A Model Based Thermal Systems Engineering Approach

- SECESA 2012 -

17-19 October 2012

Alameda Campus of IST / Technical University of Lisbon
Lisbon, Portugal

Johannes Gross[1], Christian Messe[1], Stephan Rudolph[1]

[1]Institute for Statics and Dynamics of Aerospace Structures, University of Stuttgart
Pfaffenwaldring 27, 70569 Stuttgart, Germany
Email: {gross, messe, rudolph}@isd.uni-stuttgart.de

INTRODUCTION

Several severe problems in space industry as pointed out at prominent lieu at the Virtual Spacecraft Design (VSD) presentation in May 2012 at ESA Noordwijk are originating from inconsistent system data, late problem detection and difficult information handover between stakeholders. The idea of a central data model (e.g. the VSD initiative) to represent all system engineering data in one model storage can therefore be seen as an attempt towards resolving the aforementioned problems. Due to such a digital representation of the design parameters the traditional systems engineering (SE) methods can be brought to a model-based engineering (MBE) level. Thus model-based systems engineering (MBSE) is the next logical step in this ongoing digitization effort. When thinking further about the MBSE approach and its potential evolvement in the future, the current triangle of **methods**, **processes** and **tools** in today's space systems engineering however must be reconsidered. Due to the enhanced properties of digital models the MBSE method can now be applied in a much more comprehensive and streamlined way in the design process.
An example of one of the new **tools** enabling such an advanced use of a model-based design paradigm is the Design Compiler 43 [1] which is used to compile a graph-based design language on the basis of UML (Unified Modeling Language). By supporting the model-based systems engineering **method** by means of graph-based design languages, the **processes** in satellite design are however as well subject to change. Traditionally, satellite design phases as commonly used in industry (i.e. phases 0, A-D) are characterized by different model fidelity. We want to show in this paper that the models from the later design phases can now be made available in earlier design phases when using design languages. Thus it is possible to profit excessively from modern front-loading techniques and further shorten the design process while steadily increasing design model quality and reliability.

Convention: UML Class names are written in *Italic*
UML Attribute names are written in `courant`.

PROBLEM DESCRIPTION

One explanation why the current process of creating high fidelity models is so time-consuming may be found by looking at the tools that are used in satellite design. Typical commercial Engineering Suites for CFD or FEM are designed for a huge amount of different communities from several engineering disciplines. This allows their developers to invest time and money to create clear and simple user interfaces. When it comes to thermal analysis of satellite structures for example, the situation however is different: available Software Suites like ESATAN-TMS, or it's American counterpart NASA TRASYS/SINDA G, are designed for small communities, which forces their developers to concentrate their capacities to satisfy the high standards of software stability and validity needed for aerospace applications. In this context, user interfaces for those rather exotic kinds of software are naturally far less advanced than those of "big" software suites, forcing system designers to spend many working hours into rather perfunctory tasks of model creation and adaptation. It is our belief, that these tasks can be far more automatized and simplified by improving the relation between abstract model-based engineering and concrete modelling in modern design tools, which are set on top of the classical validated engineering suites. This belief is underlined by recent publications ([2], [3], [4], [5]), where a design language also comprehends detailed information about the system in a reusable form.
In this work it is shown, that also the amount of manpower (in terms of man hours) required for model creation can be reduced dramatically by the means of a design language. The prerequisite for saving manpower is a higher level information representation in a design language. From this representation, the detailed analysis models in the different engineering applications can be automatically derived. In this work, the basic mechanisms for the derivation of a thermal simulation model from a design language representation are explained. The design language used to exemplify this derivation is developed within the PhD-project of the first author. It is a hypothetical satellite mission for the detection and monitoring of forest fires called FireSat. The FireSat mission is described in more detail in literature (e.g.

[5], [2], [3]). In the design language, the geometrical and functional aspects of this satellite are modelled. These aspects can be combined to generate many aspects of the thermal simulation executed in a commercial software (ESATAN-TMS in this case). The paper explains first the modelling of the geometry in UML and the additional thermal attributes required for a simulation. For a more detailed simulation of the thermal properties, the integration of functional aspects, such as the power consumption of the different electronic boxes is shown. From these aspects FORTRAN code is generated to simulate the power production of the solar array in this example. The paper concludes with a display of the results of a detailed thermal simulation of the FireSat.

