MODELING AN ATTITUDE AND ORBIT CONTROL SYSTEM USING SYSML

Alessandro Gerlinger Romero
Instituto Nacional de Pesquisas Espaciais (INPE/CSE), S. J. Campos - SP, Brasil, romgerale@yahoo.com.br

Mauricio Gonçalves Vieira Ferreira
Instituto Nacional de Pesquisas Espaciais (INPE/CCS), S. J. Campos - SP, Brasil, mauricio@css.inpe.br

Abstract: *This paper presents an approach for development process of an Attitude and Orbit Control System (AOCS) software for a satellite using SysML (Systems Modeling Language). The development process starts analyzing context diagram, stakeholder and their needs. So it is derived system requirements and measure of effectiveness (MoEs). After this a functional analysis is accomplished using use cases. Constraints and parametric diagrams are described. Physical aspects are modeled and considered to model behavior using sequence diagrams. Finally, model is refined to become a PSM model from software viewpoint, allowing code generation.*

Keywords: *SysML, system engineering, MDA, PIM, PSM, code generation*

1 Introduction

AOCS's (Attitude and Orbit Control System) are often the most complex subsystem on board a satellite (Pasetti and Brown, 2001). The AOCS is a typical embedded hard real-time control system, which main task is to periodically collect measurements from a set of sensors and convert them into commands for a set of actuators. Its nature, with control algorithms, device interaction and software intensive use, makes AOCS a very good candidate to use SysML (Systems Modeling Language).

Previously SysML, many systems engineering processes tend to be document-intensive (a.k.a. document centric) and employ a motley mix of diagram techniques that are frequently imprecise and inconsistent. To address this issue, OMG (Object Management Group) and INCOSE (International Council on Systems Engineering's) have specified SysML. It defines a general-purpose modeling language for systems engineering applications that supports the specification, analysis, design, verification and validation of a broad range of complex systems. These systems may include hardware, software, information, processes, personnel and facilities (OMG, 2010). SysML is defined as an UML 2 profile.

A SysML model can be considered a PIM (Platform Independent model), according MDA (Model-Driven Architecture). And in the simplest form of MDA (OMG, 2003), it can be refined generating a PSM (Platform Specific Model) from software viewpoint. Then PSM can be used to generate, at least, structure for the software, creating classes, attributes and methods. This is the approach followed by current paper to development process of an AOCS. The development process starts analyzing context diagram, stakeholder and their interests in the system. So it is derived system requirements and measure of effectiveness (MoEs). After this a functional analysis is accomplished using use cases. Constraints and parametric diagrams are described. Physical aspects are modeled and considered to model behavior using sequence diagrams. Finally, model is refined to become a PSM model, allowing code generation. Figure 1 shows graphically followed approach.

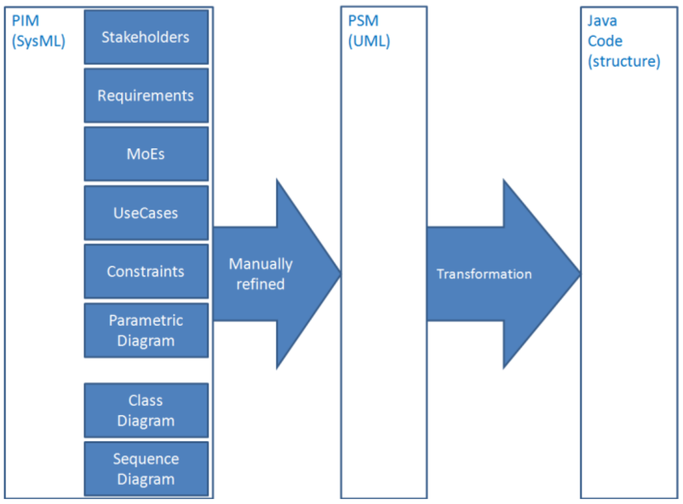


Figure 1. Paper approach to model AOCS.

Next sections will present scope, important diagrams used to analyze and to specify AOCS. Finally, some generated and manually code will be showed.

2 AOCS Scope

As sad previously, AOCS main task is to periodically collect measurements from a set of sensors and convert them into commands for a set of actuators. Current paper simplifies control modeling defined in Moreira (2006) considering bellow assumptions:

1. It uses three-axis technique, using three gyroscopes (sensor) and three reaction wheels (actuator);
2. It has only nominal mode;
3. It only sends telemetry;
4. It does not receive telecommands;
5. It has fault detection functionalities, but there are no fault isolation or recovery procedures;
6. It can be simulated without hardware, so there is a plant simulator. Plant simulator does not consider external forces or space environment, only its dynamic and cinematic;
7. It uses a PID (Proportional-plus-Integral-plus-Derivative) controller.

3 Requirements elucidation

AOCS was modeled starting from a context diagram, see Figure 2., defined using block definition diagram. Block definition diagram is available to show block definitions and follows the graphical conventions of a UML class diagram showing block, their properties and their relationships (OMG, 2010). A Block is a modular unit of system that encapsulates its contents, which include attributes, operations and constraints (OMG, 2010).

Figure 2. defines main relationships from AOCS and others satellite subsystem. It declares scope of current system under development.

