A Systems Architecting Approach to Automation and Optimization of Satellite Design in SPIDR

Lucy HOAG, Tatiana KICHKAYLO and David BARNHART
Information Sciences Institute, University of Southern California
Marina del Rey, CA 90292 USA

ABSTRACT

The design of satellites is a labor-intensive, costly, and time consuming process. It is highly multi-disciplinary, involving several diverse and interdependent subsystems all critical to mission success. In the typical approach, subsystem experts gather and flesh out design concepts over the timescale of weeks, melding analyses from multiple heterogeneous modeling tools and iterating until a sound design is settled upon. However, due to the need for collocation, the incongruity of design information from disparate tools and the inefficiencies inherent in the iterative process, this process remains expensive, error-prone and slow. Technology that can automate and optimize the spacecraft design process to lower costs while maintaining reliability has been identified as one potential solution. However, due to the prohibitively expansive searchable space implied in spacecraft design, most tools of this kind can consider only a very limited subset of design variables and can be unreliable, inefficient and prone to generating unrealistic or sub-optimal designs. The Systems Platform for Integrated Design in Realtime (SPIDR)[i], a project underway at USC's Information Sciences Institute and Space Engineering Research Center (SERC), presents a unique approach to this problem. SPIDR's search and optimization engine utilizes innovative applications of artificial intelligence (AI) techniques to quickly generate provably optimal results. The application of AI planning techniques enables consideration of the entire design space by starting search from a high-level problem specification, while constraint propagation offers an analog to the concept of iteration, where all variables are defined as interval domains and shrunk according to rules and constraints until provably optimal single point solutions are discovered. With a unique architecture that utilizes a knowledge-based information system, a powerful combination of mathematical approach and optimization algorithm, and the inclusion of socio-technical factors specific to spacecraft development, SPIDR demonstrates the potential to be a valuable addition to the toolkits of engineers in both industry and academia.

Keywords: Satellite, design, optimization, automation, rules and constraints

1. INTRODUCTION

The design of satellites is known to be a labor-intensive, costly, and time-consuming process. It is highly multi-disciplinary, involving several diverse and interdependent subsystems all critical to mission success (Figure 1). In addition, it differs from the design processes of most other highly technical systems in that once the satellite is launched, it is unreachable. Troubleshooting, debugging, and maintenance are virtually impossible [1]. For this reason, these highly complex systems generally must also be designed and built with a high degree of reliability.

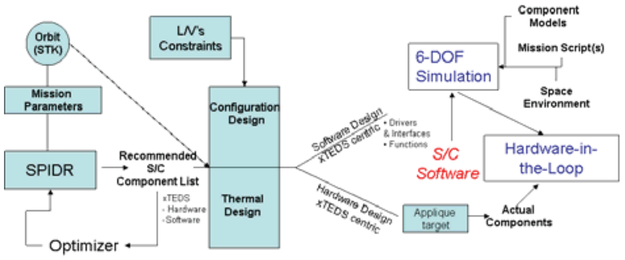


Figure 1. SPIDR and the design process within the entire spacecraft development lifecycle.

Motivation

The typical design approach within the community to-date has been to gather single discipline experts together into a physical facility, examples are the Aerospace Corporation's Concept Design Center [2] or the Jet Propulsion Laboratory's Project Design Center [3], where system and subsystem experts gather to generate and evaluate concept designs. The process occurs on the timescale of weeks, and involves melding analyses from multiple heterogeneous modeling tools and iterating repeatedly until a sound design or set of possible designs is settled upon. However, due to the need for collocation, the incongruity of design information from disparate tools, the inefficiencies inherent in the iterative process, and non-uniformity in the subsystem design tools from one expert to another, and this process remains expensive, cumbersome in solutions and slow. The result is a progression of unnecessarily inefficient development cycles: as the high system reliability required in satellite design drives up costs and schedule, the cost associated with spacecraft failure also increases, only requiring better reliability and creating a cycle of positive feedback in the space industry that continues to draw out schedules and fuel exorbitant budgets.

