Institute of Aeronautics and Astronautics: Scientific Series Band 2

Karsten Gordon

A flexible attitude control system for three-axis stabilized nanosatellites

Universitätsverlag der TU Berlin

Technische Universität Berlin

Karsten Gordon

A flexible attitude control system for three-axis stabilized nanosatellites

Die Schriftenreihe *Institute of Aeronautics and Astronautics: Scientific Series* wird herausgegeben von:
Prof. Dr.-Ing. Dieter Peitsch,
Prof. Dr.-Ing. Andreas Bardenhagen,
Prof. Dr.-Ing. Klaus Brieß,
Prof. Dr.-Ing. Robert Luckner,
Prof. Dr.-Ing. Oliver Lehmann,
Prof. Dr.-Ing. Thomas Grund

Institute of Aeronautics and Astronautics: Scientific Series | 2

Karsten Gordon

A flexible attitude control system for three-axis stabilized nanosatellites

Universitätsverlag der TU Berlin

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Universitätsverlag der TU Berlin, 2018
http://verlag.tu-berlin.de
Fasanenstr. 88, 10623 Berlin
Tel.: +49 (0)30 314 76131 / Fax: -76133
E-Mail: publikationen@ub.tu-berlin.de

Zugl.: Berlin, Techn. Univ., Diss., 2017
Gutachterin: Prof. Dr.-Ing. Sabine Klinkner
Gutachter: Prof. Dr.-Ing. Klaus Brieß
Die Arbeit wurde am 5. Dezember 2017 an der Fakultät V unter Vorsitz von Prof. Dr. Andreas Bardenhagen erfolgreich verteidigt.

Druck: Pro BUSINESS
Satz/Layout: Sebastian Grau, Karsten Gordon

Umschlagfoto:
Fotomontage: Marc Lehmann
Hintergrundbild:
NASA | https://commons.wikimedia.org/wiki/File:Sun_Glint_over_Atlantic_Ocean.jpg
Public domain

ISBN 978-3-7983-2968-3 (print)
ISBN 978-3-7983-2969-0 (online)

ISSN 2512-5141 (print)
ISSN 2512-515X (online)

Zugleich online veröffentlicht auf dem institutionellen Repositorium der Technischen Universität Berlin:
DOI 10.14279/depositonce-6415
http://dx.doi.org/10.14279/depositonce-6415

Contents

Appendices

List of Figures

List of Tables

Glossary

CEP

The celestial ephemeris pole (CEP) is the true celestial pole. It differs from the true rotational axis of the Earth with an amplitude of 0.01 arcsec [1]. 157, 158, 160

Continuous Integration

Continuous integration (CI) is a development practice, where "newly developed features are frequently integrated into the main project, which converts the integration process from a time-consuming and very complex task into a series of small steps, each applied very closely after the implementation of a new feature" [2], [3]. 18, 19, 69, 141

Embedded System

An embedded system is an applied computer system which performs a dedicated function, often within a system of electromechanical devices [4]. 34

Equation of the Equinoxes

The relation between GMST and GAST is called equation of the equinoxes.

$$GAST - GMST = \Delta\psi \cos\epsilon$$

It is given by the obliquity of the ecliptic and the length of the vernal equinox. xii, xiii

GAST

Greenwich Apparent Sidereal Time (GAST) is the angle between the IERS reference meridian and the true equinox. The relation between GMST and GAST is given by the equation of the equinoxes [5]. xii, xiii, xxii

GMST

Greenwich Mean Sidereal Time (GMST) is is the angle between the IERS reference meridian and the mean equinox. The relation between GMST and GAST is given by the equation of the equinoxes [5]. xii, xiii

IERS

The International Earth Rotation and Reference Systems (IERS) is an international organization, which provides data and standards related to Earth rotation and reference frames. 161

IGRF

The International Geomagnetic Reference Field (IGRF) is a mathematical model of the geomagnetic field which was developed by the IAGA. The current version of the model is IGRF12 [6]. 45, 53, 95, 124, 134

J2000

J2000 is the reference epoch of January 1st, 2000 at 11:58:55.816 UTC and corresponds to the Julian Date of 2451545.0. 157

Julian Date

The Julian Date (JD) is the number of days since January 1st, 4713 BC at 12:00 [5]. xiii

Modified Julian Date

The Modified Julian Date is the Julian Date substracted by 2400000.5 [5]. 42

SGP4

Simplified General Perturbations (SGP) is a mathematical model to determine the orbit of a spacecraft [7]. 46, 53, 133, 158

SLPC

The Sun low precision coordinates (SLPC) model calculates the Sun's position in the EME2000 coordinate system according to Montenbruck and Gill [5]. The accuracy of the model is 0.1–1 %. 45

State

The state includes all inner and outer quantities such as angular rate or magnetic field which are used for attitude determination and control. 7, 8, 15, 24, 41, 43–45, 47–49, 51–60, 65, 70, 76, 87, 102–105, 114, 115, 117, 119, 121–123, 125, 127, 133–135, 139–143, 156

Acronyms

ADC Analog-to-digital converter 130

ADCS Attitude determination and control system 5–8, 13–18, 21, 24, 25, 29, 30, 33, 34, 37, 38, 40, 41, 44, 47–49, 51, 53–56, 58, 60, 62, 65–70, 72–76, 85, 89, 96, 101, 111–114, 116–118, 121, 122, 126, 129–132, 134, 137–143, 166

AiL Algorithm in the loop 67, 68, 70, 72

ARW Angular random walk 117, 125

BB Building block 27, 29, 132

BC Before christ xiii

BEESAT Berlin Experimental and Educational Satellite 22, 24, 113, 155

BRITE Bright Target Explorer 4

CAD Computer-aided design 29

CAN Controller Area Network 11, 14, 24, 25, 38, 71, 121, 139

CBSE Component-based software engineering 9, 34, 35, 41, 140, 142

CDR Critical design review 69, 141, 142

CEP Celestial ephemeris pole xii

CI Continuous integration xii, 68, 142

CiL Controller in the loop 69, 72

CMOS Complementary metal-oxide-semiconductor 30, 91

COG Center of gravity 161

COM Communication system 24, 27

COTS Commercial off-the-shelf 13, 19, 138

CYGNSS Cyclone Global Navigation Satellite System 5

DCM Direction cosine matrix 77, 79, 80, 82, 83, 96, 98

DTM Detumbling Mode 118, 119

ECSS European Cooperation for Space Standardization 17, 19, 66, 67, 111, 141

EGSE Electrical ground support equipment 20, 21, 71, 72

EKF Extended Kalman filter 94

EME2000 Earth Mean Equator and Equinox of Epoch J2000 xiv, 156

EPS Electrical power system 24, 27

EQM Engineering and qualification model 27, 73

ESA European Space Agency 17

ESOQ Estimator of the Optimal Quaternion 46, 97

FDA Fluid-dynamic actuator 30, 112, 118

FDIR Fault detection, isolation and recovery 11, 15, 16, 24, 44, 45, 55, 56, 60, 62, 63, 65, 66, 139, 143

FIPM Fine Accuracy Inertial Pointing Mode 127, 134, 135

FLX Fluxgate magnetometer 130, 131

FM Flight model 27

FNPM Fine Accuracy Nadir Pointing Mode 127, 134, 135

FOAM Fast Optimal Attitude Matrix 46, 97, 98

FOR Fiber optic rate sensor 55–57, 89, 112, 113, 116, 117, 122, 125

FUSE Far Ultraviolet Spectroscopic Explorer 21

GAST Greenwich apparent sidereal time xii

GMST Greenwich mean sidereal time xiii

GNB Generic Nanosatellite Bus 4

GPS Global Positioning System 10, 32, 46, 102, 103, 130–134

GUI Graphical user interface 73

GYR Gyroscope 55, 57, 89, 113, 114, 116, 117, 125

HiL Hardware in the loop 21, 69, 73, 74

HISPICO Hochintegrierter S-Bandsender für Picosatelliten 30, 112

I^2C Inter-Integrated Circuit 11, 38, 63, 113

IAGA International Association of Geomagnetism and Aeronomy xiii

IC Integrated circuit 23, 31, 32, 43, 44, 65, 113, 122, 131, 134

ICRF International Celestial Reference Frame 156, 157, 161

ICRS International Celestial Reference System 156

IERS International Earth Rotation and Reference Systems xii, xiii, xvii, 158, 160

IGRF International Geomagnetic Reference Field xiii

IOD In-orbit demonstration 7, 30

IPM Inertial Pointing Mode 118, 119, 126

IRM IERS reference meridian xii, xiii, 158, 160

IRP IERS reference pole 160

ISB Integrated Sensor Board 113–115, 134

IT Information technology 18, 142

ITRF International Terrestrial Reference Frame 160, 161

ITRS International Terrestrial Reference System 160

JD Julian date xiii

LEO Low earth orbit 10, 22, 85

LVLH Local Vertical Local Horizontal 105, 162, 163

MEMS Micro electro-mechanical system 10, 11, 31, 32, 55, 89, 99, 112, 113, 116, 117, 125

MFS Magnetic field sensor 44, 45, 55, 63, 113–115, 124

MicroMAS Micro-sized Microwave Atmospheric Satellite 5

MOI Moment of Inertia Coordinate System 82–85, 104, 105, 161

MT Magnetic torquer 113, 119, 126

MTS Magnetic torquer system 110, 114, 119

N/A Not applicable 33, 42

NASA National Aeronautics and Space Administration 2, 5, 157, 159, 160, 162, 165

NIMA National Imagery and Mapping Agency 158

NORAD North American Aerospace Defense Command 158

NPM Nadir Pointing Mode 118, 119, 126

OBC On-board computer 24, 27

OOD Object-oriented design 14, 18, 59, 141, 142

OSCAR Orbiting Satellite Carrying Amateur Radio 2

PC Personal computer 71, 73

PCB Printed circuit board 27, 40, 113

PDR Preliminary design review 68, 141

PEF Pseudo Earth Fixed 158, 160

PPS Pulse per second 25, 40, 132

PSD Position sensitive device 113

QUEST Quaternion Estimator 45, 57, 97, 102, 117, 125

RCM Rate Control Mode 127

RODOS Realtime Onboard Dependable Operating System 26, 27, 71

RW Reaction wheel 122, 126–128

RWS Reaction wheel system 110, 112, 118, 121, 126, 128, 130

SAT Satellite-fixed Coordinate System 42, 105, 161, 162

SEU Single event upset 62

SFL Space Flight Laboratory 4

SGP Simplified General Perturbations xiii

SiL Software in the loop 27, 68, 72, 73, 75, 141

SLPC Sun low precision coordinates xiv

SNAP Surrey Nanosatellite Applications Programme 3

SOLID Solar Genarator based Space Debris Impact-Detector 30

SPI Serial Peripheral Interface 11, 38

SPM Suspend Mode 118, 119

SSO Sun-synchronous orbit 30–32

SSS Sun sensor system 55, 113–115

SSTL Surrey Satellite Technology Ltd 6

STR Star tracker 55, 121, 124–126, 130–132

TAR Target Coordinate System 104, 105

TBC To be confirmed 32, 155

TC Telecommand 13, 21, 31, 32, 40, 72, 73, 75, 140

TCP/IP Transmission Control Protocol / Internet Protocol 72

TDD Test-driven development 18, 19, 142

TEMED True Equator and Mean Equinox of Date 158

TLE Two Line Elements 158

TM Telemetry 13, 21, 31, 32, 40, 72, 73, 75, 140

TOD True Of Date 42, 82–85, 104, 105, 133, 157, 158, 160, 162, 163

TPM Target Pointing Mode 134, 135

TROPICS Timed-Resolved Observations of Precipitation structure and storm Intensity with a Constellation of Smallsats 5

TU Technische Universität xx, 1–3, 6, 7, 22, 31, 69, 129, 139, 155

TUBIN TU Berlin Infrared Nanosatellite 8, 29–31, 33, 34, 111, 118, 121, 126, 129, 130, 132–134, 139, 141–143, 155, 164, 166, 167

TUBiX TU Berlin inovative neXt generation satellite bus 7, 8, 18–27, 29–31, 33, 37, 38, 40, 41, 43, 44, 47–49, 51, 56, 62, 65, 66, 70–72, 75, 85, 88, 89, 102, 111, 121, 124, 129–132, 135, 139, 141, 143, 164

TUBSAT TU Berlin Satellite 3, 21, 22, 155

UART Universal Asynchronous Receiver Transmitter 11, 14, 38

UDP User Datagram Protocol 71

UHF Ultra high frequency 24, 30–32

UML Unified Modeling Language 35, 41, 119

USB Universal Serial Bus 71

UTC Universal time coordinated xiii

UTIAS University of Toronto, Institute for Aerospace Studies 4

VLBI Very long baseline interferometry 156

WGS84 World Geodetic System 1984 158, 160

WLAN Wireless Local Area Network 71

Symbols

ϵ	Obliquity of the ecliptic 158
λ	Eigenvalue 97
$\vec{\Omega}$	Reaction wheel angular rate vector 108, 110
$\vec{\omega}$	Angular rate vector 41, 42, 83, 91, 93, 100, 104–106, 125
$\vec{\omega}_B$	Angular rate vector bias 99
$\dot{\vec{\omega}}$	Angular acceleration vector 42, 91, 100
Π	Matrix of pole displacement 160
ψ	Vernal equinox 158
ρ	Density 87
$\vec{\tau}$	Torque vector 42, 86, 106, 107
Θ	True sidereal time (GAST) 158
A	Area 87, 96
$\vec{B}$	Geomagnetic field vector (inertial) 42, 45, 95, 101
$\vec{b}$	Geomagnetic field vector (body-fixed) 42, 43, 45, 86, 96, 101, 107, 115
c	Speed of light 87
$\vec{F}$	Force vector 86, 87
$\vec{g}$	Gravity vector 86
$\vec{I}$	Inertia tensor 84, 86
$\vec{m}$	Magnetic dipole vector 42, 86, 107, 110

N	Nutation matrix 157
$\vec{n}$	Normal vector 87
P	Precession matrix 157
q	Quaternion 41, 42, 83, 85, 95, 97, 104, 106
$\vec{r}$	Position vector 42, 46, 86, 87, 96, 133
$\vec{S}$	Position vector of the Sun (inertial) 42, 45, 101
$\vec{s}$	Direction vector to the Sun (body-fixed) 42, 45, 92, 101, 115
T	Temperature 42, 43
t	Time 42, 46, 52
$\vec{v}$	Velocity vector 42, 87

1 Introduction

The first chapter of this thesis discusses the motivation and the objectives for the research. To give the reader an introduction into the field of work and also to define the state of the art, the technological evolution of nanosatellites is reviewed briefly in the beginning. Thereafter, the individual goals are formulated and their pursuit is outlined.

1.1 The Technological Evolution of Nanosatellites

Starting with a brief discussion of the term *nanosatellite*, this section retrospects the emergence of nanosatellites in the early days of space flight. After several decades without any nanosatellite missions at all, the technological revolution of micro electronics made nanosatellites more capable and hence attractive for space missions again. Finally, the role of nanosatellites for science and Earth observation constellations and the growing interest of private companies in nanosatellite constellations is reviewed.

1.1.1 Small Satellite Categorization

The commonly used terms for small satellite categories are derived from the unit prefixes used in the metric system, e. g. *micro* or *nano*. Most commonly, the allocation of a satellite to one of these categories is based on its launch mass, where the actual numerical values for the category boundaries are not standardized. A widely used assignment is used by Bouwmeester and Guo in [8], while the TU Berlin follows a slightly different allocation (cf. Table 1.1). The approach to classify satellites according to their mass, however, has been criticized as arbitrary [9] and unsuitable [10], and new standards for categorization have been proposed [10]. Nevertheless, most surveys and databases on nanosatellites use this mass-based categorization and do not list satellites with a mass greater than 10 kg, which therefore also reflects

in the data available for the evaluation undertaken in the following sections. In the subsequent investigations, the author follows the classification by TU Berlin and satellites with a launch mass of 4–20 kg are also considered as nanosatellites.

Table 1.1: Satellite Classifications Based on Launch Mass [kg]

Category	Surveys [8], [11]	TU Berlin [12]
Microsatellite	100 – 10	120 – 20
Nanosatellite	10 – 1	20 – 4
CubeSats		4 – 1
Picosatellite	1 – 0.1	1 – 0.1

1.1.2 Nanosatellites Among the Pioneers of Space Flight

The first nanosatellite was in fact the second satellite in space at all. Four months after Sputnik (83.6 kg), Explorer 1 [13] was successfully launched by NASA on February 1st, 1958, and had a mass of only 13.9 kg. Others followed as part of the Explorer and Vanguard projects until 1961, and the first amateur radio satellites OSCAR 1 (1961) and OSCAR 2 (1962) [14] had a launch mass of 5 kg and 10 kg, respectively. After 1962, however, no nanosatellites were launched until several decades later in 1996, as can be seen from the survey of Bouwmeester and Guo [8], which includes all pico- and nanosatellite launches until 1998. An overview is given in Figure 1.1, where the large gap between the early sixties and the late nineties is clearly visible.

Bouwmeester and Guo suggest that the reason for the small satellite launch mass in the early years of spaceflight was a result of the limited payload capabilities of the launch vehicles. Their survey states that the first satellites were very simple and after the launch vehicles became more capable, the satellites also became larger and more advanced. The need for nanosatellites with very limited capabilities was no longer given after the first successful demonstration missions, since the advancing technology soon became too big, and hence the launch mass of satellites needed to be continuously increased for their integration.

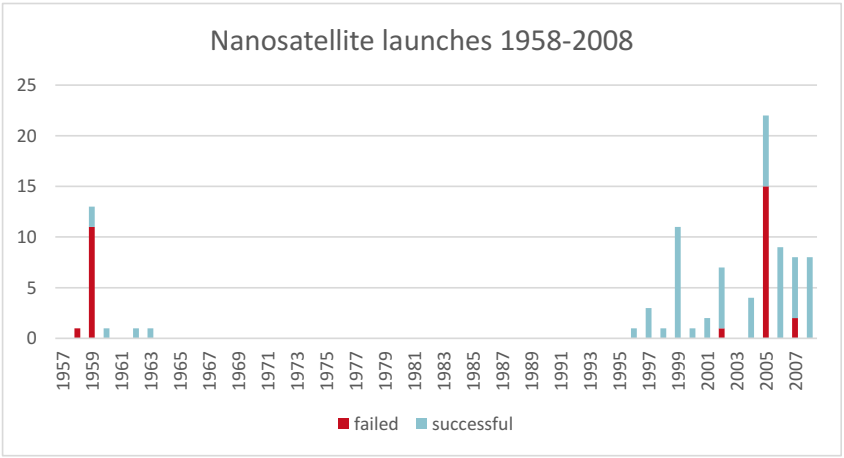

Figure 1.1: Nanosatellite Launches 1958–2008 (adapted from [8])

1.1.3 Technology Miniaturization

One of the first nanosatellite missions after a long break was carried out in 1998 by TU Berlin, whose TUBSAT-N/N1 satellites [15] demonstrated technology miniaturization by successfully performing the same communication experiments as the microsatellite TUBSAT-A nine years earlier [16]. By that time, energy-efficient micro-electronics had become available, which allowed the realization of more advanced missions also for smaller satellites. In 1996, Müncheberg, Krischke, and Lemke noticed a "new way of thinking in terms of satellite design", which resulted in a "tendency towards smaller, simpler units" [17]. Microsatellites had been introduced for civil and commercial satellite systems and a miniaturization beyond the microsatellite scale was predicted. Due to the reduced launch costs, this was at first especially interesting for universities, and after the CubeSat standard [18] was first released in 1999, the number of picosatellites began to increase immensely. One year after the CubeSat standard, the first missions to demonstrate a nanosatellite platform including three-axis-stabilization, SNAP-1, was launched [19]. In this period, the pico- and nanosatellites' mission objectives included

predominantly technology demonstration (71 %), operational use (52 %) and education (52 %), while scientific experiments were – if part of a mission – only limited [8].

1.1.4 Nanosatellite Constellations for Science and Earth Observation Missions

With the fast advance in mobile phones and other consumer electronics, the miniaturization of technology suitable for space applications increased momentum in the field of pico- and nanosatellites, which reflects in the vast increase of pico- and nanosatellite launches after 2010, as shown in Figure 1.2. The extension of the CubeSat standard for multiples of its form-factor entailed also larger, standardized deployment facilities and paved the way for 3U and 6U CubeSats [20]. By then, miniaturized satellite technology was capable of supporting complex science or Earth observation missions to a full extent. Due to their vast increase of performance on the one hand, and the comparatively low launch costs on the other, constellations and formation flying missions offered new opportunities for progressive and yet affordable missions.

Based on the nanosatellite technology demonstration with CAN-X1 (2003) and CAN-X2 (2008), the UTIAS/SFL developed the Generic Nanosatellite Bus (GNB) as a basis for the BRITE satellites, a nanosatellite constellation that monitors the brightness and temperature variations of stars [21]. It includes six spacecrafts which were launched in 2013 and 2014. In the same year, the CAN-X4 and CAN-X5 satellites [22], also based on the GNB, demonstrated formation flying with nanosatellites. At the same time, private companies also started their own nanosatellite programs. After its first technology demonstration mission called Dove-1 in 2013, the American company Planet launched Flock 1, a constellation of 28 3U CubeSats [23] in 2014. With more launches to follow soon after, Planet operates the largest constellation of satellites in space. The flock constellation provides a complete image of Earth with an optical resolution of 3–5 m.

The European Commission's FP7 NANOSAT project [11] researched ongoing and future nanosatellite projects ranging from 1998 until the 2020s. As can be seen from its database, the extremely rapid development in the field of nanosatellites will continue (cf. Figure 1.2). Here, the decrease in 2016 is due to launch delays, which also explains the great increase in 2017, where the

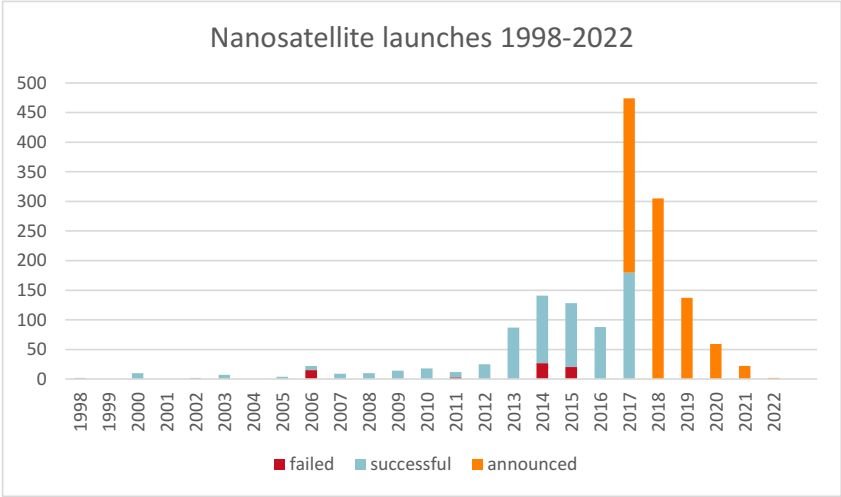

Figure 1.2: Nanosatellite Launches 1998–2022 (adapted from [11])

first half already sets the record for the most pico- and nanosatellites launches ever.

1.1.5 Present and Future Pico- and Nanosatellite Mission Programmatics

The CYGNSS is one example for a nanosatellite constellation contributing to NASA's Earth science program. Launched in December 2016, the 8 satellites with a mass of 18 kg each are utilized to predict weather, climate, and natural hazards [24]. The ADCS of CYGNSS facilitates three-axis-stabilization with a 2.1° knowledge and 2.8° control using horizon sensors, a magnetometer, a pitch momentum wheel, and magnetic torquers [25]. TROPICS is NASA's first Earth science CubeSat constellation mission and "involves twelve 3U-CubeSats (4 kg each) to determine the relationships between rapidly evolving storm structures and storm intensity" [24]. Based on the MicroMAS-2 CubeSat, three-axis stabilization with 0.5° accuracy is achieved [26].

From the perspective of the private sector, "the combination of cost-effective satellites, and the increased value of up-to-date imagery that is largely driving the commercial interest in Earth observation imagery, whether that is for agriculture, insurance, mapping or surveillance", as stated by Silva Curiel, Cawthorne, Sills, *et al.* [27] from the British company SSTL. As one example of a platform to support nanosatellite constellations, particularly for commercial applications, the SSTL-12 targets flexibility regarding different mission scenarios. In terms of attitude control, a performance range of 2° to 0.01° is specified.

With this growing interest of the private sector in pico and nanosatellites and the increasingly challenging missions which have become feasible using miniaturized spacecrafts, their requirements regarding dependability and performance increase steadily due to the pressure of competition. Hence, keeping the established technological solutions up to date has become a key business factor for space-related products or services. In the field of science, the technological advances accelerate the community's efforts to carry out even more ambitious missions in a shorter amount of time. Both trends entail the need for a reliable and yet adaptable design, which offers both a long-term compatibility for future mission scenarios and also a rapid update of technology to sustain a competitive platform. In this context, this work investigates a concept for a flexible ADCS for three-axis stabilized nanosatellites. The detailed thesis objectives are presented in the following section.

1.2 Thesis Objectives

Attitude determination and control systems for three-axis stabilized satellites incorporate multiple different sensors and actuators as well as data processing and control strategies which must be selected carefully considering a mission's requirements and constraints. Scientific research in this field offers various topics to focus on and different dissertations have been written at TU Berlin. While accuracy [28], agility [29] and robustness [30] have been worked on in the past, this thesis investigates an approach for the flexible design and verification of an attitude determination and control system (ADCS) for nanosatellites. Before the concept is elaborated and its application is presented, the following paragraphs will firstly clarify the motivation for the research.

In general, flexibility is characterized as "a ready capability to adapt to new, different, or changing requirements" [31]. As shown in the retrospective of their technological evolution, the requirements for nanosatellites have continuously increased with the new areas of application. Especially in the last decade, this trend was accelerated by advances in miniaturized technology and satellite constellations and formations emerged to carry out commercial and scientific missions. These missions generally impose high requirements in terms of an ADCS's accuracy, dependability, but also its economic efficiency and time to market.

Consequently, many nanosatellite designers establish platform concepts which are then applicable for different projects. The requirements for specific missions are, however, diverging with the broader range of application areas. One mission may consist of a large constellation of Earth observation nanosatellites for commercial purposes and require a high pointing accuracy at moderate unit costs and development effort, but also an especially short development time for continuous technology updates. A different mission may carry a scientific instrument demanding dedicated pointing modes or control maneuvers and permanent availability. On the other end of the scale, an in-orbit demonstration (IOD) mission's payloads may formulate only moderate ADCS requirements and hence low costs and development effort are the design drivers for the platform configuration. To be able to support such different missions, a platform concept must be able to adapt to varying requirements, that is, be flexible.

This thesis investigates a new concept for the flexible design and verification of an attitude determination and control system for such a nanosatellite platform. Chapter 2 discusses the impact of the technological evolution on the requirements in regard to the performance and design. Based on observations of the space market and ongoing satellite missions, criteria are derived to clarify in which parts of the design process or the systems architecture flexibility is required and why. In this context, TU Berlin's TUBiX20 nanosatellite platform is introduced, which is designed with the objective of modularity, reuse and dependability and therefore provides the basis for flexible subsystem design. In the role of systems engineer for software for the platform, the author contributed to shape the system's architecture accordingly. Chapter 3 presents the concept for the ADCS in detail, while the theoretical background for the state estimation and control techniques is subsequently provided in Chapter 4. The investigated ADCS concept is put into practice for the two

ongoing missions based on the TUBiX20 platform, namely TechnoSat and TUBIN, which the author describes in Chapter 5. Finally, the objective and motivation for this thesis are summarized and placed in relation to the outcome in the last chapter of this work.