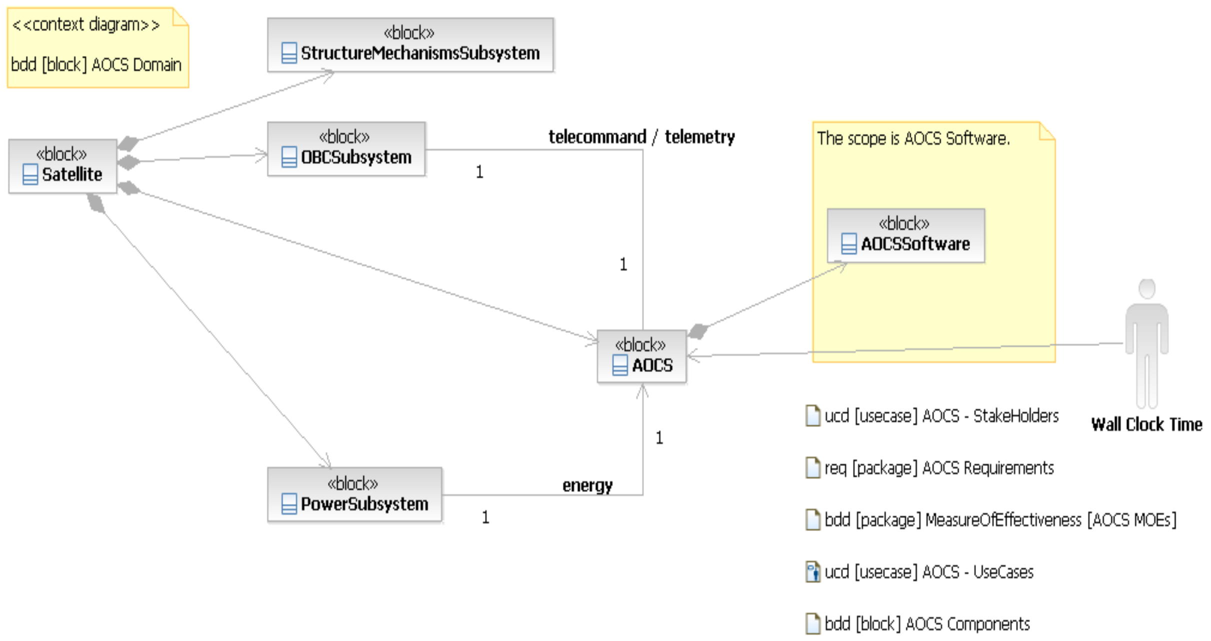

<<context diagram>>
bdd [block] AOCS Domain
«block» StructureMechanismsSubsystem
«block» Satellite
«block» OBCSubsystem
telecommand / telemetry
1
1
The scope is AOCS Software.
«block» AOCSSoftware
«block» AOCS
Wall Clock Time
1
«block» PowerSubsystem
energy
1
ucd [usecase] AOCS - StakeHolders
req [package] AOCS Requirements
bdd [package] MeasureOfEffectiveness [AOCS MOEs]
ucd [usecase] AOCS - UseCases
bdd [block] AOCS Components

Figure 2. AOCS Context Diagram.

After context diagram, stakeholders are identified and documented in a use case diagram, see Figure 3. Main purpose here is to identify and analyze stakeholders needs and their relationships. To describe relationship between stakeholder and blocks a *trace* relationship is used, defining a need. Stakeholders are the individuals or organizations that have concerns about the system (OMG, 2010).

Stakeholders needs generate stakeholders requirements, which are refined into system requirements. A requirement may specify a function that a system must perform or a performance condition that a system must fulfill (OMG, 2010). These system requirements are analyzed and described using requirements diagram, see Figure 4. Modelers can categorize requirements by modifying their predefined properties, which include id, source, text, kind, verifyMethod and risk (OMG, 2010). Modelers can define requirements relationships using composition or *derive* dependency.

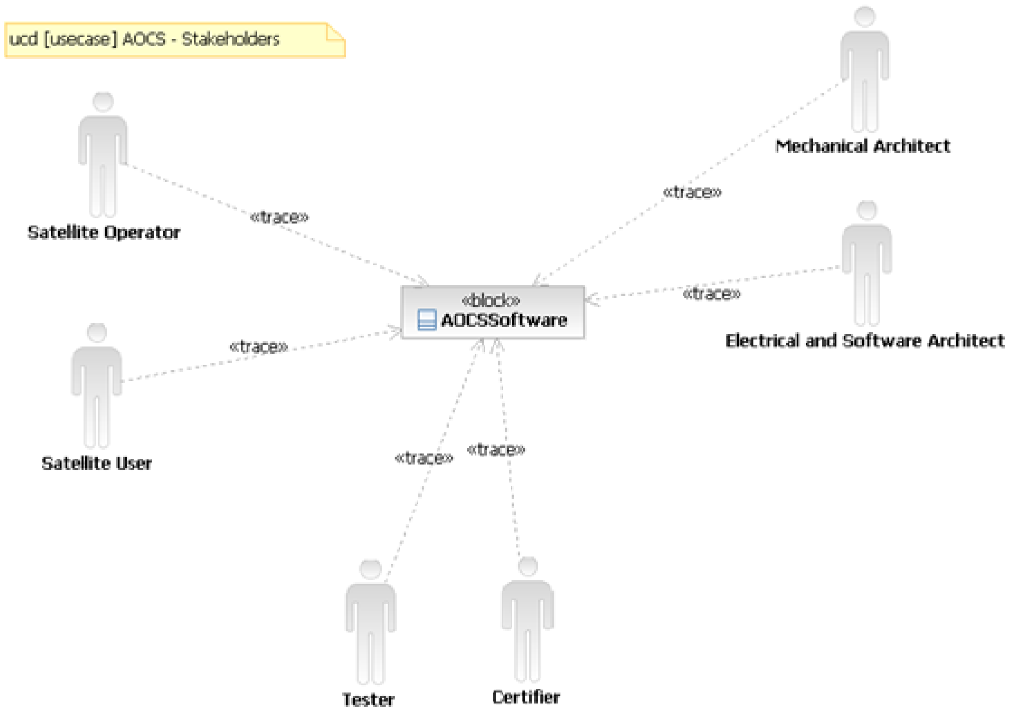


Figure 3. AOCS Stakeholders.

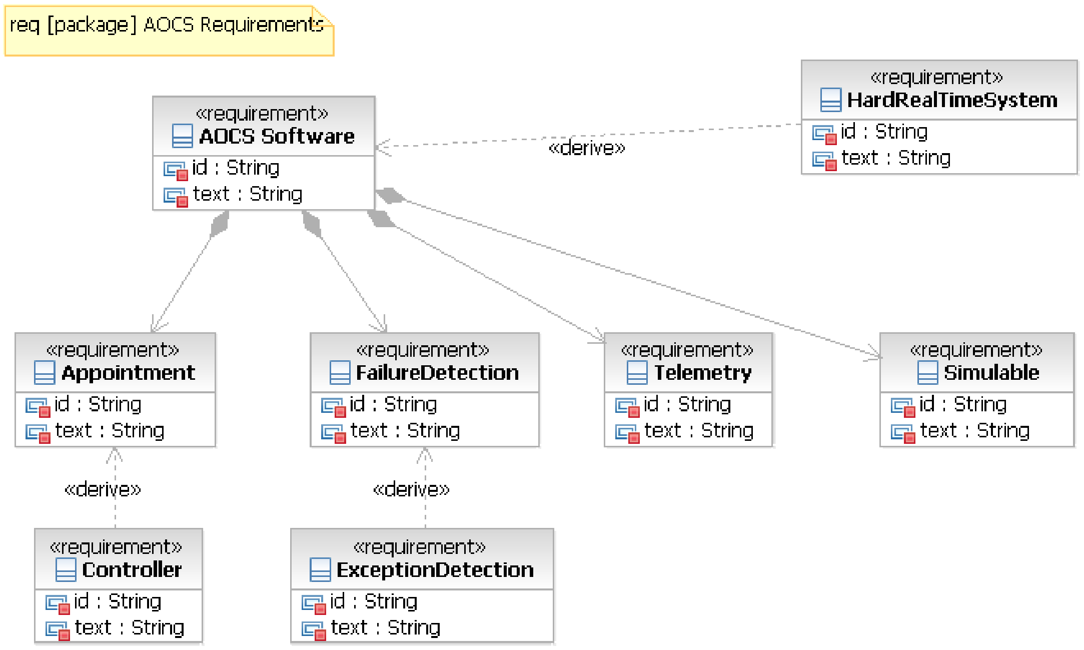

req [package] AOCS Requirements
«requirement» AOCS Software
id : String
text : String
«requirement» HardRealTimeSystem
«requirement» Appointment
«requirement» FailureDetection
«requirement» Telemetry
«requirement» Simulable
«requirement» Controller
«requirement» ExceptionDetection
«derive»

Figure 4. AOCS System requirements.

