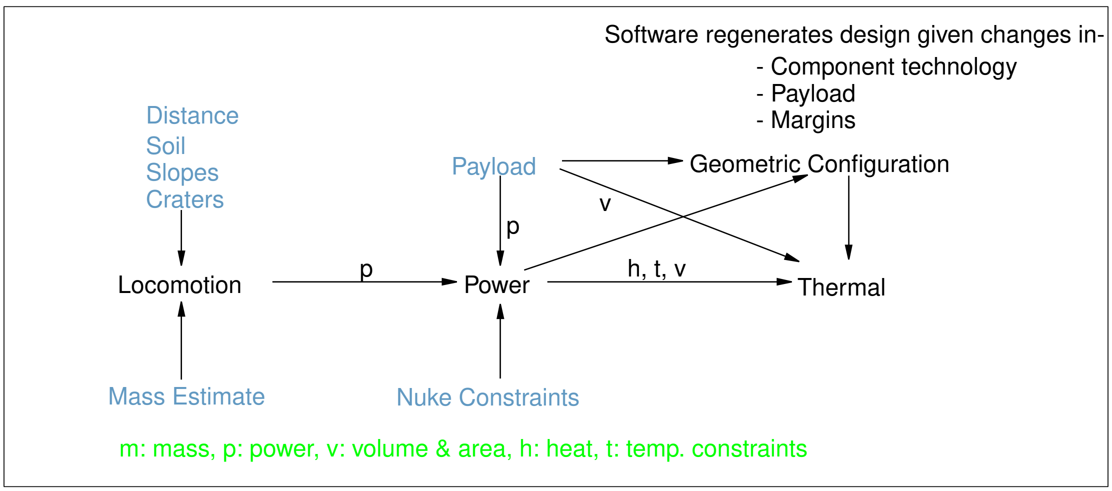

Figure 5-2: Flow of budgets affected by Synergy in generating lunar rover designs

undertaken using its optimization facilities to generate a satisficing design. This process outlined below is summarized in Table 5-2:

- Nominal design of discovery proposal. Although adequate at first glance, actual numbers from Delta II specialists revealed that maximum mass to TLI is 1315 kg. In addition, the communication gain is significantly below the desired gain margin. Hence the design is not viable due to insufficient margins.
- Since the ice discovery goal is one of science, the imaging frequency for visualization and navigation can be reduced from 1 Hz to 0.1 Hz. Although 0.1 Hz will cause operator fatigue and increase risk, low image rate is common in planetary exploration. This change satisfies the communication gain needs but does not significantly affect rover mass.
- A large mass fraction is due to battery designed for sorties into permanent polar shadows. The designed duration is 30 hours and can be reduced to 20 hrs and still have sufficient energy for deep sorties. The resulting spacecraft mass of 1143 kg is well below 1315 kg and has an apparent mass margin of 40% over the lander-rover mass.

However, synergy flags a “low margin error”. This is due to the available spacecraft mass being 1216 kg due to high CG.

- A final relaxation is done by optimizing the communication link and adapting the transmission power to match the desired gain budget. The resulting spacecraft mass of 1118 kg has a mass margin of 0.25, above the desired 20%.

The design process repeatedly hit the payload diameter constraint due to large wheel sizes. These wheels are so sized to have sufficient flotation (< 6kpa ground pressure) as recommended by lunar surface analyses. Hence, volume is a limiting resource for lunar rovers, making launchers like Ariane more attractive.

Table 5-2: Sequence of optimizations affected by synergy and relaxation of requirements

Traded Quantity	Old Value	New Value	Major Quantity Affected	Delta Change, New Value	Rover & spacecraft Mass
Baseline Design					314 kg, 1311 kg
Imaging Speed	1 Hz	0.1 Hz	Com gain	6.7 dB, 0.8 dB	312 kg, 1302 kg
Cold sortie hrs	30 hrs	20 hrs	Battery	-18 kg, 30 kg	265 kg, 1143 kg
Comm Xmit power	15 W, fixed	12 W, var	Com power	-10 W, 40 W	258 kg, 1118 kg

Edutainment Traverse

Lunar edutainment rover design has been the subject of intensive study involving about 30 designers and a total of 6 man years of effort. The knowledge acquired during this effort was used by synergy to improve on designs generated by this team by 20%. In addition, several trade-off studies and qualitative analyses are rationalized using synergy and presented below and summarized in Table 5-4:

- Processor comparison: This trade is fuelled by two observations: most of robot processing requirements can be fulfilled by vector computing of the kind that DSPs deliver, and most of the rover processing is non-critical or immune to upsets (rand bit flips). Hence using high-end radiation tolerant computing is vastly better than radiation immune “space qualified” processors. As the data point in the table suggests using even highly advanced space qualified processors over current processors like the PowerPC results in significant increase in rover mass (177 kg). This justifies the excess cost accrued to flight ready such processors.
- Use of solar power source vs. RTG (Plutonium power) causes vast hibernation energy needs which have to be met by a Radioactive Heating Units (RHU) or a battery pack. Using a solar array/RHU combination results in a dramatic reduction in mass, while raising issues of duty cycle and reliability.
- Safeguarding needs of shadow extraction call for a one hour battery reserve in the Solar/RHU scenario. This results in more than halving the mass advantage over a RTG power source.
- The above trade used Super-NiCd batteries, using other batteries like NiH2 with higher energy density though advantageous in mass results in impractical radiator size and

more than doubles the required RHU mass, which flags constraints of geometry and legal certification.

- Considerations of revulsion to use of isotopes call for examining the use of batteries to provide night survival heat. This attempt to do away with RHUs examines the use of LiIon with densities 3x conventional NiCd. However, the resulting rover design approaches that of a truck is impractical for space.

In summary, battery temperature limits have a significant effect on the rover when using solar arrays, state-of-the-art CPUs are needed to enable the high quality imagery and the task is feasible.

Table 5-4: Design trade-off for the lunar edutainment rover

Traded Components	Old Config	New Config	Major Quantity Affected	Delta Change	Delta Rover & spacecraft Mass
Processor	R1000/DSP	PowerPC6xx	Power	177 W	117 kg, 405 kg
Power/Heat Source	RTG/RTG	Sol/RHU	Power sys	-20 kg	-46 kg, -159 kg
Power/Heat Source	RTG/RTG	Sol-SNiCd/RHU	Battery	8 kg	-21 kg, -76 kg
Power/Heat Source	RTG/RTG	Sol-NiH2/RHU	Radiator	3.4 m^2	-5 kg, -16 kg
Power/Heat Source	RTG/RTG	Sol/Li-Ion	Battery	1272 kg	3354 kg, NA
Comm transmitter	Phased array	parabolic dish	comm M,P	4 kg, 4 W	16 kg, 53 kg

In situ (ISRU) mining rover

The design of a lunar strip mining ISRU (In Situ Resource Utilization) rover is different from others due to the 100 kg sled that it pulls on relatively smooth terrain. The sled places significant power demands on the rover and analysis reveals non-intuitive design choices as summarized in Table 5-6:

- RTG power sources are unsuitable for high power ISRU applications due to lower watt/kg than solar power sources.
- A first examination of a polar operation results in a moderate increase in rover mass due to the annular comm link being unsuitable for the large downlink of the ISRU robot.
- However, optimization reveals that the pointed parabolic dish formerly designed for equatorial operations is ideal for polar links and results in the best ISRU configuration. Even the RHU (isotope) mass is lower by 100gm due to smaller sun angles causing smaller radiator size resulting in lower heat leakage at night. The optimized link resulted in an unusual 5 cm dish with a half-power beam width of 40^{o}.