The ADCS design is aligned with the TUBiX20 systems architecture to obtain maximum synergies for the development. However, the approach is completely applicable for other platform solutions or individual satellites. Especially the description of the attitude determination and control techniques in Chapter 4 may be read as a stand-alone collection of algorithms and serve other researchers as a reference. Throughout the presentation of the concept and its realization, special emphasis is placed on the consistency and integrity of information. Therefore, all symbols and formats for the different state quantities, coordinate systems and diagrams are used uniformly throughout the whole document including all glossaries to allow their traceability from abstract concepts to detailed mathematical descriptions.

2 Design Approaches for a Flexible Attitude Control System

This chapter presents investigations on design approaches for a flexible attitude control system. The first two sections derive design considerations to further clarify the objective of this thesis. To provide a theoretical basis for the research carried out in Chapter 3, Section 2.4 introduces the concept of component-based software engineering.

2.1 Observations of the Satellite Market and Missions

In the following section, criteria for the design and verification of a flexible attitude determination and control system are identified. Firstly, observations from the state of the art in nanosatellite missions in general and attitude determination and control systems in particular are recorded. As a second step, design criteria are derived from these observations, which serve as a reference for the conceptual design which follows in the next chapter.

2.1.1 Commonly Used Sensors and Actuators

As can be seen from the range of nanosatellite missions examined in Section 1.1, there is a common set of sensors and actuators which is utilized in most nanosatellite missions. According to the survey of Bouwmeester and Guo [8], Sun sensors and magnetometers are the most commonly used sensor type within all pico- and nanosatellites between 1957 and 2009. Nearly 30 % of all missions in their database were equipped with either one or both of these sensor types. However, there is a large variety of different designs. Sun sensors range from simple photocells to digital sensors with integrated signal processing, which in turn reflects in a wide range of accuracy from several degrees to arc minutes. Obviously, they only provide measurement data

when illuminated by sunlight, which also allows self-sufficient power supply. Magnetometers are light-weight, energy-efficient and relatively easy to use. Since most pico- and nanosatellites are launched into a low earth orbit (LEO), measurements of sufficient strength compared to the sensor's accuracy are permanently available. With Sun vector and geomagnetic field measurements, basic attitude estimation is possible (cf. Section 4.4.6). During an eclipse, however, only the magnetic field measurements are available. Here, angular rate measurements are often used to predict the attitude. As a low-cost and power-efficient realization, MEMS gyroscopes are widely utilized [8]. Apart from attitude prediction, the angular rate measurements are also used for attitude control directly. A different sensor providing vector measurements – yet less often used than Sun sensors or magnetometers – is an Earth horizon sensor. Mostly based on infrared cameras, a wide range of products is available. For pico- and nanosatellites, predominantly low cost and coarse versions are selected.

In terms of attitude control, only a minority of all pico- and nanosatellites before 2009 performs active stabilization (40 % according to [8]). The most common control principle is magnetic actuation, either via magnetic coils or torque rods (magnetic torquers). This may be explained by the moderate pointing requirements, but also by their robustness, low energy consumption, low costs and simply a lack of alternatives: other actuator types like reaction wheels or thrusters have only been available recently, which will be examined in the following section.

2.1.2 Miniaturization of High-Accuracy Sensors and Actuators

As discussed earlier, nanosatellite missions for Earth observation or scientific payloads did not emerge before the beginning of this decade, when the technological progress led to miniaturized payloads. By then, the limited resources regarding power supply or dimensions complied with the reduced requirements of nevertheless complex instruments. Miniaturized reaction wheels and star trackers were developed and could be verified in-orbit around the same time. As a result, nanosatellites capable of high-accuracy attitude determination and control became available. Especially for formations or constellations, highly accurate position information is crucial, Therefore, many nanosatellites are nowadays also equipped with GPS receivers.

2.1.3 Growing Supply and Demand for Components

Apart from the increasing variety in different sensor types, e. g. the emergence of nano reaction wheels, star trackers and miniaturized thrusters, the technological evolution of components and the increasing interest in pico- and nanosatellites has lead to a growing range of products offered by an increasing number of manufacturers. Even the integral parts of nearly every nanosatellite mission today – Sun sensors, Magnetometers and MEMS gyroscopes – are available in a broad range of different versions from simple and cheap to complex and high-performing. In this very active market, manufacturers release new products frequently or discontinue existing ones, which often comes with modifications in data interfaces such as UART, CAN, I^2C or SPI, but also different configuration procedures as well as communication protocols.

2.1.4 Evolution of Mission Objectives

As can be seen from the examples reviewed in Section 1.1, nanosatellite projects have developed from simple demonstration missions to very complex constellations with several hundred individual satellites. Apart from the increasing requirements regarding a single satellite itself, the operations for such missions have also undergone a radical change. The in-orbit demonstration of one or more payloads may be achieved via a limited set of dedicated commands and is therefore of manageable complexity even for a small team working with predominantly manual operations. A large constellation of satellites, on the other hand, is only manageable with a high grade of autonomy for the individual satellite, since there are simply not enough resources for gradual command sequences. Furthermore, formation flying and inter-satellite communication require new algorithms and operational modes, extending the traditional, single-satellite concepts which are often limited to pointing towards nadir or a target on ground.

To achieve a high degree of autonomy, complexity must be transferred from operations to the spacecraft. This applies for conducting experiments or maneuvers, but especially for fault detection, isolation and recovery (FDIR). Here, the individual satellite must be capable of maintaining its availability without extensive operations from ground for recovery. Therefore, this intelligence must be implemented into the satellite's software.

A different, yet important aspect is the sensor calibration. While for a mission with one or a few satellites, a significant percentage may be done on ground, this is not feasible for large constellations. Here, in-orbit calibration must be implemented. To this end, the software must allow the integration of in-orbit estimation techniques as well as autonomously performed calibration maneuvers.

2.1.5 Corporate Environment

Surveys show that today a large part of pico- and nanosatellites come from private companies (40.4 % of all missions examined in [8], cf. Section 1.1.4). These companies either offer a service, i. e. bringing a customer's payload into space, or sell data recorded by their own instruments as a product. In both cases, failure comes with financial loss, either as penalty for a not fulfilled contract or a decreased offer of marketable data. Therefore, a high degree of reliability and availability is demanded for the satellites. To be able to compete, a quick entry to the market and hence a short development time is mandatory. However, private companies usually have sufficient budgets to acquire the resources demanded to carry out such a development.

2.1.6 University Environment

Although the missions carried out by universities have recently become equally aspirational with regard to the technological challenges, the environment here is still quite different. University missions generally conduct research, often as technology demonstration of novel components, while complex payloads are sometimes introduced by partners. The development and verification of the satellite from design to launch and the commissioning as well as subsequent operations and evaluation of scientific experiments are sometimes not performed by a single institution. However, the challenge here is the development of a complex spacecraft with only limited resources. Depending on funding, university missions generally have a small budget and hence the most capable hardware or lab equipment is not always available. Moreover, university projects are often carried out by young researchers or students who may leave the university after the project, which makes the transfer of knowledge from one mission to another more difficult.

2.1.7 Integrated Solutions for Complete ADCS

A trend which is observable on the market of satellite components is the development of integrated solutions for attitude determination and control [32], [33]. Those integrated systems incorporate all required sensors and actuators and furthermore a processing unit which runs the control software and offers a data interface to the rest of the satellite. Using such an integrated solution within a nanosatellite mission obviously simplifies the mission's ADCS design; only the interfaces to the rest of the system, i. e. telecommand and telemetry need to be implemented. On the other hand, their manufacturers need to carefully design their systems architecture to meet the requirements of the market. This perspective is shown in the following paragraph.

The component market situation described in Section 2.1.2 and Section 2.1.3 is a major influence factor for the design of such systems. To support both highly demanding as well as cost-efficient missions, a company has to offer a broad range of products which are preferably all based on the same modular design to allow a comprehensive and consistent product development. Since new components are released frequently while others are discontinued, the system must allow the extension of new components or removal of deprecated ones, preferably in short life cycles and with minimum development effort. To maintain their competitiveness, companies are under a high innovation pressure. The capabilities and accuracies of satellite components increase continuously, alongside with the requirements of a mission. Therefore, frequent updates or upgrades are of great importance for integrated solutions.

2.1.8 Software Development Techniques

In recent years, software has become a substantial part of the spacecraft development process. With the growing requirements, the complexity of on-board tasks is increasing drastically and therefore a well-structured and sophisticated software design is needed. With the technological evolution of micro electronics, high-performing and yet small and power-efficient microcontrollers have become available. Especially in pico- and nanosatellite projects, these commercial off-the-shelf (COTS) components are often preferred to large, energy-consuming but space-proven on-board computers. Regarding the software design, different techniques such as code generation [34] and

object-oriented design [35] have been introduced into space science, however, a long time after being applied in other sectors like consumer electronics or the automotive industry. In the past years, Linux [36] and open source software projects [37] have also become of increased influence. When it comes to design objectives, modularity is often a key factor, especially for attitude control systems' software [38], [39].

2.2 Derivation of Flexibility Criteria

In this section, criteria for flexible design and verification are derived from the observations of the satellite market and missions discussed in the previous section. In this context, the term flexibility is referred to as the "capability to adapt to new, different, or changing requirements" [31], which applies to the development, verification and in-orbit operation of the ADCS as part of a reusable nanosatellite platform.

2.2.1 Integrate Updated or Novel Technology

As pointed out in Section 2.1.1, most attitude determination and control systems incorporate a basic set of the same sensors and actuators. However, the integration of different components is still important due to the following reasons. Firstly, the supply of these component types may change. The market is in motion, and although the same component types may be used for a new mission, some novel products may have been released, promising more accuracy or better performance. On the other hand, manufacturers may discontinue a certain product and hence a replacement must be found (cf. Section 2.1.3). However, these newly selected components may require different hardware interfaces for communications (e.g. CAN instead of UART) and therefore hardware adaption may be inevitable. This in turn affects the software, since new drivers need to be implemented to adapt to new hardware interfaces and communication protocols. Furthermore, a new component is often operated differently in terms of operational modes, (register) settings or calibration procedures. The subject of rapid technology update is discussed in more detail in collaboration with the author in [40].

Furthermore, novel components are introduced such as miniaturized star trackers or reaction wheels (cf. Section 2.1.2) which provide a valuable addition to the components already in use, since new application areas for nanosatellites demand for steadily improving performance. However, the integration of such new technology should preferably come with minimum integration effort.

As stated in Section 2.1.7, integrated solutions for complete attitude determination and control have become a new trend on the market. These systems offer a cost- and time-efficient way to realize ADCS functionality, but may however need to be extended with additional sensors or actuators for certain missions, e. g. due to demanding accuracy requirements or control techniques needed. Moreover, these integrated solutions still need to be incorporated into the overall platform and interact with other subsystems such as orbit control, but also require interfaces to telecommand, telemetry and FDIR mechanisms.

2.2.2 Enable Scalability

To meet different mission requirements with a single solution, a platform should be scalable regarding development effort, performance and resources. For one mission, coarse attitude control may be sufficient, yet the mission should be realized within a short time frame and a limited financial budget. Here, it is desirable to derive a configuration from the platform which is reduced to the essentials and only integrates components which are really necessary, since not only procurement costs, but also integration and qualification effort may be saved for both hard- and software. On the other end, a different mission may require high-precision attitude control and high reliability. In this case, a configuration incorporating additional, highly precise sensors and actuators is required. However, utilizing insights and existing implementations previously made with a basic configuration is beneficial: building on the same foundation, re-use will shorten the development time and increase the reliability, especially if experiences with in-orbit operations have been made.

Besides adding novel components, the ability to integrate new algorithms for sensor calibration, state estimation and control as well as actuator control increases an ADCS's flexibility. In case a novel sensor such as a star tracker extends the existing hardware set, the software to process its measurement data and integrate these into the state determination concept needs to be added

as well. The same applies for a new actuator, since it may require different control techniques. For instance, the algorithms for three-axis-stabilization with reaction wheels are different from those using magnetic torquers. Even if a satellite already uses three reaction wheels, a fourth one may be added to provide single failure tolerance, and hence a new algorithm to distribute the control torque is required (cf. Section 4.5).

Although the utilized hardware stays the same in some cases, one may still desire to correct or improve implemented algorithms. This applies for both the reuse of a platform for a new mission as well as the development and even operational phase within the same mission. Here, new ideas may be tried to improve performance, autonomy or robustness. The possibility to experiment with new algorithms is valuable within both private companies and universities. While the former may achieve better product quality (e. g. due to better pointing accuracies or reduced jitter) or increase reliability of their services by implementing improved FDIR mechanisms, the latter have the drive for innovation as part of their research objectives.

Considering the long-term development of a satellite platform, its capability to integrate new functionality is important to keep up with the evolution of mission objectives described in Section 2.1.4. To operate a satellite based on this platform within a constellation demands new functionality for inter-satellite communication and orbit control, which in turn result in new ADCS requirements. Furthermore, sensor calibration may be performed on ground for a single satellite, but for large constellations this is not feasible and in-orbit solutions need to be implemented.

When modifying a software system's functionality, the developer may run into the danger of increasing its complexity. Here, a well-structured systems architecture which partitions the functionality into building blocks facilitates a good overview. It further allows the introduction of new features by the replacement of individual building blocks or their extension. This way, the modifications to the existing code are kept low which is particularly important if the system is flight-proven already (cf. Section 2.2.4). However, the system's ability to scale is also relevant in the other direction: the reduction of complexity. For missions with moderate requirements, but a smaller financial budget and/or a shorter timeline, the absence of components such as expensive high-precision sensors reduce the costs for procurement and shorten the development and qualification time. From the perspective of a platform which

has been extended throughout several missions and has evolved to a highly accurate but complex and costly system, such a removal of features is only feasible if the modifications are small. Adding code to a software project is usually easy, while reducing code is rather difficult unless it is partitioned into building blocks whose removal do not imply modifications elsewhere.

2.2.3 Develop and Verify Gradually

Space missions usually follow a process model such as the ECSS space project life cycle by the European Space Agency (ESA) to ensure a systematic project management approach. The ECSS process is divided into Phase 0 to Phase F, guiding the space project from feasibility studies and design via production and verification until utilization and disposal, as can be seen in Figure 2.1.

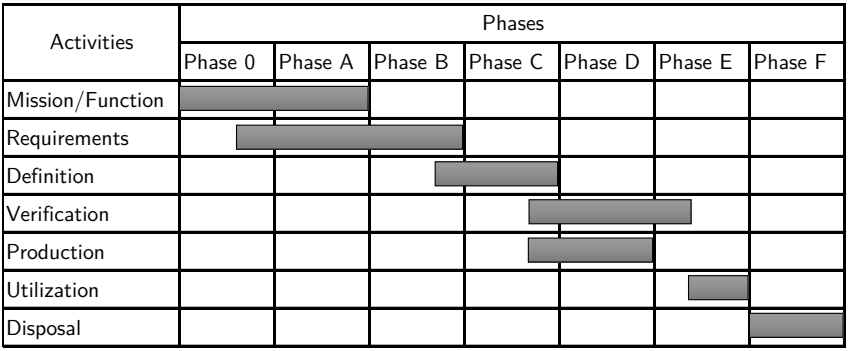

Figure 2.1: Typical Space Project Life Cycle According to ECSS [41]

However, this life cycle only provides a rough time framework for the project's progress and its individual phases may last a considerable amount of time. Here, a detailed breakdown of the tasks performed is helpful to identify which goals can be achieved at which time. A satellite's development process is very complicated due to many dependencies between the different subsystems and there is a potential risk that a delay in one subsystem affects the progress of the others. For instance, when lab facilities such as test beds or even hardware for the ADCS are developed within the project, their manufacturing may be

delayed and hence the available time for implementation and verification of the ADCS may be shortened. Moreover, the procurement of components may be delayed as well. It is therefore important to decouple the different subsystems' development, e. g. by using simulated hardware. As a result, the ADCS software may be designed and implemented before the hardware is available and therefore the feasibility of its concept can already be proven to some extent at an early stage of the project. By the time all components are available, the integration time is then shortened significantly. Moreover, insights during model or software development may create synergy effects for the hardware development if performed concurrently, as could be observed in the development of the TUBiX20 platform (cf. Section 2.3).

Due to the dependencies of hardware and software development, the software is usually under ongoing development until the end of a spacecraft project and is often in the critical path. With the challenges due to new areas of application for nanosatellite missions and supported by the increasing processing capabilities of modern hardware, the software complexity grows drastically. However, the approach towards software development often does not reflect these increasing requirements and the expense is often underestimated. Therefore, it is important to transfer the advances in software development techniques and processes from the information technology sector into the field of space technology. While design techniques such as object-oriented design or frameworks result in effective and yet well-structured code, innovative design processes shorten the development time while increasing the reliability. Generally applied in other industries for a long time, test-driven development and automated test frameworks offer a comprehensive verification. Here, new software features are coded in small, immediately testable parts. To execute such tests, frameworks such as ECatch [2] may be then used together with build-servers (e. g. Jenkins [42]) for the automated build, test and deployment of the software. This way, the integration process is transformed from a very complex and time-consuming task into a series of smaller steps, ensuring that the software is already well tested at an early stage and hence reduces time spent debugging the software on the target. This practice is often referred to as continuous integration [3]. Apart from the increase of software quality, the development process becomes more traceable and hence the progress may be evaluated more accurately.

Considering the evolution of a satellite platform software over several missions, the continuous integration practice yields further benefits. Since the software

is partitioned into small parts each verified by unit tests, modifications may be tested easily, always assuring that the software as a whole is still correct. Once again, this is important for both the private and the university environment. For the former, a reduced development time due to the minimized modification effort and also a direct provability of correctness is essential. As described in Section 2.1.6, the latter is affected by frequent staff turnover and hence knowledge transfer is a problem. Here, the unit tests help to understand the system's structure and functionality despite its complexity. The test-driven development process often results in a detailed documentation and the instant testability increases confidence in the code even after its original developers have left the institution, so it is more likely that follow-up missions will reuse the code instead of starting from scratch.

For more detailed information, practical experience gained during the application of continuous integration for the software development of the TUBiX20 platform is shared by the author and the software team in [2].

2.2.4 Minimize Modifications to Verified Hard- and Software

Qualification consumes a significant amount of financial resources and development time in every space project. Due to the hazardous environmental conditions, extensive tests have to be performed in order to prove that the satellite withstands radiation, vacuum and extreme temperature changes in-orbit as well as vibration and shocks during its launch. Depending on the model philosophy followed, there are different stages of qualification with separate models. Detailed information on testing can be found in the ECSS standards [43].

Generally, tests are executed firstly on component level to examine new, not yet space-proven hardware, and later on system level to qualify the overall spacecraft. Obviously, the expenses for such tests increase with the amount of new technology introduced. By selecting only space-proven components, the qualification effort on the component level is minimized. For satellite platforms following a COTS philosophy, it is therefore important to reduce the modifications to hardware and software verified in-orbit to a minimum. Hence, the qualification effort is kept low while still benefiting from the agility of the COTS market.

2.2.5 Support Concurrent Mission Design

The life cycles of different missions based on the same platform may overlap and therefore different variants need to be maintained at the same time. While one mission might be in its in-orbit operations phase and performance is enhanced or secured with software uploads, another satellite may be already under development. Derived from the same design, yet independent implementations with diverging adaptions require maintenance at the same time. The configuration management has to support these projects in terms of different hardware versions and compositions as well as software functionality and parameters. However, the change set between both should be kept at a minimum to reduce development effort and benefit from operational insights for both missions at the same time. This is especially important when flight experience was gained in-orbit, as discussed in the previous section. The TUBiX20 platform design responds to this requirement with a flexible configuration management which supports building the target-specific software, yet targeting maximum re-use. This will be presented later in Section 2.3.3.

2.2.6 Use Synergies

For both hardware and software, the overall scope of work goes far beyond the satellite under development itself. During its planning and development, but also integration and verification, a broad range of support equipment needs to be developed. First mission planning, dimensioning and design ideas are usually generated and elaborated using simulation models. During the detailed design, manufacturing and verification phases, these models need to be gradually refined and updated to be of further use. Once established, they may be used in a following mission from the start and therefore provide comprehensive information of the system. However, their design must allow scalability in both directions and they must be updated on a regular basis in order to avoid becoming obsolete.

The electrical ground support equipment (EGSE) is required for the operation of the satellite during qualification. Generally, developing hardware or software which is only used within a certain project phase comes with an overhead which is to be avoided. It is therefore advisable to aim at the final solution as directly as possible. A good example for such an approach is the modular

EGSE developed for TUBiX20, as presented in collaboration with the author in [44]. Here, a server-based architecture provides access to the platform's central data bus. An interface board translates message formats and allows the connection to the university's ground support software for telecommand and telemetry in all stages of the development process. From a purely virtual simulation of the satellite's software to operations during functional verification or qualification tests, the same EGSE is used. Besides the reduced overhead in software development, this architecture provided helpful insights into the satellite's behavior from early stages of development and hands-on experience in operating the system from the very beginning of the project [44].

2.2.7 Reconfigure In-Orbit

While the preceding criteria have discussed a platform's evolution and adaption to different mission requirements from one mission to another, this final section highlights the benefits of its flexibility regarding reconfiguration in-orbit. Rearranging functionality might be necessary due to different reasons.

The integration of new algorithms into the system is already discussed in Section 2.2.2. Once verified by simulations and potentially in hardware in the loop tests, such new algorithms may be deployed within the current mission via a software upload. Once again, it is desirable to keep the modifications to the software to a minimum and therefore the partitioning of functionality is crucial. Another case for in-orbit reconfiguration is the loss of a component, which may be compensated by implementing a different control strategy. An example for such an event is given by the FUSE satellite. Two and a half years after its launch in 1999, mechanical failures of two out of four reaction wheels left only two reaction wheels operational. The implementation of a hybrid control strategy using the existing magnetic actuators and the remaining reaction wheels restored the ADCS performance back to sub-arcsecond pointing accuracy and stability [45].

2.3 The TUBiX20 Nanosatellite Platform

Technische Universität Berlin has a history of space science missions of more than 25 years. Since the university's first satellite, TUBSAT-A, was launched

in July 1991, a total number of twelve satellites were brought successfully into orbit, while another nine are under development. Table A.1 in the appendix shows a complete list of all TU Berlin missions and further information may also be found in [46]. The missions are assigned to three different research programs: the TUBSAT series (1991–2007) included seven satellites which provided communication services and introduced interactive attitude control for Earth observation. In 2009, the BEESAT picosatellite series was initiated to advance research in the field of miniaturization. So far, four BEESAT satellites have been launched. The university's latest research program is TUBiX. This platform series is developed in two different scales to support satellites with an approximate mass of 10 kg (TUBiX10) and 20 kg (TUBiX20), respectively. Within the S-Net mission [47], four TUBiX10 satellites will demonstrate inter-satellite communication for a distributed satellite system. The research of this thesis was carried out in the context of the development of TUBiX20 [48]. Therefore, this platform will be presented in more detail in the following section. In the role of systems engineer for software within the development of TUBiX20, the author contributed to shape its architecture according to the requirements for a flexible nanosatellite platform. Its design approaches were then transferred and elaborated by the author as a basis for the investigation into a flexible attitude determination and control system conducted in this thesis.

The TUBiX20 nanosatellite platform's design objective is to meet different LEO mission requirements. To achieve a high level of flexibility regarding diverging mission scenarios, a generic, single-failure tolerant systems architecture has been developed. The key design considerations for this architecture are modularity, reuse and dependability [49]. While for the majority of satellite projects the hardware is developed first and the software is then adapted to its finalized design, the TUBiX20 team obtained a consistent architecture by combining modularization with the hardware/software co-design approach. Hence both domains are well-coordinated from the beginning. Insights during software design could be transferred to the hardware development and vice versa, and therefore synergy effects could be exploited.

The system is partitioned into self-contained functional modules on the smallest reasonable level with mutually matched interfaces in hard- and software, which results in an early abstraction of the system's functionality from its realization. Consequently, adaptions in one domain implicate minimum modifications in

the other. For example, if an electronic component should be replaced, only its according software driver needs to be adjusted.

Like nearly all satellite systems, the TUBiX20 platform is divided into subsystems to perform its fundamental tasks. Due to the modularized approach, the structure is further defined by a functional hierarchy which consists of four levels, as shown in Figure 2.2. The lowest platform level is called a component and is used for single hardware entities like power switches or magnetic torquers, which are individually controlled by software drivers. To form a functional unit, one or more components are aggregated to a device. For example, three (uniaxial) magnetic torquers and corresponding power switches and current sensors form a magnetic torquer device. Each component is in turn either assigned to the user layer or the service layer. The components of the user layer provide the functionality for a specific task (e. g. the magnetic torquers), while the components of the service layer form the infrastructure to operate them (e. g. power supply control and current surveillance). The next platform level comprises the subsystems of the satellite and may consist of several computational nodes and devices. Finally, the system level includes all functionality for high-level operations such as mode control or time-tagged command distribution.

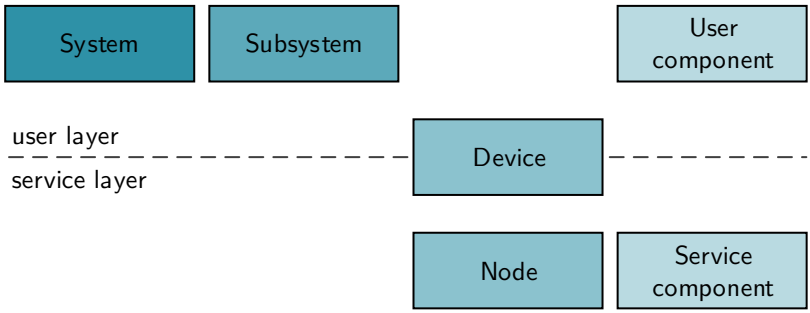

Figure 2.2: TUBiX20 Platform Levels (adapted from [50])

The distinction between user and service layer allows the identification of general solutions for recurring implementation issues: service components like power switches or surveillance sensors are decoupled from the individual setting they are used in. For example, the same ICs, circuitry and software

drivers may be used here, regardless of the user layer component which is controlled, be it, for example, an angular rate sensor or a magnetic torquer. This does not only enable extensive reuse, but also allows for a comprehensive and yet flexible fault detection, isolation and recovery (FDIR) strategy, which is investigated later in Section 3.3.5. The concept of devices and components, in turn, is the basis for a distributed attitude determination and control system. This will be discussed as the first aspect of this thesis's research in Section 3.1.

To give the reader a general overview, the description of the platform focuses on the subsystem level in this section. Each subsystem is assigned to a self-contained, cold redundant computational node, following the objective of flexibility. To minimize the design effort, all nodes are based on the same generic architecture, which will be presented in more detail in Section 2.3.1. All nodes are contained in an electronics box which provides their mechanical interface, basic radiation shielding and their connection to the redundant power bus and the redundant data bus. As data bus standard, CAN is chosen for two reasons. Firstly, CAN provides multi-master communication for distributed networks and is therefore suitable for the TUBiX20 architecture. Secondly, the standard targets high reliability and has been used successfully in the BEESAT satellites. Matching this distributed approach, the software is implemented as a network of building blocks communicating via a middleware, which aligns with the modularized hardware and decentralized communication. This software infrastructure is described later in Section 2.3.2.

There are four main nodes which perform the platform subsystems' tasks. The electrical power system (EPS) node controls power generation and distribution as well as the redundancy of all nodes and the power and data bus. It therefore runs in a worker/monitor configuration in hot redundancy. The on-board computer (OBC) node is responsible for all system level activity such as mode control, time-tagged command distribution and telemetry management, while the communication system (COM) node facilitates transmissions with the ground station using two redundant UHF receivers [49]. Finally, the attitude determination and control system (ADCS) node runs the core algorithms for state determination and control and furthermore connects a basic set of sensors and actuators directly to form the basic ADCS configuration of the platform. Additional sensors and actuators may then be connected to the data bus to extend the ADCS's capabilities. Therefore, the ADCS is adjustable towards diverging mission requirements. The detailed concept of the flexible

ADCS is presented in Chapter 3. Figure 2.3 depicts a block diagram showing the four main nodes and their power and data bus connection.