Trade studies are a basic activity for many system engineers. Given a set of requirements, it is frequently problematic to satisfy all requirements. Consequently, trade-offs must be made to arrive at a design that is optimal in a global sense (OMG, 2010). In order to determine this "global optimum" one typically uses a set of "Measures of Effectiveness" (MoE).

A MoE states an optimization condition that a system must satisfy. Whereas the requirements for a system define the domain of the solution, the solution space, the Measures of Effectiveness drive the solution to a particular region in that space. Each MoE has a weight attribute to reflect its relative importance, a score attribute to capture its value based on the alternative under investigation and the direction of the optimization (OMG, 2010). Figure 5. shows MoE considered in current paper.

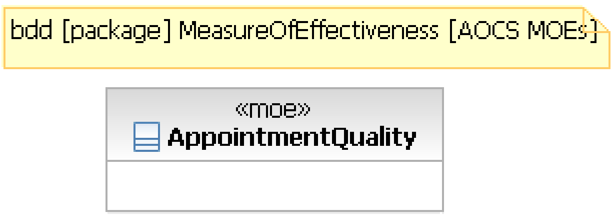


Figure 5. AOCS MoEs.

4 Functional analysis

Now that requirements are defined and agreed, it is time to do functional analysis. What functions system must do to satisfy requirements?

Using SysML, functions are analyzed and specified through use cases. According OMG (2010), a use case diagram specifies actions that a system can perform by interacting with outside agents (actors) to provide service transactions (use cases). Actors may represent users, external systems, or other environmental entities.

Figure 6. shows AOCS use case diagram. As a real time embedded software (RTES) main actor is the Wall Clock Time, clock tick starts the most functions. Functions are divided in four packages, driving next steps on development. There are associations between use cases and blocks that means that these use cases depends on some information provided or required by block. There are *satisfy* relationship between requirements and use cases. A *satisfy* relationship is dependency between a supplier requirement and a client model element that fulfils the requirement (OMG, 2010). There are too *realization* relationship between use cases and collaborations. Collaborations stores class and sequence diagrams for each use case.

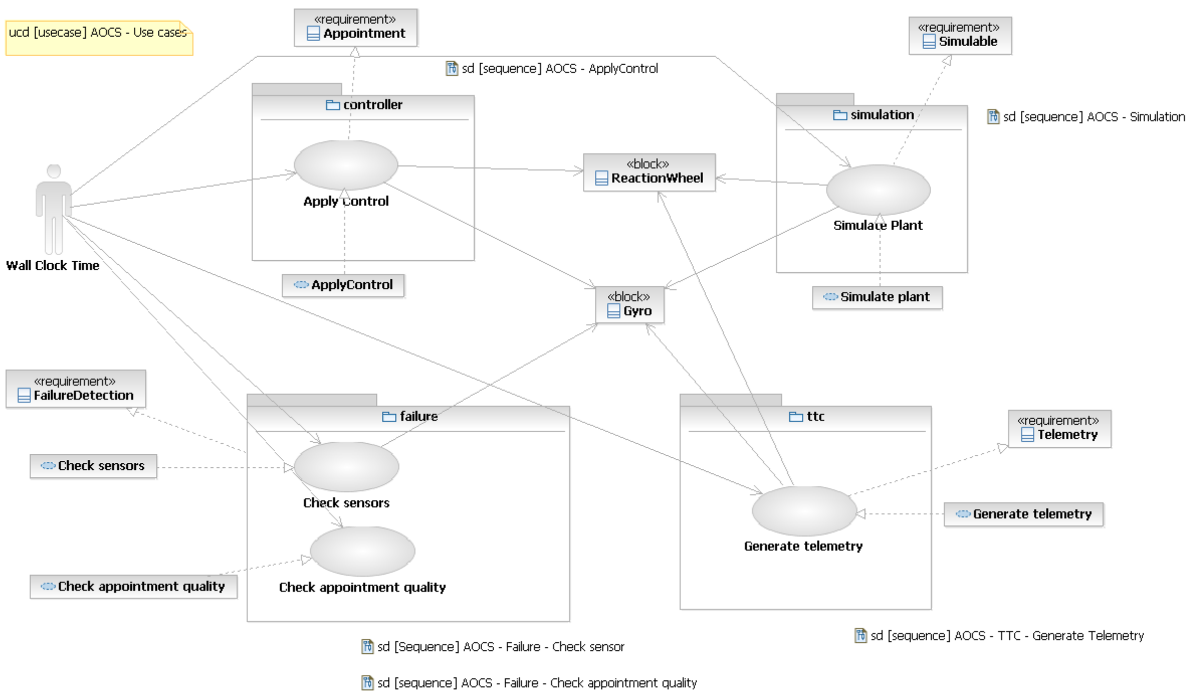

ucd [usecase] AOCS - Use cases
«requirement» Appointment
sd [sequence] AOCS - ApplyControl
«requirement» Simulable
controller
simulation
sd [sequence] AOCS - Simulation
Apply Control
«block» ReactionWheel
Simulate Plant
Wall Clock Time
ApplyControl
«block» Gyro
Simulate plant
«requirement» FailureDetection
failure
ttc
«requirement» Telemetry
Check sensors
Check sensors
Generate telemetry
Generate telemetry
Check appointment quality
Check appointment quality
sd [Sequence] AOCS - Failure - Check sensor
sd [sequence] AOCS - TTC - Generate Telemetry
sd [sequence] AOCS - Failure - Check appointment quality

Figure 6. AOCS Use Cases.

Once use cases are defined, it is possible to analyze and specify collaborations. But first, it is important to accomplish the architectural analysis.

5 Architectural analysis

After functional analysis, architectural analysis aims to identify physical interfaces where energy, goods or data flow between environment and system under development. It is time that a function becomes real, considering interfaces and physical restrictions.

Figure 7. shows each block that is involved on system. Figure 8. models how physical devices are mapped (derived) to software blocks, through real time entities, and defines that these blocks will publish telemetry data.

