UML FOR SPACE SYSTEMS: FROM SPECIFICATION TO DESIGN AND IMPLEMENTATION

IAC-13,D1,3,7,x18418

UML FOR SPACE SYSTEMS: FROM SPECIFICATION TO DESIGN AND IMPLEMENTATION

M. Rizwan Mughal, Anwar Ali, Haider Ali, Leonardo M. Reyneri,
Department of Electronics and Telecommunications,
Politecnico Di Torino, Italy
{muhammad.mughal, anwar.ali, leonardo.reyneri}@polito.it

Unified Modeling language (UML) is a high level modeling language which comes with new approaches in the modeling, design, documentation, testing of embedded systems. It is an upcoming approach borne in the software community which is nowadays extending to other industrial domains and recently entering the space community. A key element of the AraMiS architecture (A small satellite built by the students of Politecnico Di Torino) is the specification, design, testing and documentation based on UML. Each subsystem has been associated with one or more students who fully developed it from specifications to implementation and testing again using UML. A variety of UML diagrams (use case, class, sequence, requirement, etc.) have been utilized for the development of each subsystem. A teamwork server allows every designer to work independently from anywhere and commit his work which can be reviewed by the project manager and updated in the main project and therefore, the other team is also fully aware of the updated task. Complete documentation report and software code of the system can be generated at the end of the design. The UML design approach is helpful to better understand the functionality of the system and overcome design limitations on the early stages of the system. Design complexity and development time is reduced by using proposed approach. At the end, we get a fully documented, tested, fault tolerant space system thanks to UML design approach.

I. INTRODUCTION

Designing a space system is a very complex task. It must be split into simpler functions and subsystems. Each function and subsystem has to be specified, designed, reviewed, validated, implemented, tested, deployed, and most importantly documented. Development requires a wide number of tools, techniques and approaches.

Designing a low cost or university satellite has to cope with a number of additional tough constraints: Firstly, the Individual designers and teams mostly consist of students who are available for a short period of time. They often appear and disappear quickly. Secondly, the designers and teams are often inexperienced and do not have much exposure to the vast variety of practical tools. Therefore, someone has to lay down specifications for heterogeneous teams. Innovative metholodigies are necessary that allow the integration of more and more complex systems with innovative design, testing and documentation.

This paper proposes the design technique of initialization, requirements, documentation, and modeling of subsystems of small satellites as a test case, all using UML [1-4].

In particular, AraMiS is an innovative modular architecture alternative to CubeSats, for larger and more demanding applications, with a modularity at mechanical, electronic and testing levels, to be used mostly in LEO Satellites (600 – 800 km). Modularity helps to share costs among multiple missions, to reduce manufacturing and testing costs and to easily increase redundancy. Actual satellite technology leads to high costs for space missions. The use of COTS reduces development time, cost and size. Reduced reliability of COTS can be compensated by proper design and redundancy.

A key element of the ARAMIS architecture is the design and documentation methodology used, which is based on an extension of UML, as described here.

The aim of this paper is to introduce an approach to this problem using UML as a high level specification, description and documentation language. The purpose of this approach is to obtain a complete development flow for mixed-systems able to produce, on one side, documentation always close to real project implementation and, on the other, a fast and reliable method for reducing time- to-market in developing these objects.

The design of all the major subsystems of AraMiS has been carried out by using UML. Initially developed in 1995 for designing software, the UML was optimally adapted to the description of systems made of both hardware and software. It is based on the representation of entities involved in the system functioning and all interactions among them. There are many advantages of using this language with most important being

- Make easier the project understanding, even by people external to the project, thanks to a graphical/conceptual representation of the elements that make the system (components, subsystems, signals, functions...) starting from a high level description to a specific one.
- Simplify and improve the description of system functionalities and the specification definition providing a common basis in the approach of designing the units forming the whole system.
- Make exportable the system building blocks (which are independent form each other) such that they can be re-used in other projects so implementing the modularity concept.

The UML design approach for AraMiS comes from the evolution of the University Satellites at Politecnico di Torino:

- PICPOT, launched on July 2006, but launcher blew up during flight
- ARAMIS, heavily modular architecture for more demanding applications.