In response, there was an experimental shift towards "faster, better, cheaper" metrics, at one time perceived as mutually exclusive in aerospace, but then touted as parameters that could be simultaneous, even symbiotic, goals [4]. As space programs tried to respond to this push, though, it was not without consequence. For example, many view the failures of the NASA Mars Climate Orbiter and Mars Polar Lander in 1999 as issues of management problems and insufficient funding that stemmed directly from this lean initiative [5]. Clearly, to revolutionize any aspect of the traditional spacecraft design process is a non-

[i] Previous papers have referred to SPIDR as "Spacecraft Portal for Integrated Design in Realtime"

trivial challenge and must take into account the large number of opposing constraints in a fashion that can be mediated optimally. As the space industry has matured, so has the technology, both in the actual spacecraft components as well as in the modeling and analysis that goes into the production and verification of a design. Together, this growth offers significant opportunities for space programs to become more efficient and maintain high levels of reliability but with lowered costs and shorter schedules. Potential solutions have taken many forms, including through the actual spacecraft hardware and/or software (e.g. the use of Commercial Off The Shelf "COTS" components), through modularization (e.g. "Plug n Play" components and integration [6] and standardization (e.g. the picosatellites and picosatellite launchers in the CubeSat/PPOD program [7]), and through technology to improve the design environment or process (e.g. through synthesis of modeling tools [8] and optimization [9,10,11,12,13,14] This paper will analyze the Systems Platform for Integrated Design in Realtime, or SPIDR, a tool in development at USC's Information Sciences Institute and Space Engineering Research Center aimed at automating the spacecraft design and optimization processes and generating high level of confidence satellite design concepts quickly and cheaply. SPIDR accomplishes this by combining innovative software technologies in a dynamic system architecture with validated satellite design knowledge and a database of existing, real-world spacecraft hardware.

Related Work

Several varying attempts have been made to create computer-aided tools for facilitating faster, cheaper and more efficient spacecraft design processes. However, due to the prohibitively expansive searchable space implied in spacecraft design, most tools of this kind are subject to significant drawbacks. First, they often can consider only a very limited subset of design variables. Search algorithms are also generally blind or random, leading to inefficient search paths that may revisit or altogether miss potential designs. Finally, most design methodologies are linear, forcing the process to be highly iterative. As a result, most tools can be unrealistic and prone to generation of sub-optimal designs.

In 1995, NASA's Jet Propulsion Laboratory created MIDAS, the Multi-Disciplinary Integrated Design Assistant for Spacecraft, which aimed to capture engineers' design knowledge graphically to aid the design process [9] but relied on a separate tool, the Optimization Assistant, and massive parallel computing to apply generic metaheuristic optimization algorithms [10]. Also out of JPL was the Design Evolver, an evolutionary optimization approach also used with MIDAS [11], which by its nature presented the same drawbacks. In 1998, the University of Colorado's Spacecraft Concept Optimization and Utility Tool attempted to replace the typical iterative "change design" approach with an "intelligent search" [12]. This improved the process by automating the methodology of searching the space of many top-level designs and intelligently selecting the most optimal, instead of selecting some more well-defined design as a starting point and optimizing on a select number of constraints. Design_Net Engineering investigated use of Dynamic Programming and the *OptQuest* software to mathematically optimize designs from their Small Satellite Optimization and System Analysis Model in 2000 [13]. Jilla and Miller took a similar approach to multidisciplinary design optimization (MDO) in 2004 but with Distributed Satellite Systems [14]. In 2004 Schaefer and Rudolph attempted a Design Compiler based on satellite design "grammars" to perform design automation, but focused on geometry [15]. Gross et al continued this work with a UML-based central product model in 2009 [8]. Other efforts have focused on optimization of single subsystems, for example thermal [16], orbit design only [17], database use to standardize reusable space mission elements like telemetry [18], and processes for streamlining simulation and verification [19].

2. SPIDR APPROACH

Traditional approach to design of complex engineering systems such as spacecraft involves multiple people iteratively refining a design. A change to the design usually has ripple effects that need be evaluated by other team members, who will then make appropriate changes to the design (Figure 2, left).

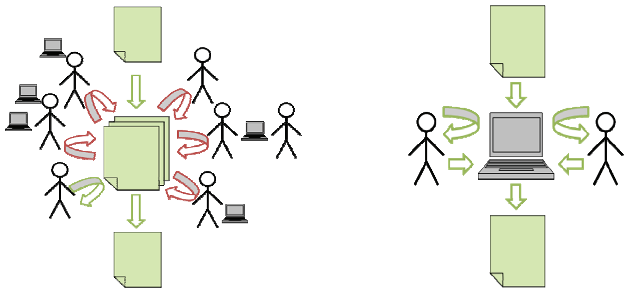


Figure 2. Traditionally, change to a design produces ripple effects, which are evaluated by other team participants (left). SPIDR evaluates ripple effects automatically based on a database of engineering knowledge, allowing users to quickly evaluate ideas (right).