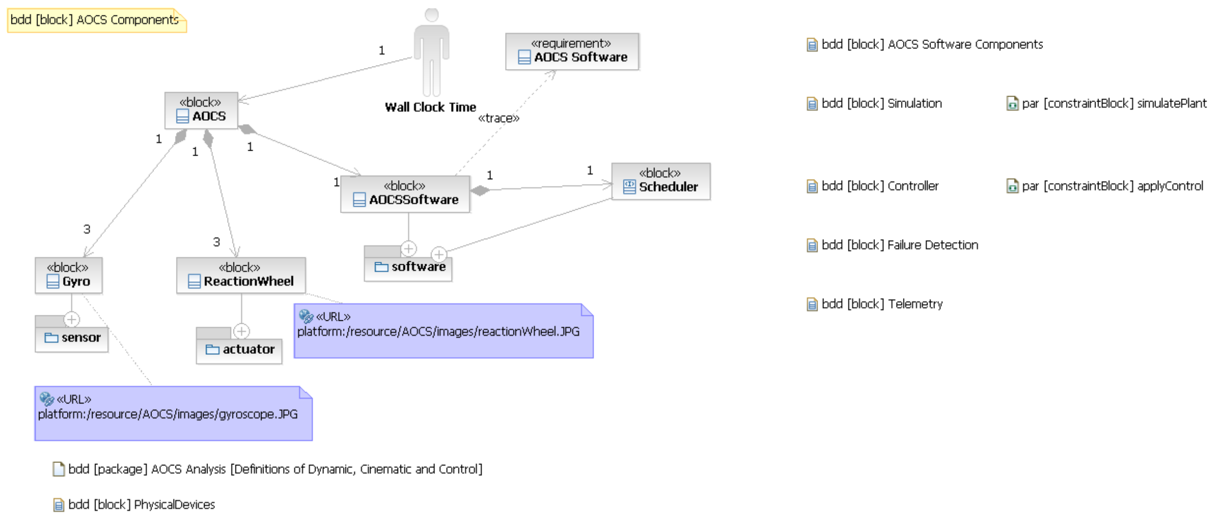

bdd [block] AOCS Components
1
Wall Clock Time
«requirement» AOCS Software
«trace»
«block» AOCS
1
1
1
«block» AOCSSoftware
1
1
1
«block» Scheduler
3
3
«block» Gyro
«block» ReactionWheel
software
sensor
actuator
«URL» platform:/resource/AOCS/images/reactionWheel.JPG
«URL» platform:/resource/AOCS/images/gyroscope.JPG
bdd [package] AOCS Analysis [Definitions of Dynamic, Cinematic and Control]
bdd [block] PhysicalDevices
bdd [block] AOCS Software Components
bdd [block] Simulation
par [constraintBlock] simulatePlant
bdd [block] Controller
par [constraintBlock] applyControl
bdd [block] Failure Detection
bdd [block] Telemetry

Figure 7. AOCS Components.

It is common to use internal block diagram to describe physical architecture. Internal block diagram is available to show the internal structure of a block and follows the graphical conventions of a UML composite structure diagram showing internal structure (parts, ports and connectors) of the subject block (OMG, 2010).

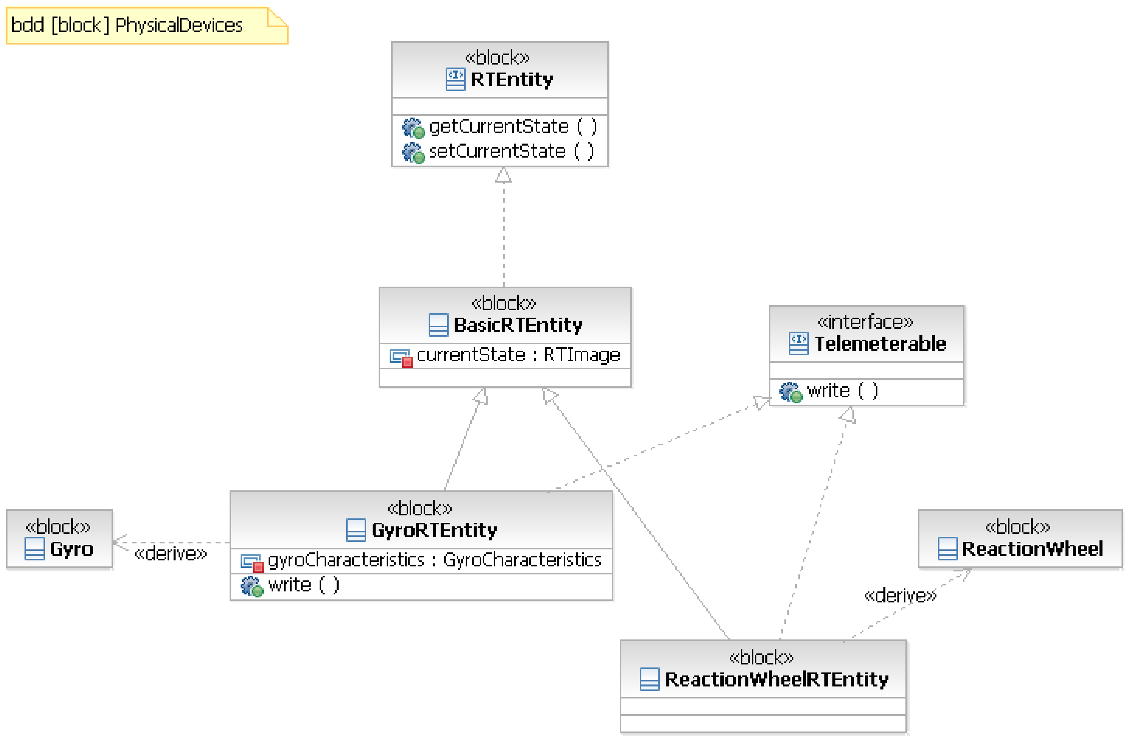


Figure 8. AOCS Physical Devices.

6 Constraint analysis

Modeling a dynamic system, such as AOCS, encompasses describing differential equations. To support this important issue SysML defines constraints. A constraint block includes the constraint, such as {F=m*a}, and the parameters of the constraint such as F, m, and a (OMG, 2010). The constraint can be described either formal statements in some language, or informal statements using text. This expression can include a formal reference to a language in braces, common languages are Modelica (Johnson *et al.*, 2007), MathML (Espinosa *et al.*, 2008) or MARTE-VSL (Espinosa *et al.*, 2008). Figure 9. shows a block definition diagram containing constraints defined on AOCS described here. Constraints are described using s-domain and time domain (Ogata, 1997).