GENERAL APPROACH

The basic idea of a design language is that all needed design recipes can be implemented in an automated routine, so that ideally both geometry and functionality of the system can be deduced directly from it's system requirements. Of course, such an ideal design language does not (yet) exist, and automatically generated models will always require adaptation and evaluation by human beings. On many (lower) levels however, perfunctory tasks can be automatized. For example, if the mission requirements call for a specific orbit and downlink rate, there is a finite design space with sensible combinations of antennas, amplifiers and so on, from which the system designer must decide if an optimum of mass, power consumption or something in between of both should be achieved. At the base of lodged design recipes and a predefined set of possible hardware combinations, an optimal result is suggested by the design compiler. In this work, we focus on the relation between the geometry modeling, which describes the physical appearance of the satellite, and the functional modeling, which depicts power and heat budget for the designed system.
After the user has created the implicit definitions of functionality and geometry in the design language, the compiler creates an explicit UML model and exports all relevant information to a format that can be interpreted by a commercial engineering suite. In the case of ESATAN, the geometry is exported in the scripting language ESARAD, and the functionality exported into a MORTRAN code is generated (an extended FORTRAN77 variant). The creation of the high level model in the design language in this example is based on a design language for geometry.

GEOMETRY MODELING

In [7] the modelling of geometry in a design language is explained in greater detail. For this work, it is sufficient to mention just the most important elements used in the geometry model. In Fig. 1 an excerpt of the geometry class diagram is given. The topmost class, *TopologyElement*, defines the topological `sub` links of *Components* and their *Positions*. *Components* can have as children *Shapes* or other *Components*. If a *Component* has a *Shape* as child, it represents a Part. If a *Component* has another *Component* as child, it represents an Assembly (or Product). A *Position* can be placed between a *Component* and its child *Component* and thus the child will be translated by the values `dx`, `dy` and `dz` and rotated by the angles `phi`, `theta` and `psi` given in the *Position* instance in degrees.
Shapes can be `shapes` or `primitveShapes` or both to a *Component*. This allows for the creation of two parallel geometry models with different model fidelity. A Component can have a `shape` for the geometry visualisation in e.g. OpenCascade or CATIA and a different `primitiveShape` for the thermal model representation. Below the *Shape* there are three exemplary geometrical primitives. The first is a simple *Cuboid* with the dimensions `lengthX`, `lenghtY` and `lenghtZ` along the three axes. The second is a *Paraboloid* defined by the attributes `focalLength`, `height` and `thickness`. The third is a *Cylinder* with the `radius` and the `height` given. All attributes are in millimetres.

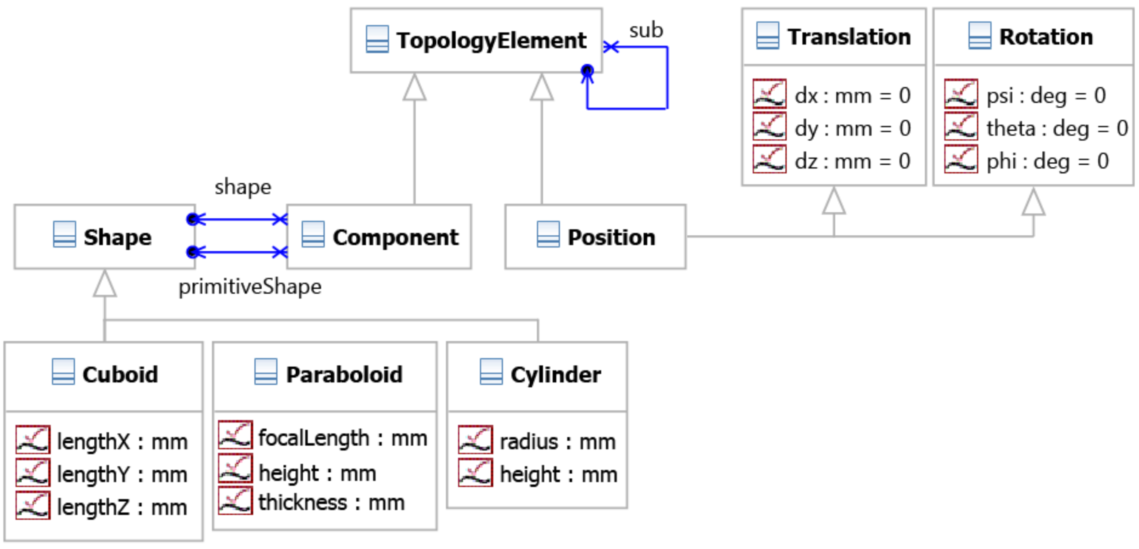

Fig. 1. Class diagram for the geometry model in UML

In the work of [4] a UML model for the definition of a thermal simulation was developed. In Fig. 2 an extract comparable to Fig. 1 of the geometry model is given. Since the thermal simulation is based on a surface model the central element in this model is a *SHELL*. It contains also the concept of assemblies since a *SHELL* can contain other *SHELLs* by the `linkSHELL`. The *SHELL* can also be translated by a *TRANSLATE* and rotated by a *ROTATE* instance.

