IAC-09-D1.5.11

CONFIGURATION MANAGEMENT IN NANOSATELLITES PROJECTS: EVALUATION OF DELFI-C³ AND CONSEQUENT ADAPTATION FOR DELFI-N3XT.

M.E.A. Genbrugge
Delft University of Technology, The Netherlands
Genbrugge.Mattias@gmail.com

R. Teuling
Delft University of Technology, The Netherlands
Teuling@gmail.com

J.M. Kuiper
Delft University of Technology, The Netherlands
J.M.Kuiper@tudelft.nl

G.F. Brouwer
Delft University of Technology, The Netherlands
G.F.Brouwer@tudelft.nl

J. Bouwmeester
Delft University of Technology, The Netherlands
Jasper.Bouwmeester@tudelft.nl

ABSTRACT

The Delfi-C³ nano-satellite was launched in April 2008 and the development of its successor, the Delfi-n3Xt, is ongoing. Both of these projects are conducted by students from the Delft University of Technology as a corporation between the Aerospace Engineering Faculty, the Faculty of Electrical Engineering, Mathematics and Computer Science and several industrial partners. This paper provides a description and evaluation of the configuration management that was applied at the Delfi-C³ nano-satellite project. It was done by comparing the initial intentions with the final implementation. More specifically, the interface control, configuration control, requirements management and the documentation management are discussed. An evaluation of the configuration management of the Delfi-C³ project was appropriate, as based on the lessons that were learned an improved configuration management approach for the Delfi-n3Xt project had to be defined. The need for simplification and increased consistency on all the aspects of the configuration management were the main conclusions of this evaluation. Basically, deciding on a configuration-strategy and controlling its consistency as early as possible creates a framework on which the rest of the systems engineering tasks can be build. To provide some insight in the practical implementation of the approach, it will be presented how it was used for the initial design of the Electrical Power System (EPS). Finally, the benefits and remaining issues will be discussed.

FULL TEXT

1. INTRODUCTION

A growing number of universities are recognizing the possibilities of small satellites, more specifically nanosatellites (satellites with a mass in the order of 1-10 kg). Such satellites can be both capable and low-cost. Especially since a structural format was standardized in the form of a CubeSat: a CubeSat has a 100x100x100 mm volume. This format was developed by California Polytechnic State University and Stanford University in 1999. Standardization of the outer dimensions and conformance to certain preset regulations enables a simplified acceptance process by the launch providers, who provide piggy-back launch opportunities. This standard effectively led to a low cost and accelerated accessibility to space. This is an ideal formula for universities or small companies.

Consequently, Delft University of Technology initiated its own nano-satellite program in November 2004 by starting the development of the Delfi-C³ which is a triple unit CubeSat (a stack of three CubeSats). On April the 28th 2008, the first satellite built in the Delfi Programme, has been successfully launched and has been operating well. The three payloads of Delfi-C³ [Fig. 1.1] are Thin Film Solar Cells (TFSC) from Dutch Space, the Autonomous Wireless Sun Sensor (AWSS) from TNO and a transceiver that has been built by the university itself. This transceiver is the Radio Amateur Platform (RAP) that has a linear transponder function for use by the radio amateur community.
Apart from providing the Dutch space industry with a platform for the qualification of novel micro-technologies, the programme provides students with the opportunity to prepare optimally for their careers as aerospace engineers. It is important to point out here that it was entirely developed, assembled and tested by more than 70 students (work-ratio students/staff is 3 to 1) over this three year period from the Faculty of Aerospace Engineering and the Faculty of Electrical Engineering, Mathematics and Computer Science.