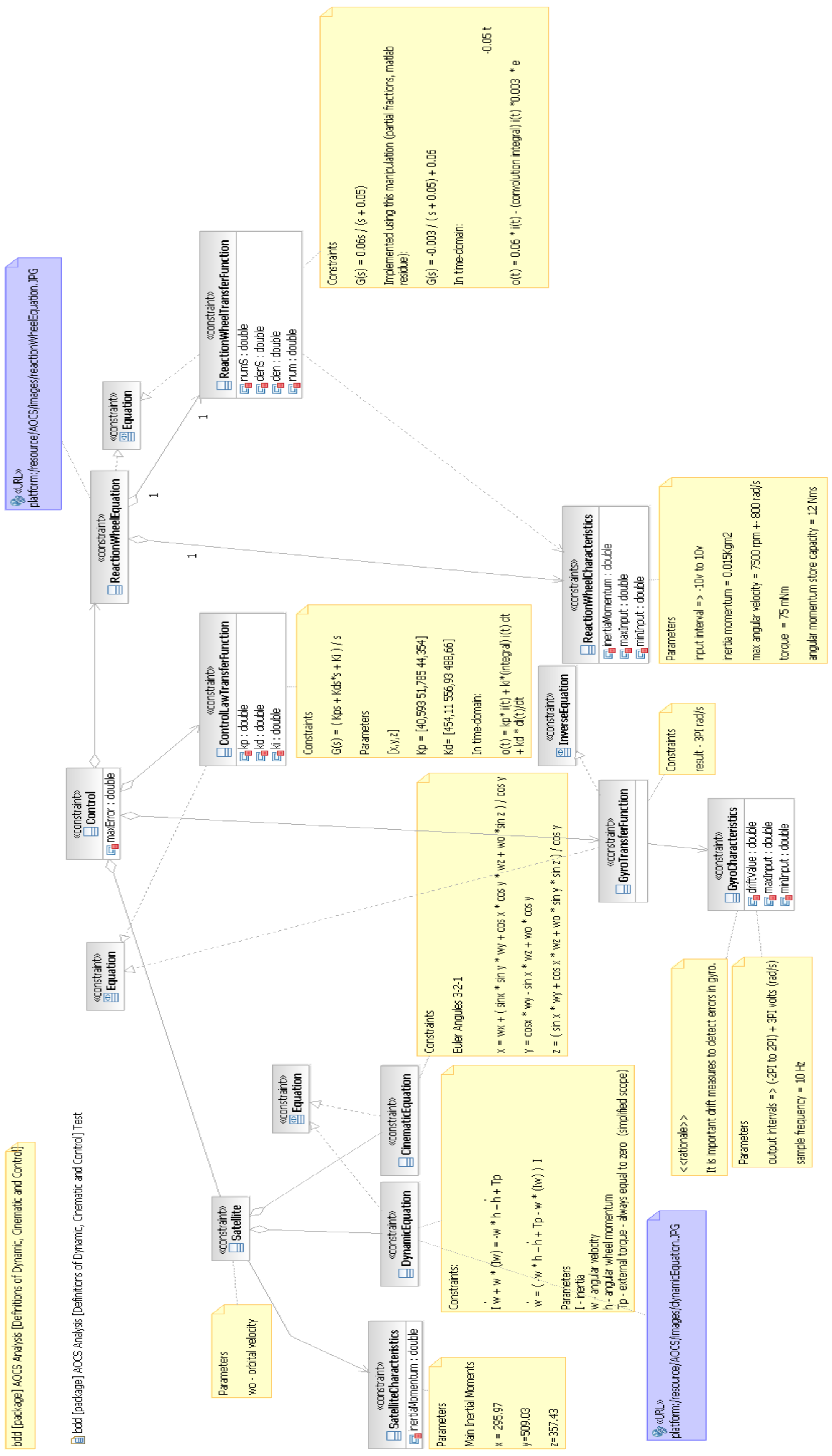

bdd [package] AOCS Analysis [Definitions of Dynamic, Cinematic and Control]
bdd [package] AOCS Analysis [Definitions of Dynamic, Cinematic and Control] Test
«URL» platform:/resource/AOCS/images/reactionWheelEquation.JPG
«constraint» Equation
«constraint» ReactionWheelEquation
«constraint» ReactionWheelTransferFunction
numS : double
denS : double
den : double
num : double
Constraints
G(s) = 0.06s / (s + 0.05)
Implemented using this manipulation (partial fractions, matlab residue):
G(s) = -0.003 / (s + 0.05) + 0.06
In time-domain:
o(t) = 0.06 * i(t) - (convolution integral) i(t) *0.003 * e^-0.05 t
«constraint» ReactionWheelCharacteristics
inertiaMomentum : double
maxInput : double
minInput : double
Parameters
input interval => -10v to 10v
inertia momentum = 0.015Kgm2
max angular velocity = 7500 rpm +- 800 rad/s
torque = 75 mNm
angular momentum store capacity = 12 Nms
«constraint» Control
maxError : double
«constraint» ControlLawTransferFunction
kp : double
kd : double
ki : double
Constraints
G(s) = (Kps + Kds^2 + Ki) / s
Parameters
[x,y,z]
Kp = [40,593 51,785 44,354]
Kd= [454,11 556,93 488,66]
In time-domain:
o(t) = kp* i(t) + ki*(integral) i(t) dt + kd * di(t)/dt
«constraint» InverseEquation
«constraint» GyroTransferFunction
Constraints
result - 3PI rad/s
«constraint» GyroCharacteristics
driftValue : double
maxInput : double
minInput : double
«rationale» It is important drift measures to detect errors in gyro.
Parameters
output intervals => (-2PI to 2PI) + 3PI volts (rad/s)
sample frequency = 10 Hz
Constraints
Euler Angules 3-2-1
x = wx + (sinx * sin y * wy + cos x * cos y * wz + wo *sin z) / cos y
y = cosx * wy - sin x * wz + wo * cos y
z = (sin x * wy + cos x * wz + wo * sin y * sin z) / cos y
«constraint» Equation
«constraint» CinematicEquation
«constraint» DynamicEquation
Constraints:
I w + w * (Iw) = -w * h - h + Tp
w = (-w * h - h + Tp - w * (Iw)) I
Parameters
I - inertia
w - angular velocity
h - angular wheel momentum
Tp - external torque - always equal to zero (simplified scope)
«URL» platform:/resource/AOCS/images/dynamicEquation.JPG
«constraint» Satellite
Parameters
wo - orbital velocity
«constraint» SatelliteCharacteristics
inertiaMomentum : double
Parameters
Main Inertial Moments
x = 295.97
y=509.03
z=357.43
1
1
1

Figure 9. AOCS Constraints.

But to define constraints it is not enough, it is mandatory to establish relationships between these constraints. To address this SysML defines parametric diagrams. A parametric diagram is defined as a restricted form of internal block diagram. A parametric diagram may contain constraint properties and their parameters (OMG, 2010). Figure 10. shows a parametric diagram, defining relationships between constraints used by controller.