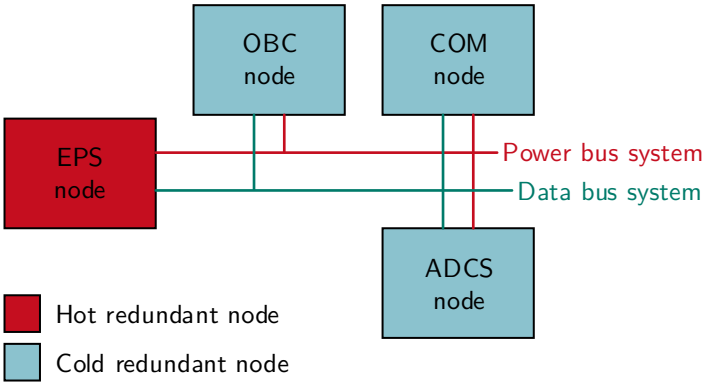

Figure 2.3: TUBiX20 Main Computational Nodes [49]

2.3.1 Generic Hardware Concept

As mentioned above, modularity is one of the key design criteria for the TUBiX20 platform. In terms of the hardware nodes, this goal is put into practice by reusing a set of reference components such as microcontroller, watchdog, power switches or CAN transceiver for all nodes – including their correspondent circuitry. The connection of nodes to the network is achieved via a standardized interface connecting the power and data bus as well as a pulse per second (PPS) signal for time synchronization. This template-based approach enables the extension of the platform's functionality via additional nodes while keeping the design effort low.

Since components like sensors and actuators from different manufacturers may use different electrical interfaces, their integration into the platform may need additional hardware for interface translation. While in monolithic architectures, the – maybe even flight-proven – hardware needs to be modified, which comes with a high effort and risk, the TUBiX20 hardware architecture allows integration via interface nodes. Such an interface node may then reuse the reference components and circuit templates for its integration into

the network. Here, there is no interference with the other hardware and component and its interface node may even be removed effortlessly when outdated. Self-evidently, this concept is also applicable for the integration of payloads, as has been presented with contribution of the author in [51].

2.3.2 Network of Software Applications

The generic hardware concept focusing on modularity and reuse provides a basis for its counterpart in the software domain: a distributed network. Since the hardware nodes share the TUBiX20 interface and use the same microcontroller as well as a common set of components, the same processor architecture is used and the identical hardware-dependent code may be reused on all nodes. Consequently, all TUBiX20 nodes' software run on the same operating system, which is called RODOS [52]. Developed especially for satellites, it provides a middleware for task communication using the publisher-subscriber protocol.

All functionality may be partitioned into applications (also referred to as building blocks) which exchange their data only via the topics of the middleware. Therefore, these building blocks are independent from each other and may be added or removed without modifications required. Thus, implementing the functionality dedicated to a hardware component, either connected directly to the data bus or using an interface node, represents the matching counterpart of the modularized hardware described in the previous section.

All code required for the reference components reused on every node such as (hardware) watchdog trigger, time synchronization mechanisms or command interface is implemented in a library of global applications [53]. Components which are compliant with the TUBiX20 hardware interface regarding power and data bus connection, but do not run on the RODOS operating system, may be integrated into the software network via dedicated building blocks for protocol translations. As the data bus is centralized, these are independent from hardware and hence may run on an arbitrary node of the network.

2.3.3 Software Build Configuration Management

This section will present a comprehensive strategy for the software build process to provide complete backwards and forwards compatibility throughout

multiple missions based on the platform and hence support concurrent mission design (cf. Section 2.2.5). After different aspects for variations for hardware and software are named, it is shown how the build management supports different configurations for its according adoption.

As described previously, the different TUBiX20 nodes are based on the same generic hardware and software architecture, and hence all nodes have a global, common set of hardware components included on their PCBs and run applications from the global library. However, each node obviously implements a specific set of functionality due to its subsystem's task and therefore may connect additional, node-specific hardware components and run node-specific software applications. Moreover, the development process of a node results in different revisions, since for example pin assignments may change from the first prototype to the flight model version. The RODOS operating system offers various ports for different controller architectures, including its emulation as guest process on a Linux installation. This is used for software in the loop simulations, which will be described later with the verification process in Section 3.4. Throughout the complete satellite life cycle, different models such as engineering and qualification model and flight model need to be supported, which may slightly differ from calibration parameters to completely exchanged component revisions. Intended as a platform for different missions for the university, the configuration management should support software development for multiple satellites at the same time in all phases of their individual life cycle.

Figure 2.4 shows the TUBiX20 configuration management for the software build process. In the horizontal direction, the build parameters *node*, *revision* and *use case* are given, which realize the configuration of the software for a dedicated target. The node distinguishes the actual TUBiX20 node as they have been presented earlier, such as EPS, COM or OBC. The second parameter, revision, selects a specific release from this node's PCB history, while the use case distinguishes the model version (e. g. EQM, FM) and the mission. In the vertical direction, the different layers of software abstraction are depicted, beginning from the operating system as the lowest level at the bottom. Depending on the level of software abstraction, the specialization of the target takes place in different stages. While the interfaces for the global building block (BB) library, global drivers and the operating systems are unique throughout the system, there is a different set of driver interfaces for the node-specific hardware and also a different building block library for each

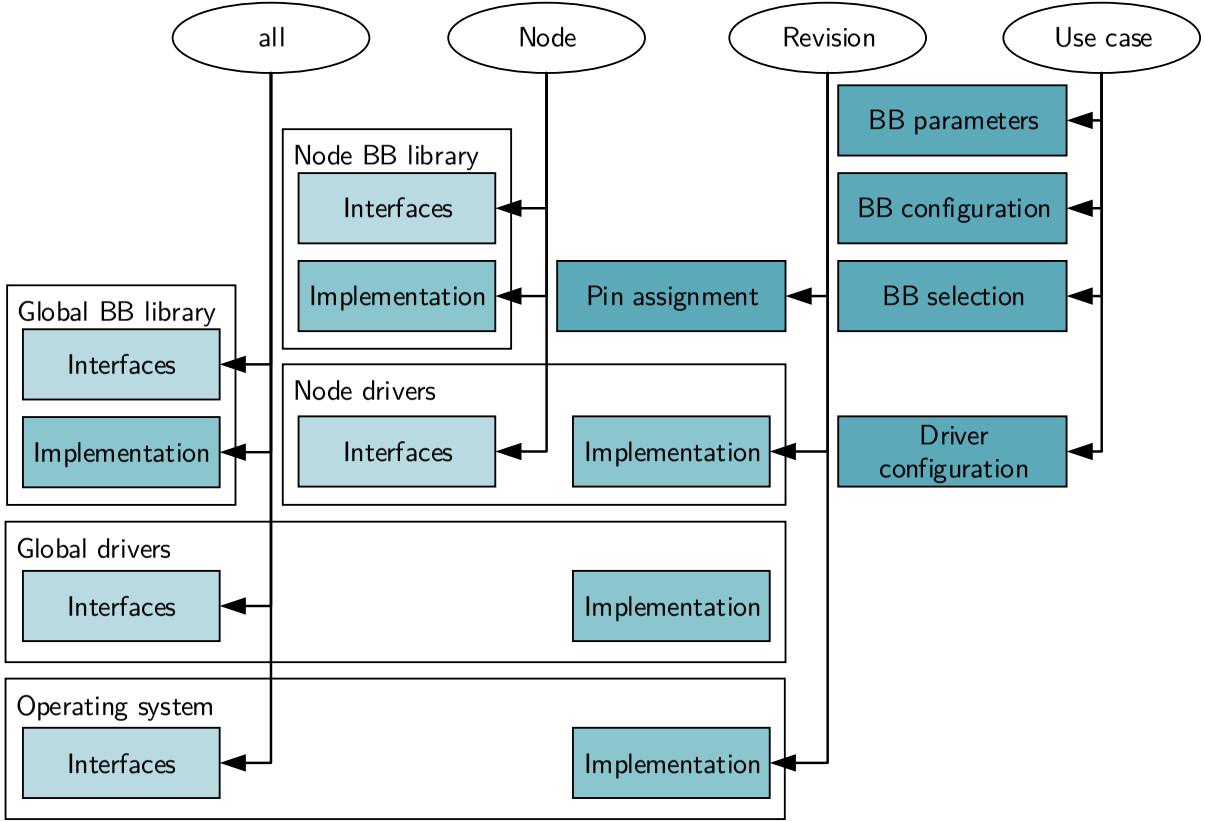

Figure 2.4: Software Build Configuration Management

node, offering implementations for the tasks assigned to that node. Unlike the building block implementation, which does not depend on the hardware target, the implementation for both global and node-specific drivers differs, since different components may be used. The same applies for the operating system. Here, the selected target implies its hardware-dependent port, e. g. for a certain controller or the Linux emulation. Finally, the building blocks actually compiled into the software image varies for the use case: one mission may not incorporate a device previously used or a new one is added. If the same building block is re-used, it may still offer a mission-specific configuration, e. g. operational modes. Moreover, different missions or models require different parameters for device calibration, which is therefore also specified by the use case.

2.3.4 Missions based on TUBiX20

There are currently two missions based on the TUBiX20 nanosatellite platform in progress: TechnoSat and TUBIN. To give the reader a first impression, CAD renderings of both satellites are shown in Figure 2.5 and Figure 2.6, respectively. Despite the fact that both are based on the same nanosatellite platform and also that their outer appearance is very similar, a closer look into these missions will show that their requirements regarding attitude determination and control are very different. Therefore, these two missions serve as examples for the tailoring of the flexible ADCS concept elaborated in this thesis.

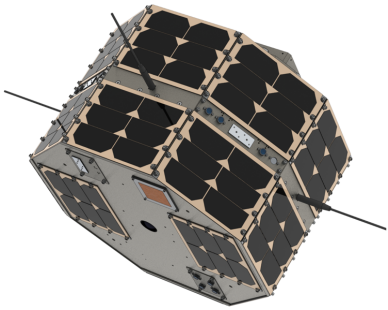

Figure 2.5: TechnoSat [51]
Image credit: Merlin Barschke

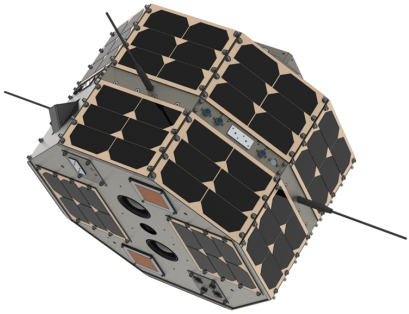

Figure 2.6: TUBIN [54]
Image credit: Merlin Barschke

Following the generic hardware and software design presented earlier, the concept allows for its adaption to the diverging requirements. While the flexible design approach will be discussed in detail in Chapter 3, the realization of the platform configurations for TechnoSat and TUBIN are presented in Chapter 5. The following subsections give a short overview of these two missions. Subsequently, the individual requirements regarding attitude determination and control are compared.

TechnoSat

TechnoSat is a mission for in-orbit demonstration of novel nanosatellite technology [55]. Launched on July 14th, 2017, into a 600 km SSO, the 20 kg satellite carries seven different payloads:

- STELLA, a star tracker for nanosatellites [56]
- the particle detector SOLID [57]
- a CMOS camera
- fourteen laser ranging retro reflectors [58]
- the S-band transmitter HISPICO [59]
- a reaction wheels system with four wheels
- the fluid dynamic actuator FDA [60].

TechnoSat's second mission objective is the development and verification of the TUBiX20 nanosatellite platform. The main parameters of the TechnoSat mission are listed in Table 2.1. Section 5.1 discusses the requirements and realization of the TechnoSat ADCS in detail.

While the S-band transmitter payload may also be used for payload data downlink, telecommand and telemetry transmissions are predominantly performed via UHF. Three of these payloads are verified in-orbit to be used within the TUBIN mission: the S-band transmitter, the star tracker STELLA and the reaction wheel system.

Table 2.1: Main Parameters of the TechnoSat Mission (adapted from [55])

Orbit	600 km SSO
Launch date	July 14th, 2017
Launcher	Soyuz with Fregat upper stage
Design lifetime	1 year
Spacecraft mass	20 kg
Spacecraft volume	$465 \times 465 \times 305$ mm
TM/TC link	Four channel UHF system
Attitude sensors	IC magnetometers MEMS gyroscopes Sun sensors Fiber optic rate sensors
Attitude actuators	Torque rods

TUBIN

The TU Berlin Infrared Nanosatellite (TUBIN) is the second mission based on TUBiX20. Its primary mission objective is the demonstration of a commercial infrared microbolometer for wildfire remote sensing on a nanosatellite [54]. TUBIN's mass is comparable to TechnoSat (20 kg) and it is planned for launch one year after its predecessor into a similar orbit. The payload consists of two infrared microbolometer cameras and one camera which is sensitive in the visible range of the light spectrum. Payload data downlink is facilitated via S-band.

To operate the payloads, TUBIN requests high-precision attitude determination and control within arc minutes accuracy. Hence, "the TUBIN mission will demonstrate the platform's ability to support a challenging Earth observation mission" [54]. The specific requirements regarding attitude control as well as implementation details are presented in Section 5.3. For a brief overview, Table 2.2 lists the main parameters of the TUBIN mission.

Table 2.2: Main Parameters of the TUBIN Mission (adapted from [54])

Orbit	600 km SSO (TBC)
Launch date	H2 2018 (TBC)
Launcher	Soyuz with Fregat upper stage
Design lifetime	1 year
Spacecraft mass	20 kg
Spacecraft volume	$465 \times 465 \times 305$ mm
TM/TC link	Four channel UHF system
Position sensors	GPS receiver Passive laser retro-reflectors
Attitude sensors	IC magnetometers MEMS gyroscopes Sun sensors Fiber optic rate sensors Star trackers
Attitude actuators	Reaction wheels Torque rods

Comparison of Requirements for TechnoSat and TUBIN

The two TUBiX20 missions TechnoSat and TUBIN are a good example for the evolution of mission objectives described in Section 2.1: while both are based on the same platform, TechnoSat's primary objective is technology demonstration and TUBIN demonstrates Earth observation on a nanosatellite. The evolution towards the support of complex mission scenarios is explicitly formulated in the secondary mission objectives. Here, TechnoSat targets the development and verification of the TUBiX20 nanosatellite platform, which is continued within the TUBIN mission by enhancing the ADCS to achieve high-precision. Table 2.3 shows a comparison of the ADCS requirements, their detailed derivation is given in Appendix C.

Table 2.3: TechnoSat vs. TUBIN ADCS Requirements

	TechnoSat	TUBIN
Mission objective	Technology demonstration	Earth observation
Pointing knowledge	N/A[1]	11.5 arcmin
Pointing accuracy	27.9°	200.3 arcmin
Pointing stability	N/A[1]	27.9 arcmin/s
Angular rate knowledge	0.1 °/s	N/A[1]
Max. angular rate	0.3 °/s	N/A[1]

Notes:

[1] no specific requirement formulated

Due to the more demanding payloads and mission objectives, the requirements regarding pointing knowledge and accuracy increase from TechnoSat to TUBIN. On the other hand, requirements formulated by particular TechnoSat payloads do not persist for TUBIN. The research for flexible attitude determination and control carried out in this thesis aims to respond to both. It will be shown that the concept supports tailoring the platform to individual mission scenarios by the extension, removal and reconfiguration of functionality.

Since TechnoSat is the precursor mission for TUBIN and does not only verify the platform in general, but also carries the S-band transmitter, a star tracker and the reaction wheels intended to be used for TUBIN, the reconfiguration may even be performed in-orbit: once the experiments with these three TechnoSat payloads have been carried out successfully, they may be used for platform services. Therefore, the ADCS takes the integration of the star tracker and the reaction wheels into account from an early stage. Nevertheless, the concept does not rely on these components to meet all requirements for the TechnoSat mission. By following this approach, the ADCS concept and performance for TUBIN may already be demonstrated within TechnoSat to a large extent.

2.4 Component-Based Software Engineering

An attitude determination and control system is a typical example for an embedded system: software running on a specialized microcontroller to perform a dedicated function, incorporated in a system of electro-mechanical devices. Due to the space environment and the high degree of autonomy, the software needs to fulfill stringent requirements regarding quality and reliability. On the other hand, an ADCS is usually a complex distributed system, since its sensors and actuators are usually embedded systems themselves. Moreover, the system has a heterogeneous character, as the components are often from different suppliers and may use different hardware- and software interfaces.

The technological evolution of satellite components reviewed in Section 1.1 has increased their performance and hence opened up new possibilities, but has also led to higher requirements and more complexity to handle. While earlier the software was limited by the computer's processing and memory capabilities, modern microcontrollers have overcome these limitations and the challenge is now to master the increasing complexity while still keeping the design maintainable, reusable and reliable. The significance of software development has progressed from a subordinate task to a domain with equal importance to any other work package such as mechanical structure design or electronics development.

To enable reliable and well-structured software, the design process and architectural design are crucial. Following, an introduction into component-based

software engineering (CBSE) is given. CBSE is a sub-discipline of software engineering which is applied as an engineering science and aims at the successful adaption, integration, and maintenance of complex, heterogeneous, and distributed software systems [61]. The basic idea is to develop the software as pre-produced parts (components) with the ability to reuse those parts in another context or application. Since the components are independent from each other, they allow for easy maintenance and customization to produce new functions or features.

A component is an encapsulated entity communicating only via dedicated interfaces. The interface and the implementation of the functionality are separated. To describe components graphically, the unified modeling language (UML) [62] defines special diagrams called component diagrams (cf. Figure 2.7).

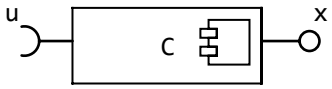

Figure 2.7: Component Diagram Example (UML)

Here, component C is marked with the component symbol in the upper right corner. While all implementation details are hidden, the interfaces are shown: the component has one required interface (input), u, and one provided interface (output), x. Since there is no direct access to any implementation details, a component may be treated as a *black box*. Based on the interfaces, components may be composed to form larger components or consist of several inner components, as may be seen in Figure 2.8. In this second example, component C is composed of components A and B. All required and provided interfaces of C are simply delegated to or from the interfaces of A and B, respectively.

The components' self-contained nature allows the development and testing of components individually, but also to swap competing implementations of the same idea and test without modifying the structure of a composition. The latter facilitates a plug-and-play design approach and the reuse of components in different contexts as well as easy updates, e. g. a new version of a component. A collection of interfaces may be extended to an application framework which

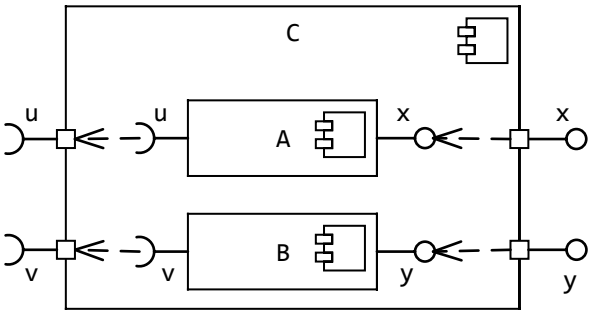

Figure 2.8: Component Composition Example (UML)

serves as an infrastructure to manage the control flow of a composition of components to form a complete application.

3 Design and Verification of a Flexible Attitude Control System

This is the main chapter of this work. It investigates a method to develop and verify a flexible attitude control system for three-axis-stabilized nanosatellites. The chapter picks up on the design considerations discussed in the previous chapter and argues how the elaborated concept is flexible in terms of the criteria identified. The approach follows the TUBiX20 strategy of partitioning the system into self-contained functional modules and hence facilitates the abstraction of functionality from realization. While this chapter discusses the ADCS architecture and development process, the subsequent chapter provides selected attitude determination and control techniques in detail to give examples for the implementation of the functional modules. Thereafter, insights into the practical realization of the concept for the first two nanosatellites based on TUBiX20 follow.

3.1 Distributed Attitude Control System

As pointed out in Chapter 2, a flexible ADCS architecture must allow for addition, removal and replacement of components due to different reasons. Since ever new products are released on the agile market of space components, rapid technology updates are mandatory to keep the platform up-to-date. While one satellite mission might require high-precision attitude determination using one or more star trackers, attitude knowledge within several degrees might be sufficient for another. The former may have a greater budget but demands the integration of newly developed components whose interfaces are not yet supported by the ADCS platform, while the latter has a stringent financial budget and a reduced project life cycle. For an ADCS architecture which is compliant with both, a high degree of modularity and abstraction is necessary. In the following section, an approach for a distributed ADCS is presented which allows for easy adaptions to a wide range of platform

configurations based on the TUBiX20 systems architecture [49], which was introduced in Section 2.3. Since the platform design comprises a network of computational nodes with standardized interfaces for hard- and software, it supports a modular and distributed ADCS design where all sensors and actuators are also considered to be network nodes. [63]. The realization of both hardware and software domain are presented in the following subsections.

3.1.1 Distributed ADCS Hardware

Figure 3.1 shows the elements of the distributed ADCS from the hardware perspective. Following the TUBiX20 platform levels introduced in Figure 2.2, the sensors and actuators are generally represented by the term *component*. They are integrated via computational nodes, which in turn interconnect via the central power (red) and data (green) bus depicted in the middle of the figure. The nodes decouple the individual components' interfaces from the systems architecture and hence provide the necessary abstraction to integrate arbitrary new components without interference to existing parts. Moreover, components may also be removed cleanly without touching the remaining network elements. While the TUBiX20 platform uses CAN redundantly as central data bus, the interface nodes connect one or more components via e. g. I^2C, SPI or UART to the network. However, if a component already complies with the central power and data bus, it may also be connected directly. Besides the interface translation, the nodes' microcontrollers also offer the capability of pre-processing sensor measurements or actuator settings and hence provide a basis for the software abstraction described in the next section.

3.1.2 Distributed ADCS Software

Analog to the distributed hardware, the software of the ADCS is also partitioned into independent modules. As described in Section 2.3, the operating system and its middleware provide a framework to run software building blocks autonomously. Consequently, the hardware-dependent software for each device is encapsulated in a single building block called *device manager*, as can be seen in Figure 3.2.

Apart from translating messages from the middleware communication topic to a device's individual data bus and communication protocol, the nodes also

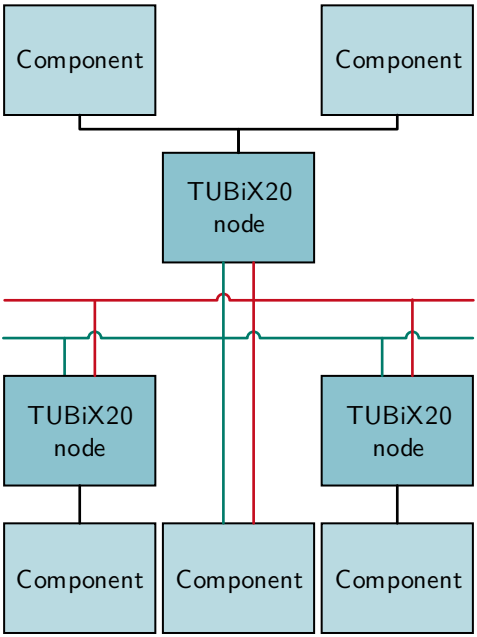

Figure 3.1: Distributed ADCS Hardware Perspective

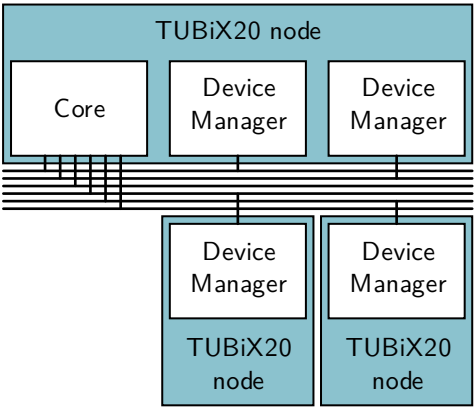

Figure 3.2: Distributed ADCS Software Perspective

perform all low-level data processing required to operate the device such as filtering high-frequent sensor measurements or calibration data management. In this manner, processor load and software image code size are shifted from the main processing node, which in turn gains more flexibility to integrate additional functionality. Resulting from the interface abstraction, devices may be added or removed without influencing the rest of the network, since removing a device implies also removing the TUBiX20 interface node and hence the software building-block related to the device which the node is hosting, as can be seen in Figure 3.1 and Figure 3.2, respectively.

When compared to a monolithic architecture, the distributed ADCS uses more microcontrollers than the required minimum. The additional nodes take up space in the electronics box and increase the power consumption, but also the manufacturing costs and development effort for the printed circuit boards. In terms of software, partitioning the code into separate applications for the different devices comes with a certain overhead, since the interfaces for TM/TC have to be implemented several times and the scheduling of the different software threads becomes more complex. However, since the overall processing load does not increase significantly and is even shared between the different nodes, the power consumption may be reduced by putting controllers into sleep mode when idle. The increased manufacturing costs and development effort in the hardware domain and the implementation overhead in the software domain are compensated during the course of several missions. While a monolithic architecture may require extensive modifications from one platform configuration to another, the distributed approach allows complete reuse of verified nodes and applications, respectively, however in a different composition. Nevertheless, the additional space required and the yet slightly increased power consumption on the one hand, and the increased code size and memory usage on the other hand make this solution predominantly useful for nanosatellites.

Since the operating system's middleware facilitates the communication between all building blocks, the core application which hosts the attitude determination and control algorithms is completely hardware-independent. It does not need to know on which node the device applications are running and vice versa. All building blocks of the ADCS software network are synchronized via a hardware pulse per second signal provided by the TUBiX20 hardware interface. However, the modularity must be supported by the attitude determination and control algorithms, as the composition of available sensors and actuators

clearly affects the control software. The modular approach for flexible attitude determination and control to respond to this necessity is presented in the next section. It shall be pointed out that this design matches with the TUBiX20 architecture, but also benefits different ADCS hardware setups, since it allows time-efficient and straightforward updates of the control software individually.

3.2 Modularized State Quantity Determination

To determine and control a spacecraft's attitude, different quantities need to be considered. The set of all relevant quantities given at a reference point in time is in this context called the state of the attitude control system. All state quantities considered for TUBiX20 are listed in Table 3.1.

Based on these state quantities, the concept investigated defines unified interfaces to encapsulate the implementation of an algorithm inside a module. These interface specifications are called *state quantity provider interfaces* and further include units and coordinate systems. Hence they contain full information about the state quantity provided. Figure 3.3 shows an example for a quantity provider interface. Here, the module called *M* implements an algorithm which provides the angular rate, $\vec{\omega}$. The module requires one interface as input: the attitude, q. Both interfaces are specified regarding their symbol, physical unit and coordinate system according to Table 3.1. The definition of the modules and interfaces is based on component-based software engineering, which was introduced in Section 2.4, and all diagrams follow the unified modeling language (UML).