Table 5-6: Progression of ISRU rover design using quantitative impact of choices

Configuration	S/C mass	Rover mass	Battery mass	Power system mass	Radiator area	RHU mass
Nominal: solar, 0^{o} lat	850 kg	181 kg	09.6 kg	36.4 kg	1.23 m^2	0.77 kg
RTG	1140 kg	265 kg	00.0 kg	76.0 kg	0.66 m^2	0.00 kg

Table 5-6: Progression of ISRU rover design using quantitative impact of choices

Configuration	S/C mass	Rover mass	Battery mass	Power system mass	Radiator area	RHU mass
80^o latitude	926 kg	203 kg	11.3 kg	35.0 kg	0.44 m^2	0.64 kg
80^o lat, pointed comm	839 kg	177 kg	10.1 kg	31.6 kg	0.39 m^2	0.62 kg

Earthrise event

The design of this rover to capture video of Earthrise above the Moon at Jan 1 2001, embodies lessons learned from the previous analyses and insights. No significant trade resulted from varying parameters like imaging frequency or other payload parameters. This is due to significant fixed-weight of sensors for housekeeping, navigation and safeguarding. For an excursion of this nature, many of these sensors are likely to be superfluous. The reduction of these ought to be directed by requirements and aided by sensitivity analysis as indicated in Table 5-8:. Sensitivity analyses for other lunar rover designs is also provided for comparison. However, these perturbations are nonlinear and are only an indication.

Table 5-8: Rover sensitivity to perturbation of payload parameters (mass, power enclosed)

Perturbation	Ice Discovery	Edutainment	ISRU	Earthrise
Mass, 1 kg	2.89 kg, 0.94 W	3.02 kg, 1.74 W	2.80 kg, 1.61 W	2.61 kg, 0.42 W
Power, 1 W	0.61 kg, 1.66 W	0.56 kg, 1.78 W	0.31 kg, 1.65 W	0.30 kg, 1.51 W
Communication, 1 kbps	0.52 kg, 1.05 W	0.03 kg, 0.06 W	0.00 kg, 0.17 W	0.01 kg, 0.04 W
component max T, -1^o C	0.02 kg, 0.01 W	0.00 kg, 0.00 W	0.30 kg, 0.30 W	0.40 kg, 0.06 W
component min T, $+1^o$ C	0.06 kg, 0.00 W	0.07 kg, 0.04 W	0.01 kg, 0.12 W	0.00 kg, 0.00 W
Drive efficiency, 1.0/1.1	3.69 kg, 10.5 W	3.38 kg, 11.5 W	8.53 kg, 50.9 W	0.26 kg, 1.76 W

Overview of Lunar Designs

Figure 5-4 and Table 5-8: give a comparative overview of the four design points considered here. These numbers generated by Synergy help designers understand the various relationships and trade-offs from which further directions in design become possible. The cost numbers listed are from estimated component costs, where many of

Table 5-10: Some of the factors of safety and margin embedded in the designs

Factor	Value	Factor	Value
CG center offset	0.1 L, 0.1 W	CPU overhead	3.0
Locom power margin	0.05	DSP overhead	0.63
electronics paking frac	0.4	Comm gain margin	5 dB
heating pow overhead	1.0	terrain sens coverage	2.0
power margin	0.2	cost margin	0.15 + 0.2
Solar power margin	2.0	Insulation overhead	1.0
Radiator absorptivity	+0.1	wheel load overhead	0.1

them may be overestimates for lack of current information. The reliability calculated seems low (0.3 - 0.4) and this is primarily due to the large number of parallel connectors for sensor data transfer. Currently all sensors are assumed non-redundant, 0.99 reliable and critical to operation. This warns the designer to carefully choose sensors and define alternate operation scenarios as one or more sensors is likely to fail during operation. Anecdotally, a edutainment rover analog (Nomad) showed such failure rates during integration and operation.

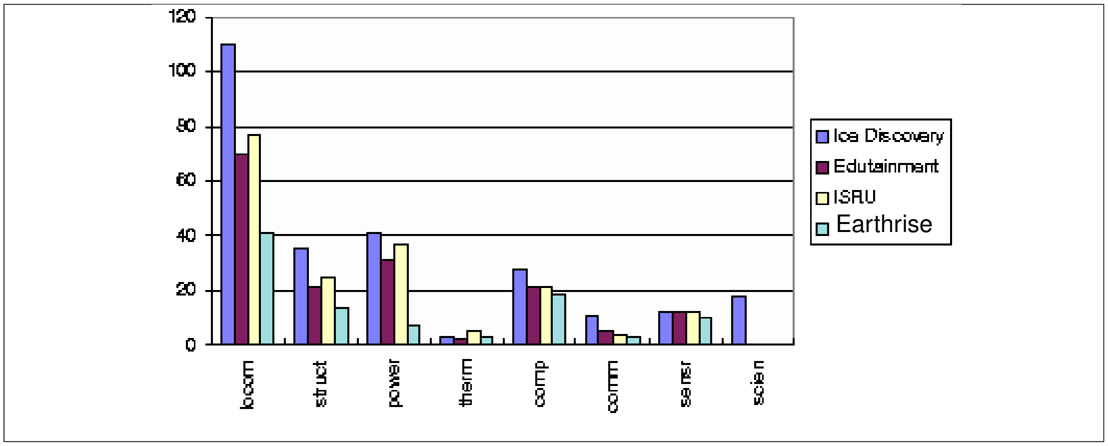

Figure 5-3: Comparative breakdown of mass budget

Simulation

Synergy's automatic simulation capability in conjunction with the controller generation capability generated landing simulation of the Mars 98 lander. The significance of this simulation is the dynamic validation of the Mars 98 lander for lunar landing, which can result in lower cost, risk and lead time to a lunar landing. A simulation such as this, goes beyond formulae and applies the laws of physics to the given design and generates a bottom up performance prediction. Such a validation is often needed for designs which are untried, vital or risky.

The largest risk in the lunar rover scenario has been the lack of low mass landers that could be used with variable payloads and launchers. The considered Mars 98 lander uses a dozen 60 lbf N2H2 thrusters for final descent. For lunar landing, a configuration with half the number of thrusters was chosen to lower mass. The simulation begins with the lander at 12.5 km altitude and a velocity of 50 m/s, the worst case accounting for uncertainties in the tracking error from the Deep Space Network and the solid rocket motor burn. The simulator shows that a soft landing is accomplished with 60.4 of fuel expended, leaving 26.5 kg for hover and 13.4 kg for margin. This 22% of the fuel expended, meeting the desired ~20% and validating the design equations. In addition, the combined dynamics of the rocket, gravity, CG/inertia, jet positions and the controller are validated as a coherent unit. Figure 5-6 shows state histories of the generated simulation.