The Delfi Programme has entered a new phase on October the 23rd (2007) with the goal of developing a second nano-satellite. It is named: Delfi-n3Xt [Fig. 1.2]. By using the design heritage and operational experience of its predecessor, the performance of the bus-systems can be increased. Increasing the capabilities of the bus enables one to fly more demanding and interesting payloads. The Delfi-n3Xt has three major bus advancements compared with the Delfi-C3: an electrical power subsystem with energy storage to ensure operations during eclipse, a high data rate link and three-axis active attitude control subsystem. With such a bus, up to five payloads will be supported:

- Micro-Propulsion System (T3µPS) developed by the Dutch Technological Research Institute (TNO), the Technical University of Twente and Delft University of Technology.
- Multi-functional Particle Spectrometer (MPS) from Cosine Research BV.
- Solar cells Degradation Measurements (SDM) from DIMES.
- Highly efficient transceiver (ITRX) by ISIS BV.
- Radiation tolerant implementation of commercial solid-state memory cards (FLASH-cards) for space applications (SPLASH), provided by the Dutch Aerospace Institute (NLR).

With all this hardware, the challenge is to remain within the envelope restrictions of the triple-unit CubeSat.

The development of complex systems by students requires a dedicated systems engineering approach. These educational projects often have specific challenges: workforce discontinuity, inexperienced people, schedule-constraints, lack of consistent documentation, etcetera. Also, the acceptance of an approach by the rest of the team is often based on the practicality and usefulness of the chosen approach. Taking all these factors into account and correspondingly tailoring the systems engineering approach in a pragmatic way before starting the development of the satellite avoids inconveniences and inefficiencies in later stages of the project.

This ‘tailoring’ of the systems engineering approach has been done during the Delfi-C3 project. In section 2.1, it will be explained how this was done. As the Delfi-C3 got launched and performed (and keeps on performing) its mission successfully, the time had come to review the efficiency of the effort (section 2.2). This evaluation led to recommendations (section 2.3) on the way systems engineering could to be applied on future student satellite projects, though this is done without providing a detailed description on the specifics of this effort.

Based on these recommendations and taking into account the specifics of the Delfi-n3Xt Project (section
3.1), an alternative systems engineering approach was developed, which was largely centred on a more thorough and unambiguous configuration control of the project (section 3.2). The preliminary design of the Electrical Power System (EPS) is used as an example to demonstrate the effectiveness of the approach (section 3.3). It is an excellent test-case, as the distributed nature of the EPS requires an increased need for consistent configuration control.

Finally, the next steps to take, a critical evaluation and some retrospective comments about the nature of the student project are provided in the section 4.

2. SYSTEMS ENGINEERING IN THE DELFI-C³ PROJECT

Note that as the author was only involved in the Delfi-n3Xt project, section 2 is primarily based on research that was done on the project documentation of the Delfi-C³ project and the information provided by the Delfi-C³ team members.

2.1. Systems Engineering Approach

The general systems engineering as it is defined in standards and applied in large ‘professional’ projects is not straightforwardly applicable to a student project with the ambition to build a nano-satellite. The specifics of such a student project differ substantially from such large projects. The procedures are just too cumbersome for a student project, which has several specific constraints which inhibit their effective execution: a human resource constraint and a time constraint.

2.1.1 Human Resources

The ideal team-structure is shown in [Fig. 2.1]. One understands that to form a complete team, quite a lot of people were needed. In reality, this ideal composition was never really achieved, due to the following reasons:

- The impossibility to schedule the arrival of new students that want to work on the project.
- The impossibility to select students on the basis of a specialization.
- The constantly changing team-composition due to the student’s personal planning.

The rapid fluctuation in team composition (students graduate, have to study for exams, internships, etc…) leads to problems of continuity in the design, and has a large impact of the systems engineering activities.

The responsibilities on the project were distributed over the team members. Each subsystem had a responsible engineer dedicated to it. But initially, it was intended to centralize the upper level systems engineering. Two persons would be responsible for different aspects:

The project manager:
- phasing and scheduling,
- organization and work breakdown,
- design data and documentation management,
- publicity and outreach,
- golden notes (informative document describing all the project issues to the team),

...
The systems engineer:

- functional & operational analysis,
- requirement definition and verification,
- system breakdown,
- configuration management,
- interface control,
- manufacturing and Integration,
- detailed design,
- technical resource budget control,
- assembly, integration and verification implementation and management,
- operations planning and execution,
- coordinating of post launch activities,
- …

Note that although the systems engineering effort was centralized, most of the activities were done in close cooperation with the rest of the team. Especially due to the constant human resource problems, the tasks of the main system engineer were mostly distributed over the team and the staff.