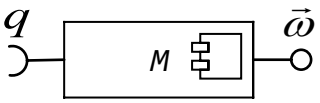

Figure 3.3: Quantity Provider Interface Example (UML)

The abstraction from the module's functionality has the following benefits. Firstly, the user does not need to know any implementation details, but only the interface definition and hence the module may be treated as a *black box*. Secondly, since units and coordinate systems for modules implementing the

Table 3.1: Quantities for State Determination

Quantity	Unit	Symbol	Coordinate system
Attitude quanterion	N/A	q	TOD $\leftarrow$ SAT
Angular rate	$^\circ$/s	$\vec{\omega}$	SAT
Angular acceleration	$^\circ/s^2$	$\dot{\vec{\omega}}$	SAT
Geomagnetic field (body-fixed)	nT	$\vec{b}$	SAT
Sun vector (body-fixed)	N/A	$\vec{s}$	SAT
Geomagnetic field (inertial)	nT	$\vec{B}$	TOD
Sun position (inertial)	m	$\vec{S}$	TOD
Position	m	$\vec{r}$	TOD
Velocity	m/s	$\vec{v}$	TOD
Torque	N m	$\vec{\tau}$	SAT
Magnetic dipole	A m^2	$\vec{m}$	SAT
Occultation [1]	N/A		N/A
Temperature	°C	T	N/A
Time (modified julian date)	d	t	N/A

Notes:

[1] Occultation by the Earth in the interval [0 1] (0 = Sunlight, 1 = Eclipse)

same provider interface are equal, additional conversions are unnecessary and mistakes are prevented. Thirdly, the modules can be easily combined, extended, added or removed. These benefits shall be illustrated with the following examples, which are selected from different layers of the distributed TUBiX20 software, from sensor calibration to data fusion to estimation algorithm alternatives. Throughout the examples, the concept is related to the flexibility criteria introduced in Section 2.2.

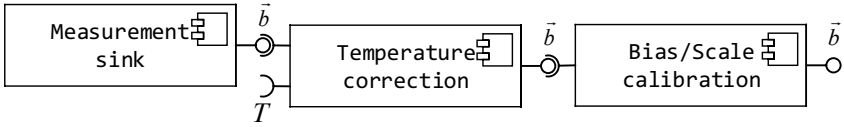

Figure 3.4: Staged Magnetic Field Sensor Calibration (UML)

The first example is taken from a device manager (cf. Section 3.1.2), which runs on a node connected to a magnetic field sensor. It shows the combination of different calibration techniques, presented in Figure 3.4. Here, three different modules provide the quantity's data, each implementing the same interface named $\vec{b}$ according to the list of state quantities from Table 3.1: the sink is simply a buffer where the hardware driver puts the measurement data. Subsequently, the temperature drift of this raw data is corrected in a first filter module, which requires a separate interface for the temperature, T. The bias and scale calibration is realized as an independent module. Here, the abstraction from functionality to implementation is clearly visible and the seamless adaption of the calibration strategy becomes obvious: since all modules implement the same interface, modules may be removed from the chain or inserted simply by re-connecting the inputs and outputs of other modules. Such an adaption might be useful, if the magnetic field sensor hardware is replaced by another type or model, which requires different calibration techniques. For example, some magnetometer ICs have an internal temperature compensation, which allows the removal of this module from the software to reduce complexity. Moreover, the calibration may be extended by additional algorithms to consider the time-varying bias of the magnetometer in-orbit (e. g. due to electronics on-board a spacecraft, as examined in [64], [65]), or to allow the simultaneous operation of the magnetic torquers to increase their duty-cycle. As can be seen from the interface, the source of

the data (sensor type, calibration algorithm, etc.) is independent from the way it is provided. Consequently, sensor fusion including different devices is straightforward, which will be demonstrated in the next example.

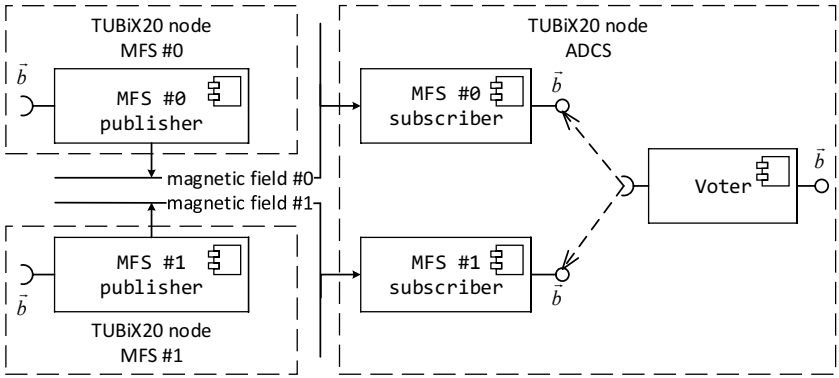

Figure 3.5: Abstraction From a Diverse Hardware Setup (UML)

The second example further illustrates the abstraction from a sensor's individual output format on the lowest level possible. Here, we assume that two magnetic field sensors (MFSs) from different manufacturers are used. One could be a small, cost- and energy-efficient magnetometer IC used for a basic ADCS configuration. The second may be a more accurate, but also more expensive, larger and heavier device such as a fluxgate magnetometer, which is added for a configuration demanding high performance. For redundancy and FDIR purposes, the first sensor may also be part of this configuration. Following the architecture of a distributed ADCS which was presented in Section 3.1, both sensors are connected to the network with generic TUBiX20 interface nodes. Hence they may be added or removed independently from each other. Figure 3.5 shows how the measurements are published from the sensors' device managers via the middleware to the ADCS core application for state determination on a third node. Here, the network messages are received by subscribers which then provide the data for the subsequent voter to perform the sensor data fusion, once again implementing the same interface.

Regarding the flexibility criteria presented in Section 2.2, the design benefits technology updates and scalability, since only minimum modifications are necessary here if components are added, removed or replaced. Different voting

strategies may be implemented according to the sensors' individual accuracy and availability. Examples for such strategies are priority voting, weighted average or arithmetic average. Moreover, cross-checks may be performed for FDIR (cf. Section 3.3.5). If the sensor setup is changed in a different mission, the voter is changed accordingly. However, for all subsequent use of the quantity magnetic field, no modifications are required. It shall be noted here, that it is irrelevant for the voter module where the different inputs actually come from, even if they are measured by different sensor types on different hardware nodes. The same applies to the beginning of the data flow: the MFS publishers may connect to any of the modules from the first example. Therefore, the first two examples together show the complete data flow from the sink receiving the sensor raw data to the final estimate for this quantity within the core application as a chain of consecutive modules. The implementation of the unified interface allows the extension or simplification of the chain via inserting or removing individual modules. While the approach may at first seem complex and imply a great implementation and processing overhead, implementing the identical interface in fact reduces the number of coordinate transformations and unit conversions. The partitioning of the functionality improves the code quality, since it benefits maintenance and individual testing.

The third example expands on the relation of different state quantities, in this case the estimation of the attitude from vector observations. By comparing the body-fixed measurements for the geomagnetic field, $\vec{b}$, and the direction vector to the Sun, $\vec{s}$ with their inertial reference values, $\vec{B}$ and $\vec{S}$, the orientation of the spacecraft is obtained. All inputs for the body-fixed and inertial vector inputs are provided by separate modules, where for each a specific method was chosen to give a concrete example. Firstly, the input for the body-fixed magnetic field is connected to the voter module from the previous example to demonstrate the continuity of the approach (cf. Figure 3.5). The Sun vector is provided by a subscriber, which receives the measurements via the middleware. The models for the inertial geomagnetic field and the inertial Sun position are the IGRF and the SLPC, which are both described in the glossary. There are different methods for the attitude estimation, which will be discussed later in Section 4.4.6. Here, the QUEST [66] method is selected. However, the implementation is treated as a *black box* and the focus is on the interfaces of its in- and outputs. It shall be clarified that all of the aforementioned modules may be exchanged or extended independently from each other. However, these

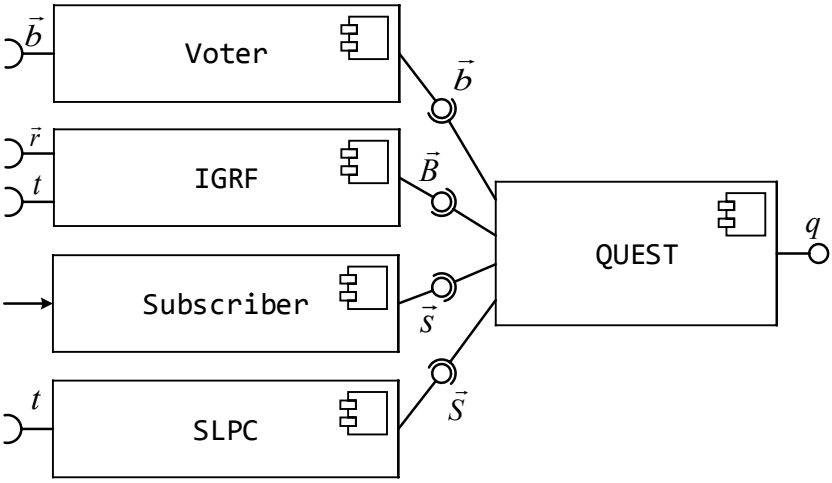

Figure 3.6: Deriving State Quantities (UML)

changes will affect the attitude estimation in terms of accuracy or reliability, since the module's inputs change.

Obviously, the method for the attitude estimation may be exchanged itself by any other method using the same interfaces, e. g. FOAM or ESOQ (cf. Section 4.4.6). Moreover, a different approach with extra inputs and outputs may be used. As an example, the Kalman filter presented in Section 4.4.7 additionally estimates the bias of angular rate measurements. If one of the magnetic field sensor nodes is removed to decrease complexity or cost, the according subscriber (cf. Figure 3.5) and the voter become obsolete and the magnetic field input for the attitude estimation is simply connected to the remaining subscriber. To increase the accuracy and reliability of the vector measurements, a Kalman filter may be inserted, as described in Section 4.4.3 and Section 4.4.4, respectively. The models for the inertial reference values are replaceable as well, but also their input sources impact the quality of the attitude estimation: the time and spacecraft position, t and $\vec{r}$, respectively, may originate from a GPS receiver or a less accurate orbit propagation model such as SGP4.

The preceding examples have shown the breakdown of the complex data processing from sensor measurements to data fusion including several state quantities into self-contained modules, which may even span different hardware nodes of the distributed ADCS. While only the state determination has been addressed so far to introduce the application of modularization in the context of the ADCS design, the same principle transfers to the state control part, which will be presented in detail in Section 3.3.4. The examples already suggest that there are numerous different methods to determine the different state quantities. Chapter 4 will provide the theoretical background for the algorithms which have been implemented as modules within TUBiX20. The next logical step is to gather these modules in a software library. For the individual realizations of different TUBiX20 missions, a selection of modules is then chosen and assembled.

When evaluated in terms of the flexibility criteria from Section 2.2, the partitioning of the ADCS functionality into self-contained modules with unified interfaces offers several benefits. The first example (Figure 3.4) is situated within a device manager and shows that technology upgrades are simplified due to the early abstraction from the devices specific properties such as communications protocol, operational modes, etc., since already the device drivers implement the same interface. Moreover, the calibration procedure may be adapted to the sensor's characteristics by adding or removing modules. The second example targets scalability (Figure 3.5): the distributed ADCS from Section 3.1 is continued to the high-level software, where the addition, removal or exchange of devices implies only minimum modification on the rest of the (eventually flight-proven) code: connecting the inputs of the voter. On this level, it is irrelevant for the state determination which network nodes the sensor is connected to. In the third example (Figure 3.6), this scalability is transferred to the recombination of state estimation algorithms. Here, upgrading the performance by introducing new estimation techniques or increasing the reliability via multiple functional redundancies is straightforward. Changing voter strategies or reconnecting the modules' inputs and outputs at runtime even allows reconfiguration in-orbit without software upload. The example highlights, that the availability of the sensors does not affect the control flow of the software, but only the result of the voting. If – as a counterexample – an ADCS software is implemented as a monolithic state determination block which merges all sensor data within a complicated structure full of forks and cross-references, it is very difficult to exchange single elements and reuse

is limited. Here, the dynamic availability of the used sensors throughout operations, depending on failure or loss, results in many case distinctions within the control flow. Depending on which sensor provides reliable data, different fusion or selection algorithms are executed, which makes it especially hard to comprehend during verification on ground or error analysis in-orbit, as discussed by the author in [63].

3.3 State Determination and Control Core Framework

The two previous sections have presented the distribution of the ADCS software to different hardware nodes and the modularization of the ADCS functionality into a library of self-contained modules. The latter shows how these modules may be used on all layers of the software, from a device manager, i. e. the part of the software related to a hardware component, to the high-level state determination and control. In this section, the framework for the ADCS core application is presented.

3.3.1 Library, Framework and Assembly

The ADCS core application combines the library of functional modules with a framework to define the control flow. For clarification, the terms library and framework are set into relation at first. A library is a collection of implementations following defined interfaces and is intended for code re-use. The library elements do not have any knowledge in which context they are used and the library does not define how and when objects or variables are instantiated or functions are invoked. On the contrary, a framework defines the control flow of an application independent from the implementation of its functionality. As opposed to procedural programming, where the user defines the control flow, the framework instantiates objects and invokes their functions. This paradigm to shift the control flow is called *inversion of control*. For TUBiX20, such a framework is designed to collect all information for state determination, invoke state control and further communicate with all devices and other subsystems, including telecommand and telemetry, via the middleware.

Both library and framework do not depend on the current mission the ADCS is adapted to: the library provides different options to perform certain tasks such as attitude estimation or control torque distribution. The framework triggers the execution of the modules without knowing the composition of modules, i. e. how attitude control is actually performed. Therefore, both can be re-used throughout all TUBiX20 missions. The framework remains the same and the library is continuously extended by new modules. To adapt the ADCS to an actual mission, a connection of library and framework needs to define which modules are included and how they are linked. This part is called the *core assembly* and is simply a list of module instantiations and interface connections.

The composition of the state determination and control framework is shown in Figure 3.7. At first, the device management applications trigger the sensor sampling and publish the measurements on the middleware topics. Subscribed within the core application, the data is then processed to determine the complete set of state quantities, which in turn forms the input for the state control. After the settings for the actuators have been calculated by the controller modules, they are transferred to the device managers of the actuators via the middleware and the control loop is closed. As shown in Figure 3.7, state determination, state control and the device managers each have an assembly of library modules (depicted in red) which is composed according to an individual mission, while the core framework (depicted in black) remains fixed.

Since the module assembly is mission-specific, but the framework is fixed, a centralized connection point is required, where the framework can access the different state quantities, independently from which modules provide them. As shown in the center of Figure 3.7, the inputs of the state control modules are all connected to the same interface, and in turn all modules' outputs for state determination are connected to a counterpart on the other side. This interface is called a *state provider*. The assembly of state provider modules, which incorporate the full set of all state quantities, is straightforward, as only matching interfaces need to be connected. An example for the interface connection is given in Figure 3.8. This state interface in based on the *facade* software design pattern by Gamma, Helm, Johnson, *et al.* [67] and bundles all state quantities in one entity. For simplification, some quantities are omitted in this figure. As a consequence, only the state provider instantiation must be known to the framework, and request for the state quantities are delegated to

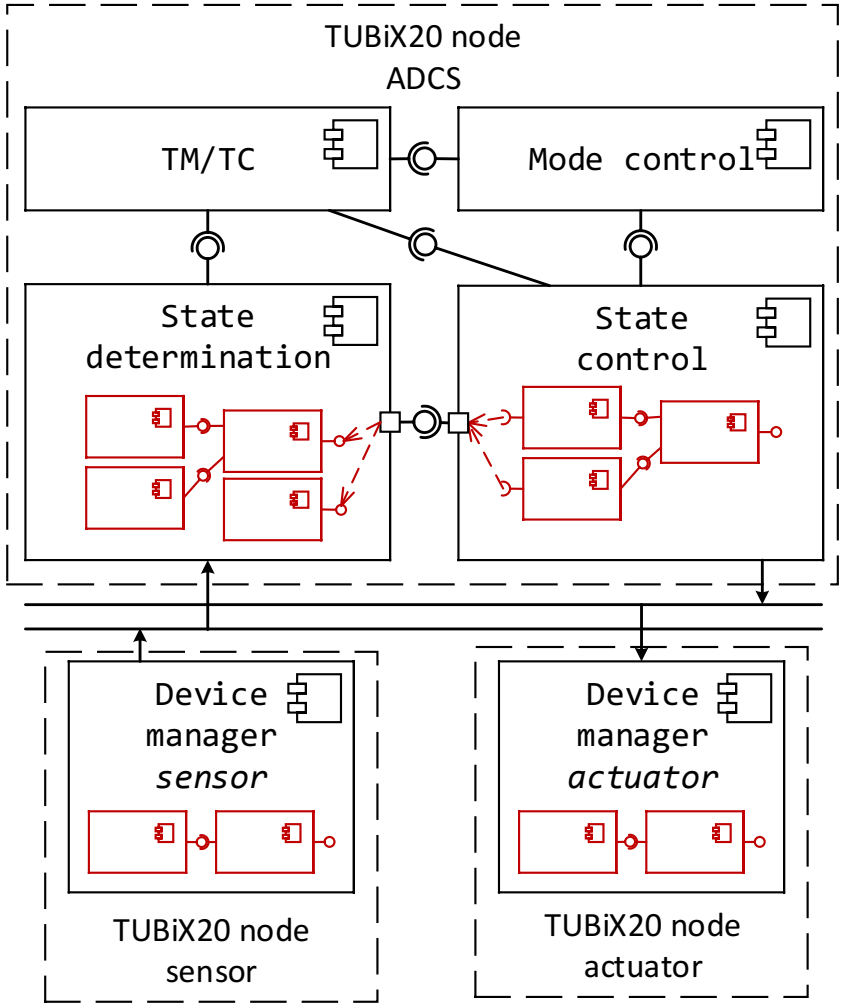

Figure 3.7: State Determination and Control Framework (UML)

the connected modules. Apart from the connection to the core framework, the concept of a centralized state interface brings further benefits: the partitioning of state determination into estimation and prediction and the benchmark of several state estimation techniques in parallel. Before these are discussed in the following two sections, the separation into library, framework and assembly is evaluated in terms of the flexibility criteria from Section 2.2.

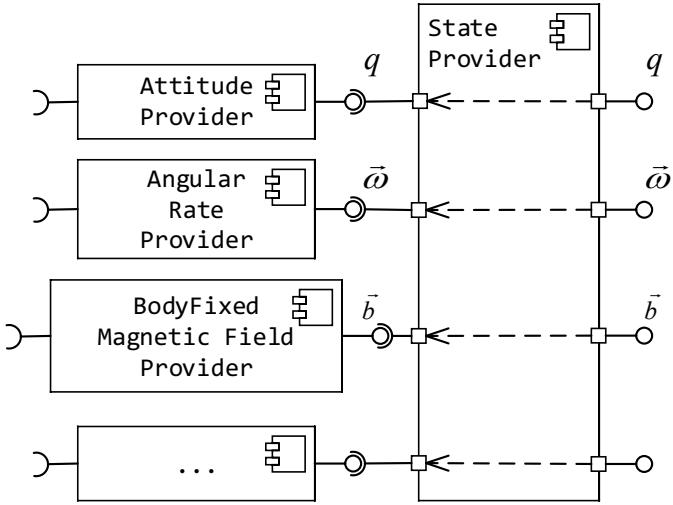

Figure 3.8: State Estimation Facade (UML)

On this higher abstraction level, the aspects of technology updates and scalability addressed in Section 3.2 become more obvious: adding, removing or exchanging devices or modules only affects the core assembly, but not the library and the framework. Hence the modifications for adaptions are kept at a minimum. Moreover, the TUBiX20 software build configuration management described in Section 2.3.3 allows software development and maintenance for multiple missions at the same time: to configure the ADCS for a mission, only a specific core assembly needs to be defined accordingly, which may then be selected as one use case of the build configuration. The further development of the module library and framework is carried out independently, which creates synergies between the different missions.

3.3.2 State Estimation and Prediction

In this section, a detailed overview on the timing reference shall put the terms state determination, state estimation and state prediction into relation. Since the attitude control system is a discrete time system with a fixed sample rate, t_s, the sensor measurements refer to the beginning of the control cycle, t_k. These measurements are processed to estimate all state quantities needed for attitude control and hence this estimation also refers to t_k. However, the control settings are applied for the following control cycle, t_{k+1}, and therefore the change of the state quantities during the control cycle time, t_s, would not be considered. To compensate this inaccuracy, the state quantities can be predicted for the end of the control cycle, as can be seen in Figure 3.9. Here, S_k denotes the state at time t_k, while S_{k+1} is the state at time t_{k+1}.

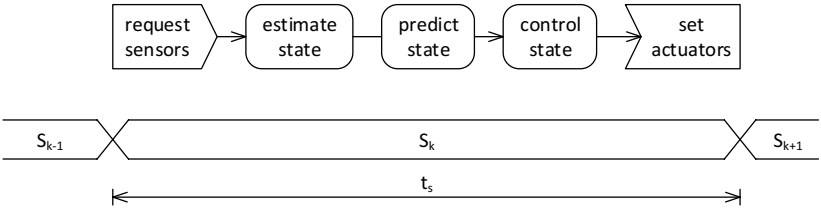

Figure 3.9: Control Cycle Time Sequence (UML)

The combination of state estimation and state prediction is called state determination. The realization of this separation in the core framework is shown in Figure 3.10. Based on the centralized state interface introduced earlier, state estimation and state prediction are realized independently from each other and may hence also be modified independently. Optionally, the state prediction may also be skipped by connecting the output of the state estimation directly to the output of the superordinate state determination module. This is possible in two ways: either the connection is re-routed at runtime via a telecommand, or the modules are not compiled into the software from the start, which benefits scalability of functionality and in-orbit reconfiguration opportunities. Since estimation and prediction are independent and the state quantities are passed via a single interface, the estimation is treated as a *black box* from the prediction's point of view: it does not need

any knowledge about the availability of the sensors or the control flow within the estimation module.

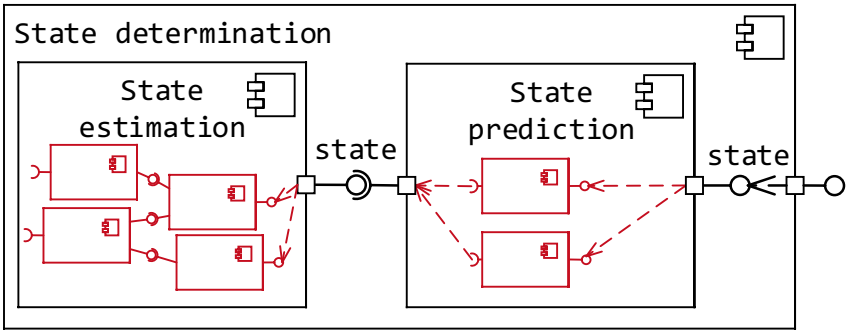

Figure 3.10: State Estimation and Prediction (UML)

Looking at the list from Table 3.1, it is obvious that not all state quantities can be predicted with the same formula or model. While the attitude is predicted by the kinematics equation, the body-fixed Sun vector and geomagnetic field vector prediction uses the dynamics equation. The applied torque depends on the actuators used and hence differs from one ADCS configuration to another. All quantities observed in an inertial reference frame cannot be predicted from estimated quantities. They may be determined from the same model as for the state estimation, e. g. SGP4 for the position and velocity or IGRF for the geomagnetic field, however for a different timestamp. Despite the different techniques used within the state prediction, there is only one module to predict each quantity.

3.3.3 Parallel State Estimation

The three examples from Section 3.2 have addressed different aspects of flexibility. Firstly, building a chain of modules implementing the same interface partitions the complex data processing into self-contained modules, which reduces the complexity and offers scalability. Secondly, the unified interfaces allow an abstraction from the hardware, and hence removing, adding or exchanging sensor or actuators implies only minor adaptions in the software.

Thirdly, state quantities may also be derived from each other and yet the module assembly is well structured and adaptable. This section will continue this last aspect and further investigate on the estimation of the complete set of state quantities from Figure 3.8 when different methods are applied at the same time to provide functional redundancies which are required due to the dynamic availability of the sensor measurements.

As implied previously, there are usually multiple ways to determine a state quantity and a wide range of different algorithms or alternative versions of the same method may be implemented: it may be measured directly by one or more sensors, estimated by a model or derived from other state quantities. Furthermore, different algorithms for filtering and sensor fusion may be used. Consequently, due to the large number of possible solutions to estimate a single state quantity, a large variety of their combinations is possible to determine the state as a whole.

Since the main objective of the state estimation is to determine the state quantities as accurately as possible, an ADCS typically uses all sensors which are fully operational. However, hardware failures as well as temporary or complete loss of a component may reduce the choices of available sensor data. There are many causes for such an unavailability which will not all be discussed here, but the following list of examples illustrates their variety:

- a star tracker is blinded by sunlight or straylight from Earth and not available for several minutes
- a component with high power consumption, e. g. a fiber optic rate sensor, is switched off to save power
- a Sun sensor's measurements are temporarily interfered by albedo effects or simply unavailable during eclipse
- a magnetic field sensor's measurements are temporarily interfered by (dynamic) dipole effects due to electric components
- a total loss of a sensor occurs, e. g. due to radiation
- the communication is disturbed and measurement data of one or more sensors cannot be transmitted.

While a (temporary) loss of a component is noted immediately, interference of a sensor by disturbing quantities may not be detected easily. Here, fault detection, isolation and recovery (FDIR) methods help to identify errors, which is discussed later in Section 3.3.5.

Considering the dynamic availability of the ADCS' components due to the reasons described above and furthermore the large variety of different methods to determine a state quantity, a conventional state estimation strategy results in a complicated structure of dependencies and case distinctions which is difficult to implement and especially to test: depending on the availability of sensor data, the control flow at runtime changes not only if certain modules are unavailable because of their invalid inputs, but also since the accuracy for the estimate of one quantity may influence the method chosen to estimate another quantity. This shall be clarified by the following example.

An Earth observation mission requires a highly accurate attitude determination and control and moreover permanent availability. To this end, the ADCS incorporates two star trackers, a fiber optic rate sensor system, a Sun sensor system, magnetic field sensors and MEMS gyroscopes. For simplification, only the state quantities attitude, angular rate, magnetic field vector and Sun vector are considered. Each quantity has at least one sensor which is particularly designed to measure it directly. Apart from filtering or averaging the direct measurements, the setup offers additional methods to derive state quantities. The following list provides some examples of these methods.

- Predict the relative attitude with high accuracy by the fiber optic rate sensor, if a start value based on the star tracker(s) is available. This is however subject to a drift and the accuracy requirements may be exceeded.

- Predict the relative attitude with low accuracy by the gyroscopes, if a start value is available. Here, the drift is increased due to the higher noise.

- Estimate the attitude from the vector observations by magnetic field sensors and Sun sensor system.

- Derive the angular rate from the temporal change of either Sun vector or magnetic field vector or both.

– If highly precise attitude information is available, transform the inertial Sun position into the body-fixed frame to obtain the Sun vector with high precision.

Given the dynamic availability of the sensors, multiple methods should be used for each state quantity to guarantee their continuous knowledge. Moreover, having information from (functionally) redundant sources enables cross-checks for FDIR. For traditional ADCS software concepts, the diverging accuracies of the individual estimates together with their dynamic availability complicates the selection of which methods to execute when, and the result is a complex structure of case distinctions where the control flow is hard to trace.

This thesis's concept for flexible attitude estimation follows a different approach. Here, the dependencies within a state estimation strategy mentioned above are resolved: the state estimation is no longer a complex and monolithic entity but an assembly of library modules which are executed concurrently. This means that the control flow is always the same and there are no case distinctions. Each method utilizes a unique set of sensors and algorithms. Hence, there are multiple candidates for each estimated state quantity, which are compared and analyzed to select the most accurate solution. In this manner, FDIR techniques can be applied easily, e. g. as a voter for the different estimation results. The estimation modules do not entail any case distinctions; if a sensor failure occurs, all modules which depend on this sensor will be marked as *not healthy*. Therefore, the individual modules have greatly reduced complexity and do not depend on one another. Both simplifies the implementation and verification.