Propagation of change ripples may involve many manual operations, which require time. If a flaw is discovered during ripple effect propagation, this time is wasted. As a result, engineers often prefer to go with tested solutions with high probability of success to minimize the number of trials, even if it leads to overall suboptimal solutions. Collaborative design systems facilitate information exchange between team members, thus reducing the time required for change evaluation. However, input from multiple participants is still required during each cycle.

Many operations performed by humans during change evaluation are relatively repetitive. Indeed, textbooks and manuals exist containing recipes for performing various design and analysis tasks. This observation provides the basis for SPIDR approach.

In SPIDR, recipes are encoded in machine executable rules. Rules corresponding to various engineering disciplines involved in the design process are added to a knowledge base. This allows SPIDR to perform evaluation of ripple effects of a change automatically, without involving other users (Figure 2, right). The automated propagation of ripple effects also opens the door for optimization: the result returned by SPIDR is the best one over alternative approaches contained in the knowledge base. Our hypothesis was that this would allow a single user to more quickly evaluate different design solutions, cutting iteration times and coming to at least an equal level of design quality than would be expected out of the traditional design team. Faster design cycles would then encourage exploration of new design choices and ultimately lead to higher quality designs. When the user arrives at a new solution or technique, he/she updates the database so that other users can use the new knowledge in their work.

The initial prototype of SPIDR focused on knowledge representation, automated propagation of implications of design decisions (ripple effects) and optimization [20]. The rest of this section provides details of this initial implementation. The next section discusses our experience of using SPIDR in a real world project.

Knowledge representation

SPIDR traces its roots to the field of constraint-based planning in Artificial Intelligence. Design-related information in SPIDR is represented in terms of variables, constraints, tokens and rules.

Tokens are typed collections of variables. Tokens may represent individual physical components, higher level systems, or even abstract concepts such as mission parameters. A token may have multiple types, where types form a taxonomy. SPIDR uses token types to determine the role of a token in a design, including its requirements and possible uses. For example, the model designer may declare that any token of type "physical component" has a variable called "mass", there should be at most one token of type "vehicle summary" in a design, and the value of variable "total mass" of the latter token should equal the sum of "mass" variables of all "physical component" tokens in the design. If the model states that "battery" is a subtype of "physical component", then the mass of any "battery" token in the design will be automatically included in the total mass of the vehicle. Token types are also used for logical markup of the model. For example, any token with type "user input" is automatically used by SPIDR for generation of the user interface.

Variables represent various quantities and properties of the design and its components. In a partial design, variables may be associated with sets of values. For example, suppose SPIDR concluded that a battery is necessary in a design, but has not yet picked up a particular model from the catalog. At this point, the value of the "mass" variable of the battery token will be an interval ranging from the lightest to the heaviest battery in the catalog. SPIDR variables can also take discrete values, e.g., to represent the type of the battery.

Constraints represent functional dependencies between variables. In the example above, summing up all component masses to compute the total vehicle mass is performed by a constraint. In SPIDR constraints may be represented as mathematical functions, simple relations (e.g., "in set"), or arbitrary computable functions implemented in some programming language.

A collection of tokens with their variables and constraints between those variables describes a design (partial or complete). Rules are small snippets of engineering knowledge describing possible or necessary modifications of a design. For example, a rule may state that any design that includes a torque rod ***should*** also include a magnetometer. Another rule may say that under certain conditions a power supply of a spacecraft ***could*** be implemented using solar panels. There may be multiple "could" rules describing various power supply technologies. Given a partial design, SPIDR would consider all such rules applicable in the design and ultimately select the one that produces the best design.

SPIDR tool

SPIDR 1.0 is a standalone Java application. The tool takes two kinds of inputs (Figure 3). The knowledge base of rules and constraints and the catalog of available components are reused over multiple missions. These information sources are slowly expanded and updated as new knowledge becomes available. The second kind of inputs includes mission requirements and optimization preferences. These are refined during the design process.