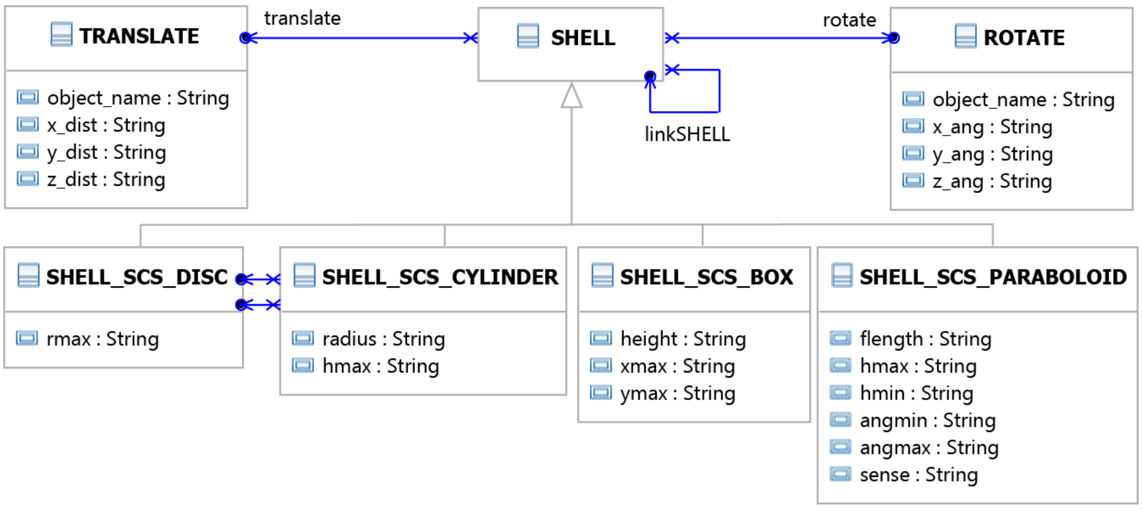

Fig. 2. Class diagram with some geometric elements of the thermal model

In the lower part of Fig. 2 some of the primitives also available in ESATAN are shown inheriting from *SHELL*. The *SHELL_SCS_BOX* is comparable to the *Cuboid* in the geometry class diagram. The `height` equals the `lengthZ` of a cuboid and `xmax` and `ymax` compare to `lengthX` and `lengthY`. The *SHELL_SCS_PARABOLOID* compares to an *Paraboloid* in the geometry model. The `flength` is the `focalLength` and `hmax` is the `height` of an *Paraboloid*. The additional parameters of the model are `hmin` for a lower boundary of the height and angles `angmin` and `angmax` to express the cut-out of a parabloid. The `thickness` of the *Paraboloid* is not given in the *SHELL_SCS_PARABOLOID* since it is a surface model. For the Cylinder in the geometry model, in the thermal model multiple instances are created. First an *SHELL_SCS_ CYLINDER* is created. This cylinder is open at the ends by default. Therefore, for every *Cylinder* two *SHELL_SCS_DISCs* are generated and transformed to the ends of the cylinder.