The approach is clarified by translating the example above into the TUBiX20 architecture, which is shown in Figure 3.11. Here, four different methods for attitude determination are assembled and prioritized according to their accuracy. A subsequent voter requests the data from all four modules and internally selects the source with the highest priority and performs FDIR cross-checks. The modules differ not only in their accuracy, but also in their availability. A voting strategy according to priorities would imply taking the star tracker average (highest priority), as long as at least one unit provides measurements. If both are unavailable (either temporarily or permanently), the attitude is taken from a prediction module which has the fiber optic rate sensor measurements as inputs. If this becomes unavailable or the drift over time exceeds a certain limit, the module will mark its data as invalid

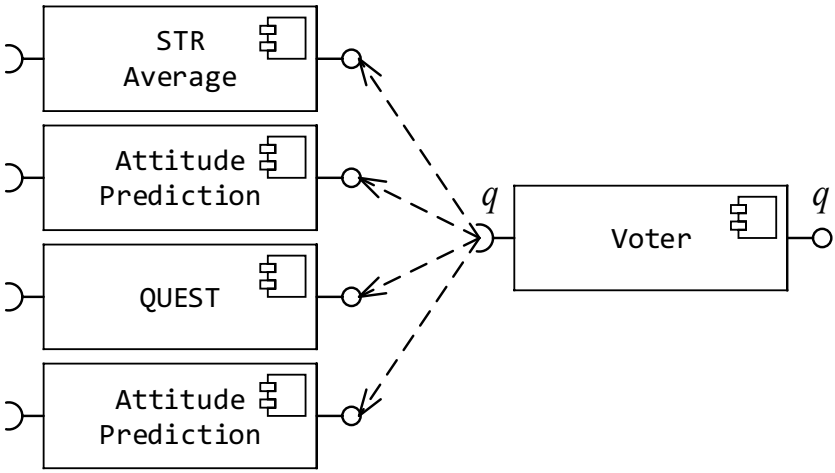

Figure 3.11: Concurrent State Quantity Estimation (UML)

and the QUEST method takes over. During an eclipse, the voter will fall back on the attitude prediction from angular rate measurements. If the fiber optic rate sensor system is switched off, the last available source is the highly drift-affected attitude prediction (lowest priority) via gyroscope measurements. Once again, the voter does not need any knowledge about its inputs apart from their interface, priorities for the selection and the number of inputs is variable. All four modules are invoked in each control cycle, so they concurrently provide up to four different attitude estimates.

Finally, the idea of concurrent state quantity estimation based on self-contained modules shown in Figure 3.11 is combined with the aggregation of all state quantities to a combined set, the state, which has been introduced in Figure 3.8. It was shown that there are various methods to estimate each quantity, but when realized as modules implementing unified interfaces, arbitrary selections of these modules may be bundled to compose a state. As a consequence, there are multiple independent state estimation concepts realized at the same time. Only one of these states is selected for further use in attitude control. However, all others provide a benchmark of different state estimation concepts at the same time, and hence one state estimation strategy may be compared to another during operations. Due to the straightforward extension of the module

assembly addressed in Section 3.3.1, new features may be introduced in the ADCS in-orbit. Since they run concurrently with the flight-proven concept in use, a new concept may be evaluated even during operations without interfering the performance. These benefits regarding in-orbit reconfiguration are discussed in more detail in Section 3.4.2.

3.3.4 State Control

Depending on the purpose of a mode, its inner structure is rather complex or simple. If active attitude control is not required permanently, the state control block may be empty for one mode. On the other end of the scale, a mode intended for payload operations may be rather complex if highly accurate three-axis-alignment is required. Due to functional abstraction, however, the structure of all modes may be generalized regarding the following characteristics to define a mode:

- Is active control requested?
- Which quantity or quantities shall be controlled?
- What is the objective of the control?
- Which control theory or control law shall be used?
- Which actuators shall be used?

The individual answers to these questions result from the ADCS's requirements and the basic concept decisions, for example which actuators are incorporated. The following list gives three examples for such mode definitions:

- "*Active* control of the spacecraft's *attitude* shall be performed to achieve *target pointing* using *state space control* and *reaction wheels*" for highly accurate pointing to the ground station for data downlink.
- "*Active* control of the spacecraft's *attitude* shall be performed to achieve *nadir pointing* using *a cross-product control law* and *magnetic torquers*" for energy-efficient pre-alignment between operational phases.
- "*Active* control of the spacecraft's *angular rate* shall be performed to achieve *detumbling* following a *cross-product control law* and using *magnetic torquers*" to reduce the initial spin rate after separation.

Setting the different characteristics in relation results in a graph structure, as shown in Figure 3.12. The five different layers of vertices correspond to the five characteristics listed above, while each path from the top vertex to one of the bottom vertices represents a control mode. Here, the paths for the three examples from above are depicted in red and more modes are added to clarify the idea. The approach is inspired by the software development concept of object-oriented design. The vertical direction (following the arrows) shows a specialization, while the other way is a generalization.

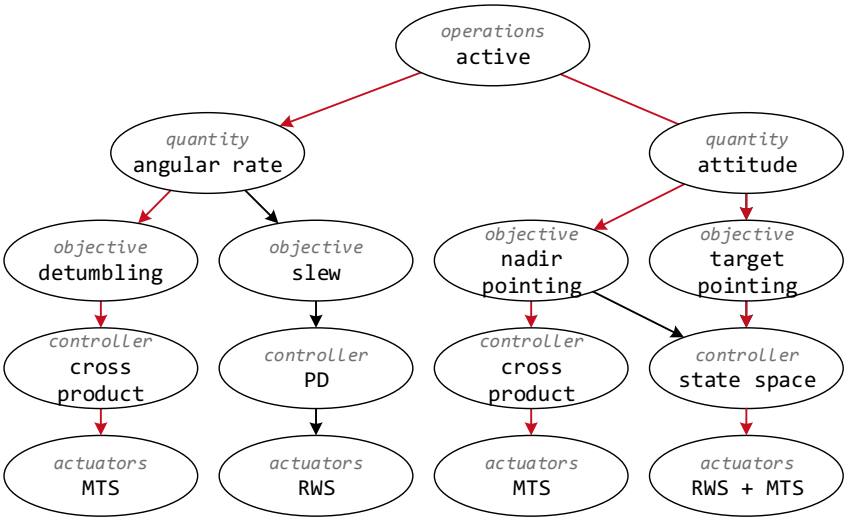

Figure 3.12: Functional Abstraction of Mode Definitions

The software architecture for a flexible realization of different state control modes continues the modular approach presented for state estimation and prediction. Therefore, the functionality shall be partitioned again into self-contained modules which are then assembled to form the state control modes and invoked automatically by the core framework as described in Section 3.3.1. Here, the characteristics defined earlier are the basis for the functional abstraction. As can be seen from the graph, some modes (i. e. paths) share common vertices. This indicates that the same module may be used for both modes. To clarify the approach, the same examples for control modes are

represented as component diagram in Figure 3.13. The pointing modes are partitioned into three different steps, which abstract the functionality from the implementation:

- control error determination
- control settings determination
- control settings distribution

While the determination of the control error differs for target pointing and nadir pointing, both may use the identical control algorithms and hardware set. On the other hand, both coarse and fine accuracy nadir pointing rely on the same module for error determination, but use different control laws and actuators. Hence, the connection of the modules in Figure 3.13 reflects the paths in the graph from Figure 3.12. Depending on the operational mode commanded to the ADCS, the core framework selects the actuator settings and passes them to the publisher to be passed on to the device manager applications.

This partitioning benefits the flexibility when compared to the criteria defined in Section 2.2. Control algorithms, e. g. the state space control algorithm, may be updated or exchanged individually which enables flexible and fast evaluation of new techniques. By adding, removing and recombining modules, the ADCS is scalable towards diverging mission scenarios, yet the modifications of the codebase is kept at a minimum. The approach reduces complexity of the task into manageable parts which may be designed, implemented and tested independently from each other for gradual development, which is addressed later in Section 3.4. The realization of the different control error determination techniques, control laws and the distribution of the control settings to different actuators is discussed in Chapter 4.

3.3.5 Fault-Detection, Fault-Isolation and Recovery

Fault detection, isolation and recovery (FDIR) is a mechanism for error correction, where a failure in the system is detected, isolated and corrected without any interference from outside the system. The implementation of a robust FDIR concept is crucial for satellites, as they have to secure their survival autonomously, especially when there is no link to the ground station.

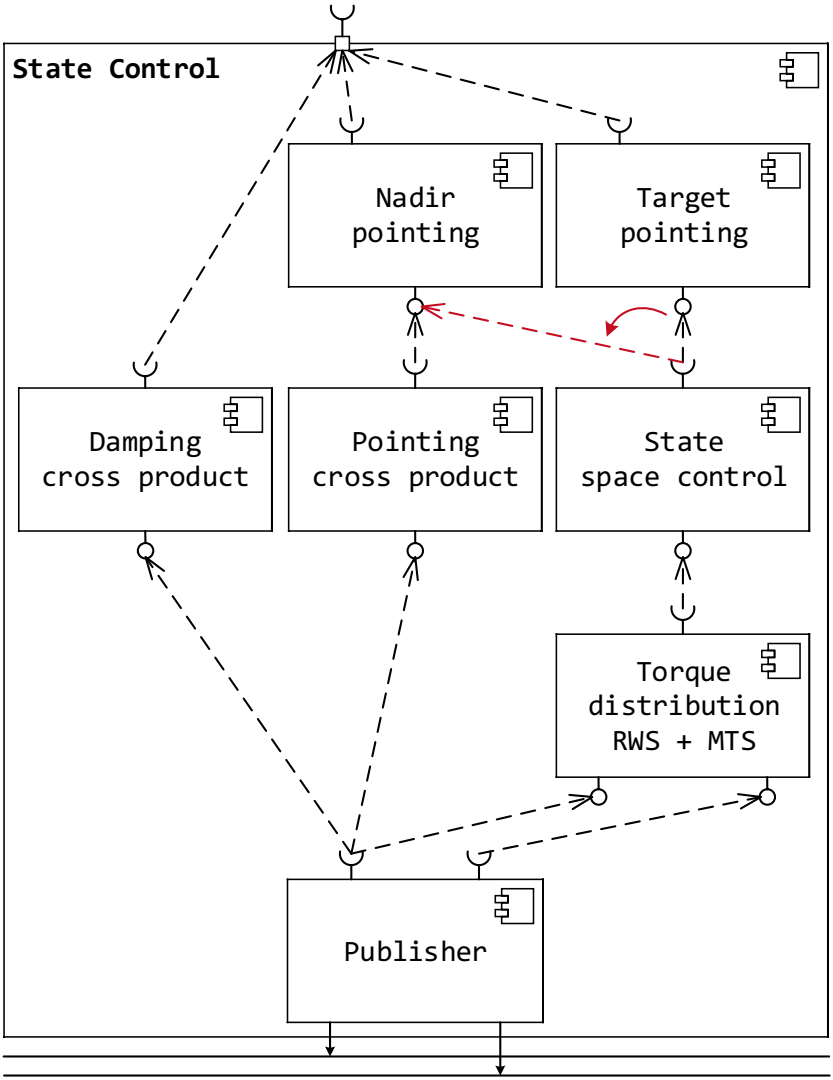

Figure 3.13: Re-Configurable State Control Modes (UML)

The system design of TUBiX20 (cf. Section 2.3) ensures the satellite's survival when tumbling, as the distribution of solar cells on each side result in a positive energy budget for an arbitrary attitude if no payload is operated. Therefore, the satellite does not need active attitude control in the satellite safe mode. Nevertheless, careful planning and implementation of FDIR techniques are necessary to enhance the availability and robustness of the system and furthermore make the attitude control system adaptable for more complex missions in the future.

The basis of successful error correction is the application of surveillance and redundancy. To meet the requirement of single-failure-tolerance, all components of the attitude control system are at least single redundant. Besides the total loss of a component, however, a large variety of failures may occur due to different phenomena like degradation, single event upset (SEU) or even software programming failures within the sensor or actuator firmware. These errors may corrupt the component's performance in different ways and are very hard to identify. Therefore, to design a robust system, it is crucial to implement also functional redundancy apart from the physical hardware redundancy. This functional redundancy is capable of detecting hardware failures or calculation errors via cross-checks of measurements and estimations.

The general FDIR concept for the TUBiX20 platform was first presented with the contribution of the author in [50] and will be introduced briefly here to provide the basis for the investigation of the specific ADCS FDIR mechanisms. Corresponding to the partitioning into four platform levels from Figure 2.2, four FDIR levels are defined. These levels define which parts of the system are affected by a fault, and therefore classify where the recovery of a fault takes place and not where the fault occurs. For example, a (temporary) failure of a component may be handled on the component level, e. g. by reinitialization, whereas the loss of a component must be propagated to the device level to manage the component's hardware redundancy. If the loss of the component also leads to the loss of the complete device, the recovery must consequently take place on the subsystem level, for example by selecting a functional redundancy for the device. The following sections describe the author's approach to apply this concept to the ADCS.

Component Level FDIR Mechanisms

The first FDIR level performs plausibility checks on the component layer. The following checks are performed for each component:

1. Is the component enabled?
2. Is the communication formally correct (e. g. I^2C)?,
3. Is the component in a valid state (e. g. are all internal status registers set like expected)?,
4. Is the measurement within an expected range (e. g. is the measured geomagnetic field vector physically possible)?

While faults regarding power status, communications or internal states may be recovered by reconfiguring the component, a fault due to invalid measurements must be propagated directly to the device level.

Device Level FDIR Mechanisms

The information of all components which form a device is merged on the device level. Here, the following checks are performed:

1. Are the current and temperature measurements for all components at an uncritical level?
2. Are comparable measurements of the redundant components (e. g. magnetic field sensor) within a reasonable range of deviation?
3. Are all components required for the device available and healthy (e. g. all uniaxial sensors which form a triaxial sensor system)?

If the overall device measurement may be determined despite a deviation of one component, this fault is recovered on the device level. Regarding the last check, a fault of a required component leads to the loss of the device and the fault must be propagated on the subsystem level.

Figure 3.14 shows an example for the FDIR structure of a device manager for a triaxial angular rate sensor. For simplicity, however, only one axis is shown. While the surveillance for temperature and current are assigned to the service layer, the checks for the angular rate measurements belong to the user layer.

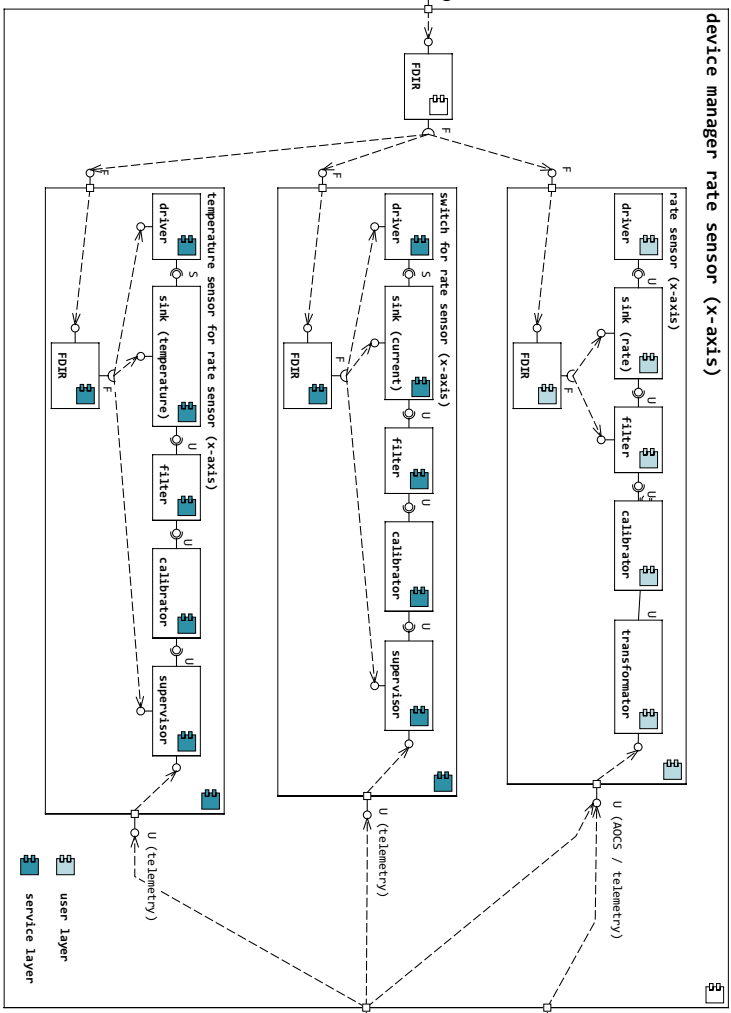

Figure 3.14: TUBiX20 Device Manager FDIR (UML) [50]

As described in Section 2.3, the service layer FDIR is unified throughout the system since the same hardware ICs and software device drivers as well as filter, calibrator and supervisor modules are used for all computational nodes and subsystems. The user layer FDIR is structured similarly, but here the data processing depends on the specific sensor's characteristics.

The FDIR mechanisms on the component and device layer are implemented in the device applications of the according device. Since there are no dependencies on other applications, adding or removing a device has no consequences for other applications. This makes the implementation flexible and clear. The fusion of information from different devices takes place in the subsystem level situated within the ADCS core framework, which is described in the following section.

Subsystem FDIR Mechanisms

The third layer of the FDIR concepts is implemented in the state estimation and control modules. TUBiX20 uses different parallel state estimation modules, which in turn use measurements from different sensors and different algorithms to estimate the state equations. Having all state quantities estimated in one module and furthermore the predicted state from the last control cycle, the plausibility of the state may be determined by cross-checking the different state quantities. The following list gives some example for these plausibility checks:

- the estimated attitude is compared to the predicted attitude from the last cycle
- the estimated geomagnetic field or Sun vector are compared to the inertial reference vectors transformed to the body frame
- the angular rate is compared to the derivation of the attitude
- the angular rate is compared to the predicted angular rate from the last cycle

It has to be noted here, that depending on the estimation method for a quantity, not all cross-checks are available. Having estimated the attitude from vector observations means using measurements and inertial reference of the geomagnetic field and Sun vector, therefore a cross-check of the measured

values with the transformed inertial reference does not bring a meaningful check, as the inputs remain the same.

To implement the various cross-checks for the different state quantities, dedicated modules may be introduced into the core assembly described in Section 3.3.1, as shown in Figure 3.15. By following this concept, the FDIR checks as well as the attitude determination and control algorithms may be implemented independently from each other; hence, they may be added, replaced or removed without interference of the system as a whole. Therefore, even complex ADCS concepts with a comprehensive FDIR strategy retain a clear and straightforward structure.

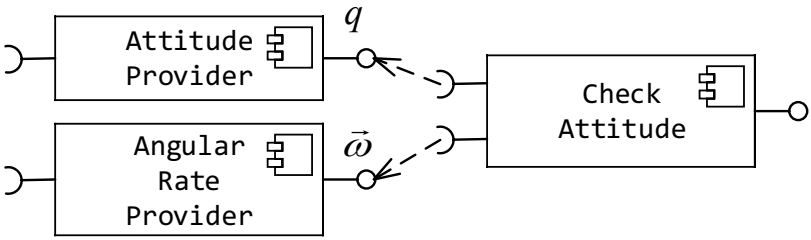

Figure 3.15: FDIR Module Integration Example (UML)

3.4 Development and Verification Process

Within TUBiX20, the author defined and applied a process model for flexible low-cost verification of the ADCS, which was first presented in [68]. The process model aligns with the ECSS product life cycle [41] and covers the ECSS phases A to D. It offers a unified procedure to subdivide all phases and consequently enables a well-structured and result-driven workflow.

3.4.1 Process Steps and Milestones

Since it is a long process from the first draft of the system's architecture until passing end-to-end tests, preferably with the spacecraft performing complex attitude control maneuvers on an air-bearing testbed, it is useful to divide

the procedure into a series of minor steps and define milestones in order to evaluate the progress objectively.

To this end, the process elaborated by the author defines four steps, which each have to pass a functional verification as a defining milestone. This functional verification proves the performance of the ADCS in a closed-loop way via simulations. These simulations, however, are performed at different stages of abstraction. Beginning purely virtually, the simulated ADCS's parts are replaced step-by-step by real software and hardware. As criteria for the verification, different simulation scenarios are deduced from the system requirements. Once defined, these scenarios are re-used in the different verification steps. The four milestones are, comparable with Eickhoff [69]:

1. The system is modeled in a close-loop simulation (0-B)
2. The software is implemented in the target language for all nodes (C/D)
3. The software is integrated in the target hardware for all nodes (C/D)
4. The system is completely integrated and verified (D/E)

Here, the according ECSS project phase is given in brackets. In each step of the process, different aspects of the system design have to be considered. However, the general procedure is very similar and always contains the sections

- analysis
- design
- implementation and tests
- verification via simulation

The structure of the process is illustrated in Figure 3.16. It is derived from the spiral model, which is often used for the development of embedded systems (cf. [70]).

In the analysis for the first step, the overall attitude determination and control concept is elaborated based on the requirements and the according hardware is chosen. During the design section, attitude control modes are defined and algorithms for data processing and control are evaluated and structured to form the high-level structure of the ADCS. The algorithms are implemented into simulation models and tested individually. Together with environment and dynamics as well as sensor and actuator models, an algorithm in the loop

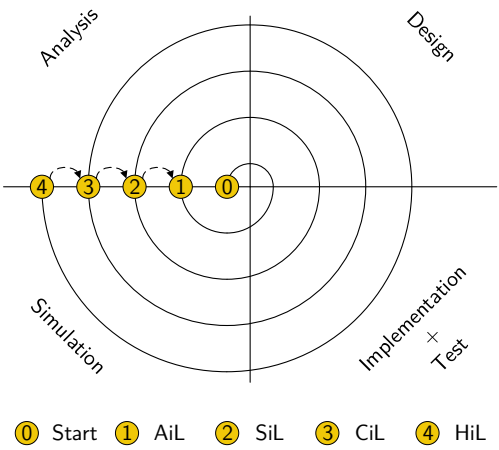

Figure 3.16: Development Process

(AiL) simulation model is composed. Its evaluation concludes phase B and the results for the different scenarios provide a good proof of concept for the preliminary design review (PDR).

The second step targets the development of the ADCS's software. Firstly, the software requirements are derived. They follow from the ADCS concept as well as the spacecraft's system design. Events, signals and actors in the software are identified thereafter. This analysis is followed by designing the software architecture, which is the base for the definition of requirements for the hardware computational nodes. Subsequently, the individual modules of the software are implemented and tested. Here, the continuous integration (CI) practice ensures that the software is already well tested within the development cycle, which leads to more robust results, as presented by the author in [2]. Once the software is implemented, it is integrated into the simulation model from the previous step, replacing all control algorithm models by the actual control software. In the following software in the loop (SiL) simulation, an early verification of the software can already take place long before the hardware is available. Here, the simulation results from the previous step (AiL) can be used as a reference to analyze the software's correctness. The software in

the loop simulation further provides useful data for the critical design review (CDR) report.

The goal of the following step is the integration of the software on the flight hardware. To this end, all hardware-specific parts of the software such as device drivers have to be implemented and tested. Furthermore, the integration is usually complex and time-consuming, since the software runs as part of an embedded system in a microcontroller environment. Once again, following the continuous integration development practice shortens the integration process [2] and benefits robustness due to remote target unit tests. For the following controller in the loop (CiL) simulation, the ADCS software runs on the target controller, while the simulation model from the previous step is modified to run in real-time and then integrated onto a spare unit of the target hardware or a development board. Hence, no additional costs for real-time simulations are required.

Subsequently, the sensors and actuators hardware is incorporated in the ADCS loop as part of the overall integration process of the satellite. The final verification is performed as hardware in the loop (HiL) simulation, where the complete ADCS is performing maneuvers defined by the simulation scenarios on an air-bearing testbed. Figure 3.17 shows the verification of the TechnoSat ADCS (cf. Section 5.1) on the testbed at Technische Universität Berlin.

The presented development and verification process was formulated by the author at the beginning of the TechnoSat mission. During the course of the project, all four steps were executed successfully and all milestones were reached. The process has thereby proven to be suitable for the gradual development of the ADCS from the first simulation model to the fully integrated system and several synergy effects could be exploited. All insights of the investigation are presented in the following section.

3.4.2 Synergy Effects

Following the process model from the previous section, synergy effects between the different steps and tools could be created and hence unnecessary effort when supplying support equipment or preparing different simulation environments could be avoided.

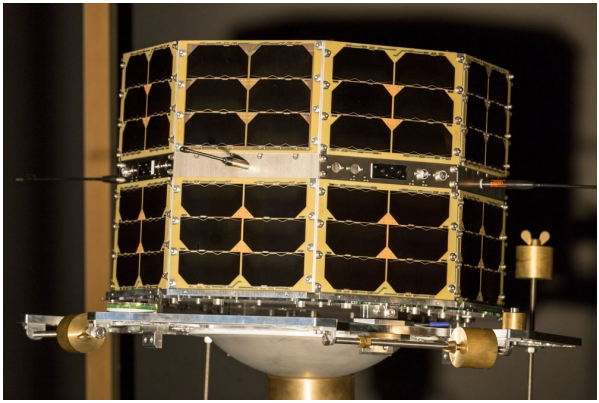

Figure 3.17: TechnoSat Flight Model on the TUBiX20 Air-bearing Testbed
Image credit: Philip von Keiser

Simultaneous Simulation Model Refinement

The first example of exploiting synergy effects is the development of the simulation models used throughout the whole development cycle of the ADCS. While preparing the algorithm in the loop simulation, the model architecture was defined. Figure 3.18 shows the top-level view of the TUBiX20 ADCS simulation model.

Apart from the evaluation of algorithms during analysis, the models also served as a useful tool for the development of the individual modules in the target language, which is C++ in the case of TUBiX20. Since simulation tools or languages like MATLAB/Simulink and Modelica offer the integration of C++ source code, simulation models and target software can be developed simultaneously from the same codebase and hence need to be verified only once. Consequently, the flight software development time is shortened since the modules have already been implemented and verified for the simulation, and, on the other hand, the simulation models are always up to date with the actual flight software. When implementing state estimation and control algorithms as simulation models directly from the flight software codebase, the closed-loop simulation, e. g. a MATLAB model, becomes more similar to the final system design. Moreover, proving to be very useful, the flight

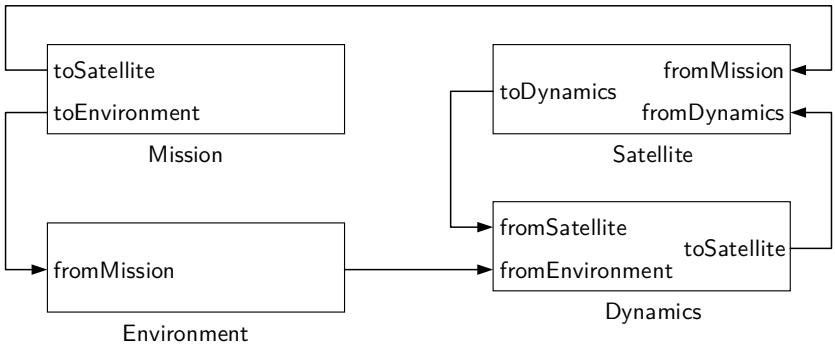

Figure 3.18: ADCS Simulation Model in MATLAB/Simulink

software algorithms can be verified with complex scenarios at low effort when also implemented as a simulation model. This approach differs from the process proposed by Eickhoff [69], where the "algorithms are mostly not yet implemented in the target language".

