IMPLEMENTATION OF A NANOSATELLITE ON-BOARD SOFTWARE BASED ON BUILDING-BLOCKS

M. F. Barschke [(1)], K. Großekatthöfer [(1)], S. Montenegro [(2)]

[(1)]Technische Universität Berlin, Marchstr. 12–14, 10587 Berlin, Germany,
+49 30 314-28743, merlin.barschke@tu-berlin.de
[(2)]Julius Maximilian University of Würzburg, Am Hubland, 97074 Würzburg, Germany,
+49 931 31-83715, sergio.montenegro@uni-wuerzburg.de

ABSTRACT

In recent years, remarkable developments within the electronic industry, especially regarding component miniaturisation and sensor technology, have broadly extended possible application areas of pico- and nanosatellites.
While state-of-the-art COTS hardware is widely used in pico- and nanosatellite projects, the approach to software is often less up-to-date. However, the full potential of a complex system such as a satellite cannot be exploited without complementing the hardware with efficient and yet flexible and well-structured software. Hence, modern software design is as important for the satellites' performance as powerful hardware components.
In this paper, we present the conceptual design of the TechnoSat on-board software in order to demonstrate the advantages of a modern approach to satellite software design. TechnoSat, due to be launched in 2015, is a TUBiX20 based satellite with the mission objective to validate several novel nanosatellite components on orbit.
The TechnoSat on-board software runs on RODOS, which is a real-time operating system based on building blocks; it was especially designed for satellite applications. While developing the software, particular attention was paid to matching the software design with the distributed node architecture of the hardware. As a result, their interaction and dependencies could be adjusted optimally to gain maximum performance and reuse capabilities.

1 INTRODUCTION

TU Berlin is currently working on a novel nanosatellite platform series called TUBiX [1]. Two versions of TUBiX are under development, TUBiX10 for 10 kg missions and TUBiX20 for 20 kg missions [2].
S-Net is the first mission to be based on TUBiX10 [3]. Within this mission, four satellites will be launched in 2016 to demonstrate multipoint inter-satellite communications in the S-band.
The first TUBiX20 mission is TechnoSat [4], which is scheduled to be launched in 2015 and carries several newly developed nanosatellite components for on-orbit verification. Furthermore, TechnoSat shall demonstrate the capabilities of the TUBiX20 platform to support its successor, the TUBIN mission.
TUBIN (TU Berlin Infrared Nanosatellite) extends the platform's capabilities to high accuracy attitude control to achieve its primary mission objective, which is to demonstrate novel Earth remote sensing technologies [5]. TUBIN carries two infrared micro bolometers, complemented by an additional camera with sensitivity in the visible spectrum as payload and is scheduled to be launched in 2016.

2 TECHNOSAT MISSION DESIGN

TechnoSat is a 20 kg satellite, whose primary mission objective is the on-orbit verification of several nanosatellite components. The secondary mission objective is the development, manufacturing, qualification and demonstration of TUBiX20, the novel nanosatellite platform of TU Berlin. Figure 1 shows a CAD Model of TechnoSat.

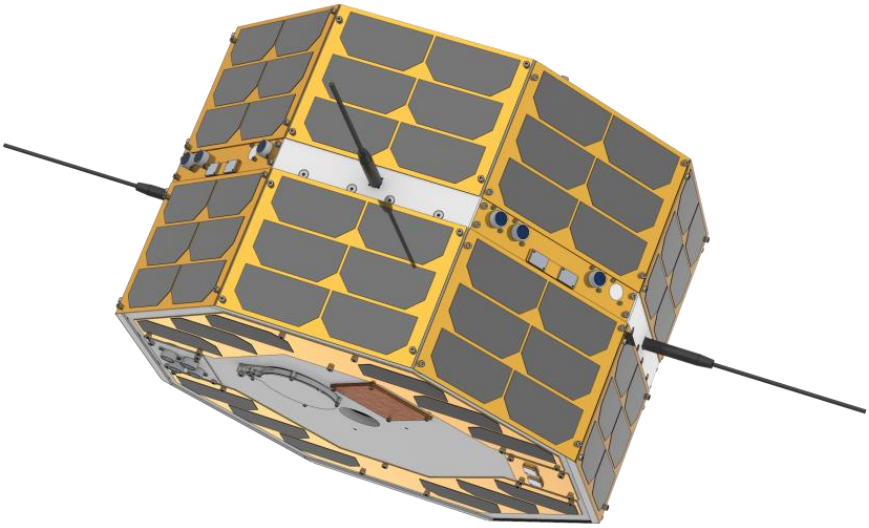

Figure 1. CAD model of TechnoSat

The following payloads will be demonstrated on orbit within the TechnoSat mission:

Black body hatch (BBH)
The black body hatch is a device for in-flight calibration of infrared sensors. It consists of a dark surface that can be moved in front of the sensor and then heated uniformly. In case of a failure in the opening mechanism, an independent emergency mechanism can be activated to open the hatch permanently. The black body hatch will be applied on the TUBIN mission, to calibrate the two infrared micro bolometer arrays.