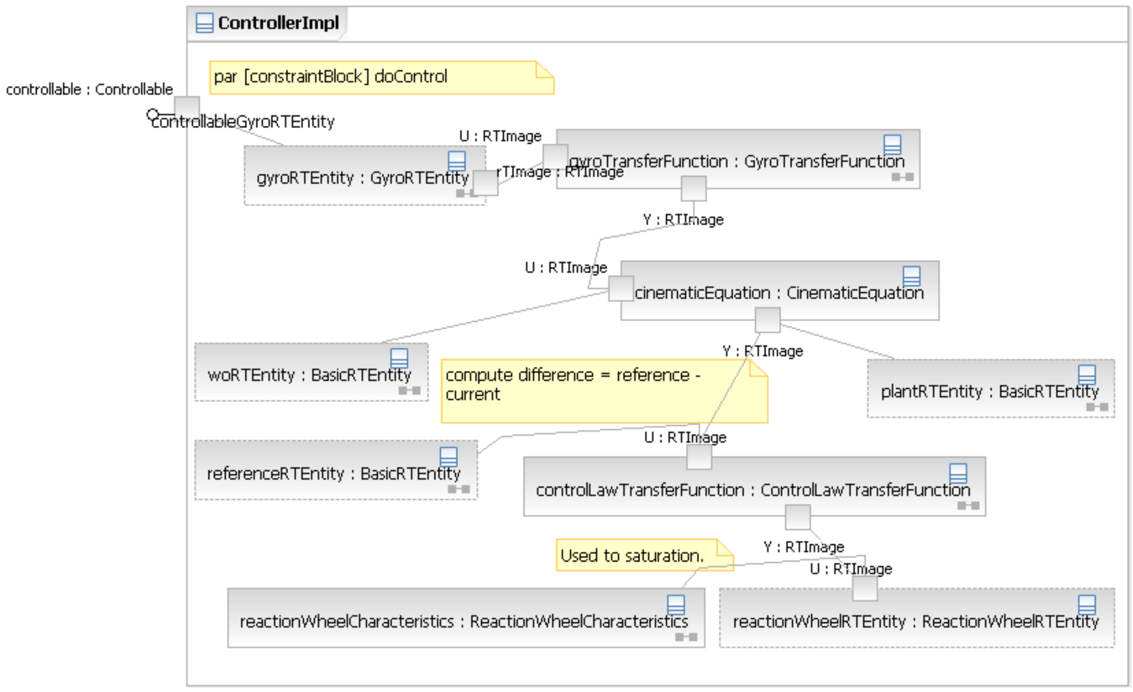

ControllerImpl
par [constraintBlock] doControl
controllable : Controllable
controllableGyroRTEntity
gyroRTEntity : GyroRTEntity
U : RTImage
rTImage : RTImage
gyroTransferFunction : GyroTransferFunction
Y : RTImage
U : RTImage
cinematicEquation : CinematicEquation
woRTEntity : BasicRTEntity
compute difference = reference - current
Y : RTImage
plantRTEntity : BasicRTEntity
referenceRTEntity : BasicRTEntity
U : RTImage
controlLawTransferFunction : ControlLawTransferFunction
Used to saturation.
Y : RTImage
U : RTImage
reactionWheelCharacteristics : ReactionWheelCharacteristics
reactionWheelRTEntity : ReactionWheelRTEntity

Figure 10. AOCS Parametric Diagram for Controller Constraints.

Parametric diagrams can be used to support trade-off analysis. A constraint block can define an objective function to compare alternative solutions. The objective function can constrain measures of effectiveness used to evaluate the alternatives (OMG, 2010).

7 Software analysis

On software-intensive systems, such as AOCS, it is possible to use SysML to model software aspects on a software viewpoint. Figures 11. and 12. shows software blocks structure defined using block definition diagram. Figure 13. shows a behaviour modelled using sequence diagram. The sequence diagram describes the flow of control between actors and systems (blocks) or between parts of a system (OMG, 2010).

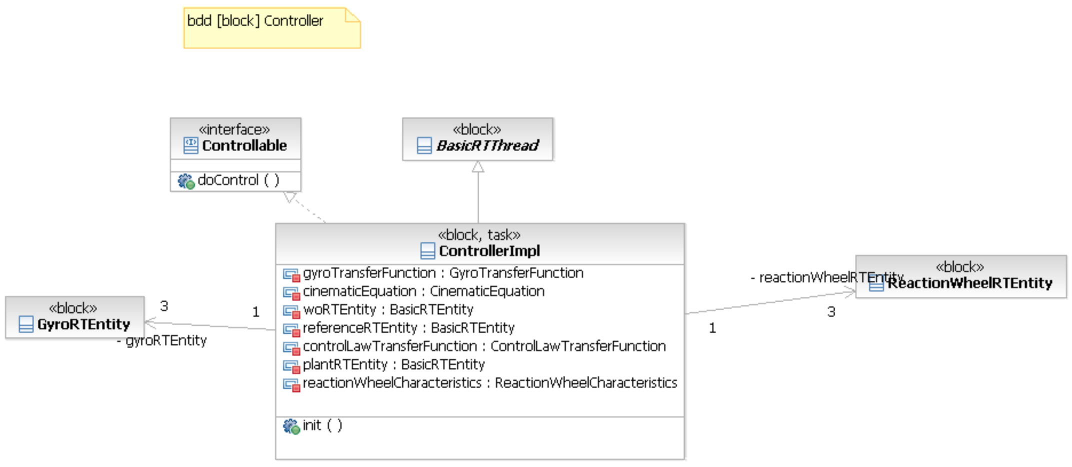

bdd [block] Controller
«interface» Controllable
doControl ( )
«block» BasicRTThread
«block, task» ControllerImpl
gyroTransferFunction : GyroTransferFunction
cinematicEquation : CinematicEquation
woRTEntity : BasicRTEntity
referenceRTEntity : BasicRTEntity
controlLawTransferFunction : ControlLawTransferFunction
plantRTEntity : BasicRTEntity
reactionWheelCharacteristics : ReactionWheelCharacteristics
init ( )
«block» GyroRTEntity
3
1
- gyroRTEntity
- reactionWheelRTEntity
«block» ReactionWheelRTEntity
3
1

Figure 11. AOCS software blocks for Controller.

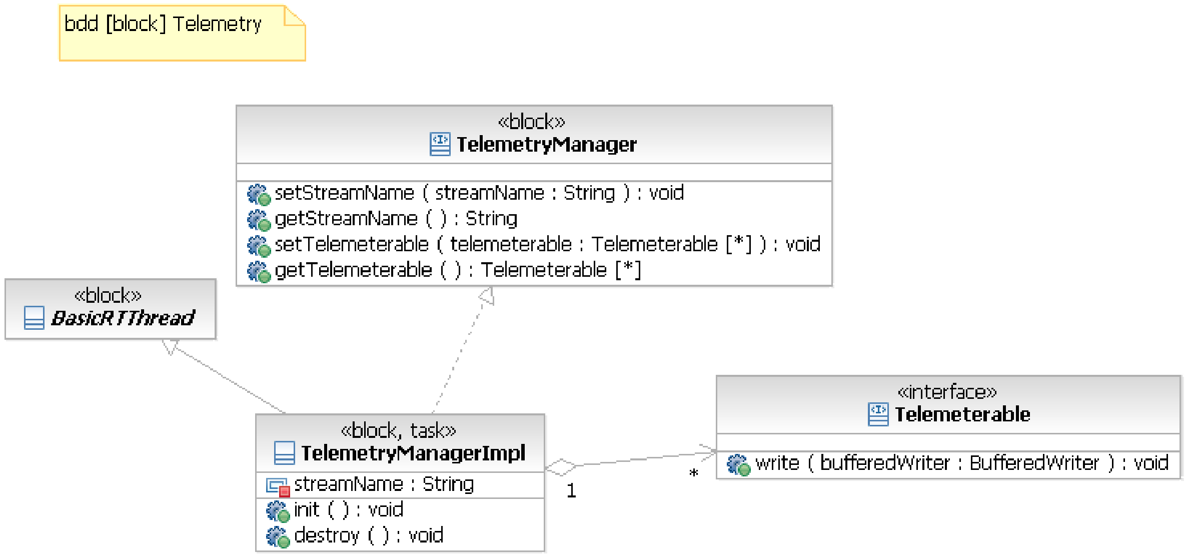

bdd [block] Telemetry
«block»
TelemetryManager
setStreamName (streamName : String) : void
getStreamName ( ) : String
setTelemeterable (telemeterable : Telemeterable [*]) : void
getTelemeterable ( ) : Telemeterable [*]
«block»
BasicRTThread
«interface»
Telemeterable
write (bufferedWriter : BufferedWriter) : void
«block, task»
TelemetryManagerImpl
streamName : String
init ( ) : void
destroy ( ) : void
1
*

Figure 12. AOCS software blocks for TelemetryManager.