Early Operations of Virtual Satellite

According to Section 2.3, the TUBiX20 platform consists of a network of computational nodes connected via a central power and data bus. Each node runs a dedicated subsystem software composed of independent building blocks. The communication of the building blocks is facilitated via the middleware of the operating system, hence it is irrelevant for the communicating building blocks where their instances are hosted in the network. Moreover, the RODOS operating system supports the emulation of the embedded software on a Linux host PC and, as opposed to [69], no additional test bed for simulations is required. While the middleware uses a redundant CAN bus on the target hardware, UDP messages are used when emulated. Either way, the physical layer is transparent for the building block. In this manner, network messages may be monitored but also induced easily without modifications to the software for both flight hardware and emulation.

The TUBiX20 electrical ground support equipment (EGSE) makes the satellite's data buses accessible via USB and WLAN. The EGSE hardware is

connected to a ground support server and therefore allows passing messages between server and satellite. Moreover, it provides several TCP/IP ports for client software, which may use any of the implemented message protocols to interact with the satellite on various levels. The EGSE server is able to work with all communication protocols and physical transmission types used in the satellite. The detailed structure of the TUBiX20 EGSE was presented with contribution of the author in [44].

Both emulation of the software and the EGSE architecture make the ground support server as well as the mission control software used for operations directly accessible without any further development effort and hence allows its usage from an early stage of the project. This entailed several benefits, which shall be explained next. First of all, the EGSE and ground software could be, after its verification, further tested in use which offered the chance of feature requests or finding errors. From the ADCS software development point of view, the same EGSE and operations software could already be used before any hardware was available.

Earlier, it the way in which the individual algorithms for attitude determination and control in the target language are transferred into simulation models was already shown. However, in the next step, the complete software was used as a whole including telecommand and telemetry interface as well as thread scheduling. Only hardware-dependent parts such as device drivers needed to be adapted to comply with the simulation. A flexible approach to combine the environment and dynamics simulation with the software was to use code-generation and then integrate the model as a building block into the software network system, as shown in Figure 3.19. While the upper half shows the derivation of simulation models from the ADCS module codebase and also the assembly of the algorithm in the loop simulation, the bottom half depicts the two following verification milestones, namely software in the loop and controller in the loop (cf. Section 3.4.1). By embedding the model assembly into a building block, the ADCS software does not need to be modified at all, since it communicates via the middleware as usual. Being part of the network, the simulation block is scheduled like all other building blocks. For the software in the loop simulation, the whole network runs in the Linux emulation environment, while in the controller in the loop simulation, the ADCS software runs on the target hardware and the simulation model block runs on a separate controller. To host the simulation on a spare unit of the target hardware or on a development kit does not entail any extra costs

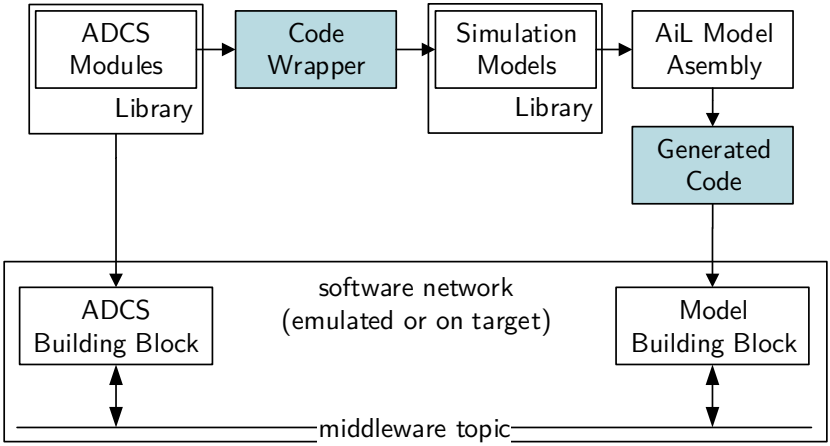

Figure 3.19: Simulation Model Integration

or development effort, since all hardware-dependent parts of the operating system and the network communication are already implemented and there is no overhead of test code at all. As a consequence, there is no additional test or support software needed and hence development resources are saved.

The possibility to operate the ADCS completely virtually and yet using the complete suite of mission control software from telecommand graphical user interface to telemetry display and database recording provided several benefits. Firstly, the verification of the software could be started before the target hardware was available and hence the development progress of the ADCS was not endangered by delays in the hardware design or manufacturing. Secondly, testing the flight software in a PC environment proved to be time efficient since no time-consuming software uploads are necessary and furthermore all debug facilities could be used instantly. Hence, the complete data and control flow could be monitored while at the same time performing simulations of complex mission scenarios. Even after the satellite was already integrated and hardware in the loop simulations were in progress, the software in the loop simulation was still used, either when the EQM of the satellite was in use by another engineer or to quickly evaluate insight from the hardware tests.

In-Orbit Verification of new Modules

Updating the software in-orbit is a complicated process which demands a lot of careful preparation. Since one of the main risks is to endanger the satellite when new software is uploaded, the modifications or new features should be tested as well as possible. Here, the process described in this section may serve as valuable, since it involves incremental verification on different levels while being flexible and time-efficient.

A newly developed ADCS module will run through the same cycle including all verification milestones. Since the simulation infrastructure is used throughout the whole project lifetime, it is still up-to-date after the satellite has been launched. The implementation of an individual module results in both flight software code and simulation model – as based on the same code, hence the different verification steps from purely virtual to on-target execution may be performed in short time while providing an already well-tested module for a subsequent hardware in the loop test. For the straightforward integration of new modules into the ADCS software, however, the systems architecture must allow for modification without interference of other code parts, which has already been addressed earlier.

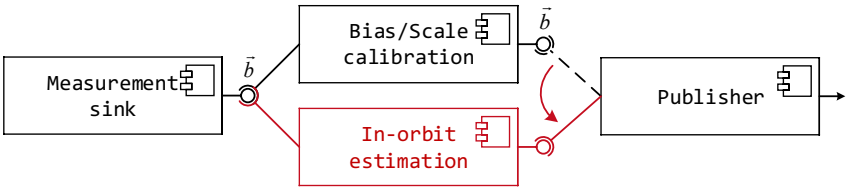

Figure 3.20: In-Orbit Selection of Calibration Methods (UML)

An example for the in-orbit verification of a new feature is given in Figure 3.20. Here, a new technique for in-orbit estimation of calibration parameters shall be evaluated. Based on the staged calibration assembly shown in Figure 3.4, a module for the new algorithm is inserted. The calibration technique applied may then be reconfigured at runtime. After the evaluation and if the in-orbit calibration is successful, both modules may remain as alternatives or one of them may be discarded and removed.

3.4.3 Early Verification Coverage

The previous sections have already shown that the design and verification process defined in this thesis allowed the early verification of the flight software with the help of simulation models. First presented by the author in [68], Figure 3.21 shows the verification coverage for the different layers of the TUBiX20 ADCS. On the lowest level, running the flight software under soft realtime conditions is already possible within the software in the loop simulation, while one step later hard real time conditions are met after the software was integrated on the target hardware. The same applied to the device driver implementation. Here, the generic driver interfaces defined by the TUBiX20 software team provide an abstraction from the hardware-dependent parts and hence allowed to replace them with mock-ups like sensor models for the virtual satellite. The modules for state determination and control were already available after the first process step, since simulation models and flight software have the same code base. The architecture of the core application, i. e. the interface to TM/TC as well as the state machine for the control modes, were introduced for the software in the loop simulation. Finally, since the operations software may be used from an early stage, the same tools and report templates for data processing could be used for all verification tests from a purely virtual satellite to orbit experiments.

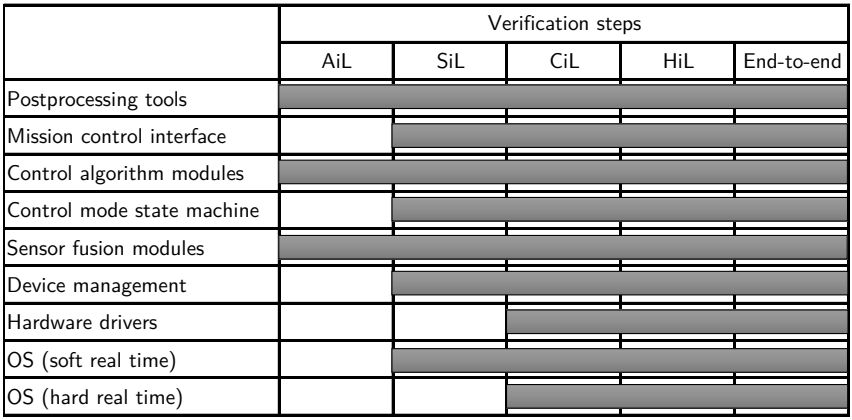

Figure 3.21: Verification Coverage at Process Milestones [68]

4 Attitude Determination and Control Techniques

While the last chapter presented the design and verification of the ADCS on an abstract level to demonstrate its adaptivity to diverging mission scenarios, this chapter provides the mathematical background for different attitude determination and control algorithms which were previously treated as *black boxes*. By now presenting different approaches for the concrete realization of their content, the concept is continued.

The techniques presented are ordered in the same way they may be used in the design and verification process. After the introduction of different attitude representations, the equations of motion for the satellite's dynamics and kinematics are stated in their general form at first and then linearized and transferred into the state space representation. At this point, the first simulation model to investigate the satellite's motion in space may be built. The modeling of the physics is completed by the subsequently introduced disturbance torques. As a second step, different techniques for state estimation, prediction and control are discussed which then serve as the mathematical background for the implementation of the module library used for the ADCS software. The description of all algorithms and equations continues the use of the symbols which were introduced for the state quantity provider interfaces in Table 3.1 to maintain the consistency of the overall concept. However, the algorithms presented in the following may also be used independently from the design architecture of the previous chapter and may serve the reader as a reference including all required steps for their implementation. Additional information regarding the coordinate systems is provided in Appendix B.

4.1 Attitude Representations

Attitude representations are discussed in nearly every thesis, article or conference paper published in the field of attitude control. As in many other

works, quaternions are used here, since they come with the advantage of less calculation effort than Euler angles or direction cosine matrices (DCMs) and do not entail any singularities. Due to the wide availability of detailed comparisons between the different attitude representations, this is not part of this thesis. Nevertheless, the notation for quaternions allows a certain ambiguity, wherefore the quaternion equations may differ from one publication to another and yet may both be correct. In the following, the quaternion notation used in this thesis is introduced and the mentioned ambiguity is pointed out. In some equations, DCMs are used, and for this reason are briefly introduced thereafter. Finally, the conversion between quaternions and DCMs is discussed. All definitions for attitude representations have been collected with contribution of the author in [71] to serve as a standard for the university's future projects.

4.1.1 Quaternions

A quaternion is defined by a scalar part, s, and a vector part, $\vec{v}$, which are assembled to a 4×1 matrix, q. The order of scalar part and vector part in the matrix differs from one publication to the other (cf. Wertz [72]). Here, the scalar part is the first element of q:

$$q = \begin{bmatrix} s \\ \vec{v} \end{bmatrix} = \begin{bmatrix} s \\ v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} q_s \\ q_x \\ q_y \\ q_z \end{bmatrix} \tag{4.1}$$

Quaternions may be applied for both, transforming a vector from one coordinate system into another or rotating a vector within the same coordinate system. In both cases, the scalar part represents the angle and the vector part represents the axis of transformation or rotation, respectively.

For a transformation from coordinate system A to coordinate system B, the according quaternion is:

$$q_{B \leftarrow A} = \begin{bmatrix} s \\ \vec{v} \end{bmatrix} = \begin{bmatrix} \cos \frac{\alpha}{2} \\ \vec{e} \cdot \sin \frac{\alpha}{2} \end{bmatrix} \tag{4.2}$$

where $\vec{e}$ is the normalized vector of the rotational axis and α is the rotational angle. The mathematical operation for the transformation is given at the

end of this section. To invert the direction of the transformation or rotation, either the angle or the axis may switch sign. It is obvious that if both switch sign, the result is a quaternion which differs from the original one and yet the represented orientation is the same.

The norm of a quaternion is given by Equation 4.3:

$$|q| = \sqrt{q_s^2 + q_x^2 + q_y^2 + q_z^2} \tag{4.3}$$

A Quaternion with the norm $|q| = 1$ is called *unit quaternion*. All quaternions for attitude representation are unit quaternions, since a norm unequal to one would modify not only a vector's orientation but also its length. To normalize a quaternion, i.e. transform it into a unit quaternion, it is divided by its norm:

$$\|q\| = \frac{q}{|q|} \tag{4.4}$$

The conjugate quaternion has an inverted vector part:

$$q^* = \begin{bmatrix} q_s \\ -q_x \\ -q_y \\ -q_z \end{bmatrix} \tag{4.5}$$

To obtain the inverse of a quaternion, its conjugate is normalized.

$$q^{-1} = \frac{q^*}{|q|} \tag{4.6}$$

For all unit quaternions, the inverse is equal to the conjugate, as they have the norm one. The product q of two quaternions q_1 and q_2 is defined by:

$$q_1 \otimes q_2 = q = \begin{bmatrix} s \\ \vec{v} \end{bmatrix} = \begin{bmatrix} s_1 \cdot s_2 - \vec{v}_1 \circ \vec{v}_2 \\ s_1 \cdot \vec{v}_2 + s_2 \cdot \vec{v}_1 + \vec{v}_1 \times \vec{v}_2 \end{bmatrix} \tag{4.7}$$

where $\circ$ is the dot product of vectors $\vec{v}_1$ and $\vec{v}_2$ and $\times$ is their cross product.

The transformation of a vector $\vec{v}$ from frame A to frame B is defined as:

$$\vec{v}_B = q_{B \leftarrow A} \otimes \begin{bmatrix} 0 \\ \vec{v}_A \end{bmatrix} \otimes q_{B \leftarrow A}^{-1} \tag{4.8}$$

$$\vec{v}_B = q_{B \leftarrow A} \otimes \begin{bmatrix} 0 \\ \vec{v}_A \end{bmatrix} \otimes q_{A \leftarrow B} \tag{4.9}$$

$$\vec{v}_B = q_{B \leftarrow A} \odot \vec{v}_A \tag{4.10}$$

where $\otimes$ is the operator of a quaternion multiplication (cf. Equation 4.7). The operator $\odot$, on the other hand, represents the transformation of a vector via a quaternion, as may be seen when comparing Equation 4.9 and Equation 4.10. The notation of the quaternion and the vector, i. e. their subscripts which state the coordinate systems, may be used as a check if the operation is applicable. In Equation 4.10, the transformation of the vector $\vec{v}$ from the coordinate system A to the coordinate system B directly reflects in the subscript of the quaternion $q_{B \leftarrow A}$ and the correctness of the operation may be checked by comparing the adjacent subscripts.

4.1.2 Direction Cosine Matrix

A direction cosine matrix is a transformation matrix which is composed of the direction cosine values between the initial coordinate system and the target coordinate system.

Let A be the initial coordinate system and B the target coordinate system of a transformation. The base vectors of A are given by $\vec{x}_A$, $\vec{y}_A$ and $\vec{z}_A$, whereas $\vec{x}_B$, $\vec{y}_B$ and $\vec{z}_B$ are the base vectors of system B. The direction cosine matrix which transforms a vector from system A to system B shall be called $T_{B \leftarrow A}$ and is defined by

$$T_{B \leftarrow A} = \begin{bmatrix} cos\sphericalangle(\vec{x}_A, \vec{x}_B) & cos\sphericalangle(\vec{y}_A, \vec{x}_B) & cos\sphericalangle(\vec{z}_A, \vec{x}_B) \\ cos\sphericalangle(\vec{x}_A, \vec{y}_B) & cos\sphericalangle(\vec{y}_A, \vec{y}_B) & cos\sphericalangle(\vec{z}_A, \vec{y}_B) \\ cos\sphericalangle(\vec{x}_A, \vec{z}_B) & cos\sphericalangle(\vec{y}_A, \vec{z}_B) & cos\sphericalangle(\vec{z}_A, \vec{z}_B) \end{bmatrix} \tag{4.11}$$

$$= \begin{bmatrix} \vec{x}_A \cdot \vec{x}_B & \vec{y}_A \cdot \vec{x}_B & \vec{z}_A \cdot \vec{x}_B \\ \vec{x}_A \cdot \vec{y}_B & \vec{y}_A \cdot \vec{y}_B & \vec{z}_A \cdot \vec{y}_B \\ \vec{x}_A \cdot \vec{z}_B & \vec{y}_A \cdot \vec{z}_B & \vec{z}_A \cdot \vec{z}_B \end{bmatrix} \tag{4.12}$$

$T_{A \leftarrow B}$ is an orthonormal matrix because the base vectors of A and B are in both cases orthogonal unit vectors. Therefore, the transpose of a DCM is the same as the DCM representing the inverse transformation.

For all transformation matrices, the transpose is equal to the inverse of the matrix:

$$T_{B \leftarrow A}^T = T_{B \leftarrow A}^{-1} = T_{A \leftarrow B} \tag{4.13}$$

and

$$det(T_{A\leftarrow B}) = det(T_{B\leftarrow A}) \tag{4.14}$$

The transformation of a coordinate system about each basis vector with a rotation angle θ is described by the following elementary transformation matrices:

$$\begin{aligned} R^x(\theta) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\ \theta & sin\ \theta \\ 0 & -sin\ \theta & cos\ \theta \end{bmatrix} \\ R^y(\theta) &= \begin{bmatrix} cos\ \theta & 0 & -sin\ \theta \\ 0 & 1 & 0 \\ sin\ \theta & 0 & cos\ \theta \end{bmatrix} \\ R^z(\theta) &= \begin{bmatrix} cos\ \theta & sin\ \theta & 0 \\ -sin\ \theta & cos\ \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{4.15}$$

The transformation of a vector $\vec{v}_A$ from coordinate system A to coordinate system B is given by:

$$\vec{v}_B = T_{B\leftarrow A} \cdot \vec{v}_A \tag{4.16}$$

4.1.3 Conversion of Attitude Representations

Since coordinate transformations are mostly expressed via DCMs, while the control theory uses quaternions due to their stated advantages, this section describes the conversion between the two attitude representations.

Direction Cosine Matrix to Quaternion

Each of the four elements of a quaternion may be calculated from the DCM's main diagonal. Thereafter, the remaining three elements follow from the

secondary diagonals. Hence there are four different ways to start the conversion, as shown in Equation 4.17.

$$\begin{aligned} q_s &= \sqrt{\frac{1}{4} \cdot (1 + T_{11} + T_{22} + T_{33})} \\ q_x &= \sqrt{\frac{1}{4} \cdot (1 + T_{11} - T_{22} - T_{33})} \\ q_y &= \sqrt{\frac{1}{4} \cdot (1 - T_{11} + T_{22} - T_{33})} \\ q_z &= \sqrt{\frac{1}{4} \cdot (1 - T_{11} - T_{22} + T_{33})} \end{aligned} \tag{4.17}$$

Depending on the element which was determined first, the calculation of the remaining elements follows from Table 4.1.

Table 4.1: Calculating Quaternion Elements from Secondary Diagonals

	q_s	q_x	q_y	q_z
q_s	q_s	$\frac{T_{32}-T_{23}}{4 \cdot q_s}$	$\frac{T_{13}-T_{31}}{4 \cdot q_s}$	$\frac{T_{21}-T_{12}}{4 \cdot q_s}$
q_x	$\frac{T_{32}-T_{23}}{4 \cdot q_x}$	q_x	$\frac{T_{21}+T_{12}}{4 \cdot q_x}$	$\frac{T_{13}+T_{31}}{4 \cdot q_x}$
q_y	$\frac{T_{13}-T_{31}}{4 \cdot q_y}$	$\frac{T_{21}+T_{12}}{4 \cdot q_y}$	q_y	$\frac{T_{32}+T_{23}}{4 \cdot q_y}$
q_z	$\frac{T_{21}-T_{12}}{4 \cdot q_z}$	$\frac{T_{13}+T_{31}}{4 \cdot q_z}$	$\frac{T_{32}+T_{23}}{4 \cdot q_z}$	q_z

The reason why all four possibilities for the conversion are presented here is due to the fact that a value close to zero for the first element determined may cause high numerical inaccuracies, since it is included in the denominator in the equations from Table 4.1. It is therefore recommended to determine all four elements independently from Equation 4.17 first, and then choose the maximum of these elements to apply Table 4.1 for the remaining three elements. The second step is important due to the ambiguity that two quaternions with opposing signs in all four elements still represent the same orientation.

Quaternion to Direction Cosine Matrix

To transform a quaternion, q, to a direction cosine matrix, T, Equation 4.18 is used.

$$T = \begin{bmatrix} q_s^2 + q_x^2 - q_y^2 - q_z^2 & 2 \cdot (q_x \cdot q_y - q_z \cdot q_s) & 2 \cdot (q_x \cdot q_z + q_y \cdot q_s) \\ 2 \cdot (q_x \cdot q_y + q_z \cdot q_s) & q_s^2 - q_x^2 + q_y^2 - q_z^2 & 2 \cdot (q_y \cdot q_z - q_x \cdot q_s) \\ 2 \cdot (q_x \cdot q_z - q_y \cdot q_s) & 2 \cdot (q_y \cdot q_z + q_x \cdot q_s) & q_s^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{4.18}$$

4.2 Dynamics and Kinematics

Before elaborating the control algorithms for a system, its equations of motion must be analyzed to provide a basis for all further design steps. In this section, the dynamics and kinematics equation for a spacecraft are reviewed in three steps. Firstly, the equations of motion are introduced as a starting point for all further steps. As these equations form a non-linear system of differential equations, it is necessary to perform a linearization in order to apply the linear control theory, which is presented secondly then. Thirdly, this linearization is expressed in the state space form, which will serve as a reference for the different state determination and control techniques discussed later.

4.2.1 Equations of Motion

The dynamics of the satellite with regard to the Moment of Inertia Coordinate System (MOI) is described by the well-known equation [73]:

$$\sum \vec{\tau}_{MOI} = \sum \dot{\vec{H}}_{MOI} + \sum (\vec{\omega}_{MOI} \times \vec{H}_{MOI}) \tag{4.19}$$

The sum of all torques, i. e. the left-hand-side of Equation 4.19 comprises internal torques and external torques. Internal torques may be applied by actuators, but also result from internal disturbances such as moving parts or sloshing liquids. The external torques, on the other hand, result from environmental disturbances, which will be discussed later in Section 4.3. With regard to the inertial TOD coordinate system, the overall angular momentum

is constant over time. This angular momentum conservation forms the basis for attitude control via momentum exchange devices such as reaction wheels:

$$\dot{\vec{H}}_{TOD} = \vec{0} \tag{4.20}$$

The dynamics in Equation 4.19 describes the rotation of the satellite. The resulting change in the spacecraft's orientation follows from the kinematics equation. To begin with, Equation 4.21 describes the kinematics differential equation for a direction cosine matrix, T, according to Wertz [72]:

$$\dot{T} = \begin{bmatrix} 0 & \omega_z & -\omega_y \\ -\omega_z & 0 & \omega_x \\ \omega_y & -\omega_x & 0 \end{bmatrix} \cdot T \tag{4.21}$$

Expressed in terms of quaternions, Equation 4.21 results in

$$\dot{q}_{TOD \leftarrow MOI} = \frac{1}{2} \cdot \Omega \cdot q_{TOD \leftarrow MOI} \tag{4.22}$$

$$= \frac{1}{2} \cdot q_{TOD \leftarrow MOI} \otimes \begin{bmatrix} 0 \\ \vec{\omega} \end{bmatrix} \tag{4.23}$$

where $q_{TOD \leftarrow MOI}$ is the transformation quaternion from the MOI to the TOD system and $\vec{\omega}$ is the angular rate of the satellite in the MOI system. The matrix Ω follows from $\vec{\omega}$ in x, y and z-direction [72].

$$\Omega = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{4.24}$$

It is important to note that the quaternion notation in [72] differs from Section 4.1.1.

4.2.2 Dynamics and Kinematics Linearization

Due to the cross-coupling of the coordinate axes, the dynamics equation from Equation 4.19 is a system of non-linear differential equations. To formulate control laws following the linear control theory, a linearization will

be derived here. This is simply achieved by neglecting the cross-coupling for the differential equation and treat it as a disturbance of the control loop instead.

Each axis considered individually, the dynamics equation is reduced to

$$\dot{H}_i = \tau_i = I_i \cdot \dot{\omega}_i \tag{4.25}$$

The subscript i stands for the coordinate axes x, y and z. From Equation 4.25, the three-dimensional linearization follows directly:

$$\dot{\vec{\omega}} = \frac{\vec{\tau}}{\vec{I}} \tag{4.26}$$

$$\dot{\omega}_i = \frac{\tau_i}{I_i} \tag{4.27}$$

with $\vec{I}$ as the inertia tensor.

For small angles, the linearization of the kinematics equations can be formulated for a single axis as

$$\dot{\theta} = -\omega \tag{4.28}$$

The sign of the angle θ is negative since the angle describes the transformation from MOI to TOD. The relation from transformation angle may be linearized for small angles and Equation 4.30 is obtained [72].

$$\begin{bmatrix} \widetilde{q}_s \\ \widetilde{q}_x \\ \widetilde{q}_y \\ \widetilde{q}_z \end{bmatrix} \cong \begin{bmatrix} 1 \\ -\frac{1}{2} \cdot \theta_x \\ -\frac{1}{2} \cdot \theta_y \\ -\frac{1}{2} \cdot \theta_z \end{bmatrix} \tag{4.29}$$

Once again, the negative sign results from the quaternion notation (cf. Section 4.1.1). Subsequently, Equation 4.28 and Equation 4.29 result in the linearized kinematics equation in quaternion notation:

$$\begin{bmatrix} \dot{\widetilde{q}}_x \\ \widetilde{q}_y \\ \widetilde{q}_z \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{4.30}$$

4.2.3 State Space Representation

The linearized equations for dynamics and kinematics, Equation 4.27 and Equation 4.29, respectively, form the state space representation, which has the general form

$$\dot{x} = A \cdot x + B \cdot u \tag{4.31}$$
$$y = C \cdot x + D \cdot u \tag{4.32}$$

Selecting the angular rate vector and the vector part of the approximated quaternion, the state space representation is

$$\begin{bmatrix} \dot{\widetilde{q}} \\ \omega \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2} \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \widetilde{q} \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \\ I^{-1} \end{bmatrix} \cdot \tau \tag{4.33}$$
$$\widetilde{q} = \begin{bmatrix} E & 0 \end{bmatrix} \cdot \begin{bmatrix} \widetilde{q} \\ \omega \end{bmatrix} \tag{4.34}$$

Since the linearization is only valid near the control working point, the selected working point should be chosen accordingly. The quaternion $q_{TOD \leftarrow MOI}$ is therefore not feasible. An appropriate solution will be discussed later in Section 4.5.1.

4.3 Environmental Disturbance Torques

The interaction of the spacecraft with the space environment results in different disturbance torques which affect the attitude control accuracy. Following, the four predominant environmental disturbance torques are presented in order of their magnitude for a low earth orbit.

In most cases, the disturbance torques are estimated in terms of their absolute value for worst case studies. However, due to their different origin and dynamic characteristics, a more realistic result is achieved when analyzed within a three-dimensional simulation. For the TUBiX20 platform, the disturbance torques were simulated as such to design and verify the ADCS using the following equations.