Fluid dynamic actuator (FDA)
As a novel actuator highly agile attitude control, the fluid dynamic actuator provides torques of up to 20×10^{-3} Nm. It consists of a ring shaped tube with a diameter of 300 mm that is filled with Galinstan, a liquid metal alloy. The metal can be accelerated within the tube by means of an electromagnetic pump for momentum storage [6].

S-band transmitter (HISPICO)
HISPICO is a highly integrated S-band link for pico- and nanosatellites that provides a data downlink with a rate of 1 Mbps [7].

Camera (CAM)
The camera shall provide payload-data for the verification of the HISPICO S-band transmitter. Furthermore, it shall gather flight heritage for a lens, which will also be applied for a camera payload of the TUBIN mission.

Nanosatellite laser ranging (NSLR)
TechnoSat is equipped with several commercial 10 mm laser retro reflectors for ground based high precision orbit determination [8].

Reaction wheel system (RWS)
The reaction wheel system consists of four newly developed nanosatellite reaction wheels in tetrahedron configuration. The same wheels will be used as primary actuators on the TUBIN mission.

Solar generator based space debris impact detector (SOLID)
SOLID is an impact detection system, whose detector surfaces are located behind the satellites solar cells. It allows for detecting objects of a size of 100 µm to 1 mm [9, 10].

Star tracker STELLA
STELLA is a nanosatellite star tracker that provides an accuracy of 0.01 degree around the pitch and yaw axis and 0.04 degree around roll [11].

Table 1 lists the details of the TechnoSat mission, which is currently concluding phase C of the development and is scheduled to be launched on a soyuz-fregat in 2015.

Table 1. TechnoSat mission details

Mission objectives	1st: on-orbit verification of novel nanosatellite components 2nd: development and on-orbit verification of the TUBiX20 platform
Design lifetime	1 year
Launch	2015
Satellite mass	20 kg
Satellite Volume	450 × 450 × 300 mm
Communication link	UHF transceiver, experimental S-band transmitter
Attitude sensors	Fibre-optic rate sensors, Sun sensors, SMD magnetic field sensors, MEMS gyros, experimental star tracker
Attitude actuators	Magnetic torquers, experimental reaction wheels

3 TUBiX20 PLATFORM DESIGN

The TUBiX20 research is focussed on a system wide optimisation approach; instead of optimising every subsystem individually, the entire system is considered while developing hard- and software of the platform to maximise its overall performance. Additionally, hardware and software share a common design approach. Their architectures are carefully matched to optimise their interactions, while keeping the development effort low at the same time.

3.1 Systems Architecture

The TUBiX20 systems architecture is heavily influenced by the requirement of single-failure tolerance for the platform. The architecture is based on a system of distributed microcontroller nodes that fulfil required computation tasks and connect components to the central data-bus system.

While the distribution of basic subsystem functionalities on four main nodes follows a rather traditional approach, certain other functionalities, such as the distribution of telecommands or new software images, are implemented in a decentralised manner. Figure 2 shows the four main nodes of TechnoSat, as well as their connection to power and data busses. In the following it is described, which tasks are associated to each of the main-nodes.

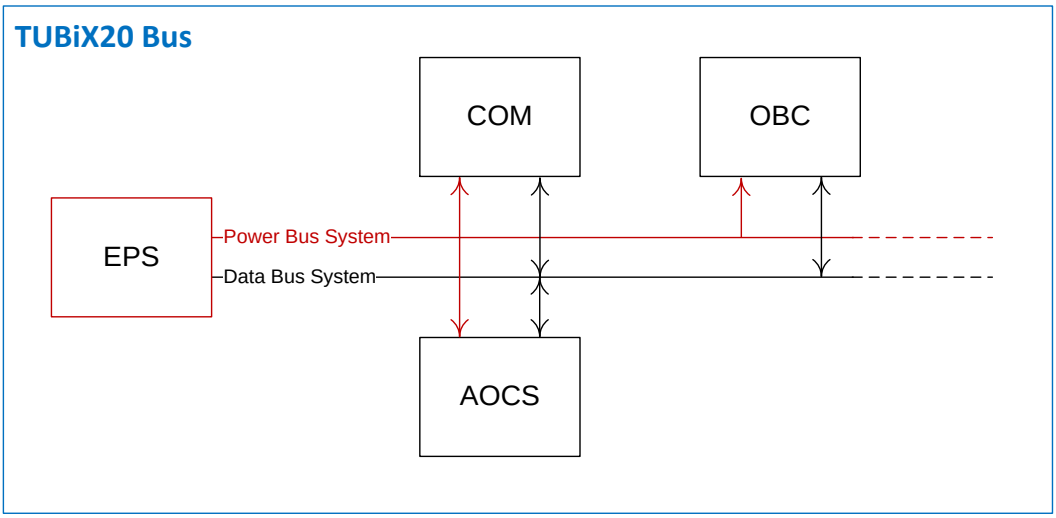

Figure 2. TUBiX20 main nodes.