An innovative approach to teamwork design of complex system was needed:

- From mission- to circuit- and component-level
- Compatible with very heterogeneous and inexperienced teams and with a variety of CAD tools (mechanical, electrical, software, etc.)

Among all possible utilities that UML offers, there are certain types of diagrams used in UML including, but not limited to use case diagram, class diagram and sequence diagram.

II. UML DIAGRAMS

II.I. SPECIFICATIONS: USE CASE DIAGRAMS

Use case diagrams show main function of the system (use cases) and the entities that are outside the system (actors). Use case diagrams show how the class and objects of the class relate and hierarchical associations and object interaction between classes and objects. These diagrams allow us to specify the requirements of the system and show interaction between system and external actors. These diagrams are the starting point in the system modeling and consist of actors and use cases.

II.I.I. Actors

Actors are generic entities, human users, other systems or the external environment, which interact with the system under design and implements one or more use cases. They are usually shown as sketched people with a short name which identifies the role in the system. They are associated with a detailed documentation. The list of actors is fundamental to understand all entities which might interact with the system. The actors are very fundamental entities and missing an actor will miss all the interfaces and functions associated with it. Therefore they are critical in the design and the designers need to put more time in identifying the possible actors during early stages of design.

Let's imagine a situation where the designer forgot to add an actor "tester" in the early stage of the design. The possible consequences may be

- There might not be possibility of test connector to test the satellite
- There will be no internal access node for debugging
- No software will be added for detailed testing
- There might not be special satellite mode to allow testing before launch, etc.

II.I.II. Use Cases

The major work of the actors is to interact with different subsystems of the satellite. The main concerning points are

What does and actor expect from a system?
What does a system expect from an actor?

All this is detailed by Use Cases, which are usually described by an oval with a name which shortly describes it. Building up the list of use cases means starting to specify the functions of the satellite or its subsystem and therefore thinking to the mission.
There exist several kinds of relations between use cases and actors including generalization, inclusion, extensions, associations etc.

This section details the use cases of the magnetic attitude subsystem of the AraMiS architecture. The use case diagram defines the functional specifications of the project and is shown in Fig.1 for magnetic torque actuator subsystem. The central mission controller and central attitude controller actors interact with the system and perform many tasks. The central mission controller is an entity (likely a software routine running on the onboard computer) in charge of satellite supervision, fault detection and management and emergency management whereas the central attitude controller is an entity (likely a software routine running on the OBC in charge of managing the attitude control subsystem in nominal operation. Fault and emergency handling are left to the Central Mission Controller. All other use cases implement most of the magnetic actuation and control functions such *as attach/ detach* coil, get voltage

and current levels of coil and housekeeping sensors etc. All the housekeeping functions are performed by using data handling technique described in [1].

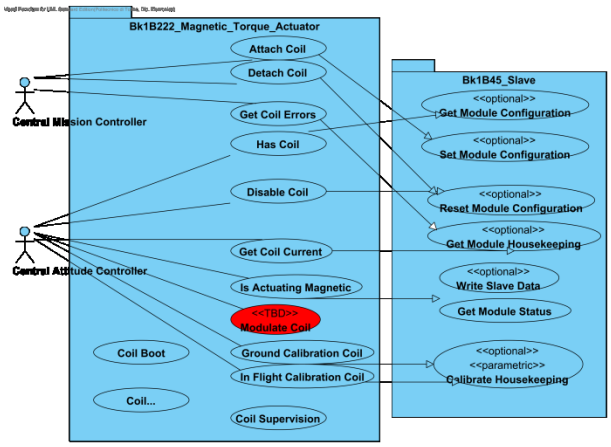

Fig.1 Use case diagram of magnetic attitude control class

III. ARCHITECTURE AND DESIGN

III.I. Classes

The objects have tendency to know things i.e. they have attributes and they do things i.e. they have methods. All objects of the same type are represented by a class. In UML notion, classes are depicted as boxes with three sections, the top one indicates the name and stereotype of the class, the middle one lists the attributes of the class, and the bottom one lists the methods.