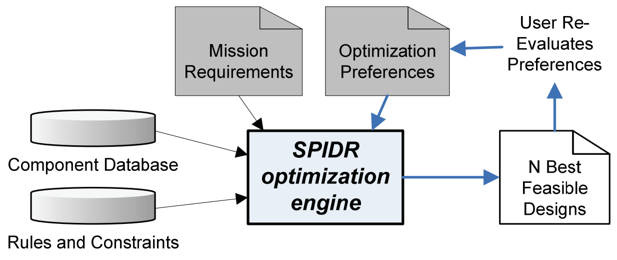
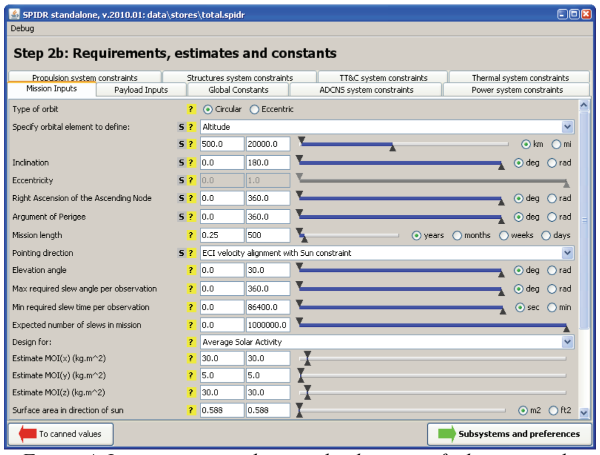


Figure 3. Inputs and outputs of SPIDR optimization engine and the refinement cycle of the design process

The computational core of the tool consists of a rule engine and an optimization engine. The rule engine is responsible for enforcing "should" rules and generating alternative solutions using "could" rules. The optimization engine orchestrates the search over multiple alternatives and free parameters.

As mentioned earlier, variables in partial designs may have sets of values (which are represented as intervals for numeric variables). This feature is used by the algorithm to speed up search. It is also useful for iterative refinement of design requirements. The user initializes the design process by specifying hard mission requirements, such as orbit and payload parameters. The SPIDR tool then guides the user through a sequence of steps, automatically rolling out implications of information specified so far and requesting more information. During the first pass, the user may leave some of the parameters flexible by specifying intervals instead of single values (Figure 4). The user selects a metric function among several relevant for the problem, and SPIDR generates several best designs consistent with the knowledge base. The user can use these designs as an illustration of what is possible and modify requirements (e.g., by placing an upper bound on the total cost of the spacecraft) and/or choose a different optimization metric. This cycle is repeated until a satisfactory design is produced.

Designs produced by SPIDR are saved as XML files containing information about applied rules, the set of tokens in the design, and values of all variables. XSLT translation files are used to convert the original design XML into a variety of text-based formats. In addition, application-specific translators may be used for binary formats. For example, SPIDR includes a translator that inserts design values into cells of an existing MS Excel spreadsheet.


SPIDR standalone, v.2010.01: data\stores\total.spidr
Debug
Propulsion system constraints
Structures system constraints
TT&C system constraints
Thermal system constraints
Mission Inputs
Payload Inputs
Global Constants
ADCNS system constraints
Power system constraints
To canned values
Subsystems and preferences

Step 2b: Requirements, estimates and constants

Type of orbit	?	[✓ Circular] Eccentric		
Specify orbital element to define:	S ?	Altitude		
	S ?	500.0	20000.0	[✓ km] mi
Inclination	S ?	0.0	180.0	[✓ deg] rad
Eccentricity	S ?	0.0	1.0	
Right Ascension of the Ascending Node	S ?	0.0	360.0	[✓ deg] rad
Argument of Perigee	S ?	0.0	360.0	[✓ deg] rad
Mission length	?	0.25	500	[✓ years] months weeks days
Pointing direction	S ?	ECI velocity alignment with Sun constraint		
Elevation angle	?	0.0	30.0	[✓ deg] rad
Max required slew angle per observation	?	0.0	360.0	[✓ deg] rad
Min required slew time per observation	?	0.0	86400.0	[✓ sec] min
Expected number of slews in mission	?	0.0	1000000.0	
Design for:	?	Average Solar Activity		
Estimate MOI(x) (kg.m^2)	?	30.0	30.0	
Estimate MOI(y) (kg.m^2)	?	5.0	5.0	
Estimate MOI(z) (kg.m^2)	?	30.0	30.0	
Surface area in direction of sun	?	0.588	0.588	[✓ m2] ft2

Figure 4. Input screens with several values specified as intervals.