Electrical Power System (EPS)
The electrical power system is responsible for power generation, conditioning and distribution. Furthermore, it performs redundancy management for all cold redundant nodes and the power- and data-bus system.

Communication System (COM)
The UHF link to the ground station is managed by the communication system. It transmits received transfer frames to the ground and distributes telecommands received from the ground station using the central data bus.

On-Board Computer (OBC)
Certain functionalities, such as telemetry storage and time-tagged telecommands management are handled in a centralised manner by the on-board computer.

The Attitude and Orbit Control System (AOCS)
The attitude and orbit control system is responsible for the determination and control of the satellite's attitude using the available set of sensors and actuators.

3.2 Design Guidelines
In order to fully exploit the advantages of the distributed node architecture, while keeping the overall implementation efforts low, consistent system wide standardization is applied. Therefore, several design guidelines were established and consistently followed in the development of hard- and software of the platform.

Modular design
On the on hand, the modular design is an essential factor for the platform's adaptability to different missions, payloads and orbits. On the other hand, it simplifies the development process of the platform, as self-contained modules can be developed and tested separately.

Separation of functional units
While implementing a modular design, the level of modularity is of great importance. Basically, TUBiX20 implements modularisation on the basis of functional units. One example for such a functional unit is an integrated sensor board, which accommodates several sensors for attitude determination. These sensors were not integrated in the solar panels, despite a possible reduction of mass and volume, but integrated on a separated board. Here, the separation of the two functional units i.e. the integrated sensor board and the solar panel is of greater importance, as it is necessary to exploit the advantages of the modular design.

Unification of interfaces
Another key aspect of a modular design is the system wide unification of hard- and software interfaces. Clearly defined interfaces allow for developing subsystems or even components to serve these interfaces, without requiring a detailed knowledge of the functioning of other systems of the platform.

Extensive reuse
Every component, circuit and software mechanism used on the satellite must be developed and then tested thoughtfully. Furthermore, the modular approach of the platform implies certain overhead, as all hard- and software must serve the standard interfaces. To minimise the development effort, extensive reuse of components, circuitry and software is executed. The approach is based on the fact that each node in the system requires a set of standard hard- and software elements. In terms of hardware, these include, among others, the microcontroller, the data bus interface, mass memory and a watchdog circuit. In software, applications for communication with other nodes, software upload and for triggering the watchdog are required on every node. All these components and applications are reused for every node in the system, hence they need to be developed and tested only once.

3.3 Telemetry and Telecommanding

Telemetry contains measurements and status data of platform and payloads. Here, telemetry can be divided into real-time and historical telemetry and at the same time into standard and extended telemetry.

- *Real-time telemetry* is collected by the on-board computer and immediately sent to the communication system for downlink to the ground station. Real-time telemetry is only generated in response to a telecommand.
- *Historical telemetry* is periodically sent to the on-board computer by the different nodes and stored for later downlink.
- *Standard telemetry* contains an essentialsubset of available measurements and constants. The standard telemetry allows for verifying the overall health of the satellite, it is exclusively generated by the four main-nodes of the platform (cf. section 3.1 Systems Architecture).
- All additional telemetry generated by platform and payload, which is not included in the standard telemetry, is referred to as *extended telemetry*.

To prevent loss of telemetry data and reduce the system’s complexity, the satellite will only transmit as a direct response to a telecommand. An exception is a morse code beacon, which can be activated to be sent periodically.
A telecommand received by the satellite is either just acknowledged, or responded to with a single telemetry transfer frame. Telecommands that are published within the satellite by the communication system are received by all nodes, which independently act upon the command if necessary.

3.4 Data-Upload

Data, such as new software images or configuration data, which is uploaded to the satellite from the ground station, is made available on the central data bus by the communication system and is then directly received by the target node, so that no central processing unit is required.

4 TECHNOSAT SYSTEM DESIGN

Figure 3 gives an overview over the TechnoSat subsystems and payloads, as well as their connections to the power- and the data-bus systems. TechnoSat features seven TUBiX20 nodes, i.e. nodes that follow the TUBiX20 hard- and software design guidelines. Three of the payloads are so-called alien nodes, which are connected to the redundant data bus system, but do not adhere to the TUBiX20 hard- and software design guidelines (cf. section 3.2 Design Guidelines). They each require a software application that runs on one of the TUBiX20 nodes and translates their custom protocols to the TUBiX20 protocol and furthermore coordinates the telemetry generation.