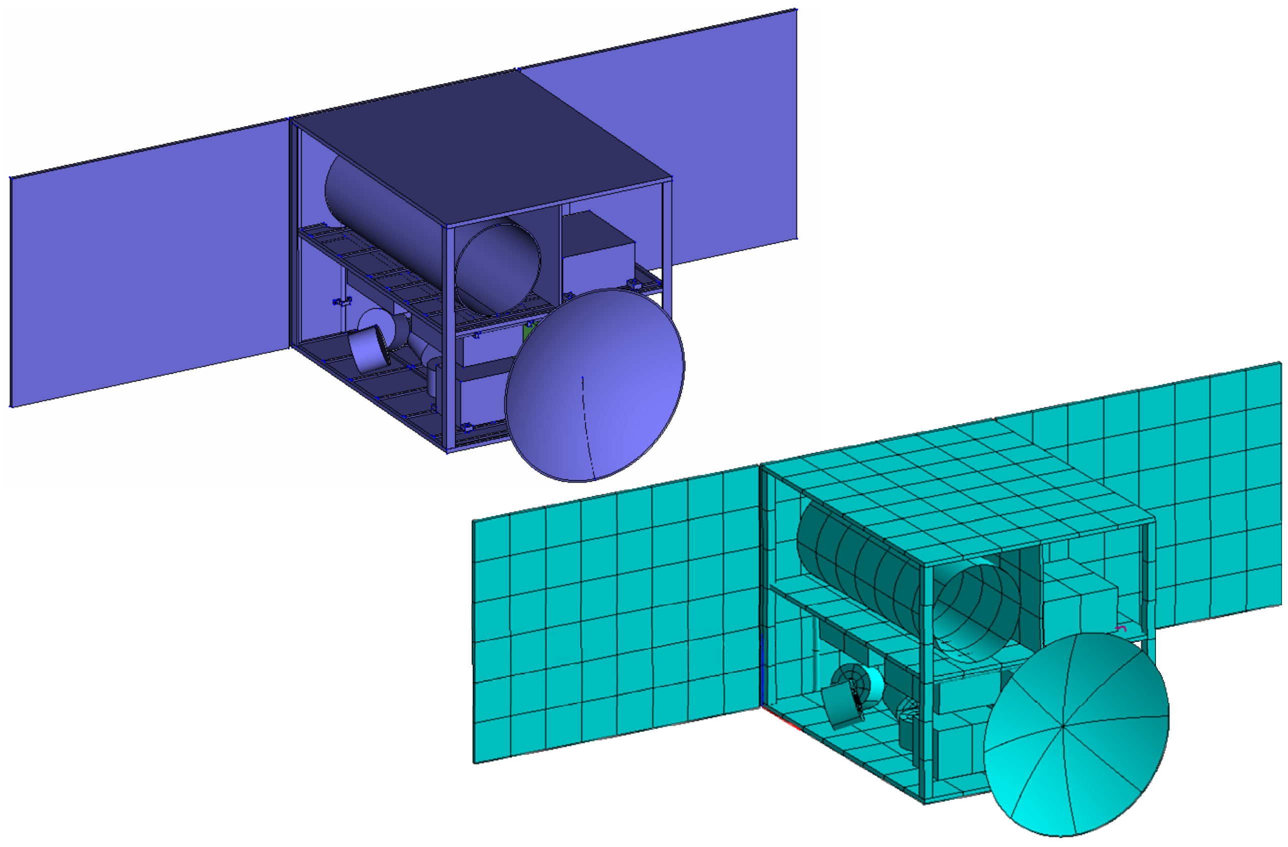

Fig. 3. Geometry Model and thermal model generated from the design language

With the primitives shown in Fig. 2 it is already possible to create a thermal geometry model as shown in Fig. 3 on the right hand side. On the left hand side the according geometry model is shown. The geometry model includes some more details as for example the decks in the satellite have stiffening corrugations and the boxes have mounting links. These details are not exported to the thermal model since they are not relevant to a thermal simulation.

THERMAL MODELING

For modelling the specific parameters of a thermal simulation the UML model from [4] contains further classes. In Fig. 4 a few more classes for the definition of a *SHELL* are given. According to the model, a *SHELL* has a *BULK_MATERIAL*, a *MESH* and two sides, *SIDE1* and *SIDE2*. The *BULK_MATERIAL* specifies the assumed thickness of the surfaces for the calculation of the heat capacity (`thick`). It has also a link `bulk` to an instance specifying the material properties for the *SHELL*. The *MESH* class specifies the number of thermal nodes on the *SHELL* in the different axis directions. The *SIDE1* and *SIDE2* classes give with `nbase` a base node number from which the nodes are counted by an `ndelta` increment. Furthermore the two sides specify the optical properties of the surfaces and indicate if the side is active in the simulation by a Boolean value.