Quantity	Units	Ice Discovery	Edu-tainment	ISRU	Earthrise
Max Launch mass	kg	1219	1260	1267	1315
Spacecraft mass	kg	1118	790	850	557
Available Mass margin	%	0.25	1.74	1.41	4.42
SRM solid rocket mass	kg	706	519	553	385
lander mass	kg	151	107	115	75
lander fuel mass	kg	100	67	74	45
Rover mass	kg	257.5	163.4	180.6	95.9
Locomotion mass	kg	110.4	70.0	77.4	41.1
structure mass	kg	35.6	21.4	24.6	13.3
powersystem mass	kg	41.0	31.0	36.4	7.2
thermal mass	kg	2.8	2.3	5.1	3.4
computronics mass	kg	27.1	21.6	21.3	18.0
communication mass	kg	10.8	4.9	3.5	3.0
sensors mass	kg	12.2	12.2	12.2	10.0
science mass	kg	17.5	0.0	0.0	0.0
Power margin	%	0.2	0.2	0.2	0.2
rover power	W	273.2	250.0	573.0	131.9
locomotion power	W	74.1	75.3	368.2	13.8
powersystem power	W	29.9	26.2	30.7	17.7
computronics power	W	33.8	31.1	26.2	32.4
communication power	W	40.4	36.3	22.3	16.9
sensors power	W	29.5	39.4	31	29
science power	W	15	0	0	0.0
rover length	m	2.17	2.16	1.50	0.99
rover width	m	1.81	1.16	1.25	0.82
wheel diameter	m	0.60	0.60	0.42	0.27
wheel width	m	0.60	0.39	0.42	0.27
spacecraft CG	m	1.0	0.98	0.97	0.88
Rover CG	m	0.8	0.66	0.56	0.38
electronics box size	m x m x m	0.42	0.39	0.52	0.36
radiator area	sq m	0.28	0.43	1.23	1.09
No of sensors	-	31	23	23	22
downlink data rate	kbps	87	768	370	432
computing power need	Mips	34	751	82	420
computer boards	-	5	3	4	4
safeguarding latency	msec	667	767	130	70
Worst case reliability	-	0.29	0.40	0.40	0.41
spacecraft cost	$ Milliions	214	211	214	203

Figure 5-4: Vital statistics and budgets of lunar rover designs

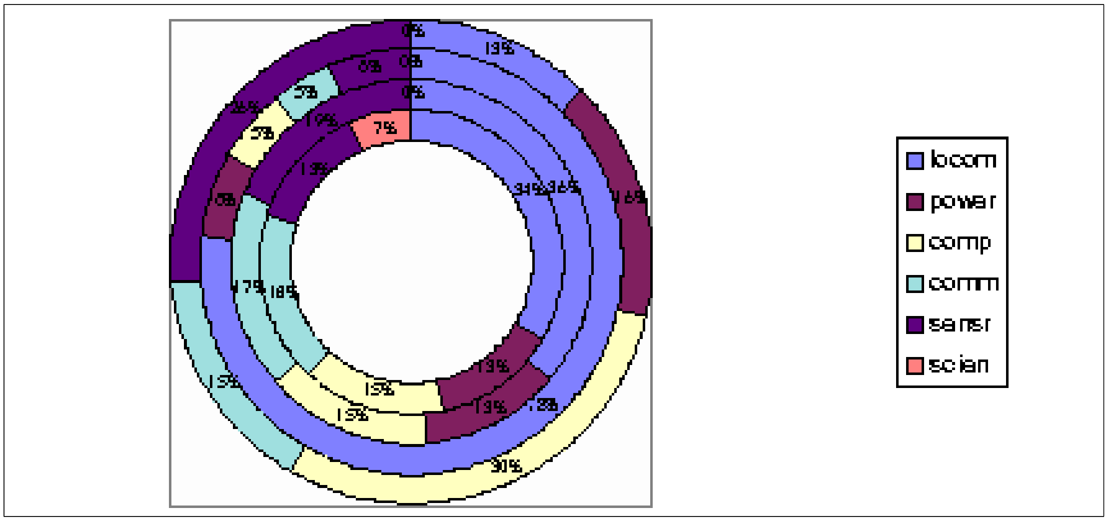

Figure 5-5: Comparison of fractional power usage, inner most is Ice Discovery, outermost is Earthrise

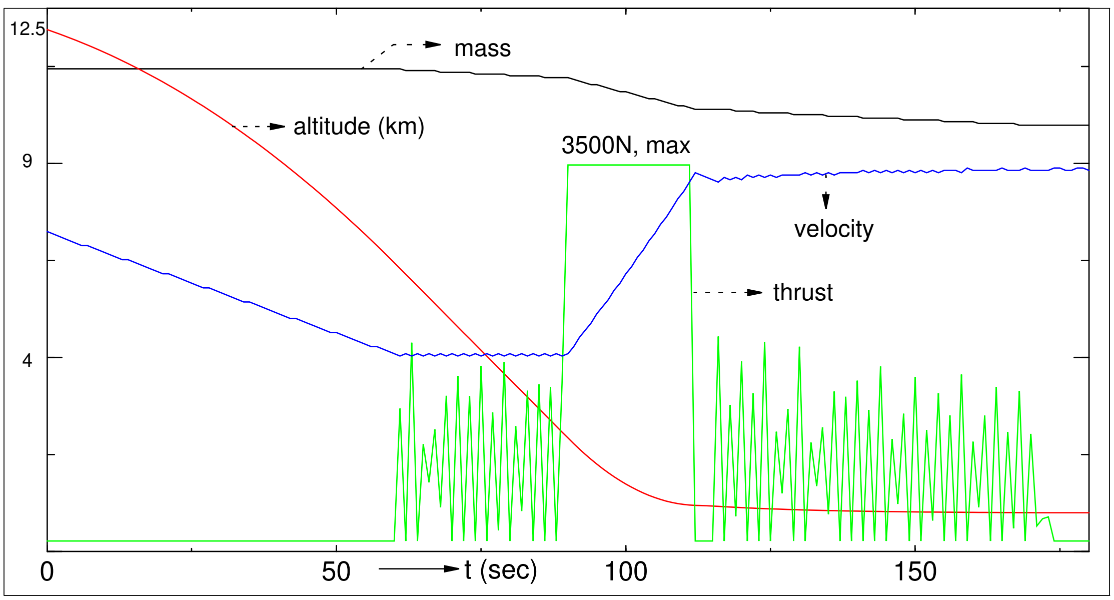

Figure 5-6: Profile or multidimensional landing simulation generated by Synergy

Optimization

Lunar rover design used all three optimization approaches to generate optimal configurations and size the designs.

Rule Based Optimization

Choices like RTG vs. Solar, RHU vs. battery have been studied carefully and can be condensed to simple rules. These rules triggered the following changes based on mission needs. The alternates can be examined to validate these rules. Some of the rule switching that synergy executed while generating the lunar designs are:

- Compression algorithms switch from static to motion flow as frame rates increase or maximum speeds fall.
- Radiator louvres switch from open to close during day-night transition analysis.

- RHU and battery automatically sized as a function of the power source when night power is insufficient (for e.g. when using solar power).
- Night heating and power mechanisms enabled only when the mission calls for night operations and/or survival.

Root Finding

Root finding is used to ensure that design constraints are met while minimizing the use of resources. This feature proved essential to enable rapid analysis of design/ technology changes, answering "what-if" questions and doing perturbation analysis. For example, a requirement change like frame-rate of imaging sensor trigger changes in computing and communication budgets. This leads to a recalculation of the number boards and power needs. Simultaneously the communication hardware and power is also resized. These effects propagate to the electronics box (hence thermal) and power module. All these changes affect sizing of locomotion module. Rover mass changes propagate to lander fuel mass and solid rocket motor sizing. This may trigger rules to switch launchers. Decides that RHU not needed as the battery designed for cold sortie operation is sufficient to supply night heat for the 3 "day" (24 hrs) night at the target mission location and time.