4.3.1 Magnetic Field Disturbance

The disturbance torque, $\vec{\tau}_M$, results from the interaction of the spacecraft's residual dipole, $\vec{m}_D$, with the Earth's magnetic field, $\vec{b}$ [74].

$$\vec{\tau}_M = \vec{m}_D \times \vec{b} \tag{4.35}$$

4.3.2 Gravity Gradient Disturbance

Since the mass of the satellite is not distributed equally, the gravity gradient causes the disturbance torque, $\vec{\tau}_G$, according to [72], [75]:

$$\vec{\tau}_G = 3 \cdot \frac{\mu}{|\vec{r}|^3} \cdot [-\|\vec{r}\| \times (I \cdot \|\vec{r}\|)] \tag{4.36}$$

Here, $\vec{r}$ is the vector pointing from Earth to the spacecraft, while $\mu = G \cdot M$ is the product of gravitational constant and the Earth's total mass. Finally, $\vec{I}$ is the inertia tensor. The vector of gravity, $\vec{g}$, is [5]

$$\vec{g} = -\frac{\mu}{|\vec{r}|^3} \cdot \vec{r} \tag{4.37}$$

and it follows that

$$\vec{\tau}_G = 3 \cdot \frac{|\vec{g}|}{|\vec{r}|} \cdot [\|\vec{g}\| \times (I \cdot \|\vec{g}\|)] \tag{4.38}$$

4.3.3 Solar Radiation Disturbance

The disturbance torque due to solar radiation pressure is calculated from the effective force, $\vec{F}_s$, and the vector from the center of gravity to the force application point, $\vec{r}$ [76]. For the three-dimensional case with n illuminated faces, it follows that

$$\vec{\tau}_s = \sum_{1}^{n} \vec{r}_i \times \vec{F}_{s,i} \tag{4.39}$$

The effective force is

$$\vec{F}_{s,i} = \frac{\vec{s}}{c} \cdot A_i \cdot (1 - \alpha) \cdot (\|\vec{s}\| \cdot \|\vec{n}_i\|) \tag{4.40}$$

Here, $\vec{s}$ is the radiation vector, c the speed of light, A the area of the illuminated face, α the adsorption factor, and $\vec{n}$ the normal vector of the illuminated face.

4.3.4 Atmospheric Drag Disturbance

The atmospheric drag causes a disturbance torque similar to the solar radiation pressure. The aerodynamics force, $\vec{F}_a$, and the vector from the center of gravity to the force application point, $\vec{r}$, create the disturbance [76]:

$$\vec{\tau}_A = \sum_{1}^{n} \vec{r}_i \times \vec{F}_{a,i} \tag{4.41}$$

with

$$\vec{F}_{a,i} = -\frac{1}{2} \cdot \rho \cdot c_D \cdot A_i \cdot |\vec{v}| \cdot \|\vec{v}\| \cdot (\|\vec{v}\| \cdot \|\vec{n}_i\|) \tag{4.42}$$

Similar to the previous section, A is the area of the face and $\vec{n}$ its normal vector. Moreover, ρ is the atmospheric density, c_D the drag coefficient, and $\vec{v}$ is the velocity of the spacecraft.

4.4 State Quantity Determination

This section presents different techniques to determine the state quantities addressed in Section 3.2. Following, different algorithms for state estimation and state prediction are discussed. The relation of these terms was addressed earlier in Section 3.3.2.

4.4.1 Merging Multiple Vector Measurements

Merging multiple sensor measurements not only improves the estimation accuracy of the vector quantity, e. g. by averaging them, but also allows the detection of invalid sensor measurements. The following method described is referred to as the Ding algorithm (cf. Ding, Chen, Xing, *et al.* [77]) and was developed to identify faulty sensors in large sensor networks. For its application for vector quantities, the description from Munir, Gordon-Ross,

and Ranka [78] was modified. While the computational effort is low, the algorithm proved to reliably detect faulty measurements when tested with the TUBiX20 sensor boards. Here, it was used for fault detection of the Sun vector, magnetic field vector and angular rate measurements.

While the algorithm originally targets large sensor networks, the sensors aboard a nanosatellite may all be considered as neighboring sensors, hence the first step of the algorithm, "identify neighboring sensor nodes", is omitted. The remaining steps are (derived from [78]):

1. Compute d_i for each sensor S_i using Equation 4.43
2. Compute y_i for each sensor S_i using Equation 4.46
3. If $|y_i| \geq \Theta$, consider S_i as faulty, otherwise consider S_i as healthy
4. To obtain the estimation of the vector quantity, calculate the average of all healthy sensors

To compute d_i, the original algorithm uses the median of the sensor measurements. However, the median cannot be applied for three-dimensional quantities, since the median for the individual axes might belong to different measurements. Therefore, the coefficient for each sensor is not the difference from the median of the neighboring nodes, but the absolute value of the (vector) difference from the mean value:

$$d_i = \left| \vec{x}_i - \frac{1}{n} \sum_{i=1}^{n} \vec{x}_i \right| \tag{4.43}$$

The mean, μ, of all coefficients d_i is [77]

$$\mu = \frac{1}{n} \sum_{i=1}^{n} d_i \tag{4.44}$$

and the standard deviation, σ, follows as [77]

$$\sigma = \frac{1}{n-1} \sum_{i=1}^{n} (d_i - \mu)^2 \tag{4.45}$$

Finally, a standardized value, y_i, is computed for each sensor:

$$y_i = \frac{d_i - \mu}{\sigma} \tag{4.46}$$

As a threshold, Θ, for the third step of the algorithm, a value of 1.1 showed good results with the TUBiX20 hardware.

4.4.2 Pre-Processing Angular Rate Measurements at High Frequency

To follow the dynamics of a spacecraft, angular rate sensors such as fiber optic rate sensors or MEMS gyroscopes generally use a high measurement frequency. For the sensors used within TUBiX20, their frequencies are within 20 Hz and 100 Hz. However, the control cycle of the ADCS is much slower. For TUBiX20, it is 2 Hz. As a consequence, the sensor measurements must be pre-processed so that the instantaneous angular rate is available for every control cycle. A simple averaging of the measurement data would not fulfill this purpose: if the angular rate changes during the control cycle, the average value cannot equal the instantaneous value.

However, since the actuator settings hold for the complete duration of the control cycle, and therefore the applied torque remains constant during that time when neglecting disturbances, a linear Kalman filter may be composed to estimate the instantaneous angular rate based on the highly frequent input data. This Kalman filter will further compensate the noise of the sensor to some extent.

The state space equation for the general formulation of a linear Kalman filter is [79]:

$$\dot{x}(t) = Fx + Gu + w \tag{4.47}$$
$$z(t) = Hx + v \tag{4.48}$$

Here, F is the system matrix and G is the input matrix. The vectors x and u describe the state quantities and the input quantities, respectively. Furthermore, H is the observation matrix and z the vector of the measured quantities. The vector w holds the disturbance within the system description in form of white noise and v is the measurement disturbance and also modeled by white noise. The matrices Q and R express this system and measurement noise and are defined as:

$$Q = E(ww^T) \tag{4.49}$$
$$R = E(vv^T) \tag{4.50}$$

As a first step for the implementation, the fundamental matrix is formed via the inverse Laplace transformation [79]

$$\Phi(t) = \mathcal{L}^{-1}[(s \cdot E - F)^{-1}] \tag{4.51}$$

or approximated by a Taylor series.

$$\Phi(t) = E + F \cdot t + \frac{(F \cdot t)^2}{2!} + ... + \frac{(F \cdot t)^n}{n!} \tag{4.52}$$

where E is the identity matrix.

The Kalman filter consists of two phases: prediction and update. The prediction estimates the state quantities for a point in time, k, based on the updated value from the previous one, $k-1$, using the fundamental matrix, which describes the system's dynamics to predict the values over the sample rate t_s:

$$\begin{aligned} \hat{x}_k^- &= \Phi_k \hat{x}_{k-1}^+ + G_k u_{k-1} && (4.53)\\ P_k^- &= \Phi_k \cdot P_{k-1}^+ \cdot \Phi_k^T + Q_k && (4.54)\\ & && (4.55) \end{aligned}$$

$\hat{x}_k^-$ is the predicted state vector at k and P_k^- the covariance matrix of the prediction. In the next step, the predicted quantities are updated via a measurement z_k.

$$\begin{aligned} K_k &= P_k^- \cdot H^T \cdot (H \cdot P_k^- \cdot H^T + R_k)^{-1} && (4.56)\\ P_k^+ &= (E - K_k \cdot H) \cdot P_k^- && (4.57)\\ \hat{x}_k^+ &= \hat{x}_k^- + K_k(z_k - H\hat{x}_k^- - HG_k u_{k-1}) && (4.58)\\ \hat{x}_k^+ &= \hat{x}_k^- + K_k \cdot i_k && (4.59) \end{aligned}$$

The covariance matrix after the update is P_k^+, while $\hat{x}_k^+$ holds the updated state quantities. The matrix K_k is called Kalman gain matrix, and i_k is the innovation.

The general equations for the linear Kalman filter are now used for the state space equations to estimate the angular rate, $\vec{\omega}$, and angular acceleration, $\dot{\vec{\omega}}$, as follows:

$$x = \begin{bmatrix}\omega_x & \omega_y & \omega_z & \dot{\omega_x} & \dot{\omega_y} & \dot{\omega_z}\end{bmatrix}^T \tag{4.60}$$

$$u = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T \tag{4.61}$$

$$f(x) = \int \dot{\omega} \tag{4.62}$$

$$h(x) = x \tag{4.63}$$

From Equation 4.60 to Equation 4.62, it follows that

$$F = \begin{bmatrix}0 & E \\ 0 & 0\end{bmatrix} \tag{4.64}$$

and Equation 4.63 provides

$$H = E \tag{4.65}$$

For the implementation, Equation 4.64 must be inserted into Equation 4.52. Subsequently, Equation 4.53 to Equation 4.59 may be used.

To reduce the computational effort, this filter should directly process the sensor raw data, and the filter outputs may then be used for further on-board calibration or merging such as the technique discussed in Section 4.4.1.

4.4.3 Filtering Sun Vector Measurements

There is a large variety of sensors from very simple solar cells [80] to more complex digital CMOS sensors [81]. While the latter provide a high accuracy with low noise and are less subject to Earth albedo disturbances, a filter may reduce the relatively large noise of the former. Moreover, disturbances like albedo or reflections from the satellite structure may be detected internally when comparing the measurement with a Kalman filter's internal prediction value. Finally, the prediction provides an estimate for the Sun vector even when no measurement data is present during an eclipse.

Since the inertial position of the Sun does not change significantly over time when compared to one control cycle, the filter for Sun vector measurements

may be implemented as a linear Kalman filter, and hence the general equations from the previous section are applied and only the state space equation and the calculation of the fundamental matrix need to be rewritten.

The state space equations simply include the body-fixed Sun vector, $\vec{s}$:

$$x = \begin{bmatrix} s_x & s_y & s_z \end{bmatrix}^T \tag{4.66}$$

$$u = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \tag{4.67}$$

$$h(x) = x \tag{4.68}$$

To determine the fundamental matrix, Φ, the differential equation for the temporal change of a vector, $\vec{v}$, measured in body frame subject to the angular rate of the spacecraft is required:

$$\dot{\vec{v}}_{MOI} = \dot{\vec{T}} \cdot \vec{v}_{TOD} + \vec{T} \cdot \dot{\vec{v}}_{TOD} \tag{4.69}$$

With regard to the duration of one control cycle, the measured vector varies only to a very small extent in the inertial frame, so that the right part of the right hand side of Equation 4.69 may be neglected and it follows that [82]:

$$\dot{\vec{v}}_{MOI} = \begin{bmatrix} 0 & \omega_z & -\omega_y \\ -\omega_z & 0 & \omega_x \\ \omega_y & -\omega_x & 0 \end{bmatrix} \cdot \vec{v}_{TOD} \tag{4.70}$$

$$= -\vec{\omega}_{MOI} \times v_{MOI} \tag{4.71}$$

Since the applied control torque is constant within one control cycle, the angular rate is assumed to be constant when neglecting the disturbance torques. Therefore, the fundamental matrix may be determined from Equation 4.71. To determine the fundamental matrix, the inverse Laplace transformation is applied (cf. Equation 4.51):

$$(s \cdot E - F) = \frac{1}{c} \cdot \left(E \cdot s^2 + S(\vec{\omega}) \cdot s + \begin{bmatrix} \omega_x^2 & \omega_x\omega_y & \omega_x\omega_z \\ \omega_x\omega_y & \omega_y^2 & \omega_y\omega_z \\ \omega_x\omega_z & \omega_y\omega_z & \omega_z^2 \end{bmatrix} \right) \tag{4.72}$$

with

$$c = s^3 + s\omega^2 \tag{4.73}$$

and

$$S(\vec{\omega}) = \begin{bmatrix} 0 & \omega_z & -\omega_y \\ -\omega_z & 0 & \omega_x \\ \omega_y & -\omega_x & 0 \end{bmatrix} \tag{4.74}$$

as the skew-symmetric matrix of the angular rate $\vec{\omega}$.

The transformation to time space gives:

$$\Phi(t - t_0) = E \cdot c_1 + S(\vec{\omega}) \cdot s + \begin{bmatrix} \omega_x^2 & \omega_x\omega_y & \omega_x\omega_z \\ \omega_x\omega_y & \omega_y^2 & \omega_y\omega_z \\ \omega_x\omega_z & \omega_y\omega_z & \omega_z^2 \end{bmatrix} \cdot c_2 \tag{4.75}$$

with

$$\begin{aligned} c_1 &= \cos(\omega t) \\ c_2 &= \frac{\cos(\omega t) - 1}{\omega^2} \\ s &= \frac{\sin(\omega t)}{\omega} \end{aligned}$$

For small angles, Equation 4.75 may be linearized ($\sin(\omega t) = \omega t$, $\cos(\omega t) = 1$), which results in:

$$\Phi(t - t_0) = \Phi_k = E + S(\vec{\omega}) \cdot (t - t_0) \tag{4.76}$$

Just like in the previous section, the fundamental matrix Φ and the observation matrix H are then used in Equation 4.53 to Equation 4.59 to implement the linear Kalman filter.

4.4.4 Filtering Magnetic Field Measurements

Magnetic field sensors are widely used in pico- and nanosatellites [8] for two different reasons: to determine the spacecraft's attitude from vector observations (cf. Section 4.4.6) and also as an input for attitude control using magnetic torquers (cf. Section 4.5.6). Hence a good knowledge of the magnetic field vector benefits both attitude knowledge and attitude control accuracy directly. However, the sensors are subject to noise, but also disturbances caused by the satellite's residual dipole.

Unlike the inertial Sun position, however, the inertial geomagnetic field changes drastically over one orbit. Therefore, the linear Kalman filter presented in the last section is not applicable and an extended Kalman filter must be used.

For non-linear state equations, the state space formulation is defined as

$$\dot{x}(t) = f(x) + w \tag{4.77}$$
$$z(t) = Hx + v \tag{4.78}$$

The system matrix F and the observation matrix H are determined via partial differentiation:

$$F = \left.\frac{\partial f(x)}{\partial x}\right|_{x=\hat{x}} \tag{4.79}$$
$$H = \left.\frac{\partial h(x)}{\partial x}\right|_{x=\hat{x}} \tag{4.80}$$
$$\tag{4.81}$$

To determine the fundamental matrix, the Taylor series already described in Equation 4.52 may be used. The prediction of the state quantities is performed via the non-linear differential equation:

$$\bar{x}_k^- = \bar{x}_{k-1}^+ + \dot{\bar{x}}_{k-1}^+ t_s \tag{4.82}$$

with

$$\dot{\bar{x}}_{k-1}^+ = f(\bar{x}_{k-1}^+) \tag{4.83}$$

The state space equation for the magnetic field sensors is given in the following equations. It is similar to the one for the Sun vector filtering from the previous section, and hence the fundamental matrix is the same.

$$x = \begin{bmatrix} b_x & b_y & b_z \end{bmatrix}^T \tag{4.84}$$
$$u = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \tag{4.85}$$
$$h(x) = x \tag{4.86}$$

For this application, the prediction vector from Equation 4.82 is:

$$\bar{x}_k^- = \bar{b}_{SAT,k} = \bar{q}_{SAT \leftarrow TOD,k} \otimes \bar{B}_{TOD,k} \tag{4.87}$$

Here, $\vec{B}_{TOD}$ is the inertial vector estimated by the IGRF model, while $q_{SAT \leftarrow TOD}$ is the estimated quaternion stating the attitude of the satellite. It needs to be noted that the accuracy of the reference model affects the accuracy of the filter output. As both reference models are further subject to the satellite's position, the accuracy of the orbit determination must be considered as well.

4.4.5 Kalman Filter for Angular Rate Estimation

Apart from the Kalman filter for angular rate measurements based on the assumption of constant acceleration during one control cycle presented earlier in Section 4.4.2, there is a different way to formulate the filter equations. This time, the state space equations are taken from the equation of motion for the satellite's dynamics (Equation 4.19). When used subsequently after the other Kalman filter, the effect is rather small, and moreover the quality of the filtering depends also on the knowledge of the applied torque (cf. Section 4.5.10). However, the filter equations are also useful to predict the angular rate, which is presented later in Section 4.4.9.

The Kalman filter equations are:

$$x = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^T \tag{4.88}$$

$$u = \begin{bmatrix}\tau_x & \tau_y & \tau_z\end{bmatrix}^T \tag{4.89}$$

$$f(x) = -\vec{\omega} \times (I \cdot \vec{\omega}) \tag{4.90}$$

$$g(u) = \frac{\tau}{I} \tag{4.91}$$

$$h(x) = x \tag{4.92}$$

The partial derivative of Equation 4.90 and Equation 4.92 gives

$$F = \begin{bmatrix} 0 & -\omega_z \cdot \frac{I_z - I_y}{I_x} & -\omega_y \cdot \frac{I_z - I_y}{I_x} \\ -\omega_z \cdot \frac{I_x - I_z}{I_y} & 0 & -\omega_x \cdot \frac{I_x - I_z}{I_y} \\ -\omega_y \cdot \frac{I_y - I_x}{I_z} & -\omega_x \cdot \frac{I_y - I_x}{I_z} & 0 \end{bmatrix} \tag{4.93}$$

and

$$H = E. \tag{4.94}$$

Moreover, the input matrix is

$$G = I^{-1}. \tag{4.95}$$

The state prediction uses Equation 4.90:

$$\bar{x}_k^- = \bar{x}_{k-1}^+ + \frac{1}{I}\left[\tau_k - \bar{x}_{k-1}^+ \times (I \cdot \bar{x}_{k-1}^+)\right] \cdot t_s \tag{4.96}$$

4.4.6 Attitude Estimation from Vector Observations

If an ADCS does not include a star tracker, the spacecraft's attitude may also be estimated from at least two vector observations. This results in an over-determined set of equations, which can be solved by a least-squares approach. In 1965, Wahba [83] first formulated the equation for the loss function, L, thereafter called Wahba's problem:

$$L(A) = \frac{1}{2}\sum_{i=1}^{n} w_i \left|\vec{b}_i - A \cdot \vec{r}_i\right| \tag{4.97}$$

where $\vec{b}_i$ is a vector in the body fixed frame and $\vec{r}_i$ the corresponding reference vector in the inertial system. All vectors are unit vectors and represent a direction (e. g. geomagnetic field or Sun direction). A is the spacecraft's orientation as DCM and w_i is the weight of the measurement i. All weights $i = 1..n$ for $n \in \mathbb{N}$ sum up to a value of one.

Note that Equation 4.97 uses the original symbols from [83], which are not to be confused with the usage of $\vec{b}$, $\vec{r}$ and A in the rest of this thesis.

Davenport found out that expressing the direction cosine matrix A as quaternion leads to an eigenvalue problem of the form [72]:

$$K \cdot q_{opt} = \lambda_{max} \cdot q_{opt} \tag{4.98}$$

with

$$K = \begin{bmatrix} S - E \cdot tr(B) & Z \\ Z^T & s \end{bmatrix} \tag{4.99}$$

$$B = \sum_{i=1}^{n} w_i \cdot \vec{b}_i \cdot \vec{r}_i^T \tag{4.100}$$

$$Z = \sum_{i=1}^{n} w_i \cdot \vec{b}_i \times \vec{r}_i \tag{4.101}$$

$$S = B + B^T \tag{4.102}$$

$$s = tr(B) \tag{4.103}$$

and $tr(B)$ as the trace of the matrix B. The optimal quaternion estimate, q_{opt}, is the eigenvector to the maximum eigenvalue, λ_{max}. There are several robust algorithms to solve an eigenvalue problem, yet all of them are computationally demanding.

A practical solution for Davenport's problem came with the observation of M. D. Shuster [66], who stated that for small values of the loss function L, the maximum eigenvalue is close to the sum of the weights:

$$\lambda_{max} \approx \lambda_0 = \sum_{i=1}^{n} w_i \tag{4.104}$$

Using λ_0 as an initial value, a few Newton-Raphson iterations of a characteristic equation formulated by Shuster provide a sufficient approximation for q_{opt} [66].

Presented in 1981, QUEST is the most frequently used algorithms for attitude determination from vector observations. Following alternative algorithms, such as FOAM [84] or ESOQ [85], all base on Equation 4.104, but use different characteristic equations for the Newton-Raphson iteration. A detailed review of the different methods is given by Markley and Mortari in [86].

For the particular case of exactly two measurement sources, there is a vast reduction of the computational effort for all mentioned methods and all of them satisfy the same condition for λ_{max} [66], [86]:

$$\lambda_{max} = \sqrt{a + b}. \tag{4.105}$$

with

$$a = w_1^2 + w_2^2 \tag{4.106}$$

$$b = 2 \cdot w_1 \cdot w_2 \cdot [(\vec{b}_1 \cdot \vec{b}_2) \cdot (\vec{r}_1 \cdot \vec{r}_2) + |\vec{b}_1 \times \vec{b}_2| \cdot |\vec{r}_1 \times \vec{r}_2|] \tag{4.107}$$

According to FOAM, the optimal attitude matrix can be determined directly [86]:

$$\begin{aligned} A_{opt} = \vec{b}_3 \cdot \vec{r}_3^T &+ \frac{a_1}{\lambda_{max}} \cdot [\vec{b}_1 \cdot \vec{r}_1^T + (\vec{b}_1 \times \vec{b}_3) \cdot (\vec{r}_1 \times \vec{r}_3)^T] \\ &+ \frac{a_2}{\lambda_{max}} \cdot [\vec{b}_2 \cdot \vec{r}_2^T + (\vec{b}_2 \times \vec{b}_3) \cdot (\vec{r}_2 \times \vec{r}_3)^T] \end{aligned} \tag{4.108}$$

with

$$\vec{b}_3 = \frac{\vec{b}_1 \times \vec{b}_2}{|\vec{b}_1 \times \vec{b}_2|} \tag{4.109}$$

$$\vec{r}_3 = \frac{\vec{r}_1 \times \vec{r}_2}{|\vec{r}_1 \times \vec{r}_2|} \tag{4.110}$$

This method is not applicable if $\vec{b}_1$ and $\vec{b}_2$ are parallel or antiparallel.

4.4.7 Kalman Filter for Attitude Estimation

Kalman filters are widely used for attitude determination of spacecrafts. Although all attitude representations (cf. Section 4.1) may be used in the filter equations in principle, the implementations may vary due to their different characteristics. The use of a DCM brings a high calculation effort and further requires the consideration of the redundant matrix elements. Euler angles are also inefficient due to the trigonometric functions and also bring singularities (due to gimbal lock) to deal with [87]. Here, quaternions provide a calculation-efficient alternative, since their use does not involve trigonometric functions and even linear state equations can be used.

Nevertheless, the definition of the unit quaternion entails the dependency of the four quaternion elements. This leads to a singular covariance which might result in the filter's instability due to rounding errors. This problem is discussed in detail in [87] and different solutions are proposed. Here, the

multiplicative extension of the Kalman filter is formulated, as done previously in [88], [89].

This approach does not consider the deviation of the attitude as arithmetic difference of the real and estimated quaternion but by their product which represents the error quaternion. A second state quantity of the filter is the bias of the angular rate measurement, $\vec{\omega}_B$, which makes the filter especially useful when rate sensors with high bias such as MEMS gyroscopes are used. The state vector is defined as follows:

$$x = \begin{bmatrix} \delta q & \omega_B \end{bmatrix} \tag{4.111}$$

with

$$\delta q = \hat{q}^{-1} \otimes q \tag{4.112}$$

The angular rate sensor is modelled by

$$\omega_m = \omega - \omega_B - \eta_1 \tag{4.113}$$

$$\dot{\omega}_B = \eta_2 \tag{4.114}$$

with the uncorrelated white noise processes η_1 and η_2.

The system matrix is

$$F = \begin{bmatrix} [\omega_m - \omega_B]_\times^T & -\frac{1}{2}E \\ 0 & 0 \end{bmatrix} \tag{4.115}$$

and the input matrix

$$G = \begin{bmatrix} -\frac{1}{2}E & 0 \\ 0 & 0 \end{bmatrix} \tag{4.116}$$

The matrix of the process noise is [90]

$$Q = \begin{bmatrix} (\eta_1^2 \cdot t_s + \frac{1}{3} \cdot \eta_2^2 \cdot t_s^3) \cdot E & -\frac{1}{2}\eta_2^2 \cdot t_s^2 \cdot E \\ -\frac{1}{2}\eta_2^2 \cdot t_s^2 \cdot E & \eta_2^2 \cdot t_s \cdot E \end{bmatrix} \tag{4.117}$$

Here, $[v]_\times$ is the skew-symmetric matrix which is equivalent to the cross product with a vector $\vec{v}$, and t_s is the sampling rate. The prediction is achieved by Equation 4.120:

$$\hat{q}_k^- = \Theta \hat{q}_{k-1}^+ \tag{4.118}$$

$$\omega_{B_k}^- = \omega_{B_{k-1}}^+ \tag{4.119}$$

$$P_k^- = \Phi_k P_{k-1}^+ \Phi_k^T + GQG^T \tag{4.120}$$

The closed-loop solution for the prediction of the quaternion via the matrix Θ is presented in [72] and, amongst others, used in [30]:

$$q(t_{n+1}) = e^{\frac{1}{2}\Omega_n t_s} q(t_n) \tag{4.121}$$

Ω_n is the well-known angular rate matrix (Equation 4.24) at time t_n. Assuming a constant angular rate over t_s, the calculation effort may be reduced by using the average value of the control cycle, $\bar{\Omega}$.

$$\bar{\Omega} = \begin{bmatrix} 0 & -\bar{\omega}_x & -\bar{\omega}_y & -\bar{\omega}_z \\ \bar{\omega}_x & 0 & \bar{\omega}_z & -\bar{\omega}_y \\ \bar{\omega}_y & -\bar{\omega}_z & 0 & \bar{\omega}_x \\ \bar{\omega}_z & \bar{\omega}_y & -\bar{\omega}_x & 0 \end{bmatrix} \tag{4.122}$$

and

$$\bar{\omega} = \sqrt{\bar{\omega}_x^2 + \bar{\omega}_y^2 + \bar{\omega}_z^2} \tag{4.123}$$

follow according to [72]:

$$q(t_{n+1}) = \left[\cos\left(\frac{\bar{\omega} \cdot t_s}{2} \right) \cdot E + \frac{1}{\bar{\omega}} \cdot \sin\left(\frac{\bar{\omega} \cdot t_s}{2} \right) \cdot \bar{\Omega}_n \right] \cdot q(t_n) \tag{4.124}$$

The error of the approximation in Equation 4.121 is in the order of t_s^3, and it vanishes completely if $\vec{\omega}_n$ and $\dot{\vec{\omega}}_n$ are parallel. Therefore, the approximation corresponds to the closed loop solution even if the angular rate changes over t_s, provided that the axis of rotation is constant [30].