Support of the team is indispensable for the effectiveness of a systems engineering approach. This is related to another aspect that is inherent to a student project that is being led by students: a lack of authority, as both the project manager and the systems engineer were students. They had certain responsibilities within the project, but not the authority to enforce their personal opinions. This had as a positive side-effect that decisions were taken based on consensus. Therefore students were stimulated to come up with thoroughly thought plans to be able to persuade their peers. For a team of inexperienced students, this is educationally a very valuable and stimulating approach.

2.1.2 Project Phases/Scheduling

One of the parameters that determine the success of the mission is time, or better the correct managing of it. Defining and controlling the project’s phasing and scheduling is a difficult but critical task. The phasing is normally a responsibility of the systems engineer, as it is tightly related to the technical design process. But in the Delfi-C³ project, it was chosen give all those responsibilities to the project manager. This involves also the setting up of meetings and reviews. The fluctuations in the team-composition posed a challenge to the schedule of the project. The time- constraints were a cause for the abandoning of several systems engineering activities during the course of the project. This time-constraint is clearly an important constraint on the amount and type of systems engineering activities that can be performed.

2.1.2 Systems Engineering Philosophy

The systems engineering effort was based on the input from these previously described project constraints combined with the available standard methods in for example: systems engineering courses, INCOSE handbooks [Ref. 2] and ECSS standards.

The setup and development of SE tools must have a positive effect on the overall process efficiency. However, the impact on the short term development progress cannot be too high [Ref. 3].

The two previously mentioned aspects cover the sensitive balance between the usefulness of the time-investment in the system engineering and the technical...
engineering where it is all about. The short development time (2.5-3 year) and the worsening aspect of the constant shortage of manpower create a large constraint on the amount of systems engineering that can be applied.

In the technical design, the system complexity should be reduced as much as possible. The SE process and tools should reflect this strategy by avoiding too complex procedures and extensive intervention from the management team [Ref. 3].

This aspect is also closely related to the two previously described aspects. It is all about fine-tuning the procedures and approaches in order to obtain an as efficient approach as possible, related to the time constraint. The level of formality of the approach was perceived as a matter that could be used to ‘tweak’ the process. It was realized [Ref. 3] that a strict and formal approach leads to a well documented project, enabling the traceability of the design-choices. This is an important feature in a student project, as the team has a very changing composition. But on the other hand, formal documentation was believed to be very time-consuming. Based on this, it was decided to limit the amount of documentation wherever possible without comprising the consistency of the design [Ref. 3]. Formal documentation was considered less needed, due to the integrated systems development approach that was applied. All students of the team were located in the same room in the Faculty of Aerospace Engineering. On the same floor, all the technical facilities were available to the team. It must be noted that in contrast with this, also a small group of students was working in the faculty of electrical engineering on the electronics.

2.2. Evaluation

It is hard to determine the relation between systems engineering activities and the success of a project. Therefore, the effectiveness of a systems engineering activity will be determined based on how and if it was used in the project.

At the Delfi-C project, the intentions were to apply a flexible, on-the-fly systems engineering approach, as was explained in the previous section. This means that the formality of the systems engineering activities was to be limited. The pragmatic approach was to start using it only when it was really needed at the time. So no complete technical management plan was conceived in the beginning of the project. But during the development multiple systems engineering activities were initiated, based on needs arising in the project.

The main observation that can be made is that the intended activities were described theoretically in detail, but problems occurred with the practical implementation of them.