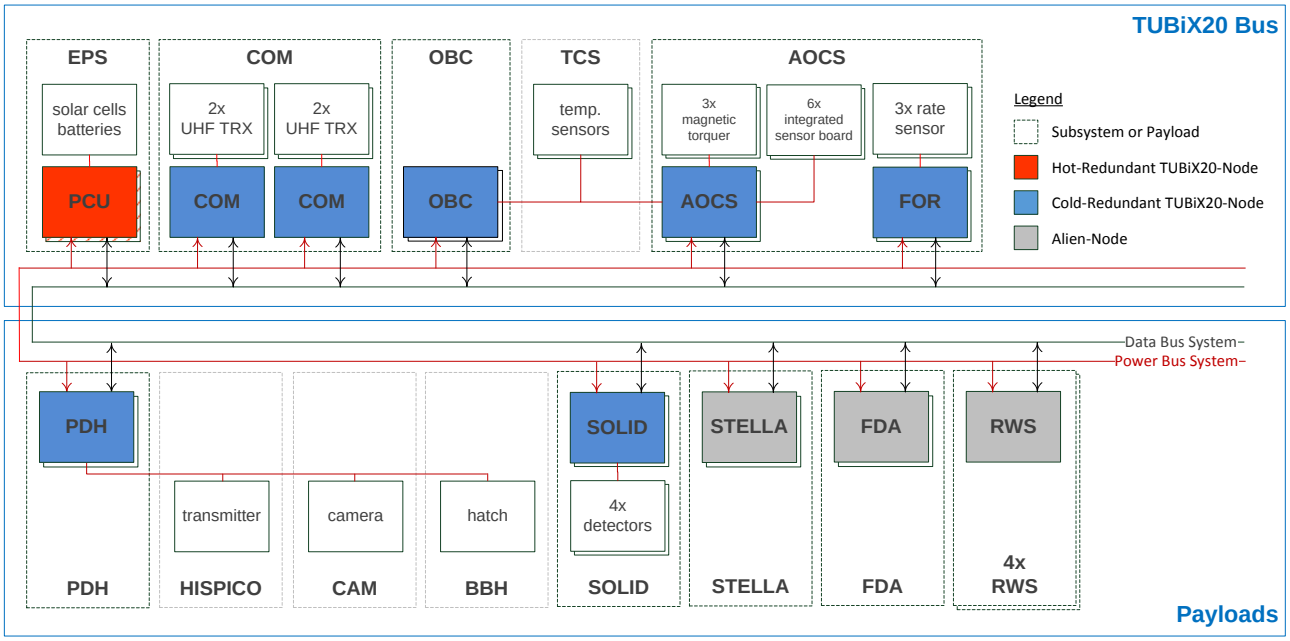

Figure 3. TechnoSat system overview.

5 TECHNOSAT ON-BOARD COMPUTER HARDWARE

The TechnoSat on-board computer hardware is a straightforward implementation of a TUBiX20 node. Central element of the on-board computer is a cold redundant microcontroller unit (MCU). Each MCU features flash memory and non-volatile static random-access memory (nvSRAM) and is supervised by an external hardware watchdog (WD). Both MCUs are connected to each of the CAN busses. Furthermore, a number of temperature sensors (TMP) are connected to the I2C buses of the MCUs.

Figure 4 shows a schematic of the on-board computer, containing its just mentioned main components and their data interconnections. Solid lines in the diagram indicate the active parts and dashed lines indicate the inactive parts of the node, for MCU1 being the active unit