- Sizes the comm link to meet the gain requirements exactly, avoiding any excess margins and consequent oversizing of the rover.
- Similar sizing behavior is triggered by budget trackers for temperature (which determine temperature limits for the electronic box based on its composition and subsequently size the thermal control), mass, power, energy, volume and computing.

Genetic Search

The cardinality of many choices exceeds that of the design constraints, allowing a space of multiple viable solutions. These include choosing a combination of antenna size and power to meet a constant gain need, or sizing wheel width and diameter to meet flotation requirements. This problem is handled by search techniques like the genetic algorithm implemented in Synergy. The problem of sizing the phased array was considered illustrative as the gain calculation from number of antenna elements and element power is highly non-linear and has no "best" recipe. In addition, this problem contains a continuous and a discrete value variable, typical of real design problems. The genetic search was used to evolve a heuristic which was eventually used to size the designs described above. The objective function used is Max(rover-mass * (1+||invalid constraints||)). The genetic search uses a crude heuristic to initialize the search with a population of 20. Table 5-4: summarizes the results of a typical run until convergence sets in. These runs repeatedly maximized the power of the array indicating an advantage in maximizing antenna power (25 mW and 154 elements) for this particular design. These simple searches can be extended to complex problems are they are presented. A drawback is the high computing requirement for handling large problems. This is expected to alleviate as the software is stabilized and optimized.

Table 5-4: Genetic search to find the optimal phased array configuration

Generation	Rover Mass
1	186 kg
2	177
3	176
4	165
5	166

Generation of Control Software

Control software for lunar landing was generated to support the simulation described above. An objective function of Max(landing mass) was used with a velocity constraint of < 1 m/s at a landing of 0.25 m - 0.0 m. The controller is represented as a an interconnected net of input sensor pose estimates and output thruster firing commands. A set of weights and biases forms the state vector that is evolved using the genetic algorithm. The genetic algorithm manipulates and evolves the state vector, guided by the simulator. For each candidate state vector (chromosome) generated by the search, a complete landing simulation is performed and the resulting objective function value (fitness) is used to select and mutate the next generation. The search ends when the landing is sufficiently soft and the best design remains constant over several generations.

Chapter 6

Conclusion

This thesis advances the state-of-the-art in robot design environments and demonstrates it by applying Synergy to lunar rovers, generating comprehensive configurations, optimized designs, and simulations with auto-generated control software. During the course of this thesis, many insights are distinguished leading to possible research directions. True success of this thesis, however lies in its ongoing development and use for robot design, and though such environments will be inevitably developed in the future, it is hoped that Synergy motivates and influences the nature and effectiveness of such developments.

Contribution

Synergy's contribution is the formulation and demonstration of the viability and power of a comprehensive robot design environment. This contribution can be distinguished as follows:

- First general grammar and implementation of a language to comprehensively encode designs of general robots. A spreadsheet implementation of this language which allows automatic generation of designs from knowledge and constraints of the technology and the requirements. This spreadsheet also implements a framework that supports agents for simulation, optimization and software generation.
- First robot design software implementation and methodology to automate rational sizing, analysis, simulation, optimization and software generation, in a unified environment using a single representation of the robot.

Accomplishments

In applying Synergy to design lunar rovers, several results are achieved that were not possible with existing design environments:

- Showed that the Mars 98 lander is by coincidence suitable as a lunar lander by automatically generating the controller and simulation of the 6-DOF landing dynamics, sensing, propulsion and fuel depletion
- Encoded lunar rover design knowledge in Synergy and used to auto-generate several designs. Synergy ensured that these designs are consistent and met all the defined constraints.
- Synergy used these designs to automatically generated simulations, optimized designs and generated control software.
- The turnaround time for creating a new lunar rover (preliminary) design reduced from weeks to days. The turnaround time for modifying or optimizing an existing lunar rover design given new information reduced from weeks to hours. Similar reduction in time visible for simulation or coding control software.

Possible Applications and Advantages

Synergy in its current incarnation can be immediately used to improve robot design in the following ways:

- Designers are relieved from having to mentally juggle all the technology involved in creating the robot. Instead they can transfer that expertise to the spreadsheet which melds and adapts that knowledge to the specific design. This property extends life of the knowledge across design cycles and allows its effective transfer and reuse.
- Designs represented in Synergy are relieved from having to custom code first simulations or optimizers, reducing resource usage and expertise needed.
- Designs and technology encoded in synergy can be easily transferred to other designs and upgraded ongoingly with new information, reducing the effort in understanding, reconfiguring existing knowledge for new designs, or in incorporating the effect of new technology.

Insights and Lessons Learned

Several angles of a robot design environment became clear during the research. Although these insights were not investigated deeply and followed up with implementation, it is important to consider them when contemplating Synergy's use and development:

- Synergy currently suffers from a lack of geometric understanding of the robot design, resulting in the designer hand coding this knowledge when needed. This drawback also prevents automated analysis of optical issues like visibility, sensor configuration, solar power, thermal configuration and communication.
- The above observation can be generalized to say that upgrading knowledge or adding simulation agents reduces the burden of knowledge and analysis from the designer,

making the designs simpler to represent and encode, while increasing the confidence and realism of the designs.

- This thesis used components to represent robots for reasons of modularity and reonfigurability. The very same approach also maximizes the reuse and transfer of knowledge. The success of a robot design environment will eventually rest on availability of state-of-the-art information on technology and components. Hence, it is worth noting the convergence of internet and data mining technology in this context. The “Finite Component” method for design and analysis is vital in the long run.
- Computing efficiency of the design software is essential as robot and component complexity grow with computing ability. Hence, unless the software and algorithms are efficient, the design environment will always be slow and unwieldy, irrespective of growth in computing power. It is notable that the advances in component and robot technology are governed and constrained partly by the computing power available for analysis and simulation. For example, even though the capacity of chips is doubling every 18 months, the time taken to simulate one remains relatively constant (~1 day).

Possible Directions

This investigation provides the motivation and direction for several advances that Synergy or any other robot design environment will benefit from. The identified advances are outlined below in an approximate order of decreasing priority. The research and effort content of these directions varies and will play a role in pragmatic implementation priority:

- Synergy purpose is to enable better robots. It will hence benefit significantly from application to a complete concept-prototype-deployment cycle of a real-world robot such as a lunar rover. This recommendation arises from the focus and direction derived from its current application to a preliminary design.
- Techniques to improve computational efficiency like fuzzy encoding or memory based learning will need to be investigated. Many such techniques exist in single dimensional simulators and would need to be ported to Synergy’s agents. This will allow multi-resolution modeling and analysis critical to using a common representation to search millions of designs cheaply while analyzing a few in increasing detail.
- A geometric engine to automatically derive component properties like volume, mass, CG, interference, relative distances and surface areas. This foundation will allow synthesis of other knowledge like automated thermal modeling, wireless link simulation, optical simulations and automated sizing of connectors, heat pipes, structural members and other physical links.
- An effective user interface is required to make Synergy usable. Such an interface would allow graphical assembly of components, and analysis of the resulting dependency networks and resource flow. Additional visual feedback of simulations in their native form is useful to reduce designer fatigue. “Native form” implies representing data in a familiar form - example electronics as a circuit, color coding force levels, dials and meters for torques and temperatures.