To understand this, one needs to go back to the basics of the technical management. The intention of technical management is to support, maintain and control the design process. The initial approach was to control a baselined design. This description of the baselines was fixed in the documentation. Therefore controlling the baseline became rather a matter of controlling the documentation. This resulted in the complex documentation structure with different document-types, document control sheets, approval sheets and etcetera. So the focus was not on the design, but rather on the documentation describing this design. This generated a lot of extra work, for both the systems engineers and the other team members. On top of that, duplication in the documentation effort was introduced by the necessity to describe the baseline in standard formats for each review that came up. Another source of duplication was the generation of theses. Also, as new team-members joined the team, they introduced their own (sometimes inaccessible) documentation, without updating the documentation generated by their predecessors. Combining this with the scattered nature of the documentation due to a large number of different document-formats and the non-existence of a unique well-maintained documentation archive, the control of the design by controlling the documentation was largely ineffective.

Configuration management was introduced later in the project to get a better grasp on the design. The configuration management became a merger of the initial baseline control using the documentation structure and a systems breakdown in configuration items. Again an extensive control system was foreseen, this time by using a forum. In the end, this structure was also not maintained. It is the opinion of the author that this complex structure inhibited the acceptance of the approach by the rest of the team. When later in the project the time pressure rose, this complex structure documentation structure was abandoned completely.

The baselines were not only described by the configuration breakdown, but later in the project also by a more hardware oriented breakdown. As the configuration breakdown was based on a functional description, both breakdowns were not easily interchangeable. Different systems engineering activities became based on different breakdowns. The configuration breakdown was used to allocate the requirements, while the hardware oriented breakdown was used to manage the interface control.

As the requirement definition remained limited to high-level configuration items, the use of the generated requirements remained limited. If they would have been
based on a hardware oriented breakdown, they would become more concrete and therefore useful in the verification phase. Although the interface definition and control process was initially based on the hardware breakdown, this process never really got off the ground. Such an extensive centralized process was not supported by the rest of the team. Especially for such processes that affect all team members, acceptance by them is critical for the implementation. Decentralized initiatives by team members themselves did work out, for example software ICDs.

Design-maturity control is a typical activity that serves the need to control the design. But the baselines were defined by configuration items, which was an impractical functional breakdown. This is another example of wrongly applied systems engineering tools.

2.3. List of Recommendations

From all the observations that were made in the previous section, a few important recommendations can be made:

- Only one breakdown should be used for all the different technical management activities. It can act as the backbone of all of the systems engineering efforts.
- The systems breakdown and control should start as soon as possible in the project. To keep the breakdown useful in later stages of the project, the breakdown should be hardware oriented. Functional breakdowns can be useful for development purposes, but they should not be used for technical management purposes.
- The design should be controlled based on a description of the design by using the hardware oriented system breakdown, not based on documentation-structures.
- Requirement, interface, budget and maturity control must be based on the same system breakdown.
- The documentation structure has to be unique in its form (no hard-copies or different locations of file-storage).
- The documentation structure must be as simple as possible. The number of documentation-types must be limited. And the structure of the documentation can be linked to the system breakdown.
- To ease the transition of responsibilities and increase the usefulness of the documentation, their format (layout and structure) should be subjected to strictly maintained project-standards. At the same time, this will increase readability for all the team-members.
- Responsibilities for documentation and system breakdown items are to be linked to team-functions, not to people. When people leave the project, there documentation becomes the responsibility of their successors.
- Centralized versus distributed systems engineering responsibilities: The design of the system is done by the complete team, which consists of inexperienced students. Giving a single student as a systems engineer the responsibility to do all of the systems engineering tasks in a centralized manner is considered too much for the capabilities of that single student. The strength of a student-team is based on the team, not on the individuals. If one considers the team as a network of communicating elements, systems engineering can be distributed over those elements as long as it remains clear for everyone where the responsibilities lay and no ambiguity in the communication exists.
- Avoiding ambiguity should be the main task of the systems engineer. He could be the stimulating and controlling factor in different systems engineering tasks, like controlling the systems breakdown and interface and requirements definition. But the definition should remain a team-responsibility. In this way, systems engineering becomes a useful team activity rather than a chore subjected to them. This responsibility distribution can lead to effective systems engineering, rather than the generation of large academic systems engineering efforts.
- Budgets are an example of how such an approach works. They can perfectly be controlled and maintained by the team-member responsible for the specific subsystem, without the involvement of the main systems engineer. As long as they all adhere to the same systems breakdown and the responsible team member can find out who is responsible for which part of the system. Controlling this would still be the responsibility of a main systems engineer.
- Requirement definition must be based on the same systems breakdown. The usefulness of this task for later phases of the development (e.g. verification) is determined by their level of concreteness. Though this does not mean that one needs to go too far: requirements need to describe the system, not the design.
- Making one central system that describes all interfaces is inefficient and hard to maintain. It should be attempted to define the different
interface-types that are relevant to control, and consequently customize different interface control process depending on the interface-type. Responsibility for the interface definition and control can also be allocated to the team-members that already have the responsibility of a subsystem that is related to the interface-type. This is also the most efficient way to use the experience of the different team members.