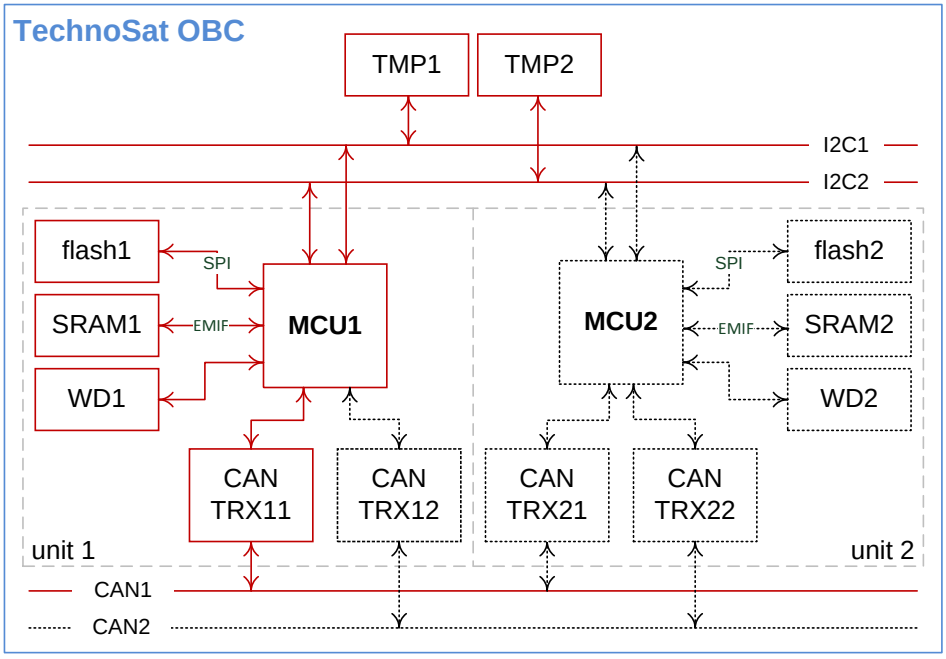

Figure 4. Schematic of the OBC electronics hardware.

6 RODOS OPERATION SYSTEM

The Real-time Onboard Dependable Operating System (RODOS) is an operating system for embedded systems based on building blocks, which was designed for application domains demanding high dependability [12, 13]. RODOS has its roots in the operating system BOSS and was developed at the German Aerospace Center (DLR). It is applied, among others, within the current micro satellite program of DLR. The open source project is currently coordinated by the Julius Maximilian University of Würzburg, Germany.
RODOS is a framework that features a multilayer structure. While the first layer is responsible for directly controlling the embedded hardware, a middleware runs on the second layer. This middleware is responsible for the communication between applications and components that form the top layer. The operating system is implemented object-oriented in C++, complemented by C and assembly code for hardware specific tasks.
Throughout the development of RODOS, special attention was paid to implement its various features in a simple, nevertheless robust way. Unnecessary complexity was avoided to provide the user with a straightforward, clearly arranged system. The RODOS core and its middleware were implemented in less than 3500 lines of C++ Code. In the following, some of RODOS features are listed.

- Object-oriented C++ interfaces
- Ultra fast booting
- Real time priority-controlled preemptive multithreading
- Time management
- Thread safe communication and synchronisation
- Event propagation
- An integrated publish/subscribe middleware

The basic idea of RODOS is composing a complex software system out of a number of simple components, the so-called building blocks. The aim is to define an environment where each

application can be developed independently from all other applications in the system. For this purpose, three kinds of global interfaces need to be defined.

- The applications interface to the operating system
- The interface to the middleware, for interconnection of applications
- The interface to the application manager to be able to migrate applications

Figure 5 shows a building block with its interfaces to the operating system, the middleware and the application manager.

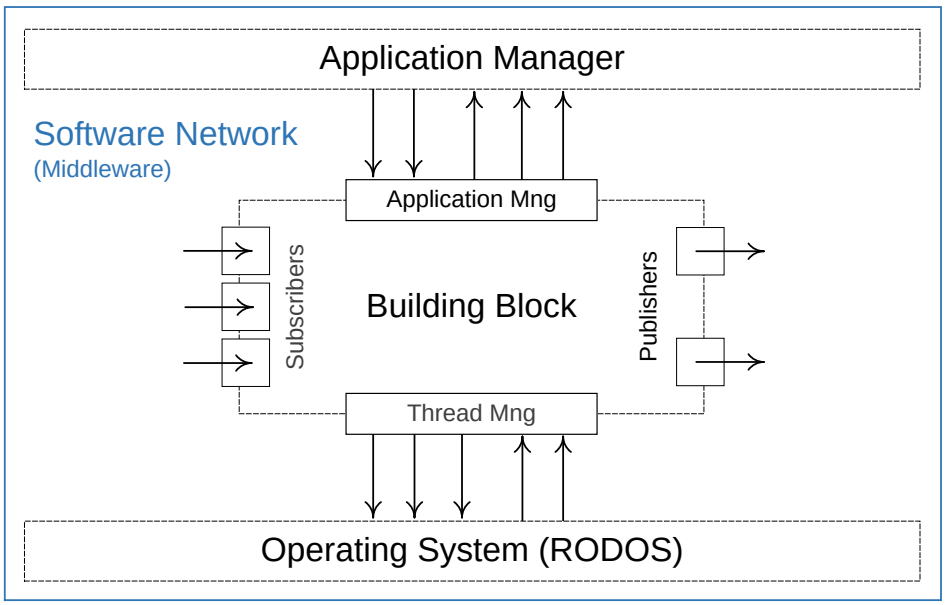

Figure 5: Interfaces of a building block.

Internally, a building block may consist of several threads, semaphores, event mangers, publishers and subscribers, However, this internal structure is not relevant for other building blocks, as all communication takes place via standardised interfaces. In the following, these interfaces are described.