- Improvements in simulation agents are needed to enable detailed design in Synergy. These include a using SPICE as the electronics agent, expanding the dynamics agent to simulate terra-mechanics.
- The optimization agent can be improved by creating operators that can use knowledge encoded in the design to better direct the search. Other operators for specific uses like scheduling and geometric configuration will be useful. Beyond traditional optimization, techniques of artificial knowledge manipulation, innovation and creativity could be used to effectively generate design alternatives.
- The language needs to better represent stochastic variables and relationships in order to increase its usefulness. Other related modifications like encoding and propagating the granularity and uncertainty of parameters will allow robust designs and more effective simulation and optimization.
- The simulation needs to be expanded to interface with software engineering tools and real-time communication architectures so that engineers can use Synergy as a virtual robot prototype to develop, test and integrate robot software. Other useful analysis includes real-time software modeling through Markov models and automated calculation of computing needs and frequencies.
- Many language features need to be refined including better automated elimination of circular dependencies so that the resulting design iterations are guaranteed to be stable. Further graphical feedback of such dependencies and modes of elimination is needed for the designer to understand the propagation of design changes.
- Industry standards (e.g. IEEE standards) for representing component knowledge need to be created and encouraged (as either data or pre-compiled objects to protect proprietary information) for dramatic improvements in robot design effectiveness. Further, mechanisms to access this information including universal IDs and web (e.g. XML) language standards would need to be established in parallel. This recommendation stems from observing that components constitute a bulk of the design space and that existing component information is sparse and insufficient for design of mechatronic artifacts.

Epilogue

This thesis represents a first robot design environment that creates a comprehensive design language and establishes the viability of automating design sizing, analysis, simulation, optimization and control software generation. There are several areas of research that open up on looking beyond the rudiments and focusing on metrics of fidelity, efficiency, comprehension and ease of use.

The vision of Synergy is a design environment that self generates, verifies robot designs and software from simplistic configurations and specification of the requirements. Such an environment would have the capability to create a robot in a month with help from application experts, technical artisans and one (or no) roboticist, clearing the path for an explosion in robotic applications.

In the context of this dream, Synergy is primitive and may be compared to the first drafting (CAD) software. The scope of this work is vast and ongoing. Although a large part of this scope is implementation and would be commercially driven in the future, the initial demand, motivation and direction would need to be fuelled by the research community.

Glossary

The following vocabulary is used in this thesis. They appear in the text, or are of general significance. Words found below in *this font (glossary type)* have their own entry.

Agent	A "memory-less" software object with knowledge pertinent to a specific design aspect like a *dimension* of *robot simulation*, *optimization*, or *control* software generation.
Accuracy	The quantitative difference between a generated *simulation* versus a referential reality, model, or *simulation*.
Artifact	Any artificially created physical or software device.
Component	A functional and structural unit of the *robot* usually representing a physical part or a software module.
Comprehensive	Referring to the set of *robot dimensions* that define the *robot's form*, so that its *functionality* can derived solely from these *dimensions,* the *environment* and a knowledge of physics.
Configuration	Assembly of technology and *components* defining gross geometry, interrelations, and functionality of the *robot*.
Consistent	Refers to *robot designs* where the *component elements* are sized to match each other, so that all the *constraints* and *relationships* are satisfied, with minimal or no excess usage of resources like mass and power.
Constraint	A restriction imposed on the *robot, sub-system* or a *component* to ensure that it performs as designed or engineered.
Control	In the context of robot software, refers to the mapping of sensory and goal input to actuator commands to achieve a particular behavior like path tracking.

Design (1) A quantitative and explicit description of robot configuration with sufficient detail to enable quantitative function/performance analysis and prototyping.

Design (2) The art of *engineering* a (*robotic*) *artifact* from concept to prototype, from defined *objectives.*

Dimension A facet of the complete *robot* model that usually segregates it from the rest of the model due to a particular distinction in governing laws. Examples include mechanical dynamic behavior, electronic behavior and thermal flow.

Dynamic Referring to the transient nature of a *robot/ simulation* or one of its *dimensions*, where the behavior is a function of time, the *robot*'s *state* and its changing *environment*.

Element *Component* property that defines it's parameters, state, constraints and relationships.

Engineering The practical application of science to create artifacts using existing artifacts. Often refers to the constraints and requirements imposed by individual components and their effect on the entire design.

Environment The external conditions and constraints that affect *functionality* of the *robot* or its *dimensions*

Fidelity The accuracy of a model/simulation with respect to the absolute reference of physical reality.

Form The morphological and anatomical make-up of an *artifact* (*robot*).

Function The space of behaviors that an *artifact/robot* is designed to exhibit.

Mechatronics The science of understanding and creating artifacts by fusing electro-mechanisms, computing and software.

Mission Aggregation of desired *robot tasks* that constitute successful *robot* operation.

Model A mathematical representation of physical/cognitive behavior of a *mechatronic component/system*.

Module A collection of component(s) with a common functional purpose.

Objective A quantified and desired work/data that the *robot* is expected to generate.

Optimization The task of searching through the *design* space of a *robot* to find the best *robot design* in reference to given *objectives*. *Design* space includes the components and their parameters.

Parameter A symbolic/quantitative variable that defines/sizes a particular aspect of the *robot design* and ultimately influences its *form* and *function.*

Performance Quantified metrics of *robot function* in context of desired *objective/mission*.

Relationship An *element* of a *component/system* that defines a facet of its interaction with respect to another component.

Robot	A *mechatronic artifact* that perceives and understands the world in the context of its *tasks/objectives* and generates physical work to accomplish them.
Simulation	A quantitative evaluation/prediction of a *mechatronic artifact*'s behavior using numerical methods.
State	The set of parameters (like pose, currents, temperatures) that define the condition of a *robot* or its *dimensions*, such that future behavior can be predicted using this *state* and the *environment*.
Static	Refers to *robot* analysis or model that is invariant with time, usually intended to capture functionality succinctly for ease of understanding, computing or communication to others.
(Sub)-System	A set of inter-related *components* that constitute a *functional* mechatronic entity.
Task	*Robot* behavior defined as an *objective*.

References

Design Environment

[Hobbs95] Hobbs, J., *Viper: A software environment for designing lunar rovers,* internal report, Matra-Marconi, London, 1995

[Farritor96] Farritor,S., Dubowsky,S.,"*On the design of rapidly deployable field robotic systems,*" Proceedings, ASME design engineering conference, Aug 1996, CA

[Stewart93] Stewart,D.B., Volpe,R.A., Khosla,P.K., "*Design of Dynamically Reconfigurable Real-time Software using Port-Based Objects,*" Robotics Institute, Carnegie Mellon University, CMU-RI-TR-93-11

[Gertz93] Gertz,M.W., Khosla,.K.,"*Onika: A Multilevel Human-Machine Interface for Real-Time Sensor-Based Robotics Systems,*" Proceedings, SPACE 1994

[Morrow95] Morrow,J.D., Khosla,P.K., "Sensorimotor skills for Robotic assembly skills," IEEE conference on Robotics and Automation, ICRA, 1995

[Taylor92] Taylor,M., *Coverdale on Management,* Butterworth-Heinemann, Oxford, England, 1992

[Kelly95] Kelly,A., "An Intelligent Predictive Control Approach to the High Speed Cross Country Autonomous Navigation Problem", Ph.D. Thesis, Robotics Institute, CMU, Pittsburgh, May 1995

[Coulter94] Coulter, R.C., "A Systems Engineering Approach to Electro-Mechanical Reconfiguration for High Mobility Autonomy," Technical Report, Robotics Institute, Carnegie Mellon University, CMU-RI-TR-94-27, 1994