3. USE CASE

The utility of SPIDR as a systems engineering tool for spacecraft design was tested during the development of USC's first nanosatellite project. The system engineer for the project developed a set of SPIDR rules for nanosatellite design. Multiple times during the design phase of the project, the SPIDR optimization loop described in the previous section was executed. The produced XML design files, as well as versions in several other formats, were then uploaded to the team's collaboration web site. From there, various team members downloaded the data to perform various analyses, results of which were then transferred to the system engineer and integrated into the next design iteration.

Design Sharing

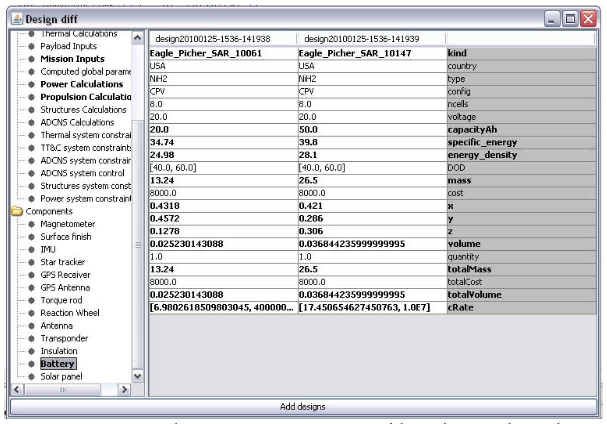
The initial prototype of SPIDR was a standalone tool that integrated various kinds of analysis into its optimization loop. In practice, many important analytical tools could not be integrated with SPIDR (due to the lack of APIs for the tools, licensing restrictions, or the need for user interaction). Therefore, the USC team used a web site to host SPIDR designs. Even though the optimization loop could not be completely closed, this provided a consistent method for configuration control, data sharing, and quicker computation that helped streamline and maintain accuracy in the design process.

Reusable Input Files

With breaking of the single-session optimization loop described above, the need for savable and reusable problem specifications became apparent. A capability was added that allows users to save all initial inputs as an XML input file, including mission and payload inputs and subsystem constraints. These files can then be loaded at the starting screen of SPIDR, allowing exact reuse or modification and re-save.

Design Solution Comparison

Another issue that surfaced was the need to compare output variables from similar design solutions. For example, due to the low-cost nature of the nanosatellite project, the exact launch and orbit were unknown. While SPIDR already has the capability to accommodate this sort of ambiguity in the use of intervals, this only provides the most optimal possible outcomes within the specified launch or orbit interval. To provide a better trade-off analysis capability, a function was implemented for a user to easily perform side-by-side comparison of design solutions from similar problem specifications (missions) and analyze the difference in results (Figure 5). This provided a quick way to identify the differences in performance variables such as communication link margin, reaction wheel saturation time, etc. based on very similar problem specifications that differed only in orbit altitude and eccentricity.

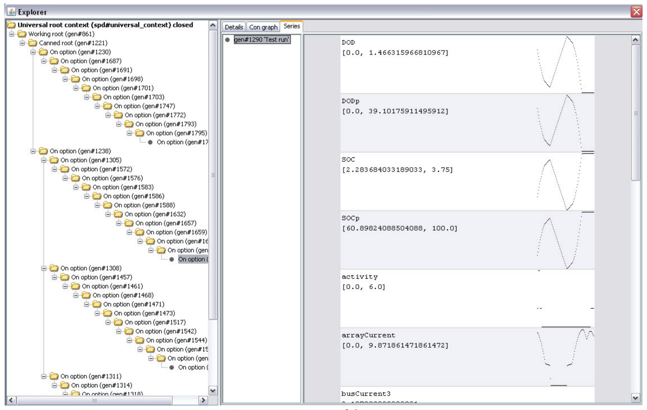


design20100125-1536-141938	design20100125-1536-141939	
Eagle_Picher_SAR_10061	**Eagle_Picher_SAR_10147**	**kind**
USA	USA	country
NiH2	NiH2	type
CPV	CPV	config
8.0	8.0	ncells
20.0	20.0	voltage
20.0	**50.0**	**capacityAh**
34.74	**39.8**	**specific_energy**
24.98	**28.1**	**energy_density**
[40.0, 60.0]	[40.0, 60.0]	DOD
13.24	**26.5**	**mass**
8000.0	8000.0	cost
0.4318	**0.421**	**x**
0.4572	**0.286**	**y**
0.1278	**0.306**	**z**
0.025230143088	**0.036844235999999995**	**volume**
1.0	1.0	quantity
13.24	**26.5**	**totalMass**
8000.0	8000.0	totalCost
0.025230143088	**0.036844235999999995**	**totalVolume**
[6.9802618509803045, 400000...	**[17.450654627450763, 1.0E7]**	**cRate**

Figure 5. Design solution comparison. Variables whose values change across designs appear in bold.