Each object in UML classes can either be associated with hardware (HW), software (SW), an analog (ANA) implementation etc. depending on the stereotype of each class. Each stereotype is labelled with a specific colour. Each subclass has objects which contain different attributes and operations.

UML classes and objects are used to specify any electronic, mechanical, software element of a system.

The attributes of the class store data for the class. The attributes can either be constant, therefore representing characteristics common to all objects of that type variable, therefore storing time-variable data which are part of the class. Classes in turn can be made of one or more other classes (hierarchical structure). The generic UML class for AraMiS is shown in Fig.2 showing unique class for every major subsystem.

The class diagram of attitude and orbit control subsystem has been elaborated as a test case. The magnetic attitude control system together with the inertial attitude control system is used to accomplish the desired rotation to the satellite, by sending commands directly from the ground. The system consists of certain sensing and control classes as shown in the Fig.3. The class diagram of magnetic attitude subsystem consist of

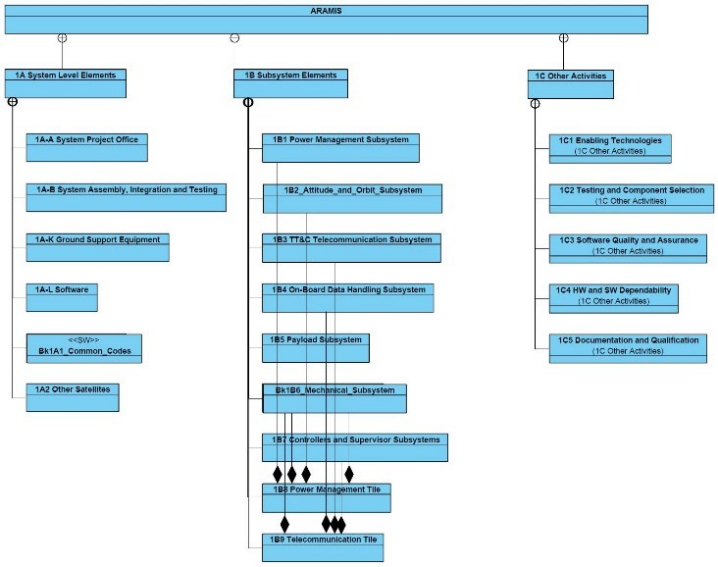

Fig. 2 . Generic class diagram for AraMiS architecture

- Bk1B222_Magnetic_Torque_Actuator block consisting of solenoid coil.

Fig.2 Class diagram of AraMiS

- Bk1B221_Magnetometer_Sensor block consisting of a magnetic field sensor and conditioning electronics.

The current flowing through the solenoid provides the tile angular momentum while the magnetometer measures the values of earth magnetic field in orbit.

The block 1B22 uses a microprocessor located in the Tile Power Management to handle the different sub-commands to perform housekeeping calculations and calibrate the telemetry data.