[Apostol97] Apostolopoulos,D., "*Analytic Configuration of Wheeled Robotic Locomotion,*" Ph.D. Thesis, 1997, Robotics Institute, Carnegie Mellon

[Kannapan87] Kannapan,S.M., Marshek,K.M., "A Parametric Approach to Machine and Machine Element Design," Technical Report, Mechanical Systems and Design, University of Texas at Austin, 198, 1987

[Kannapan89] Kannapan,S.M., Marshek,K.M., "An Algebraic and Predicate Logic Approach to Representation and Reasoning in Machine Design," Technical Report, Mechanical Systems and Design, University of Texas at Austin, 212, 1989

[JPL97] Jet Propulsion Laboratory, *"Project Design Center,"* 1997, pdc.jpl.nasa.gov

[Excel95] Microsoft Corporation, *"Excel for Windows 95,"* Version 7.0, www.microsoft.com/products/business.htm,1995

[MSProject95] Microsoft Corporation, *"Microsoft Project for Windows 95,"* Version 4.1, www.microsoft.com/products/business.htm, 1995

[Wolfram97] Wolfram Research Inc., *"Mathematica 3.0 Professional Version,"* www.wri.com, 1997

[Prata92] Prata,S., *The Waite group's C++ Primer Plus*, Galgotia Publications, New Delhi, India, 1992

VLSI design tools

[Boddu95] Boddu,J., *"UltraSparc development cycle,"* internal report, Sun Microsystems, Mountain View, 1995

[SES94] Scientific Engineering Solutions Inc., *Modules for functional block simulation,* San Jose, 1994

[SUN94] Sun Microsystems, *SAS (Sparc Architectural Simulator),* Mountain View, 1994

[CadLang94] Cadence Design Systems Inc., *Verilog-XL, language for VLSI design,* San Jose, 1994

[CadSim94] Cadence Design Systems Inc., *Verilog static analysis and dynamic simulators,* San Jose, 1994

[Synopsis94] Synopsis Inc., *Synthesizer for VLSI net-list generation,* San Jose, 1994

[ViewLogic94] ViewLogic Inc., *Motive - timing analysis tool,* San Jose, 1994

[TI94] Texas Instruments, *Chip manufacturing support & software (static Design Rule Checking),* Houston, 1994

[Sunrise94] Sunrise Inc., *ATPG (Automatic Test Pattern Generator),* San Jose, 1994

Language to Represent Robot Design

[PTC97] Parametric Technology Corporation, "Pro/ENGINEER Release 18.0," www.ptc.com, 1997

[SDRC97] Structural Dynamics Research Corporation, *"I-DEAS Master Series 5,"* 1997, www.sdrc.com

[Dassault97] Dassault Systems, "CATIA CADAM," Version 4, www.catia.com, 1997

[Autodesk97] Autodesk Inc., "AutoCAD Release 14," www.autodesk.com, 1997

[PADS97] "PowerPCB - PCB Layout and autorouting software", www.pads.com, 1997

[SubramaniE93] Subrahmanian,E., "Shared Memory in Design: Theory and Practice," Technical Report, Carnegie Mellon University, EDRC 05-72-93, 1993

[Schaefer84] Schaefer,M.J., Rehak,D.R., Fenves,S.J., "Introduction to Relational Databases using Structural Engineering examples," Journal of Technical topics in Civil Engineering, ASCE, volume 110, no. 1, pp. 1-18, May 1984

[Craig88] Craig,J.J., “Issues in the design of Off-line programming systems”, International symposium of Robotics Research, MIT Press, 1988

[Rinderle87] Rinderle,J.R., “Function and Form Relationships: A Basis for Preliminary Design,” NSF workshop on Design Process, Waldron, 1987

[Cutkosky89] Cutkosky,M.R., Brown,D.R., Tenenbaum,J.M., “Extending Concurrent Product and Process Design toward Earlier Design Stages,” Concurrent Product and Process design, ASME, DE, volume 21, PED-vol. 36, 1989

[Hardwick89] Hardwick,M., Spooner,D.L., “The ROSE Data Manager: Using Object Technology to Support Interactive Engineering Applications,” IEEE transactions on knowledge and data engineering, Volume 1, Number 2, June, 1989

[Serrano87] Serrano,D., Gossard,D., “Constraint Management in Conceptual Design,” 2nd International Conference of the applications of Artificial Intelligence in Engineering, Boston, August, 1987

[Pressman] Pressman,R.S., *Software Engineering, a practitioner’s approach*, 3rd Edition, McGraw-Hill International Editions, New York

[DeMarco79] DeMarco,T., *Structured Analysis and System Specification,* Prentice Hall, 1979

[Coad86] Coad,P., Yourdon,E., *Object-Oriented Analysis,* Addision-Wesley, 1986

[Dahl72] Dahl,O., Dijkstra,E., Hoare,C., *Structured Programming,* Academic Press, 1972

[Booch86] Booch,G., “Object Oriented Development,” IEEE Trans. Software Engineering, vol. SE-12, no. 2, pp 211-221, Feb 1986

[Tremblay76] Tremblay,J.P., Soreson,P.G., *An Introduction to Data Structures with Applications,* McGraw-Hill, 1976

[Newell62] Newell,A., “Some problems of the basic organization in problem solving programs,” Proceedings 2nd conference on Self-Organizing Systems, Spartan Books, 1962

[Craig93] Craig,I.D., “A new interpretation of the Blackboard Architecture,” Research Report 254, Department of Computer Science, University of Warwick, UK, 1993

[Petrie97] Petrie,C., “ProcessLink Coordination of Distributed Engineering,” Center for Design Research, *cdr.stanford.edu/ProcessLink/*, Stanford, 1997

[Baya97] Baya,V., Baudin,C., Das,A., Maboqunje,A., “DEDAL, Using device models to Facilitate Retrieval and Indexing of Multimedia Design Information,” Center for Design Research, cdr.stanford.edu/GCDK/dedal/, Stanford, 1997

[Atkinson91] Atkinson,C., *Object-Oriented Reuse, Concurrency and Distribution; An ADA-based approach,* ACM press, Addison-Wesely, New York, NY, 1991

[Cargill92] Cargill,T., *C++ Programming Style*, Addison Wesley, Reading, Massachusetts, 1992

Robot Simulation

[Deneb97] Deneb Robotics Inc., *ULTRAGRIP - The simulation and analysis tool for a variety of Robotic applications,* www.deneb.com, 1997

[RobCad] *RobCad robot simulation and design package,* Israel, 1996

[SILMA89] SILMA Inc., *The CimStation User's Manual,* Version 4.0, Sunnyvale, 1989

[Naher93] Naher,S., LEDA (Library of Efficient Data types & Algorithms) Version 3.5, www.mpi-sb.mpg.de/LEDA/, Max Planck Institute for Information, 1997

[Keffer94] Keffer,T., Tools.h++ Introduction and Reference Manual, Version 6, Rogue Wave Software Inc., Corvallis, Oregon, 1994

[SENSE8-93] SENSE8 Corporation, *World ToolKit Reference Manual,* Version 2.0, Sausalito, CA, 1993

[Keller92] Keller,P., "A toolkit for kinematic and graphic simulation of mobile robots on natural, fractal terrain," tech report, Field Robotics Center, Robotics Institute, Carnegie Mellon University,1992