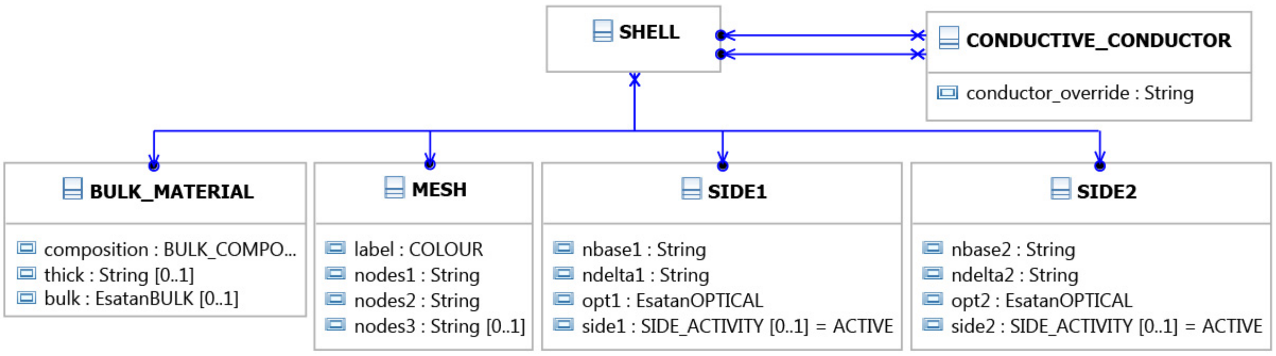

Fig. 4. Class diagram of the thermal model in UML

When the geometry and material definition for the thermal model is established, further boundary conditions such as contact conductance have to be considered. For the automatic generation of such boundary conditions the different semantic layers of the design language can be used. The satellite design language for example consists of different sub-languages. For the mounting of the satellites´ electronic boxes on the primary structure, the satellite design language uses a mounting language. In this mounting language, a *MountingPlane* has *MountingBoxes* (see Fig. 5). Since all elec-

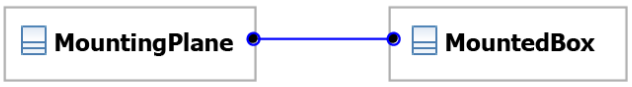

Fig. 5. Excerpt of the mounting language

tronic boxes are mounted by this mechanism on the different decks of the satellite, it is now possible to identify all boxes and their mounting plane (i.e. the side of the deck) automatically:
From the geometry language the local coordinates of the *MountedBoxes* on the *MountingPlanes* can be retrieved. From these coordinates, the node numbers where the *MountingBoxes* are located on the *SHELLs* representing the mounting planes. When the node number is retrieved, a conductive conductor can be generated. A conductive conductor represents the contact conductance between the box and the deck can be modelled. By this simple mechanism, the box can be mounted mechanically as well as thermally to the deck in one step.

FUNCTIONAL MODELING

For the definition of system-wide budgets a satellite-system language is given. In this system language the elements for a power budget are specified. The basic element of this budget is represented by a *PowerElement* which specifies the attribute `power` in Watts for every power consumer. Every satellite box which consumes power, inherits the attribute `power` from this class. In Fig. 6 an example of the on-board computer *OBC* inheriting from *PowerElement* is given.

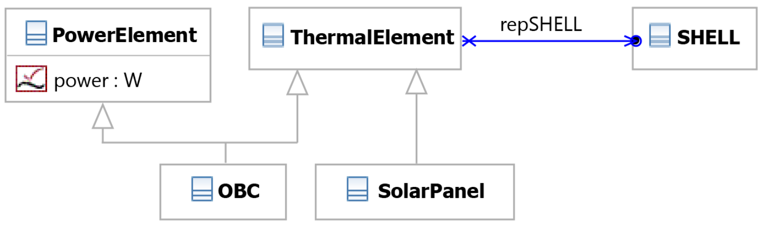

Fig. 6. Geometry Model as generated from the design language

The code for the generation of the thermal model is now able to identify, that the *OBC* consumes the given amount of `power`. For the identification of the according *SHELL* the *OBC* inherits also from *ThermalElement*. For all *ThermalElements*, a `repSHELL` link is set when the thermal geometry representation is generated. Thus the code can navigate over the `repSHELL` link to the *SHELL* of the *OBC* and generate a boundary condition for the thermal simulation with the `power` value as radiated heat from the *OBC SHELL*.
Another functional aspect in this example is the generation of code for the power simulation. The basis therefore is also the `repSHELL` link from the ThermalElement. In Fig. 6 on the right hand side, the SolarPanel class is shown. This class is instantiated for every solar panel of the satellite. In the generation of the thermal simulation now FORTRAN code is produced, with all thermal nodes of the solar array listed. An excerpt of this code is shown in listing 1. The

```
# ------------------------------------------------------------------------------
# SOLAR_POWER calculation
#
# Q_SOLAR is combined with ETA_SOLAR (ca. 27%) the power which can
# be provided by the solar array.
# ------------------------------------------------------------------------------
      SUBROUTINE SOLAR_POWER LANG = MORTRAN # Calculate Power of SolarArray
C SOLAR POWER
        Q_SOLAR = 0.0
C SOLAR PANEL 1
        Q_SOLAR = Q_SOLAR + QS3154
        Q_SOLAR = Q_SOLAR + QS3155
        Q_SOLAR = Q_SOLAR + QS3156
        Q_SOLAR = Q_SOLAR + QS3157
        ...
        ...
        ...
        Q_SOLAR = Q_SOLAR + QS22212
        Q_SOLAR = Q_SOLAR + QS22213
        Q_SOLAR = Q_SOLAR + QS22214
        Q_SOLAR = Q_SOLAR + QS22215
C Power by solar cells
        POWER_SOLAR = Q_SOLAR * ETA_SOLAR
      RETURN
      END
C
```