Thread management interface
The thread management interface allows the applications to communicate with the core operating system; the RODOS core. Through this interface the building block requests resources in order to publish its services. Examples for such resources are execution time, memory area or access to I/O devices. Vice versa, the core operating system uses the thread management interface to forward events to the application. Such events might be interrupts, timer events or device ready messages. The RODOS core provides an object orientated interface with threads, semaphores, events, various passive support classes and time information.

Software network interface
The RODOS middleware is the communication system that interconnects applications via their software network interface. An application may subscribe to topics or services and publish information and data in form of messages on the software network interface. Messages are published under a given topic and any application that subscribes this topic receives the message. A topic is represented by an ID and a data type. A message may be published only locally, i.e. only within the given node, or globally, i.e. for all nodes in the system.

Application management interface
The application manager decides which application shall be running on which node; therefore each application should provide an interface to the application manager. As applications on TechnoSat may not be migrated to other nodes, no application manager is implemented.

For the communication of one microcontroller with others, RODOS features gateways that manage the external communication. While publishing a message on a topic, an application can define, weather the message is only published within the node, or if it is made available in the entire system by the gateway.

7 TUBiX20 SOFTWARE INFRASTRUCTURE

The TUBiX20 software infrastructure implements the design guidelines (cf. section 3.2 Design Guidelines) on side of the software. It is realised to minimise the software development efforts despite the relatively high number of nodes in the system and thus complements the distributed hardware system design. The software infrastructure defines the communication interface between the nodes and provides a library with all generic software functionality which is implemented identically for all nodes. Figure 6 shows a UML package diagram of the software infrastructure.

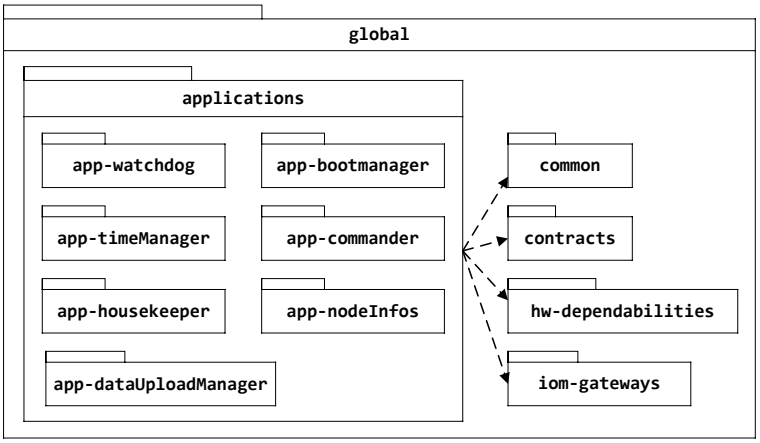

Figure 6: UML package diagram of the TUBiX20 software infrastructure.

7.1 Contracts

The contracts include system wide definitions of ID constants, classes and topics.

ID Constants
Several constants must be defined system-wide, as they are used for object identification and other information on systems level. In the following, some of these system-wide assigned IDs are listed:

- Anomaly IDs
- Application IDs
- Telecommand codes
- CAN-Bus addresses

Classes
The following class-definitions are used by all nodes and need therefore be identified on system level:

- Source Packets
- Boot Images
- Telecommands
- Switch configurations of the electrical power system

Topics
Several topics are required for system-level communication. In the following a number of these topics are describes exemplarily. However, if two or more nodes require additional topics for information or data exchange, these also need to be implemented system-wide.

- *rawTeleCommands*
  rawTeleCommands is used by the communication system to publish telecommands originated from ground station. Each node subscribes to this topic to be able to receive commands.

- *commands*
  The commands topic is used by commander application to distribute received commands within a node.

- *anomalyReport*
  Anomalies can be reported by any application. They are managed and centrally stored by the on-board computer. The applications report their anomalies on the anomalyReport topic.

- *iAmAlive*
  In order to ensure that all crucial applications on the node are healthy, each application reports its status in a periodical massage to the watchdog application on the iAmAlive topic.

7.2 Applications
The following applications are included in all nodes of the system. They provide a minimal set of functionalities that are required for the node to interact with the system.

BootManager
The boot manager application is responsible for the boot process. It manages the list of images that is used by the boot loader to decide which software image shall first be checked and loaded if applicable. Furthermore, the boot manager can check software images that are stored in the external flash after being uploaded to the satellite and copy these images to the internal flash.

Commander
On each node the commander receives all telecommands directly from the communication system on the rawTeleCommands topic. The commander initiates the collection of standard real-time telemetry, if requested. Furthermore, the it republishes the commands within the node, if the command was addressed to the node the commander is running on.