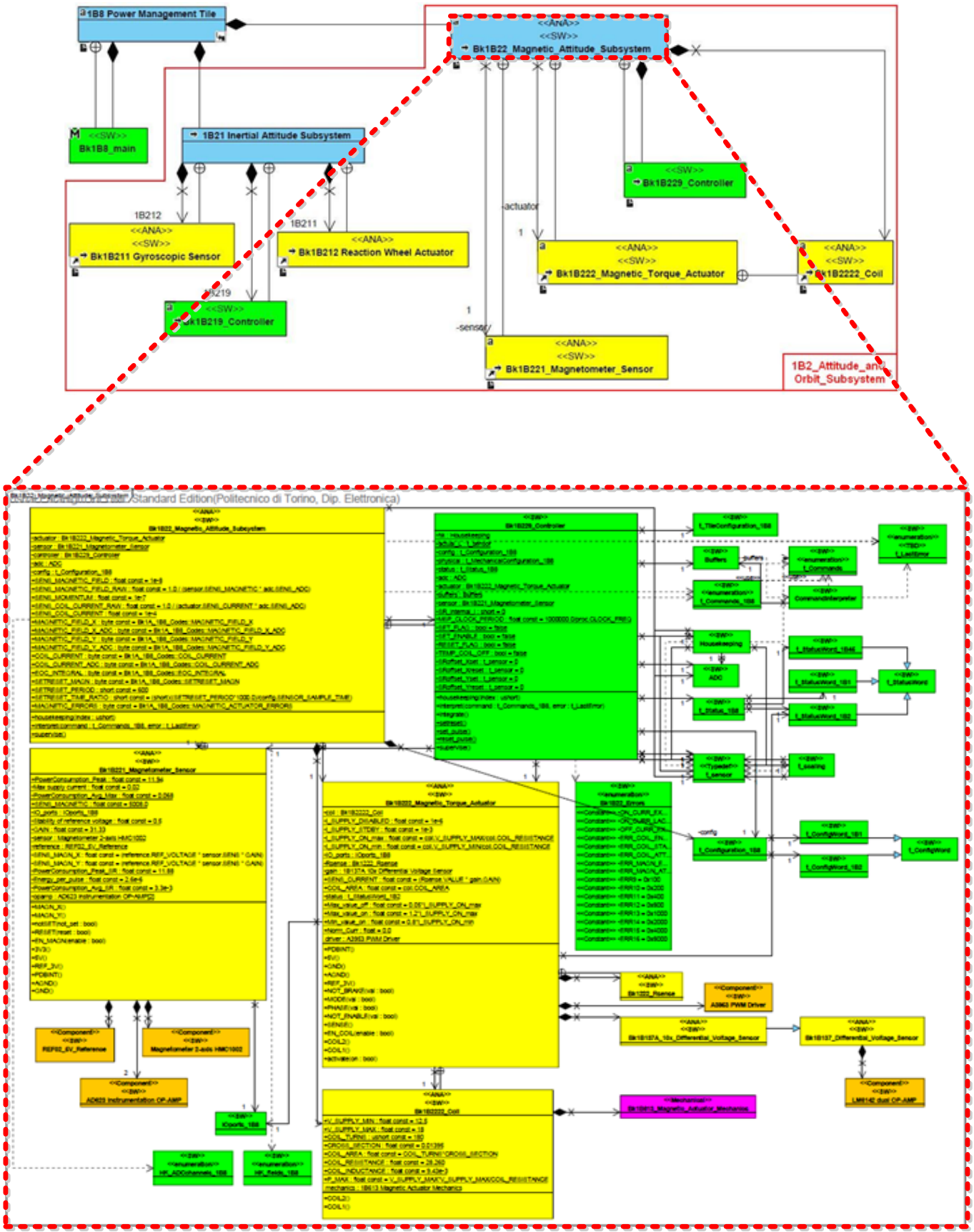

Fig.3 Magnetic attitude control class diagram

IV. SEQUENCE DIAGRAMS

Sequence diagrams describe how structural elements communicate with one another and time sequence of events. Time increases vertically from top to bottom. Fig.4 shows the sequence diagram of simple data handling example. Fig. shows sequence diagram to give the set/ reset pulses to our magnetometer. The controller sends either set or reset pulses to the magnetometer bock which ultimately passes to the magnetometer sensor.