Listing 1: Example of Code generated for the simulation of the produced power.

numbers `QS3154, QS3155 etc.` represent the node numbers of the solar array. The value `ETA_SOLAR` is the efficiency of the solar cells. The incident solar radiation is multiplied by `ETA_SOLAR` to get the power produced by the array. In the exemplary thermal simulation an detailed analysis of the whole power budget of the satellite is accomplished. The power consumption of the electronic boxes was transferred manually in this example but there is no limitation to easily generate these routines too. By these means, the thermal simulation can be refined with different modes of the satellite to simulate the different operational states in orbit.

The generated algorithm for the power and heat budget model is represented in an activity diagram in Fig. 7. In each time step the available solar power is determined from the solar radiation fluxes given by ESATAN, and it is determined which hardware components should be turned on or off, according to lookup tables which have to be specified for the investigated mission. Once the required power for the current time step is known, the algorithm decides whether the produced power from the solar array suffices to supply the power consumers. If there is not enough power produced, a delta power has to be drawn from the on-board battery. If more power than required is produced, the battery can be charged with the excess solar power. For most power consumers it can be assumed that the requested power is directly converted into heat. One exception is the travelling wave tube amplifier, where most of the energy is radiated in the generated radio signal. The heat fluxes for the Power Converter and Distribution Unit (PCDU), or the

Solar Power Converter Unit depend on the power requested by the consumers and the electrical efficiencies of the units. This power simulation can be modified and improved by the system engineer at will, as long as those modifications can be converted into the MORTRAN syntax. Possible modifications range from the addition of heaters over the simulation of heat pipes to the implementation of detailed battery heating models.