[Thomas90] Thomas,H., Wettergreen,D., Thorpe,C., and Hoffman,R., "Simulation of the Ambler Environment," in proceedings of the 23rd Pittsburgh conference on modeling and simulation, Pittsburgh, May 1990

[Hoffman94] Hoffman,R., and Simmons,R., "Simulation of Autonomous Robotic Excavation," in Proceedings of the ASCE Specialty Conference on Robotics for Challenging Environments, Albuquerque, New Mexico, February 1994

[Katragadda95] Katragadda,L.K., "A constraint based Dante walking simulator for evaluating walking strategies, and configurations," TRIWG Report, Robotics Institute, Carnegie Mellon University, 1995

[Katragadda94] Katragadda,L.K., Kelly,A., Hebert,M., "Autonomous navigation of a lunar relevant rover using predictive simulation, and active stereo perception," TRIWG annual report, Carnegie Mellon University, Sep 1994

[Kuester95] Kuester,S., Lane,J.C., "Interface Design Issues of the Ranger Telerobotics Flight Experiment," SAE 25th International Conference on Environmental Systems, San Diego, July 1995

[Piguet95] Piguet, L., Fong, T., Hine, B., Hontalas, P., and Nygren, E., "VEVI: A Virtual Reality Tool for Robotic Planetary Explorations", Virtual Reality World, Stuttgart, Germany, Feb 1995

[Hebert96] Hebert,M., Thorpe,C., Stentz,T., "Intelligent Unmanned Ground Vehicles: Research at Carnegie Mellon University", Kluwer Academic Publishers, 1996

[Pearce95] Pearce,H., Tong,D., Pratt,D., Finco,J., "The role of simulations in (Boeing) 777 FSEU development," The Boeing Company, AIAA-95-0995-CP, 1995

Modeling methodology and simulation techniques

[Kane85] Kane,T.R., Levinson,D.A., *Dynamics: Theory And Applications*, McGraw-Hill, USA, 1985

[MDI97] Mechanical Dynamics Inc, *"ADAMS Full Simulation,"* 1997, www.adams.com

[Baraff94] Baraff,D., "Fast Contact Force Computation for Nonpenetrating Rigid Bodies," SIGGRAPH 94, Computer Graphics Proceedings, Annual Conference Series, 1994

[Baraff93] Baraff,D., "Non-penetrating Rigid Body Simulation," Chapter 2, Eurographics '93 State of the Art Reports, Barcelona, September, 1993

[Bekker55] Bekker,M.G., *Theory of Land Locomotion, The mechanics of vehicle mobility*, The University of Michigan Press, Ann Arbor, USA, 1955

[Bekker60] Bekker,M.G., *Off-The-Road Locomotion, Research and Development in Terramechanics*, The University of Michigan Press, Ann Arbor, USA, 1960

[Nagel75] Nagel,L.W., "SPICE2: A Computer Program to Simulate Semiconductor Circuits," PhD dissertation, College of Engineering, University of California, Berkeley, ERL-M520, 1975

[Foley90] Foley, van Dam, Feiner, Huges, *Computer Graphics: principles and practice*, 2nd Edition, Addison-Wesley, 1990

[Ling92] Ling,R., Steinberg,L., "(Thermal) Model generation from Physical Principles: A Progress Report," Technical Report, Department of Computer Science, Rutgers University, CAP-TR-9, 1992

[Thomas80] Thomas,L.C., *Fundamentals of Heat Transfer,* Prentice Hall, Inc., New Jersey, USA, 1980

[Sukhatme79] Sukhatme,S.P., *Heat Transfer,* Orient Longman, 1979

[ONeill66] O'Neill,B., *Elementary Differential Geometry*, Academic Press, New York, 1966

[Preparata88] Preparata,F.P., Shamos,M.I., *Computational geometry*, Springer-Verlag, New York, 1988

[Cook89] Cook,R.D., Malkus,D.S., Plesha,M.E., *Concepts and Applications of Finite Element Analysis,* 3rd Edition, John Wiley & Sons, 1989

[Algor97] Algor Inc., *"ALGOR, Finite Element Design and Analysis, for design and event simulation"* 1997, www.algor.com

[MSC97] MacNeal-Schwendler Corporation, *"MSC/NASTRAN and SuperModel, Solutions for CAE Process Management and Advanced Aerospace Modeling,"* 1997, www.macsch.com

[ANSYS97] ANSYS Inc., *"ANSYS/Multiphysics, Broad analysis and design optimization package,"* Version 5.4, 1997, www.ansys.com

[Bala91] Balasubramaniam,L., Paz-Soldan,J.P., Rinderie,J.R., "Automated Modeling to Support Conceptual Design," Technical Report, Carnegie Mellon University, EDRC 24-55-91, 1991

[Rinderie90] Rinderie,J.R., Balasubramaniam,L., "Automated Modeling to Support Design," Technical Report, Carnegie Mellon University, EDRC 24-31-90, 1990

[Bourne92] Bourne,D., "Intelligent Manufacturing Workstations," Annual Research Review, Robotics Institute, Carnegie Mellon University, 1992

[Cutkosky93] Cutkosky,M.R., Engelmore,B.S., Fikes,R.E., "PACT: An Experiment in Integrating Concurrent Engineering," Computer, v26, no. 1, Jan 1993

[Shamah96] Shamah,B., "Idea to Iron - paperless development of Nomad," University review, NASA Telerobotics, Stanford, 1996

Numerical methods

[Gear72] Gear,W.C., *Numerical initial value problems in ordinary differential equations,* Prentice Hall, New Jersey, 1972

[Strang86] Strang,G., *Introduction to Applied Mathematics,* Wellesley-Cambridge Press, 1986

[Conte80] Conte,S.D., deBoor,C., *Elementary Numerical Analysis,* 3rd Edition, McGraw-Hill, 1980

Optimization of Robot Design

[Luenberger73] Luenberger,D.G., *Introduction to Linear and Nonlinear Programming,* Addison-Wesley, 1973

[Fletcher87] Fletcher,R., *Practical methods in optimization,* John Wiley & Sons, New York, 1987

[Greenberg71] Greenberg,H., *Integer Programming,* Academic Press, 1971

[Broyden67] Broyden,C.G., *"Quasi Newton Methods and their application to function minimization,"* Maths. Comp., 21, 1967

[BFGS] Fletcher,R., *Practical methods in optimization,* John Wiley & Sons, New York, 1987

[Fletcher64] Fletcher,R., Reeves,C.M., *"Function Minimization by Conjugate Gradients,",* 1964, Computer J., 7

[Fletcher85] Fletcher,R., Matthews,S.P.J., *"A stable algorithm for updating triangular factors under a rank one change,"* Maths. Comp., 1985, 45

[Neiderr95] Neiderreiter,H., Shiue,P.J., "Monte Carlo and quasi Monte Carlo methods in sceintific computing," conference proceedings, Springer, 1995

[Korf87] Korf,R.E., "Real-time (A-Star) heuristic search," Computer Science, University of California, Los Angeles, 1987

[Lupton93] Lupton,R., *Statistics in theory and practice,* Princeton University Press, New Jersey,1993

[Goldberg89] Goldberg,D.E., *Genetic algorithms in Search, Optimization, and Machine Learning,* Addison-Wesely, Massachusetts, 1989

[Davis91] Davis,L., *Handbook of Genetic Algorithms,* Van Nostrand Reinhold, New York, 1991

[Baluja95] Baluja,S., *"An Emperical Law of Seven Interactive and Evolutionary Function Optimization Heuristics,"* Carnegie Mellon, 1995, CMU-CS-TR-95-193