DataUplaodManager
Large data packets that exceed the maximum size of one telecommand frame are received by the DataUploadManager. The packets are transferred from the communication system directly to the node in question.

Housekeeper
The housekeeper is responsible for collecting the standard telemetry of the corresponding node.

TimeManager
The TUBiX20 platform features a system wide distributed pulse-per-second (PPS) signal. The time manager uses this PPS signal for synchronisation of the node-time with the hardware. Furthermore, it distributes the time within the node. The power control unit is the time-master and publishes the satellite time periodically on the corresponding topic.

Watchdog
Every TUBiX20 node features a hardware watchdog, which initiates a hard reset if not triggered by the software periodically (cf. section 5 TECHNOSAT ON-BOARD COMPUTER HARDWARE). The watchdog application supervises all critical applications of the corresponding node and only triggers the watchdog if all those applications are healthy. Furthermore, the watchdog application sends a periodic health message to the power control unit for redundancy management.

7.3 Common

Certain data and functionalities are independent of the applications and can therefore be shared within a node. Common data is for example:

- Parameters of the global applications
- The processed commands counter
- The refused commands counter

Examples of common functionalities are:

- The PPS synchronisation
- Debug functions for data visualisation

7.4 I/O-Manager of the Gateway

Besides running it on the target controller, RODOS can as well be emulated in Linux. For TechnoSat, this capability is used for development and verification purposes. Here, only one building block on a node is platform specific, the I/O-manager of the gateway. It realises the interface between the node and the data bus system. The implementation of the I/O-manager within the flight software communicates via CAN bus, while the Linux emulation uses UDP.

7.5 Hardware Dependabilities

Besides platform specific applications, there are also platform specific functions. Platform specific functions are for example:

- Reboot the current boot image
- Reboot with a different boot image
- Triggering the watchdog
- Check the node number

8 TECHNOSAT ON-BOARD COMPUTER SOFTWARE

Beside the functionality provided by the TUBiX20 software infrastructure, the TechnoSat on-board computer implements certain node-specific functionalities:

- Anomaly management
- Telemetry transfer-frame assembly
- Satellite mode management

- Time-tagged command management
- Temperature sensor readout

The applications that perform these on-board computer specific tasks are described in more detail in the following.

AnomalyReporter
Each application in the satellite can publish anomalies. The on-board computer subscribes to the anomalyReport topic and records the anomaly as telemetry.

TransferFrameComposer
The transfer frame composer assembles the transfer frames, which are then handed over to the communication system for transmission to the ground station. Each transfer frame starts with a set of standard real-time telemetry and is than filled up with either further real-time telemetry or historical telemetry or payload data.

ModeManager
TechnoSat incorporates three satellite modes, the LEOP (launch an early orbit phase) mode, the nominal mode and the safe mode, which are managed by the ModeManager. The LEOP mode is only entered upon the first activation of the on-board computer. After checking all satellite systems, the nominal mode is entered via telecommand. If any node reports a critical anomaly, the on-board computer enters safe mode. As TechnoSat implements an open mode concept, the operation of the satellite via telecommands is not restricted in safe mode.
Upon entering safe mode, however, the power control unit establishes a defined state of the satellite regarding powered components and the collection of extended historical telemetry is stopped, so that only standard historical telemetry is sampled. Figure 7 shows the diagram illustrating the TechnoSat satellite modes.

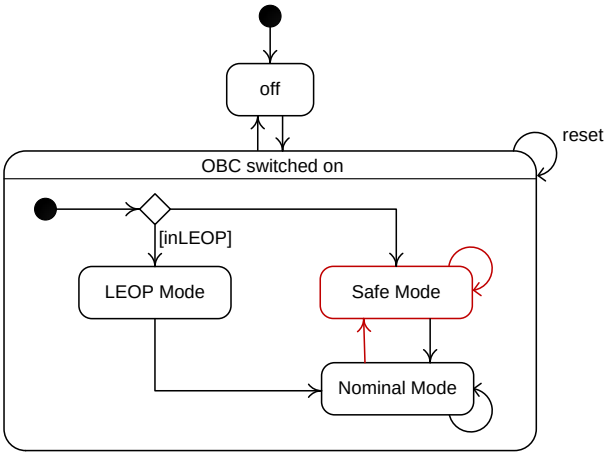

Figure 7. TechnoSat satellite modes.

TimedCommandsManager
Time tagged commands are required to perform payload experiments, when the satellite is not in contact with the ground station. Time tagged commands are managed in a centralised manner i.e. by the on-board computer. All time-tagged commands are stored in one time-tagged command list. Before this list can be executed, it needs to be activated via telecommand. The time tagged command list is deleted if a critical anomaly occurs or when the on-board computer is rebooted.