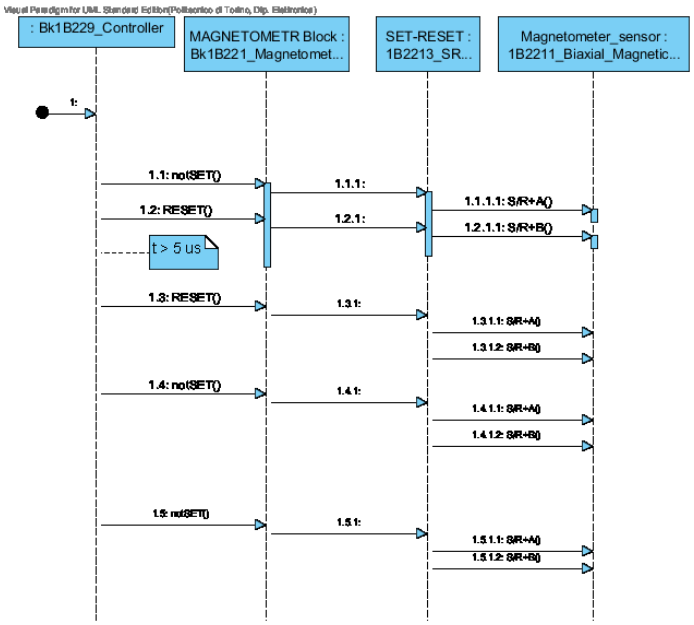

Fig. 4 Sequence Diagram of set/reset pulses of Magnetometer

The detailed analysis of using all the UML diagrams in the design, implementation and testing of any electronic system is described in the next section.

V. DESIGN FLOW

The UML diagrams are very helpful in building subsystems from conception to the final design and testing. Fig.5 shows a design sequence for magnetometer subsystem of AraMiS architecture. The use case diagrams indicate all the functions of the magnetometer managed by central mission controller and central attitude controller actors. These use cases are then realized by a set of UML classes in class diagram. The class diagram shows the subclasses for magnetometer and either hardware, software and analog components needed in order to complete the design. Each class is represented by a specific color based on its type. The classes have been divided into different types based on the attributes. The software classes contain the software codes in order to implement the specific use cases. This makes the design quite easier by placing pieces of codes in the corresponding classes and then generating the entire code at any time in the design. The detailed documentation of every use case, class and sequence diagram is written in the respective location. The built-in project documentation design offers a high level of flexibility, user control and customization. After selecting output format using the desired template, the project documentation can easily be customized according to the user needs. There is also possibility of selecting the level of detail for each element, such as including hierarchy diagrams to assist the communication of class relationships. There is also possibility to insert hyperlinks to aid the navigation in the project documentation.

After the use case and class diagrams, the final design is realized as shown in Fig.5. Every hardware and component stereotyped class corresponds to physical component. The physical schematics can then be realized by use of any CAD tool. The final design is ready to be implemented at this stage. After the design is physically available, the next step is to test it. Testing sequence is realized by sequence diagrams. The test is performed according to the sequence diagram and verified with the requirement diagrams. If the test sequence is in accordance with the requirements, the final module is ready to be integrated onto the tiles. At the end, a complete documentation report of all the steps of the design can be easily generated. The generated code for testing and realization is quite clear, readable and well documented and properly commented. Moreover, the generated code clearly depicts the system design in terms of relationship between classes, their associations, etc.