Dynamic Behavior

The design process in SPIDR initially considered only time-independent parameters, e.g. the dry mass. However, some values are dependent on the mission scenario. For example, a common analysis in the design of a power system for a satellite is that of the depth of discharge of the battery over time, which is dependent on orbit, solar panel parameters, power consumption of the satellite's components and mission operations which specify modes for these components. If the calculated depth of discharge exceeds the nominal limit for that battery at any point in some projected time period, the design is known to be infeasible, and changes to some time-independent parameters need to be made to fix the problem. SPIDR was extended with the ability to incorporate dynamic behavior, allowing users to encode in SPIDR certain processes that were previously contained in external models and necessitated out-of-the-loop analysis (Figure 6).



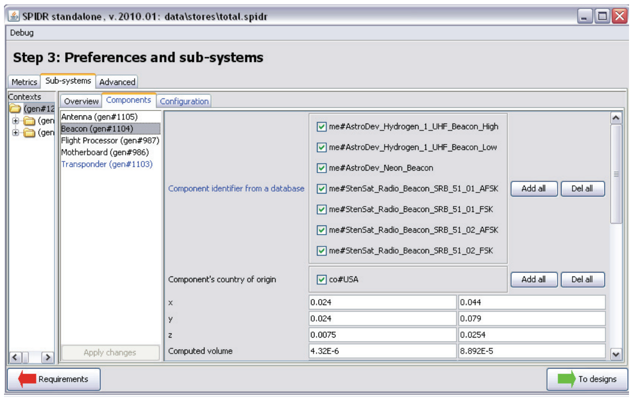
```
DOD
[0.0, 1.466315966810967]

DODp
[0.0, 39.10175911495912]

SOC
[2.283684033189033, 3.75]

SOCp
[60.89824088504088, 100.0]

activity
[0.0, 6.0]

arrayCurrent
[0.0, 9.871861471861472]

busCurrent3
```

Figure 6. Power profile time series.

By incorporating these analyses into the internal SPIDR routine, non-feasible and sub-optimal designs are able to be discarded earlier in the design cycle.

Manual Component Selection Override

The USC case proved to be an interesting one for SPIDR because the satellite is a CubeSat, or a nanosatellite "kit" that when purchased is already partially developed and functional at a baseline level. This changes the problem specification for an automated design optimization tool, as a significant portion of the subsystem hardware is already pre-selected, and the number of ways to fulfill remaining subsystem tasks is limited due to the constrained size, shape, interfaces and other unique characteristics of the satellite. To account for this, the capability was added for a user to manually select or de-select specific components after feasible design configurations have been discovered (Figure 7).


SPIDR standalone, v.2010.01: data\stores\total.spidr

Debug

Step 3: Preferences and sub-systems

Antenna (gen#1105)
Beacon (gen#1104)
Flight Processor (gen#987)
Motherboard (gen#986)
Transponder (gen#1103)

Component identifier from a database
- ✓ me#AstroDev_Hydrogen_1_UHF_Beacon_High
- ✓ me#AstroDev_Hydrogen_1_UHF_Beacon_Low
- ✓ me#AstroDev_Neon_Beacon
- ✓ me#StenSat_Radio_Beacon_SRB_51_01_AFSK
- ✓ me#StenSat_Radio_Beacon_SRB_51_01_FSK
- ✓ me#StenSat_Radio_Beacon_SRB_51_02_AFSK
- ✓ me#StenSat_Radio_Beacon_SRB_51_02_FSK

Component's country of origin
- ✓ co#USA

x	0.024	0.044
y	0.024	0.079
z	0.0075	0.0254
Computed volume	4.32E-6	8.892E-5

Figure 7. Preferences screen with option for manual component selection.

SPIDR Lite

Ideally, one generalized set of rules and constraints should be able to encapsulate the entire satellite design space, and thus be applicable to any satellite design problem, regardless of size, customer, developer, etc. With a sufficiently complex knowledge base, any set of interfaces, vendors, component lead times, etc., would be accounted for. The correct narrowing of variables and components then occurs based on user inputs, which can include qualitative options such as the existence of a predefined component or baseline bus. However, it was found that this level of generality was difficult to reach by extending the current generic satellite template, especially in a compressed schedule. Because of this, an additional template specific to CubeSats was created to perform the same optimization routine as for a generic satellite, based on much of the same knowledge base but with slightly different rules and constraints.