TemperatureReporter
As TechnoSat implements a passive thermal control system and therefore does not feature a dedicated thermal control computer, the temperature sensors are connected to existing nodes, i.e. the power control unit, the on-board computer and the attitude-control computer. This application manages the sampling of the temperature sensors which are connected to the on-board computer.

9 CONCLUSIONS

TechoSat is based on TUBiX20, a nanosatellite platform for 20 kg missions that is currently developed at TU Berlin. TUBiX20 features a single-failure tolerant platform design and follows a system wide optimisation approach, which is based on an extensive reuse of hard and software elements within the system. The satellite software is based on the modern building block operation system RODOS. All basic functionality is provided by a software infrastructure, which is identically implemented on every node of the satellite. In this manner, development and testing efforts for these functionalities are kept to a minimum, as compared to an individual implementation for every node. The on-board computer software additionally implements centralised functionalities such as time-tagged command management or telemetry transfer frame assembly.

ACKNOWLEDGEMENTS

The TechnoSat project is funded by the Federal Ministry of Economics and Technology (BMWi) through the German Aerospace Center (DLR) on the basis of a decision of the German Bundestag (Grant No. 50 RM 1219).

REFERENCES

[1] M. F. Barschke, Z. Yoon and K. Brieß (2013). TUBiX – The TU Berlin Innovative Next Generation Nanosatellite Bus. *In: Proceedings of the 64th International Astronautical Congress*, September 23 – 27, Beijing, China.

[2] M. F. Barschke, F. Baumann, W. Ballheimer, K. Grossekatthoefer, C. Nitzschke and K. Brieß (2013). TUBiX20 - The novel Nanosatellite Bus of TU Berlin. *In: Proceedings of the 9th IAA Symposium on Small Satellites for Earth Observation*, April 8 – 12, Berlin, Germany.

[3] Z. Yoon, W. Frese, K. Briess, A. Bukmaier (2014). System design of an S-band network of distributed nanosatellites. *CAES Space Journal*. 6 (1). pp. 61 – 71.

[4] M. F. Barschke, H. Adirim, O. Balagurin, W. Ballheimer, L. Dornburg, H. Kayal, D. Noak, C. Nitzschke, N. A. Pilz, H. Wojtkowiak and K. Brieß (2013). TechnoSat - A Nanosatellite Mission for On-Orbit Technology Demonstration. *In: Proceedings of the 27th AIAA/USU Conference on Small Satellites*, August 10 – 15, Logan, USA.

[5] F. Baumann, W. Ballheimer, K. Großekatthöfer, C. Nitzschke, K. Briess (2012). TUBIN – A Nanosatellite Mission with Infrared Imager Payload, *In: Proceedings of the 4S Symposium*, June 4 – 8, Portoroz, Slovenia.

[6] D. Noack, K. Brieß (2014). Laboratory investigation of a fluid-dynamic actuator designed for CubeSats. *Acta Astronautica*. 96. pp. 78 – 82.

[7] K. Brieß, R. Alavi, K. Jäckel and H. Podolski (2008). S-Band Communication for Nano- and Pico Satellites for Cross Platform Compatibility, *In: Proceedings of the 59th International*

Astronautical Congress, September 29 – October 3, Glasgow, Scotland.

[8] G. Kirchner, L. Grunwaldt, R. Neubert, F. Koidl, M. F. Barschke, Z. Yoon and H. Fiedler (2013). Laser Ranging to Nano-Satellites in LEO Orbits: Plans, Issues, Simulations. *In: Proceedings of the 18th International Workshop on Laser Ranging*, November 11 – 15, Fujiyoshida, Japan.

[9] W. Bauer, O. Romberg, C. Wiedemann, G. Drolshagen and P. Vörsmann (2012). Development of in-situ Space Debris Detector, *In: Proceedings of COSPAR-39th Scientific Assembly*, July 14 – 22, Mysore, India.

[10] W. Bauer, O. Romberg, C. Wiedemann, R. Putzar, F. Schäfer, G. Drolshagen and P. Vörsmann (2012). HVI-Test Setup of In-Situ Space Debris Detector, *In: Proceedings of the 63rd International Astronautical Congress*, October 1 – 5, Naples, Italy.

[11] O. Balagurin, H. Kayal and H. Wojtkowiak (2012). Validation and qualification of a CMOS based miniature star tracker for small satellites. *In: Proceedings of the 4S Symposium*, June 4 – 8, Portoroz, Slovenia.

[12] S. Montenegro, F. Dannemann (2009). RODOS: Real Time Kernel Design for Dependability, *In: Proceedings of Data Systems in Aerospace (DASIA)*, Istanbul, Turkey, May 26 – 29.

[13] S. Montenegro (2013). Visionary data management system for nano satellites (VIDANA), *In: Proceedings of the 9th Symposium on Small Satellites for Earth Observation*, April 8 – 12, Berlin, Germany.