As a final general comment to the previous recommendations, it must be said that although the tasks and the responsibility to initiate them are distributed over the team, it should remain the responsibility of a main systems engineer to check on the correct implementation of the work that is to be done.

3. SYSTEMS ENGINEERING IN THE DELFI-N3XT PROJECT

3.1. Delfi-n3Xt Project Specifics

In this section, specific aspects of the Delfi-n3Xt project are discussed. They are aspects in which the Delfi-n3Xt project differs substantially from the Delfi-C3 project. Therefore they are important for the definition of the general systems engineering approach, as the approach will have to be set up by keeping them in mind. Note that these aspects were already present before the systems engineering approach of the author was initiated.

3.1.1. Team Composition

The team-structure of the Delfi-n3Xt project has evolved quite naturally. Until now, there is a project manager (staff), a general systems engineer (student), a set of subsystems engineers (master students working on their thesis) and academic staff who offer support to the project. Each one has its own function and responsibilities in the project.

Within the team, no straightforward hierarchy is defined. Rather lines of influence are present. The staff-members supervise most of the team-members for their theses. The project manager has more in general more experience than a student-member and also supervises some. A subsystems engineer partly supervises a bachelor student. But between the systems and the subsystems engineers, no hierarchy at all is present. The systems engineer can not impose certain systems engineering practices to its team-members, but has to convince them of their use.

A responsibility that all team-members share is the responsibility of their documents: it remains with the author. The moment an author leaves the team, this responsibility of the living document must be assumed by its successor. Then they are responsible for all sets of documentation that are created on their specific subsystem / component.

3.1.2. Documentation Management & Control

The main activity that has been done in the project as an overall systems engineering effort is documentation management. Important improvements compared to the Delfi-C3 project have been made, mainly related to the documentation structure and control.

An effective documentation structure has been created, with the main objective to create an unambiguous and unique data-base of all the information that is gathered and generated in both the Delfi-n3Xt project as well as in the Delfi-C3 project. As the latter project its documentation was rather chaotic, Delfi-C3 project had some work to sort out the relevant information. This is an important task, if the new team wants to profit to the maximum from the work that is done by the Delfi-C3 team.

The unambiguity and uniqueness of the documentation is greatly enhanced by using a central project disk to store all relevant documentation, software and other files. Due to the integration of all networks of the Delft University of Technology into one network, this disk is reachable from every location. To add information to the disk, an approval process via the project manager is required. This ensures the quality of the information that is added. Once the information is on the disk, the documentation becomes protected from changes and is given a unique identification code.

To avoid splitting and increase controllability of the documentation, only few document types are allowed. Additionally, this forces team members to keep on updating the same document and not start new documents all the time again. The idea is to create ‘living’ documentation: this kind of documentation remains updated, and therefore will always reflect the current state of the design. Old versions will be archived for traceability reasons. If this process is correctly maintained, the project disk will always contain up to date versions of documentation.