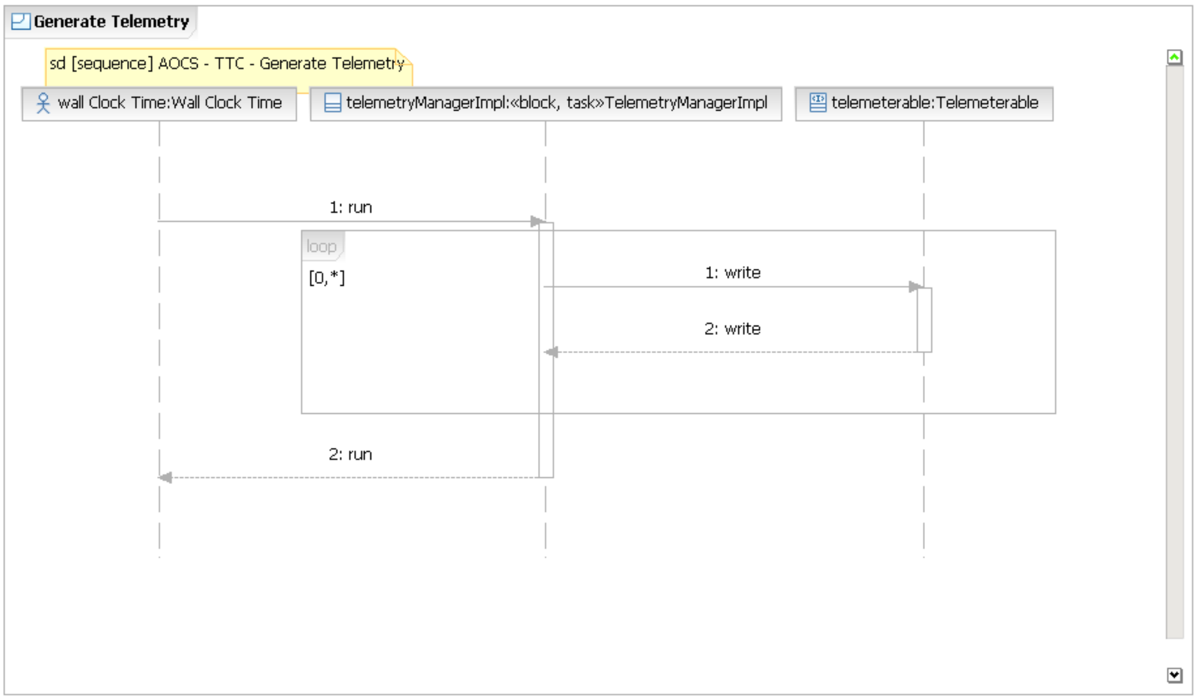

Generate Telemetry
sd [sequence] AOCS - TTC - Generate Telemetry
wall Clock Time:Wall Clock Time
telemetryManagerImpl:«block, task»TelemetryManagerImpl
telemeterable:Telemeterable
1: run
loop
[0,*]
1: write
2: write
2: run

Figure 13. AOCS Sequence Diagram for TelemetryManager.

Figure 14. shows a refined block definition diagram that has reference to a Java Interface. This kind of reference transforms PIM model on a PSM model, because now it is platform specific. In current paper original model is maintained platform independent, and a copy is refined to become a PSM.

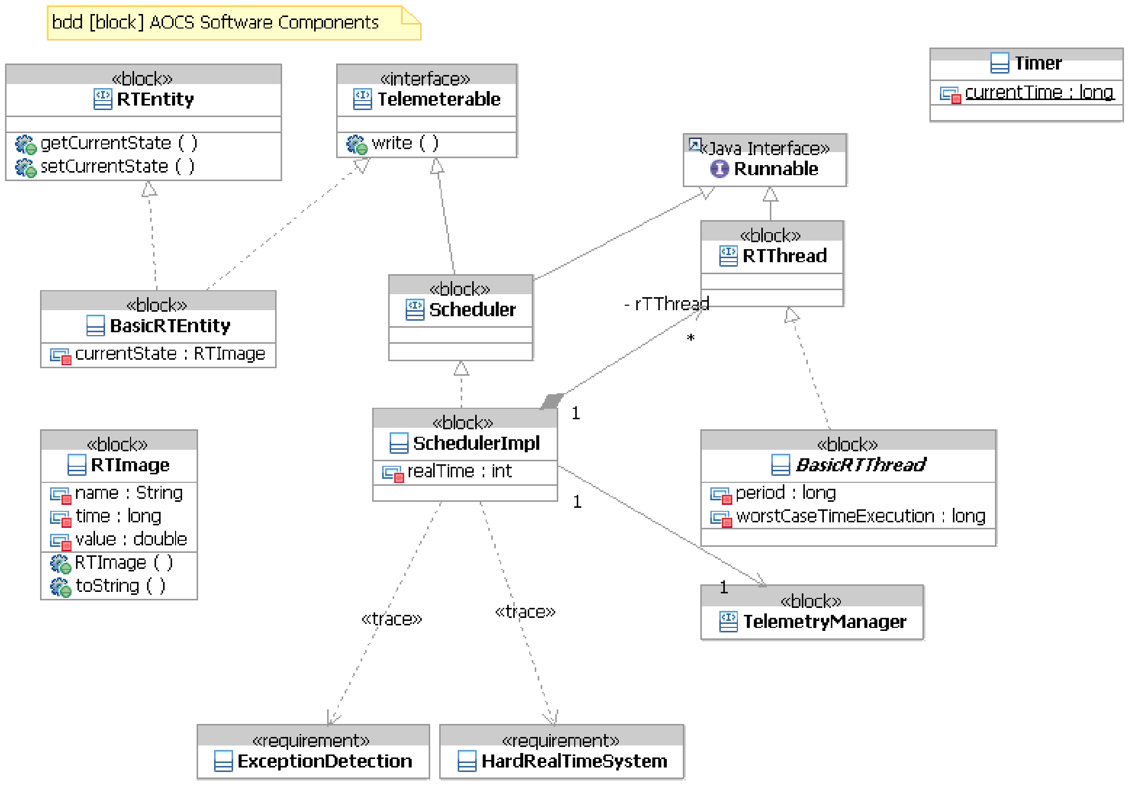
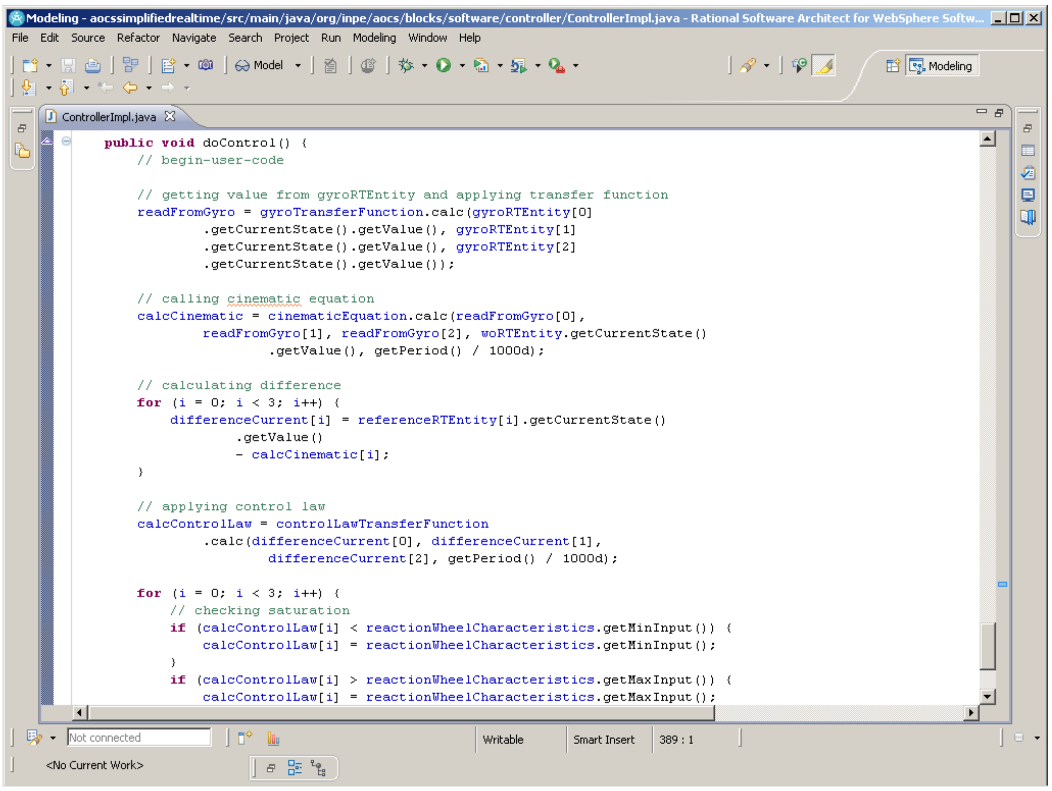