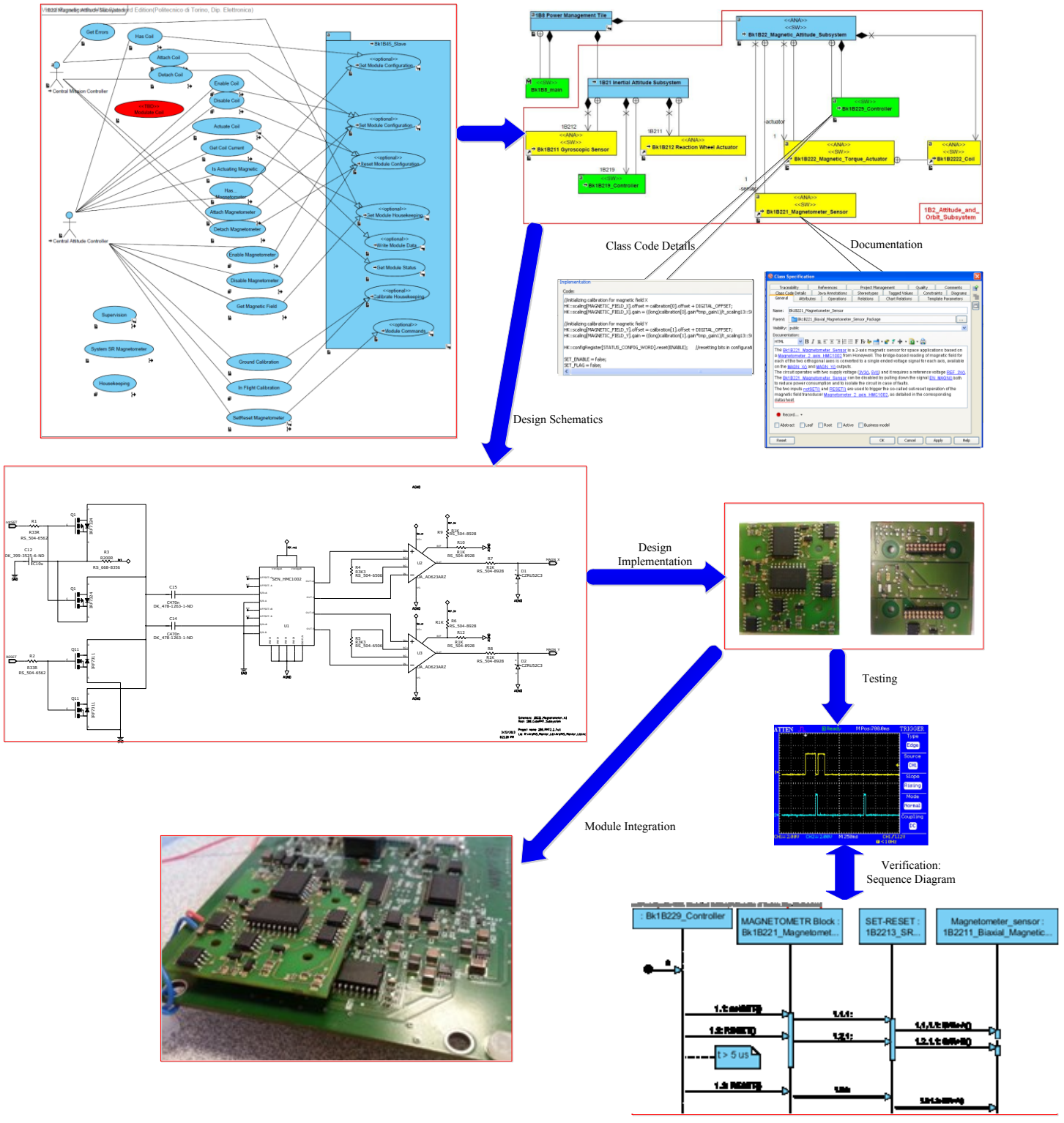

Fig. 5 Design flow of magnetic attitude subsystem

VI. CONCLUSION

This paper has introduced an object oriented approach to the design of on-board data handling subsystem of AraMiS Nano satellites based on UML. There are many advantages of UML based design over typical design approach. The UML design approach is helpful to better understand the functionality of the system and overcome design limitations on the early stages of the system. Design complexity and development time is reduced by using proposed approach. Extensive documentation of each diagram helps in understanding the system much better, reuse of UML singleton classes helps minimize probability of error. In addition, interaction between different blocks is much easier. UML tool can generate code for several programming languages. The generated code is quite clear, readable and well documented and properly commented. Moreover, the generated code clearly depicts the system design in terms of relationship between classes, their associations, etc.

REFERENCES

[1] Mughal, M.R.; Reyneri, L.M.; Ali, A., UML based design methodology for serial data handling system of NanoSatellites, Satellite Telecommunications (ESTEL), 2012 IEEE First AESS European Conference on , vol., no., pp.1,6, 2-5 Oct. 2012

[2] Reyneri, L.M.; "UML-based design methodology for designing university satellites" 1st IAA Conference on University Satellite Missions and CubeSat Workshop, Rome, Italy, 24-29 January 2011

[3] Tranchero, M.; "HW-SW design flow of a Nano-satellite using UML and CodeSimulink co-design environment," Electrotechnical Conference, 2006. MELECON 2006. IEEE Mediterranean, vol., no., pp.85-88, 16-19 May 2006.

[4] Reyneri, L.M.; "An Object Oriented Codesign Flow for low-cost HW/SW/mixed-signal systems based on UML," Electrotechnical Conference, 2006. MELECON 2006. IEEE Mediterranean, vol., no., pp.80-84, 16-19 May 2006.