To increase the readability and consistency of the documentation, their format and lay-out is strictly standardized. Rules and guidelines on how to produce comprehensible documentation are fixated.

In contrast to the large amount of data control sheets that had to be added to each document that was generated in the Delfi-C3 project, the Delfi-n3Xt project has only one: the ‘document control sheet’. This greatly reduced the workload and increases the cooperation by the team-members.
3.2. A Configuration Management Approach

The solution to effective systems engineering in the Delfi-n3Xt project is an approach based on configuration management (CM). Its definition is based on:

- Lessons learned from the Delfi-C³ technical management activities (section 2).
- Initial needs and constraints of the Delfi-n3Xt project (section 3.1).
- Standard systems engineering approaches (general literature).

The Delfi-n3Xt project is of course very similar to the Delfi-C³ project. The approach will therefore be based primarily on the evaluation the Delfi-C³ technical management approach. Of course, the Delfi-n3Xt project also has its specific needs. Although the general mission objectives are the comparable, the technical complexity of the design is a lot higher. There are more payloads, active attitude control, upgraded communication systems, etcetera. This all puts higher demands on the organization and technical management side of the Delfi project. Standard systems engineering techniques are the third pillar of the Delfi-n3Xt systems engineering approach. Of course, the specifics of student project will inspire the need of tailoring these standardized approaches. A subtle equilibrium between the extensiveness and usefulness must be found to create an efficient systems engineering approach.

The applied CM approach leads to a restricted range of tools and documents (Table 3.1), which are all fully accepted and agreed on by the team. Acceptance of the approach by the team-members is one of the main parameters that lead to success, especially within a student team. The systems engineer can exert no pressure based on hierarchy on its peers. Therefore, each part of the approach is based on long discussions with team fellow members. The result is a balanced approach that maintains the equilibrium between formality that is desired from a systems control point of view and the informality that is desired by the team-members. This conforms naturally to the social dynamics of a student team. Table 3.1 provides an overview of the list of such documents so far. If one considers the column with responsible team members, the distributed nature of the responsibilities in a CM approach becomes clear.

The basis of the CM approach is the representation of the system by an ‘AND’-tree of Configuration Items (CI): the Configuration Tree (Fig. 3.1). Each CI has been assigned to a responsible team-member. The subdivision of the system is based on the following principles:

- CI’s are conveniently defined as items that are worthwhile to control separately [Ref. 2].
- Disciplinary homogeneity within a CI is important.
- A CI should not be a combination of independent components.
- The final responsibility of the internal management of a CI should be unambiguously attributed (preferably to one person), otherwise it is very likely that in the end nobody will take the responsibility.
- A division along hardware-components (or software) is preferred.
- The combination of all the CI’s must envelop the complete Delfi-n3Xt system.

The basis of the CM approach is the representation of the system by an ‘AND’-tree of Configuration Items (CI): the Configuration Tree (Fig. 3.1). Each CI has been assigned to a responsible team-member. The subdivision of the system is based on the following principles:

- CI’s are conveniently defined as items that are worthwhile to control separately [Ref. 2].
- Disciplinary homogeneity within a CI is important.
- A CI should not be a combination of independent components.
- The final responsibility of the internal management of a CI should be unambiguously attributed (preferably to one person), otherwise it is very likely that in the end nobody will take the responsibility.
- A division along hardware-components (or software) is preferred.
- The combination of all the CI’s must envelop the complete Delfi-n3Xt system.