In addition, because the satellite is student-produced, a unique set of limitations, opportunities and socio-technical constraints were introduced to the design process. For example, the choice of transceiver for the communication system is influenced not only by technical parameters but also by choice of ground station equipment, which was unsure for the student-based USC team, and the ability to interface with global ground station networks that are only available to student projects. Unique circumstances such as these necessitated continuous redefinition of options and design pathways within the rule and constraint network that may or may not be applicable to any other satellite design problem and therefore not reusable from project to project. This prompted the question of whether hard coding the rule set for CubeSats was a helpful or time-efficient endeavor at all.

An application called *SPIDR Lite* was created to explore this question. SPIDR Lite is essentially a simplified version of SPIDR that allows a user to create an optimization project without having to hard code the rules in the SPIDR environment, facilitating a more user-friendly and easier-to-understand way for an engineer to readily exploit the power of the SPIDR search and optimization engine. Further use of this tool will help to address the utility of an optimization tool in satellite design when the problem being addressed is not a "clean" optimization problem.

Living Design

Finally, the need to integrate the design, integration and testing lifecycles became evident. Software to track tasks, task owners, timelines and data exist commercially, for example ESI Group's *VDot* technology [21] and in academia, for example the *ELogbook* software developed at Stanford in 2006 [22]. These were both considered for use in the USC project and ELogbook was chosen due to financial limitations. ELogbook provided a way to log integration activities, but did not prove useful for direct integration with design documents. A tool that could more cohesively combine the design process with work schedule and ongoing documentation would help streamline the process and reduce errors. SPIDR was identified as a baseline tool that could be extended to accomplish these tasks, treating designs and the design process as living artifacts and tying together roll-out of implications of any design modifications, direct collaboration with users and user tasks, including both integration and test activities as well as out-of-the-loop design analysis, team discussion and project management.

4. FUTURE WORK

There are a number of areas of future work that are enabled with the SPIDR architecture that USC is investigating.

- SPIDR was built and designed for the spacecraft world as an example of a highly complex arena that could benefit from it. It is possible to extend the use of SPIDR to other engineering problems that could uncover new or even unexplored design arenas. Transportation systems, energy harvesting systems, marine vehicle systems are just a few that might benefit from this tool.
- Another area within the satellite-specific realm is the larger world of acquisition. By considering SPIDR as an "contract technical acquisition" tool that could be shared between the customer and the system developers, it is possible to create a highly synergistic relationship, based on the ability of a developer to take pre-defined requirements in the form of XML inputs, and prove that their solutions can actually work at a hardware level, *before the acquisition process is completed.* Possibilities exist to drastically cut down the time for typical satellite acquisition (from years now) and better match the disparate requirements that drive satellite delivery that come inherently from the acquisition process.

- Combining SPIDR computational power with the constructs of the traditional House of Quality [23]. The ability to not only design to mission needs (not just mission requirements) and at the same time compare key mission success parameters across project "design solutions" that are fundamentally different (e.g. starting with an earth observation mission, compare single satellite versus satellite constellation versus UAV's versus balloons, etc.), would open up a larger trade space across different architectural solutions that do not share the same valuations. Most customers in cross platform solutions (i.e., Government) are constantly trying to understand the benefits of space versus air versus ground, and having a single tool that is able to evaluate each, heuristically on its own and against disparate mechanisms, would help identify the best solution to a specific problem.

5. CONCLUSION

SPIDR has proven to decrease the time it takes for a single user to generate multiple designs, and come up with an optimum design based on rules and constraints entered for development of a highly complex, subsystem interdependent satellite. While not entirely replacing the traditional system team intensive approach, this approach will provide a means to increase the fidelity of a team's starting position on a design, thereby decreasing the time to an optimal solution and lowering the costs for a design team to operate within.

6. REFERENCES

[1] Rechtin, E. (1991). Systems architecting: Creating and building complex systems. Englewood Cliffs, NJ: Prentice Hall.

[2] Aguilar, J., Dawdy, A. and Law, G. "The Aerospace Corporation's Concept Design Center" **International Symposium of the International Council on Systems Engineering Proceedings**, 26-30 Jul. 1998.