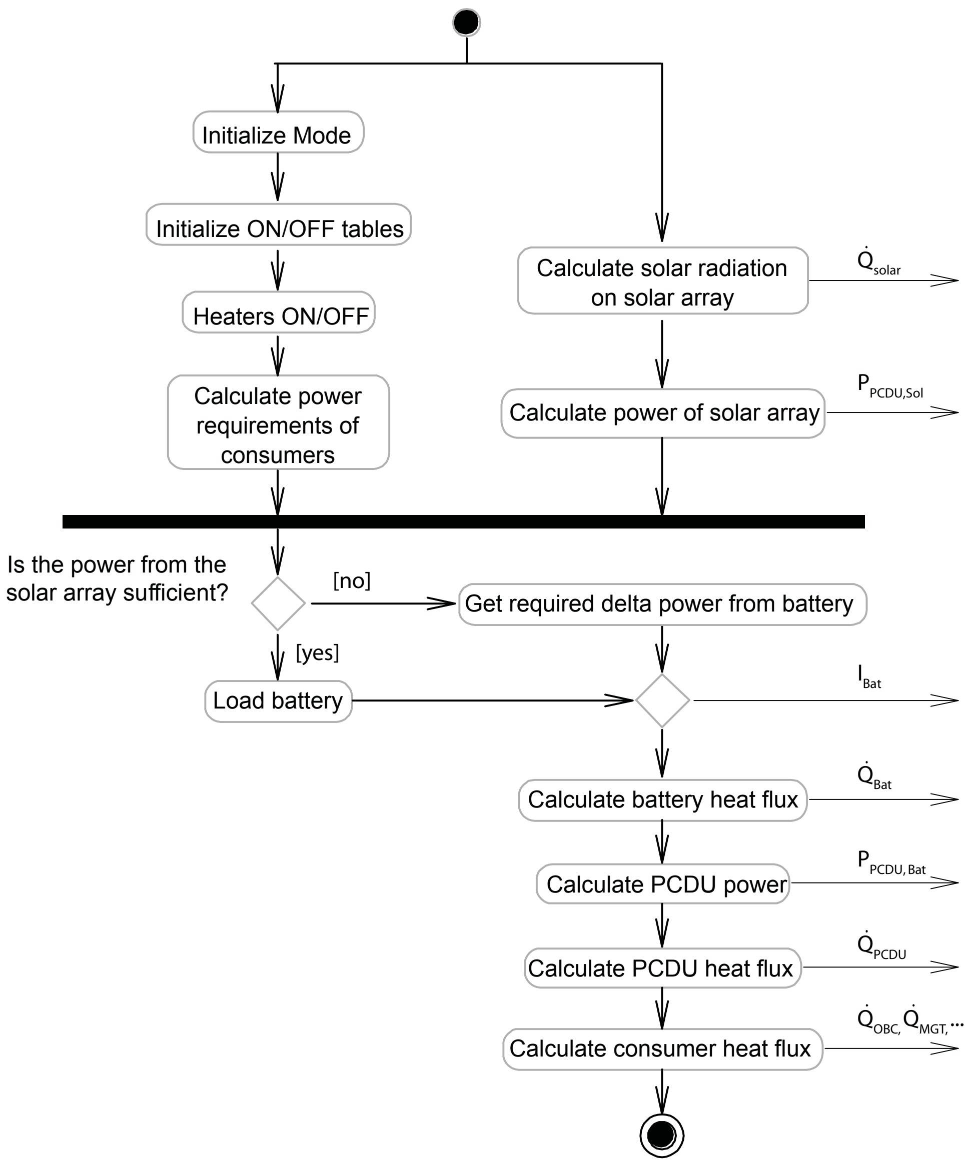
Initialize Mode
Initialize ON/OFF tables
Heaters ON/OFF
Calculate power requirements of consumers
Calculate solar radiation on solar array
$\dot{Q}_{solar}$
Calculate power of solar array
$P_{PCDU,Sol}$
Is the power from the solar array sufficient?
[no]
Get required delta power from battery
[yes]
Load battery
I_{Bat}
Calculate battery heat flux
$\dot{Q}_{Bat}$
Calculate PCDU power
$P_{PCDU,Bat}$
Calculate PCDU heat flux
$\dot{Q}_{PCDU}$
Calculate consumer heat flux
$\dot{Q}_{OBC}, \dot{Q}_{MGT}, ...$

Fig. 7. Activity diagram of the algorithm for the coupled thermal and power simulation

RESULTS AND DISCUSSION

Not claiming to be advanced ESATAN experts, the authors were able create and adapt models within hours, which would have taken at least several days, had they been created manually. Since this work is to be considered as proof of concept, the authors did not have the intention of creating a perfectly working thermal system. For example, the power converter unit turned out to create huge amounts of heat, which has not been expected when the hardware was positioned in the structural frame. This is a simple example of how a high-fidelity model in an early design stage creates sophisticated knowledge of the preliminary design. In a future design language of a thermal system, the system engineer could add one rule, that this power converter unit must be placed close to a radiator. This way, the design language would get better and better the more often it is used, allowing the system engineer to create deeper knowledge of early design variants.

Of course, this speed-up of the design process comes at a certain price: modern information representation and processing techniques find their way now directly in the core processes of engineering, i.e. modelling and processing is no longer done on the MORTRAN/FORTRAN level. Instead, modelling is done one (or even several) levels of abstraction higher by modelling the (sub-)system properties in a graph-based design language in UML format and the appropriate MORTRAN/FORTRAN model is compiled from that. While this seems to be a far too high price for the speed-up in one engineering discipline, the reader should be assured that this applies in any discipline, thus being an universal, multi-disciplinary modelling platform guaranteeing system-wide parameter and model consistency through an automated compilation process. While model compilation is just the first direct consequence of this technology, the full impact of graph-based design languages will show off when further knowledge processing techniques such as model-to-model transformations combined with background (design knowledge) ontologies will be operational as it is currently investigated in the working group of the authors at Stuttgart University.