### Table 3.1: Documents and tools that are part of the configuration management approach.

<table>
<thead>
<tr>
<th>CM Document/Tool</th>
<th>Responsible Team Member</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Management Rationale</td>
<td>Systems Engineer</td>
</tr>
<tr>
<td>Systems Engineering Team Members Manual</td>
<td>Systems Engineer</td>
</tr>
<tr>
<td>Configuration Tree</td>
<td>(Sub)Systems Engineer</td>
</tr>
<tr>
<td>Requirements Management Tool</td>
<td>(Sub)Systems Engineer</td>
</tr>
<tr>
<td>Requirements Creation Form</td>
<td>(Sub)Systems Engineer</td>
</tr>
<tr>
<td><strong>Documentation:</strong></td>
<td></td>
</tr>
<tr>
<td>Standard Documentation</td>
<td>All</td>
</tr>
<tr>
<td>Delfi Documentation Manual</td>
<td>Systems Engineer</td>
</tr>
<tr>
<td>Document Input Sheets</td>
<td>All</td>
</tr>
<tr>
<td>Delfi Standard List of References</td>
<td>Project Manager</td>
</tr>
<tr>
<td>Delfi Drawing List</td>
<td>STS Subsystems Engineer</td>
</tr>
<tr>
<td><strong>Interface Control Documents (ICD):</strong></td>
<td></td>
</tr>
<tr>
<td>SDM Payload ICD</td>
<td>Payload Interface Engineer</td>
</tr>
<tr>
<td>MPS Payload ICD</td>
<td>Payload Interface Engineer</td>
</tr>
<tr>
<td>SPLASH Payload ICD</td>
<td>Payload Interface Engineer</td>
</tr>
<tr>
<td>T3UPS Payload ICD</td>
<td>Payload Interface Engineer</td>
</tr>
<tr>
<td>ITRX Payload ICD</td>
<td>Payload Interface Engineer</td>
</tr>
<tr>
<td>Mechanical and Structural ICD</td>
<td>STS Subsystems Engineer</td>
</tr>
<tr>
<td>Electrical Wiring ICD</td>
<td>ADCS/EPS Subsystems Engineer</td>
</tr>
<tr>
<td><strong>Budgets:</strong></td>
<td></td>
</tr>
<tr>
<td>Link Budget</td>
<td>COMMS Subsystems Engineer</td>
</tr>
<tr>
<td>Power Budget</td>
<td>EPS Subsystems Engineer</td>
</tr>
<tr>
<td>Data Budget</td>
<td>CDHS Subsystems Engineer</td>
</tr>
<tr>
<td>Volume Budget</td>
<td>STS Subsystems Engineer</td>
</tr>
<tr>
<td>Mass Budget</td>
<td>STS Subsystems Engineer</td>
</tr>
</tbody>
</table>

8
Controlling the system configuration is not only done by controlling the contents and structure of the configuration tree, but also the demands (requirements) of the CI’s and their interfaces. The content of the CI’s is described in the standardized documentation, such as technical notes, reports, etc… Another advantage of the living-document concept presents itself at this stage: the same information can be found in the same official document (even if it has undergone some changes and updates), carrying the same name, during the complete design process.

Fig. 3.1: An impression of a part of the configuration tree describing a part the communication subsystem (more specifically its S-band transceivers and antenna system).

The main thought behind the setup of the CM is to maximise use the qualities within the team of students as much as possible. This by not using a centrally ‘pulled’ approach, but a distributed pushed and centrally guided approach. Around this basic framework of the configuration of the satellite, the subsystems engineers can devise their own systems engineering tasks based on their own needs. As long as the structure of the configuration and its corresponding standards are reflected within the new systems engineering tools or documents, interoperability of these tools is possible. Based on research on the Delfi-C3 project it is understood that this interoperability is critical for the continuity of these specific efforts. For example, on the base of a configuration item number, one should be able to find all the related requirements, drawings, relevant documents, interface definitions, etc… If this is not possible, the document or tool will become useless for the team. Delfi-C3 lacked this main structure and thus the continuity. Consequently, few of the systems engineering effort that was initiated centrally in the beginning of the Delfi-C3 was maintained until the end of the project.