[3] Smith, J. "Concurrent Engineering in the Jet Propulsion Laboratory Project Design Center", Paper 98AMTC-83, **Society of Automotive Engineers**, Long Beach, CA 1998.

[4] McCurdy, H. (2001) Faster, Better, Cheaper: Low Cost Innovation in the US Space Program. Johns Hopkins University Press, Baltimore, MD.

[5] "NASA reexamines faster, better, cheaper strategy" **Issues in Science and Technology**, National Academy of Sciences, 22 Jun 2000.

[6] Fronterhouse, Don; Lyke, James; Achramowicz, Steve.; "Plug-and-play Satellite", **AIAA Infotech Conference Proceedings**, 7-9 May 2007, Rohnert Park, CA.

[7] Puig-Suari, J., Turner, C. and Ahlgren, W. "Development of the Standard CubeSat Deployer and a CubeSat Class PicoSatellite", **2001 IEEE Aerospace Conference Proceedings**, 10-17 March 2001, Big Sky, MT.

[8] Gross, J., Reichwein, A. and Rudolph, S. "An Executable Unified Product Model Based on UML to Support Satellite Design" **American Institute of Aeronautics and Astronautics SPACE 2009 Proceedings**, 14-17 Sept 2009, Pasadena, CA.

[9] George, J., Peterson, J., and Southard, S. "Multidisciplinary Integrated Design Assistant for Spacecraft", **American Institute of Aeronautics and Astronautics Proceedings**, 1995.

[10] Fukunaga, A.S., Chien, S., Mutz, D., Sherwood, R., and Stechert, A. "Automating the process of optimization in spacecraft design", **IEEE Aerospace Conference Proceedings**, 1997.

[11] Fukunaga, A.S., and Stechert, D., "An Evolutionary Optimization System for Spacecraft Design", Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: Proceedings of the Tenth International Conference, 1997.

[12] Mosher, T. "Spacecraft Design Using a Genetic Algorithm Optimization Approach", **IEEE Aerospace Conference Proceedings**, 1998.

[13] Riddle Taylor, E. "Evaluation of Multidisciplinary Design Optimization Techniques as Applied to Spacecraft Design", **IEEE Aerospace Conference Proceedings**, 18-25 Mar 2000, Big Sky, MT.

[14] Jilla, C. and Miller, D. "Multi-Objective, Multidisciplinary Design Optimization Methodology for Distributed Satellite Systems", **Journal of Spacecraft and Rockets**, Vol. 41, No. 1 Jan-Feb 2004.

[15] Schaefer, J. and Rudolph, S. "Satellite design by design grammars", **German Aerospace Congress Proceedings**, 2003.

[16] Galski, R., de Sousa, F., Ramos, F., and Muraoka, I. "Spacecraft Thermal Design with the Generalized Extremal Optimization Algorithm" **Inverse Problems, Design and Optimization Symposium**, 2004, Rio de Janeiro.

[17] Jinxiu, Z and Xibin, C. "An Integrated System for Satellite Orbit Design and Mission Analysis", **AIAA Modeling and Simulation Technologies Conference and Exhibit**, 16-19 Aug 2004, Provide, RI.

[18] Simon, G., Shaya, E., Rice, K, Cooper, S., Dunham, J. and Champion, J. "XTCE: A Standard XML-Schema for Describing Mission Operations Databases", **IEEE Aerospace Conference Proceedings**, 2004.

[19] Heindricks, R. and Eickhoff, J. "The significant role of simulation in satellite development and verification", **German Aerospace Congress Proceedings**, 2003.

[20] Barnhart, D., Kichkaylo, T. and Hoag, L. "SPIDR: Integrated Systems Engineering Design-to-Simulation Software for Satellite Build", **Conference on Systems Engineering Research 2009 Proceedings**, 20 Apr 2009, Loughborough, UK.

[21] "Simulation Systems Integration" Accessed at <http://www.esi-group.com/products/simulation-systems-integration/vdot> March 2, 2010.

[22] Kavelaars, A.T, Bloom, E., Claus, R., Fouts, K., and Tuvi, S. "Electronic Logbook for Space Systems Integration & Test Operations" **IEEE Transactions on Aerospace and Electronic Systems**, vol.45, no.1, pp.167-178, Jan 2009, Menlo Park, CA.

[23] HARVARD BUSINESS REVIEW, The House of Quality by John R. Hauser and Don Clausing, May-June 1988.