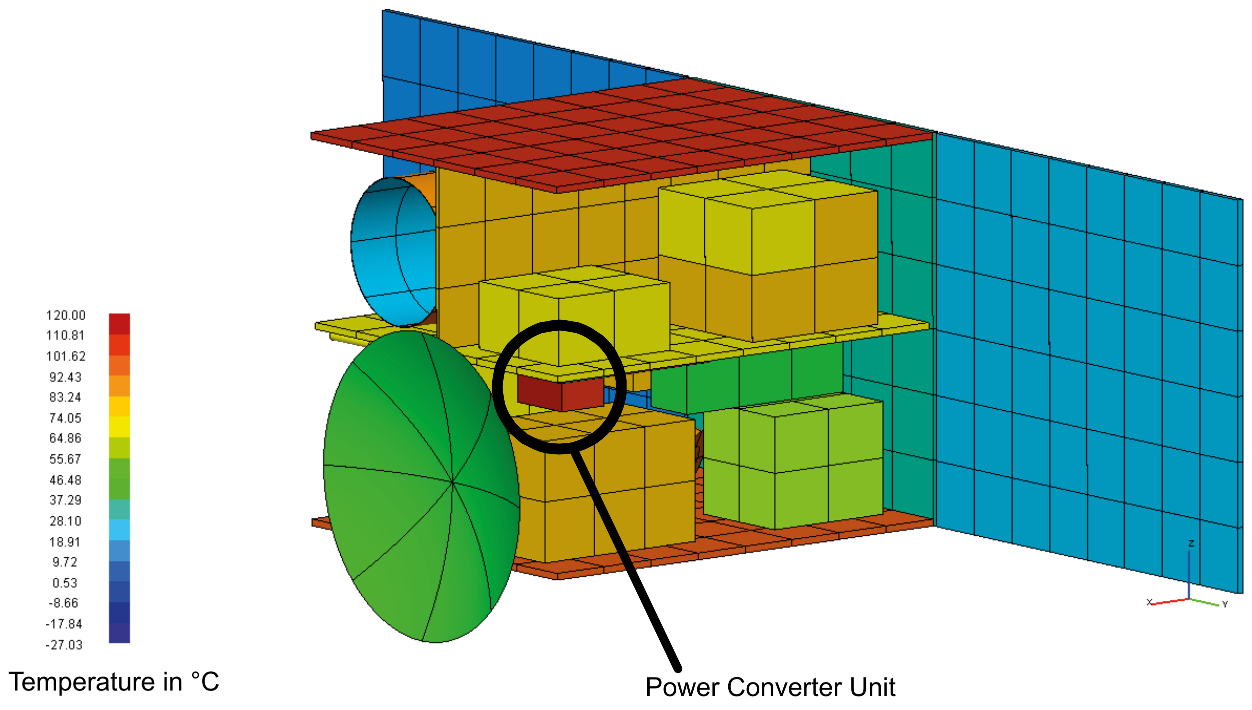

Fig. 8. Result from the generated Radiative Model

CONCLUSIONS

In this work, a proof of concept for the utilization of design languages in thermal systems engineering is given. It is shown that by simple inheritance mechanisms provided by UML a semantically rich engineering model can be created. From the different aspects of this model, relevant information for a thermal simulation model can be automatically retrieved. The possibility to eliminate the tedious task of creating simulation models is demonstrated.
This approach comprehends a high potential of early problem identification by means of multidisciplinary design analysis. This can even be extended when such a design language is coupled not only to a thermal and power budgeting, but also to FEM analysis, AOCS simulation and an automated solution of the integration problems such as packing and wiring (routing).

REFERENCES

[1] R. Alber, S. Rudolph "“43” – a generic approach for engineering design grammars", Proceedings AAAI Spring Symposium ‘Computational Synthesis’, Stanford (2003)
[2] J. Gross and S. Rudolph, "Generating Simulation Models from UML - A FireSat Example", Proc. of the Symposium on Theory of Modeling and Simulation (TMS’12). Orlando, FL (USA) 26-29 March 2012.
[3] J. Gross and S. Rudolph, "Dependency Analysis in Complex System Design using the FireSat Example." Proceedings of the INCOSE Symposium, Rome (Italy), June 2012.
[4] M. S. Kocak, “Erstellung einer Schnittstelle zur generischen Thermalsimulation von Satelliten”, *Diplomarbeit*, Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart 2010.
[5] J. W. Larson and J. R. Wertz, eds.: *Space mission analysis and design*. 3rd ed. El Segundo, California: Microcosm, 1999.
[6] C. Messe, “Entwicklung des Thermalkonzeptes für die Lunar Mission BW1”, *Studienarbeit*, Institut für Thermodynamik der Luft- und Raumfahrt, Universität Stuttgart 2012.
[7] J. Schmidt, “Eine Entwurfssprache für Geometrie”, *Studienarbeit*, Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart 2012.