[Kim92] Kim,J.O., "Task based kinematic design of Robot manipulators," PhD thesis, Carnegie Mellon University, 1992

[Murthy92] Murthy,S., *"Synergy in cooperating agents: designing manipulators from task specifications,"* Dept. of Electrical and Computer Engg., 1992, Carnegie Mellon

[Roston94] Roston,G., *"A Genetic Methodology for Configuration Design,"* Ph.D. thesis, Mechanical Engg., Carnegie Mellon, CMU-RI-TR-94-42

[Leger97] Leger,C., "Automated Synthesis of Optimal Robot Configurations," PhD proposal, Robotics Institute, Carnegie Mellon University, 1997

[Koza92] Koza,J.R., *"Genetic Programming - on the programming of computers by means of natural selection,"* Bradford Books, 1992, The MIT Press

[Sims94] Sims,K., *"Evolving Virtual Creatures,"* Computer Graphics, Annual Conference Series, pp 15-22, SIGGRAPH 1994

Design Search through Innovation and Creativity

[Rich91] Rich,E., Knight,K., *Artificial Intelligence,* 2nd Edition, McGraw Hill, 1991

[Brown83] Brown,D.C., Chandrasekaran,B., "An approach to expert systems for mechanical design," IEEE Computer Society Trends and Applications, pp. 173-180, Gaithersburg, May 1983

[Coyne87] Coyne,R.D., Rosenman,M.A., Radford,A.D., Gero,J.S., "Innovation and Creativity in Knowledge-Based CAD," *Expert Systems in Computer Aided Design,* Elsevier Science Publishers, North-Holland, Amsterdam, 1987

[Coyne88] Coyne,R.D., Newton,S., Sudweeks,F., "Modelling the emergence of Schemas in design reasoning," University of Sydney NSW 2006, Australia

[Cagan87] Cagan,J., Agogino,A.M., "Innovative Design of Mechanical Structures from First Principles," AI EDAM, 1(3), pp. 169-189, 1987

[Choy] Choy,J.K., Agogino,A.M., "Symon: Automated Symbolic Monotonicity Analysis System for Qualitative Design Optimization," Expert systems laboratory, University of CA, Berkeley

[Steinberg92] Steinberg,L., Langrana,N., Fisher,G., "MEET: Decomposition and Constraint Propagation in Mechanical Design," Rutgers University, 1992

[Murthy89] Murthy,S.S., Addanki,S., "PROMPT: An Innovative Design Tool," NSF Engineering Design Research Conference, University of Massachusetts, Amherst, June, 1989

[Mitchell89] Mitchell,M.J., Ware,G., Edythe,M., "A Computational view of Design Creativity," International Round-Table Conference, Modeling Creativity and Knowledge-Based Creative Design, Heron Island, December, 1989

Controller Generation For Robots

[Kelly98] Kelly,S.D., and Murray,R.M., "Lagrangian Mechanics and Carangiform Locomotion," Submitted, IFAC Symposium on Nonlinear Control systems Design (NOLCOS), 1998

[Latombe91] Latombe,J.C., *Robot Motion Planning,* Kluwer Academic Publishers, Boston, 1991

[Brumitt97] Brumitt,B., "A Mission Planning System for Multiple Mobile Robots in Unknown, Unstructured, and Changing Environments," PhD thesis, Robotics Institute, Carnegie Mellon, 1997

[Kodit85] Koditschek,D., *"Adaptive Strategies for Control of Natural Motion,"* 24th Conference on Decision and Control, 1985

[Chen94] Chen,V., Cannon,R.H.Jr., *"Experiments in Nonlinear Aaptive Control of Free-Flying Space Robots,"* 1994, IEEE Conference on Robotics and Automation

[Rosenblatt62] Rosenblatt,F., *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms,* Spartan Books, Washington D.C., 1962

[Rumelhart86] Rumelhart,D.E., Hinton,G.E., Williams,R.J., "Learning internal representation by error propagation," in *Parallel and distributed processing by Rumelhart and McClelland,* MIT press, Cambridge, 1986

[Fahlman90] Fahlman,S.E., Boyan,J.A., "The Cascade Two Learning architecture," Technical Report CMU-CS-90-100, Carnegie Mellon University, 1994

[Pomerleau92] Pomerleau,D.A., "Neural Network Perception for Mobile Robot Guidance," Kluwer Academic Publishing, 1992

[Davis95] Davis,I.L., "Neural Networks for Real-Time Terrain Typing," Ph.D. thesis, Robotics Institute, Carnegie Mellon, CMU-RI-TR-95-06

[Nechyba95] Nechyba,M.C., Xu,Y., *"Towards Human Control Strategy Learning: Neural Network Approach with Variable Activation Functions,"* Robotics Institute, Carnegie Mellon, CMU-RI-TR-95-05

[Dowling98] Dowling,K., "Limbless Locomotion: Learning to Crawl, snake robots that learn to locomote" Ph.D. Thesis, 1998, Robotics Institute, Carnegie Mellon University

[Barrett97] Barret,D., "Optimization of Swimming Locomotion by Genetic Algorithms," Recent trends in robot locomotion workshop, ICRA, 1997, Albuqerque, NM

[Maybeck] Maybeck,P., *Stochastic Models, Estimation and Control,* Ch 1, Vol. 1, Academic Press Inc.; www.cs.unc.edu/~welch/kalmanLinks.html

[Moore95] Moore,A.W., Atkeson,C.G., Schaal,S., *"Memory-based Learning for Control,"* Robotics Institute, Carnegie Mellon, 1995, CMU-RI-TR-95-18

[Moore97] Moore,A.W., Lee,M.S., *"Cached Sufficient Statistics for Effecient Machine Learning with Large Datasets,"* Robotics Institute, 1997, CMU-RI-TR-97-27

Lunar Exploration and Design

[Heiken91] Heiken,G.H., Vaniman,D.T., French,B.M., *"LUNAR sourcebook, a user's guide to the moon,"* 1991, Cambridge University Press

[Griffin91] Griffin,M.D., French,J.R., *Space Vehicle Design,* AIAA Education Series, Washington, DC, 1991

[Larson92] LarsonW.J., WertzJ.R., *Space Mission Analysis and Design,* 2nd Edition, Kluwer Academic Publishers, 1992

[Krotkov94] Krotkov,E., Bares,J., Katragadda,L., Simmons,R., Whittaker,R., "Lunar Rover Technology Demonstrations with Dante and Ratler," ISAIRAS, JPL, California, 1994

[Katragadda94] Katragadda,L., Murphy,J., Whittaker,W., "Rover Configuration for Entertainment-Based Lunar Excursion," International Lunar Exploration Conference, San Diego, 1994

[Klarer93] Klarer,P.R., Purvis,J.W., "A Highly Agile Mobility Chassis Design for a Robotic All-Terrain Lunar Exploration Rover", Sandia National Labs, Sandia, 1993

[Murphy97] Murphy,J., Katragadda,L., "Nomad: Technology and demonstration of planetary telepresence", SSI Conference on Space Manufacturing, May 1995

[Gump96] Gump,D., Whittaker,W.L., Katragadda,L.K., Murphy,J.R., "Japanese expedition to the Moon," project report submitted to Mitsubishi, 1996

[Whittaker97] Whittaker,W.L., Duke,M., et al "Lunar Ice Discovery," proposal to the NASA discovery program, Carnegie Mellon, Dec 1996