The function of the (general) systems engineer in the Delfi-n3Xt configuration management approach is therefore not to define or initiate all these budgets or interface documents, but rather to control their conformance to the configuration structure and standards. In this way the function of the (general) systems engineer is not overloaded with responsibilities, and can therefore also do effective engineering work on the project. Considering the fact that students are scarce goods, this is definitely advantageous from a human resource point of view. It also makes the function of the systems engineer less critical. This function could be assumed by any of the subsystems engineers working on the project, providing all the team members with the opportunity to gain some experience in systems engineering and thus maximizing the educational value of the project.

3.3. The Preliminary Design of the EPS

The EPS provides, stores, regulates, distributes and controls the electrical power of the satellite. This subsystem will function as an example on the practicality of the previously described system engineering tools.

3.3.1. EPS configuration management

The design of the EPS started with the top-level requirements from the mission objectives. These requirements were used in the high level trade-offs of the complete satellite design. These initial trade-offs resulted in a preliminary architecture of the EPS, i.e. the use of a Solar array, a 12 V regulated bus, distributed power control system and the need for energy storage. At this point in the project the configuration management approach was used to organise the detailed design. Fig. 3.2 shows the EPS breakdown in the configuration tree. The configuration items are chosen such that they are pragmatic in use and in total represent the complete system-hardware. The top level choices are visible in this tree, for example, the distributed nature of the EPS results in the many local EPS’s (SAT.2.1.4.1 to SAT.2.1.4.9) situated on every other system in the satellite. Parts of the EPS that are outsourced to external project partners were deliberately kept as a single configuration.
item. Examples are the Global EPS (SAT.2.1.3) and the Physical Power Bus (SAT.2.1.5). These items are given functional and interface requirements that are agreed upon with the external partner.

The other configuration items are designed and manufactured by the students themselves. These items are broken down into smaller items that are suitable working packages for students. For example, the Energy Storage (SAT.2.1.1) breaks down into the battery cells, control, wiring and bracket. Each item gets lower level requirements that are used in the detailed design. The battery bracket for instance, has the volume constraints of the satellite, the interface requirement with the structure and the outer dimensions of the battery cells. With this reduced list of requirements, a mechanical engineering student can design the bracket without losing time in discussions with system engineers working on other subsystems.

3.3.2. EPS Documentation Management.

In the preliminary design of the EPS the number of separate documents was kept at a minimum. A top level EPS document was created which contains all the up-to-date information about the EPS. This document contains the following items:

- The EPS top level requirements
- The EPS architecture trade-offs
- EPS breakdown into configuration items
- Lower level requirements and design of all the EPS configuration items

This top level document is updated throughout the project by the EPS system engineer. No other documents are created with information on the design of the EPS and it is accessible for all students. Although this is a rigid structure it does provide transparency on the status of design and information can easily be found and accessed.

4. CONCLUSIONS & RECOMMENDATIONS

The unique attributes of a student project as the Delfi-n3Xt Project require a unique system engineering approach. The concept of using a configuration tree and configuration items presents a clear overview of the responsibilities of each part of the satellite and effectively helped the systems engineers to develop a coherent system, while providing them with an overview of the rest of the system. The system works in a way that greater consistency and extensive simplifications are realised in the systems engineering effort. Overall the method does provide means of a structured design approach and transmitting the knowledge from the graduating student to the new student has been simplified greatly compared to Delfi-C3.

Problems did occur when system engineers had different interpretations of the system engineering tools and how to use them, especially when multiple team members are involved in the usage of one specific tool or a systems engineering effort. For example during the requirements definition, different ‘styles’ and quality-levels existed, depending on the effort that the student was willing to put in it. The main systems engineer must
therefore have the ability to enforce or rather convince his peers to introduce consistency in their effort. As the system is concurrently designed with the development of the systems engineering approach, by the time the approach is crystallizing, already inconsistencies have been introduced in the different approaches of the student-engineers working on the project. In the ideal case the tools should be ready and documented before the start of the project.

Most importantly, one needs to realise that it remains mainly an educational project. Even when things go wrong (maybe especially in such occasions) the students will benefit from the experience. Introducing staff to handle the difficult issues might benefit the course of the project, but not the students.

6. REFERENCES