Figure 14. AOCS Software components (PSM, reference to Java Interface).

8 Code and code generation

Using basic transformation UML2Java from Rational Software Architect (IBM, 2010) PSM is used to code generation. Only structure is generated, i.e., classes, blank methods and attributes. After code generation, a developer must refine behaviors models on java code (in the current paper). Figure 15. shows code implemented to materialize parametric diagram presented in Figure 10.


Modeling - aocssimplifiedrealtime/src/main/java/org/inpe/aocs/blocks/software/controller/ControllerImpl.java - Rational Software Architect for WebSphere Softw...
File Edit Source Refactor Navigate Search Project Run Modeling Window Help
Model
Modeling
ControllerImpl.java
Not connected
Writable
Smart Insert
389 : 1
<No Current Work>

```
    public void doControl() {
        // begin-user-code

        // getting value from gyroRTEntity and applying transfer function
        readFromGyro = gyroTransferFunction.calc(gyroRTEntity[0]
                .getCurrentState().getValue(), gyroRTEntity[1]
                .getCurrentState().getValue(), gyroRTEntity[2]
                .getCurrentState().getValue());

        // calling cinematic equation
        calcCinematic = cinematicEquation.calc(readFromGyro[0],
                readFromGyro[1], readFromGyro[2], woRTEntity.getCurrentState()
                        .getValue(), getPeriod() / 1000d);

        // calculating difference
        for (i = 0; i < 3; i++) {
            differenceCurrent[i] = referenceRTEntity[i].getCurrentState()
                    .getValue()
                    - calcCinematic[i];
        }

        // applying control law
        calcControlLaw = controlLawTransferFunction
                .calc(differenceCurrent[0], differenceCurrent[1],
                        differenceCurrent[2], getPeriod() / 1000d);

        for (i = 0; i < 3; i++) {
            // checking saturation
            if (calcControlLaw[i] < reactionWheelCharacteristics.getMinInput()) {
                calcControlLaw[i] = reactionWheelCharacteristics.getMinInput();
            }
            if (calcControlLaw[i] > reactionWheelCharacteristics.getMaxInput()) {
                calcControlLaw[i] = reactionWheelCharacteristics.getMaxInput();
```

Figure 15. AOCS code.

9 Future works

Important features, from SysML, such as allocation, allocation tables, test cases, viewpoints formalisms, activity diagrams and flow ports must be evaluated on next works. Another relevant research field is how to describe constraints, and how allow code generation using a MDA approach. Other interesting field is how to use SysML and UML in a cohesive way.

9 Conclusions

Development of complex systems needs robust approaches and languages. SysML is a good candidate to become a de facto standard for system engineering aiding designers on modeling and understanding system behavior and its effects on other system aspects. Current paper shows a possible, but not complete, approach to starting from context diagram and finish on software code using SysML.

References

Espinosa, H., Cancila, D., Gérard, S., Challenges in Combining SysML and MARTE for Model-Based Design of Embedded Systems. France: CEA, 2008. 18 p. Available in: <http://www.omg.org/cgi-bin/doc?syseng/09-06-08.pdf >.

IBM, Rational Software Architect Site. Available in: <http://www.ibm.com>

Johnson, T.a., Jobe, J.M., Paredis, C.J.J., Burkhart, R., Modeling continuous system dynamics in sysml, 2007 ASME International Mechanical Engineering Congress and Exposition, 11-15 November, Seattle, Washington, USA, 2007. Available in: <http://www.srl.gatech.edu/publications/2007/JohnsonParedis-IMECE2007_DRAFT.pdf>

Moreira M. L. B., Projeto e simulação de um controle discreto para a plataforma multi-missão e sua migração para um sistema operacional de tempo real. 2006. 181 p. (INPE-14202-TDI/1103). Dissertação (Mestrado em Mecânica Espacial e Controle) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, Brasil. 2006. Available in: <http://urlib.net/rep/sid.inpe.br/MTC-m13@80/2006/07.10.13.42?languagebutton=pt-BR>.

Ogata, K., Modern Control Engineering. Third Edition. USA: Prentice Hall, 1997. 987 p. ISBN 0-13-227307-1.

OMG., Model-Driven Architecture. USA: OMG, 2003. 62 p. Available in: <http://www.omg.org/mda>

OMG., Systems Modeling Language. USA: OMG, 2010. 260 p. Available in: <http://www.sysml.org/docs/specs/OMGSysML-v1.2-10-06-02.pdf>

Pasetti, A., Brown, T., Software Frameworks and embedded control systems. UK: Springer, 2002. 293 p. ISBN: 978-3-540-43189-3.