The estimation update is performed differently for the quaternion and the attitude bias:

$$\hat{q}_k^+ = \hat{q}_k^- \otimes \begin{bmatrix} \sqrt{1 - |\delta\hat{q}^+|^2} \\ \delta\hat{q}^+ \end{bmatrix} \tag{4.125}$$

$$\omega_{B_k}^+ = \omega_{B_{k-1}}^+ + \Delta\omega_B^+ \tag{4.126}$$

$$\tag{4.127}$$

The formulation of the observation matrix, H_k, and the innovation, i_k, depends on the measured quantities and hence on the used sensors. Moreover, different approaches to calculate the innovation may be used. Here, the innovation is formed via the cross-product of a vector quantity measured in the satellite

frame and its inertial reference. This approach is also used in [91], [92] and brings the advantage that both direction and the norm of the error are considered. The innovation is given by [92]:

$$i_k = \begin{bmatrix} \vec{b} \times \vec{B} \\ \vec{s} \times \vec{S} \end{bmatrix} \tag{4.128}$$

and the observation matrix is

$$H_k = 2 \cdot \begin{bmatrix} H_b & 0 \\ H_s & 0 \end{bmatrix} \tag{4.129}$$

with

$$H_b = \begin{bmatrix} b_z^2 + b_y^2 & -b_x \cdot b_y & -b_x \cdot b_z \\ -b_x \cdot b_y & b_x^2 + b_z^2 & -b_y \cdot b_z \\ -b_x \cdot b_z & -b_y \cdot b_z & b_x^2 + b_y^2 \end{bmatrix} \tag{4.130}$$

$$H_s = \begin{bmatrix} s_z^2 + s_y^2 & -s_x \cdot s_y & -s_x \cdot s_z \\ -s_x \cdot s_y & s_x^2 + s_z^2 & -s_y \cdot s_z \\ -s_x \cdot s_z & -s_y \cdot s_z & s_x^2 + s_y^2 \end{bmatrix} \tag{4.131}$$

The vectors $\vec{b}$ and $\vec{s}$ are the body-fixed measurements, while $\vec{B}$ and $\vec{S}$ are their inertial reference vectors. While there are no measurements available for the Sun vector during an eclipse, the lower half of H_k and i_k, respectively, are set to zero and the attitude is only determinated from geomagnetic field measurements. The filter may be initialized using the attitude estimation method described in Section 4.4.6 to shorten the time for the filter to converge.

Star tracker measurements may be used directly in most cases, since modern star trackers already provide full lost-in-space solutions in every cycle with high accuracy. Nevertheless, the use of the Kalman filter is still helpful to determine the angular rate measurement bias. Here, the innovation is analog to Equation 4.112 and the observation matrix is simply:

$$H = \begin{bmatrix} E & 0 \end{bmatrix} \tag{4.132}$$

4.4.8 Averaging Quaternions

If the ADCS has more than one star tracker, the measurements of all available sensors must be averaged. In doing so, one has to consider different sources

of disturbance. On the one hand, the relative orientation of the star trackers may vary over time, due to displacements of the structure induced by thermal influences. These variations may result in an error with a magnitude of several ten arc seconds, even if the structural design is done carefully [93]. On the other hand, a star tracker has anisotropic noise distribution. In fact, the measurements along the boreside axis of the sensor are 3–5 times less accurate than across this axis [30]. Therefore, averaging the star tracker quaternions without taking the anisotropy into account may even worsen the attitude knowledge [93]. Cheng, Markley, Crassidis, *et al.* [94] presents an algorithm for the weighted averaging of quaternions via a loss function. This method further leads, analogously to the Q-method [86], to the numerically intense solution of an eigenvalue problem, which may be approximated using the QUEST method presented in Section 4.4.6. However, these methods induce a singularity of the solution for certain rotations, whose circumvention brings more calculation load.

As an alternative, an algorithm formulated by Romans [95] is less computationally intense and is applicable for small deviations of the star tracker measurements. Therefore, this algorithm was also used by Yoon for the TET-1 attitude control [30] and is adapted for TUBiX20. A detailed description of the algorithm can be found in [30], [95].

4.4.9 State Quantity Prediction

As described in Section 3.3.2, the state prediction provides the forecast for the state quantities referring to the end of the control cycle. There are only a few state quantities which depend on the spacecraft's dynamics and kinematics, namely attitude and all body-fixed vector information. Apart from the spacecraft's position, all other state quantities may either be considered constant over the control cycle (such as actuator outputs) or are inertial quantities such as magnetic field and Sun position. The latter may hence be calculated as by the same environment models as in the state prediction phase, while using the control cycle end as input time. The prediction of the position depends on the source of the estimated value; if an orbit propagation algorithm is used, the same rule as for the environment models applies and the same algorithm may be used. If a GPS receiver provides high-accuracy position information, the use of a coarse orbit propagation algorithm may be much

more inaccurate than just using the GPS value, hence a high-precision orbit model must be utilized. In the following, the prediction of the spacecraft's attitude and vector information will be presented. Due to its complexity, a prediction of the position is omitted.

As will be seen in the following sections, the prediction respective step from the according Kalman filter may be used for state prediction.

Attitude Prediction

The equation to predict the attitude for one control cycle, Equation 4.124, was already presented in Section 4.4.7:

$$q(t_{k+1}) = \left[\cos\left(\frac{\bar{\omega}\cdot t_s}{2}\right)\cdot E + \frac{1}{\bar{\omega}}\cdot\sin\left(\frac{\bar{\omega}\cdot t_s}{2}\right)\cdot\bar{\Omega}_n\right]\cdot q(t_k) \tag{4.133}$$

Angular Rate Prediction

Once again, the prediction equation can be obtained from the Kalman filter algorithm. In this case, Equation 4.96 states:

$$\vec{\omega}(t_{k+1}) = \vec{\omega}(t_k) + \frac{1}{I}\left[\vec{\tau}_k - \vec{\omega}(t_k)\times(I\cdot\vec{\omega}(t_k))\right]\cdot t_s \tag{4.134}$$

Body-fixed Vector Prediction

Finally, the prediction of body-fixed vector quantities may be derived from the prediction step of the linear Kalman filter presented in Section 4.4.3. The differential equation for the shift of a body-fixed vector is according to Equation 4.71:

$$\dot{\vec{v}}_k = -\vec{\omega}_k \times \vec{v}_k \tag{4.135}$$

which makes the vector quantity prediction:

$$\vec{v}_{k+1} = \vec{v}_k + \dot{\vec{v}}_k \cdot t_s \tag{4.136}$$

4.5 Attitude Control

This section presents different techniques for attitude control.

4.5.1 State Control Error

The attitude error is the relative deviation of the actual orientation regarding the target orientation as expressed by Sidi [73]:

$$q_E = q_S^{-1} \otimes q_T, \tag{4.137}$$

with q_E, q_T und q_S as attitude error, target attitude and actual attitude, respectively. Since all control laws are formulated with regard to the MOI coordinate system, Equation 4.137 is rewritten as

$$\begin{aligned} q_{MOI \leftarrow TAR} &= q_{MOI \leftarrow TOD} \otimes q_{TOD \leftarrow TAR} \\ &= q_{TOD \leftarrow MOI}^{-1} \otimes q_{TOD \leftarrow TAR} \end{aligned} \tag{4.138}$$

Together with the angular rate error, $\vec{\omega}_{MOI}^{TAR}$, the state error is obtained as

$$S_E = \begin{bmatrix} q_{MOI \leftarrow TAR} \\ \vec{\omega}_{MOI}^{TAR} \end{bmatrix} \tag{4.139}$$

and the target state is

$$q_{MOI \leftarrow TAR} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.140}$$

$$\vec{\omega}_{MOI}^{TAR} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \tag{4.141}$$

This is also the working point for the linearization of the dynamics equation (cf. Equation 4.30), which is an important constraint for the stability of the control loop. Based on the linearization of the quaternion for the working point, the error state from Equation 4.139 is further reduced to:

$$S_E \cong \begin{bmatrix} \widetilde{q}_{MOI \leftarrow TAR} \\ \vec{\omega}_{MOI}^{TAR} \end{bmatrix} \tag{4.142}$$

However, the spacecraft shall be aligned with the target attitude according to its geometry, e. g. the boreside axis of a camera or the cone of an antenna,

and hence the relative orientation of the SAT frame to the MOI frame must be taken into account for the calculation of the desired orientation. The target state of the spacecraft depends on the attitude control mode. In case of a desired nadir orientation, the SAT coordinate frame must be aligned with the orbit-fixed LVLH coordinate frame. Consequently, the relative orientation of the TAR frame to the LVLH frame is the same as the orientation of the SAT frame to the MOI frame:

$$q_{MOI \leftarrow TAR} = q_{SAT \leftarrow LVLH} \tag{4.143}$$

To obtain the attitude error for pointing towards nadir, the orientation of the SAT frame with regard to the TOD frame provided by the state quantity determination (cf. Table 3.1) is multiplied with the orientation of the LVLH frame with regard to the TOD frame:

$$q_{LVLH \leftarrow SAT} = q_{LVLH \leftarrow TOD} \otimes q_{TOD \leftarrow SAT} \tag{4.144}$$

The latter may be derived from Equation B.11 in the appendix, while further using Equation 4.17 and Table 4.1.

For the second part of the state error, the angular rate, however, the MOI frame is used. Here, the control error results from the angular rate provided by the state determination, $\vec{\omega}_{SAT}$, and furthermore the relative rotation of the LVLH frame with regard to the SAT frame:

$$\vec{\omega}_{MOI}^{LVLH} = q_{MOI \leftarrow SAT} \odot \tag{4.145}$$

with

$$\omega_0 = \frac{2\pi}{T} \tag{4.146}$$

$$q_{MOI \leftarrow LVLH} = q_{MOI \leftarrow SAT} \otimes q_{SAT \leftarrow LVLH} \tag{4.147}$$

and T as the orbital period. Once again, the operator $\otimes$ stands for the multiplication of quaternions (Equation 4.7), while $\odot$ stands for the transformation of a vector via a quaternion (Equation 4.10).

4.5.2 Quaternion Feedback Control

A straightforward control law for attitude control is given by Sidi [73] as

$$\vec{\tau}_c = 2 \cdot K_P \cdot q_{E_s} \cdot q_{E_v} + K_D \cdot \vec{\omega}_E \tag{4.148}$$

In compliance with Equation 4.137, q_E is the attitude error and $\vec{\omega}_E$ the angular rate error. K_P and K_D are the proportional gain and the derivative gain, respectively.

Using the notation of Equation 4.142 and considering the sign of the quaternion notation (cf. Equation 4.29), the control law can be rephrased as

$$\vec{\tau}_c = 2 \cdot K_P \cdot \widetilde{q}_{MOI \leftarrow TAR_s} \cdot \widetilde{q}_{MOI \leftarrow TAR_v} - K_D \cdot \omega_{MOI}^{TAR} \tag{4.149}$$

4.5.3 State Space Control

This section will introduce a state space control law for the linearized state error formulation given in Equation 4.142. The optimal state space controller considers time response as well as energy consumption by minimizing the criterion J:

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{4.150}$$

The control feedback matrix, K, is determined by solving the Ricatti equation, S [96]:

$$SA + A^T S - SBR^{-1}B^T S + Q = 0 \tag{4.151}$$

and then obtaining

$$K = R^{-1}B^T S \tag{4.152}$$

Given the state error from Equation 4.142, the control torque, $\vec{\tau}_c$, may then be calculated as

$$\vec{\tau}_c \quad = \quad -K \cdot \begin{bmatrix} \widetilde{q}_{MOI \leftarrow TAR} \\ \omega_{MOI}^{TAR} \end{bmatrix} \tag{4.153}$$

4.5.4 Detumbling with a B-Dot Algorithm

The most famous algorithm to detumble a spacecraft, e. g. after separation, is the so-called B-Dot controller. Here, the derivative of the body-fixed magnetic field is taken as feedback for the controller input:

$$\vec{m}_{MOI} = -\frac{K}{\|\vec{b}\|^2} \cdot \dot{\vec{b}}_{MOI} \tag{4.154}$$

As a value for the controller gain, K, the formula

$$K = 4 \cdot n \cdot (1 + \sin(i)) \cdot I_{min} \tag{4.155}$$

produced good results in the functional tests on an air bearing testbed for the TechnoSat satellite (cf. Section 5.1). Here, i is the inclination of the orbit, I_{min} is the minimum value of the principal moments of inertia, and n is the mean motion of the orbit in rad/s.

4.5.5 Detumbling with a Cross-Product Algorithm

A more elaborate approach is given in Equation 4.156 [97]. Since it takes both the magnetic field and the angular rate of the satellite into account, the performance is better than the B-Dot controller, but in turn it needs two different types of input.

$$\vec{m}_{MOI} = -\frac{K}{\|\vec{b}\|^2} \cdot \vec{b}_{MOI} \times \vec{\omega}_{MOI} \tag{4.156}$$

The controller gain, K, is the same as the one from the B-Dot controller.

4.5.6 Attitude Control Using Magnetic Torquers

The relation between the control torque, $\vec{\tau}_c$, magnetic dipole, $\vec{m}$, and the geomagnetic field, $\vec{b}$, is given in Equation 4.157 [72], [73].

$$\vec{\tau}_c = \vec{m} \times \vec{b} \tag{4.157}$$

It is clearly observable, that a torque can only be generated which is perpendicular to the body-fixed geomagnetic field, $\vec{\tau}_\perp$, which leaves an error $\vec{\tau}_E$:

$$\vec{\tau}_E = \vec{\tau}_c - \vec{\tau}_\perp \tag{4.158}$$

To adapt the quaternion feedback control law from Section 4.5.2 to this context, it is combined with the cross-product detumbling algorithm from Section 4.5.5. Similar to Equation 4.149, the control magnetic dipole consists

of two parts for the state error components from Section 4.5.1: attitude error and angular rate error.

$$\vec{m}_{MOI} = \frac{1}{\|\vec{b}\|^2} \left[K_P \cdot m_q - K_D \cdot m_\omega \right] \tag{4.159}$$

The angular rate error part is similar to the detumbling control law from the previous section, except that the angular rate error is used and not the actual angular rate.

$$\vec{m}_\omega = \vec{b}_{MOI} \times \vec{\omega}_{MOI}^{TAR} \tag{4.160}$$

The attitude error part combines the cross-product approach with the quaternion feedback control from Section 4.5.2, where the feedback is calculated from the scalar part and the vector part of the attitude error.

$$\vec{m}_q = \vec{b}_{MOI} \times [\widetilde{q}_{MOI \leftarrow TAR_s} \cdot \widetilde{q}_{MOI \leftarrow TAR_v}] \tag{4.161}$$

Once again, K is used for the controller gain. To be able to tune the control performance, the factor α is introduced as

$$K_P = \alpha \cdot K \tag{4.162}$$

$$K_D = K \tag{4.163}$$

4.5.7 Attitude Control Using Reaction Wheels

Unlike the magnetic torquers, reaction wheels can apply a torque directly around the desired axis, which is in this case the rotational axis of the motor. Due to the angular momentum conservation (cf. Equation 4.20), a torque applied around the axis of one reaction wheel will induce a rotation of the spacecraft in opposite direction. Hence, for the case of three reaction wheels whose axes of rotation correspond to the axes of the control torque, the torque is

$$\vec{\tau}_{RW} = -\vec{\tau}_c \tag{4.164}$$

To convert this torque into the required angular acceleration, $\dot{\vec{\Omega}}$, for each reaction wheel, the wheel's inertia is required:

$$\dot{\vec{\Omega}}_{RW} = \frac{1}{J} \vec{\tau}_{RW} \tag{4.165}$$

with J as the reaction wheel's inertia and $\vec{\Omega}$ as their angular rate.

4.5.8 Reaction Wheel Torque Distribution

To achieve single-failure tolerance, four reaction wheels are used for redundancy. Usually, the four wheels are arranged in a tetrahedron configuration, so that if one wheel fails, the remaining three can still create a torque in any arbitrary direction.

To distribute the control torque from the three-dimensional controller output to the input torque for the four reaction wheels, Equation 4.166 is used:

$$\begin{bmatrix} \tau_0 \\ \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = A_{wR} \cdot \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \tag{4.166}$$

The inverse transformation, from the wheels' angular momentum array to the three-dimensional angular momentum, is given by:

$$\vec{H} = A_w \cdot \begin{bmatrix} H_0 \\ H_1 \\ H_2 \\ H_3 \end{bmatrix} \tag{4.167}$$

The matrix A_w may be derived from the reaction wheels' mounting orientations; a detailed example is given in [73]. The reverse transformation matrix, A_{wR}, is slightly more complicated to determine, since Equation 4.166 is an overdetermined system. Following [72], [73], A_{wR} may be calculated from A_w as

$$A_{wR} = A_w^T \left(A_w \cdot A_w^T \right)^{-1} \tag{4.168}$$

4.5.9 Reaction Wheel Desaturation

The previous section describes the use of reaction wheels to transfer the spacecraft's angular momentum to the reaction wheels and vice-versa by the use of the angular momentum conservation. However, due to internal and external disturbances, the overall angular momentum, i. e. the sum of spacecraft angular momentum and reaction wheels, changes. Since the environmental disturbance torques (cf. Section 4.3) need to be compensated,

the angular momentum and hence the rotational rate of the reaction wheels slowly increases over time. Once the maximal rate is reached, the disturbances may no longer be compensated and the control loop becomes unstable.

Therefore, the reaction wheels must be desaturated. A common way to achieve this is to use magnetic torquers or magnetic coils. The formula to determine the desaturation torque is, similar to Equation 4.157, also based on the cross product.

$$\vec{m}_D = -\frac{K}{\|\vec{b}\|^2} \cdot \vec{b}_{MOI} \times \vec{H}_{MOI} \tag{4.169}$$

Once again, K is the same as in Equation 4.155. Due to the cross-product relation between torque, magnetic dipole and magnetic field, the torque error already described in Equation 4.158 is also existent here. However, in this case it only affects the performance of the reaction wheels' desaturation: if the torque generated by the magnetic torquers is subtracted from the torque settings for the reaction wheels, reaction wheels and magnetic torquers practically share the control torque.

$$\vec{\tau}_{RW} = -(\vec{\tau}_{RW} - \vec{m}_D \times \vec{b}_{MOI}) \tag{4.170}$$

4.5.10 Applied Torque

Following Section 4.5.6 to Section 4.5.9, the torque which is applied to the spacecraft results from the torque from the reaction wheel system (RWS) and the torque from the magnetic torquer system (MTS):

$$\vec{\tau} = \vec{\tau}_{RWS} + \vec{\tau}_{MTS} \tag{4.171}$$

$$= -A_{wR} \cdot J \cdot \dot{\vec{\Omega}} + \vec{m} \times \vec{B} \tag{4.172}$$

This information is useful as input in the dynamics equation for the Kalman filter presented in Section 4.4.5 and hence the angular rate prediction from Section 4.4.9. Both inputs, $\vec{\Omega}$ and $\vec{m}$, may be obtained from the telemetry of the reaction wheels or derived from current measurements for the magnetic torquers, respectively.

5 Practical Realization of TUBiX20 Attitude Control System Configurations

This chapter presents the practical realization of the concept for the design and verification of a flexible attitude control system described in Chapter 3 using the example of the first two missions based on the TUBiX20 nanosatellite platform, TechnoSat and TUBIN.

5.1 Basic Attitude Control for the TechnoSat Mission

TechnoSat is the first satellite to be based on the TUBiX20 nanosatellite platform and was briefly introduced in Section 2.3.4. The mission has the objectives of [55]

- in-orbit demonstration of novel nanosatellite components
- the development and in-orbit demonstration of the adaptive nanosatellite platform TUBiX20

The first step of describing the realization of the TechnoSat ADCS will be to review the requirements defined at the beginning of phase B of the ECSS life cycle. Since the focus is on requirements which affect the derivation of a platform configuration according to the concept investigated in this thesis, irrelevant requirements, e. g. regarding procurement or documentation, are omitted. The remaining requirements for the TechnoSat ADCS are:

TE-ACS-03 The ADCS design shall be single failure tolerant.

TE-ACS-05 The ADCS must damp an initial angular rate of at least $8\,^{\circ}/\mathrm{s}$.

TE-ACS-06 During STELLA experiments, the angular rate must not exceed $0.3\,^{\circ}/\mathrm{s}$.

TE-ACS-07 During STELLA experiments, STELLA must be pointed into free space.

TE-ACS-08 During FDA experiments, the ADCS must determine the relative angular rate with an accuracy of at least 0.1 °/s.

TE-ACS-09 During FDA experiments, the ADCS must not perform active control.

TE-ACS-10 During reaction wheel system experiments, the ADCS must not perform active control.

TE-ACS-11 During HISPICO experiments, the ground station must be within the 3 dB-lobe of the patch antenna for at least 90 s.

TE-ACS-07 and TE-ACS-11 define requirements regarding the attitude of the satellite during payload experiments. However, the quantification needs to be derived from these requirements, which is performed in Appendix C. It is shown that both payloads may be operated when the satellite is pointing towards nadir. The required pointing accuracy for

- STELLA experiments is 27.9°;
- HISPICO experiments is 34.1°.

5.1.1 Distributed Attitude Control System

Since the required pointing accuracy for all payload experiments are rather moderate, high-accuracy attitude determination and control is not necessary. Therefore, the author chose a cost- and energy-efficient concept using vector observations for attitude estimation and magnetic torquers for attitude control. Similar to a large majority of pico- and nanosatellites [8], TechnoSat uses magnetometers, Sun sensors and MEMS gyroscopes as sensors. To meet the high requirements regarding the knowledge of the angular rate for the verification of the fluid-dynamic actuator and the reaction wheel system, a fiber optic rate sensor system composed of three sensors complements the basic sensor set.

The Sun sensor design is based on a position sensitive device (PSD) and was utilized for all satellites of the BEESAT satellite series from 2009 to 2016 (cf. Table A.1 in the appendix). The layout was adapted for TechnoSat to be put on a common PCB including also the MEMS gyroscopes and the magnetic field sensor ICs. In the following, these boards are referred to as integrated sensor board (ISB). Each board contains two sensors of each type. Consequently, the Sun sensor system is single failure tolerant with two sensors in every spatial direction, while the magnetic field sensors and gyroscopes have 12 redundant units. Since their power consumption is not more than several milli ampere, they may even be operated continuously. This multiple redundancy offers different benefits:

- even simple techniques such as averaging improve the measurement accuracy and reduce noise;
- placed on the outer panels, the magnetic field sensors are less subject to the satellite's residual dipole;
- with two magnetic field sensors in every spatial direction, the remaining influence of the residual dipole may be estimated;
- cross-checks of the various measurements allow to identify invalid measurements (cf. Section 4.4.1).

On the other hand, placing the sensors on the outer structure comes with a disadvantage: the temperature gradient is much higher than inside the satellite. While the magnetic field sensors' internal temperature drift correction is sufficient, the MEMS gyroscopes' measurements drift up to several degrees per second. However, a simple two-point calibration leads to a satisfactory compensation. The only communication bus supported by all sensors is I^2C, which is therefore used on the integrated sensor boards. The bus is also redundant for single-failure tolerance. Similarly, the redundant drivers for the magnetic torquers also connect via I^2C.

Figure 5.1 shows the TechnoSat ADCS hardware, following the concept of a distributed system from Section 3.1.1. The integrated sensor boards and the magnetic torquers are connected directly to the ADCS node. The fiber optic rate sensors (FORs) are connected to a separate hardware node which contains all hardware-specific circuitry to trigger the measurements and to receive the data. Therefore, removing or exchanging this particular sensor does not interfere with the rest of the ADCS. Moreover, the microcontroller

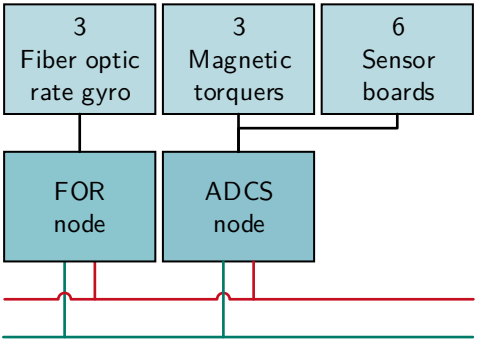

Figure 5.1: Distributed ADCS Hardware for TechnoSat

on this interface node may already pre-process the high-frequent sensor data (cf. Section 4.4.2) and only transmit a filtered solution to the core application hosted on the ADCS node. Consequently, the network of software application also separates the different device types, as shown in Figure 5.2. The ISB manager application operates the magnetic field sensors, gyroscopes and the Sun sensor system, while the magnetic torquer system (MTS) is controlled via a separate application. The core application for state determination and control is the only hardware-independent building block.

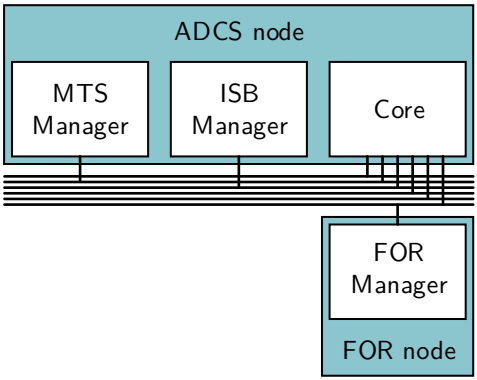

Figure 5.2: Distributed ADCS Software for TechnoSat

5.1.2 State Estimation Module Assembly

Based on the sensors incorporated for TechnoSat, adequate techniques for state estimation were selected. As described in Chapter 3, all algorithms are implemented and verified in self-contained modules which are used for simulation models as well as for the flight software. This section describes the assembly of these modules for the most relevant state quantities. Different principles introduced in Section 3.2 are picked up and explained in detail. The concrete algorithms were presented in Chapter 4 and will be referenced individually. As described in Section 3.3, the state determination and control core framework invokes the module functions and interacts with the device manager applications via the communications middleware.

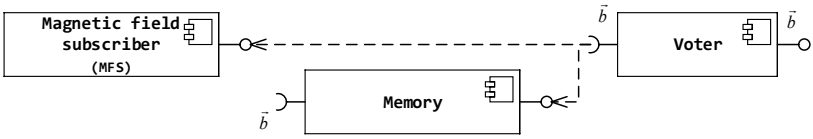

Figure 5.3: Magnetic Field Measurements (UML)

The estimation of the magnetic field is straightforward. A subscriber buffers the incoming magnetic field sensor (MFS) measurements from the ISB manager application and provides them via the magnetic field vector interface, $\vec{b}$ (cf. Table 3.1), similar to the example for the abstraction from a diverse hardware setup given in Figure 3.5. Additionally, the predicted magnetic field values from the last control cycle serve as a backup in case no data was received. A priority voter selects the measurements from the subscriber if available and falls back on the predicted data otherwise.

The second state quantity discussed is the body-fixed direction vector to the Sun, which is in this work represented by the symbol $\vec{s}$. Here, a similar combination consisting of a subscriber for Sun sensor system (SSS) measurements, the previously predicted value and a voter is used. However, a Kalman filter is added to improve the data quality: the noise is reduced and furthermore the filter compensates temporary data loss due to its internal predictor. There are different reasons why it makes sense to use a Kalman filter for the Sun vector measurements. Since the inertial Sun position changes very slowly compared to one control cycle, a rather simple linear Kalman filter such as presented

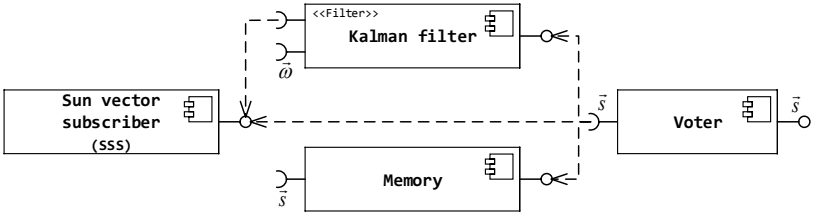

Figure 5.4: Kalman Filter for Sun Vector Measurements (UML)

in Section 4.4.3 may be used. This is not feasible for the magnetometers, since the inertial magnetic field changes drastically within one orbit. Here, an extended Kalman filter, as discussed in Section 4.4.4, is necessary. However, there is no source to obtain a reference value for the prediction step, since the attitude is estimated from vector observations – and hence the magnetic field.

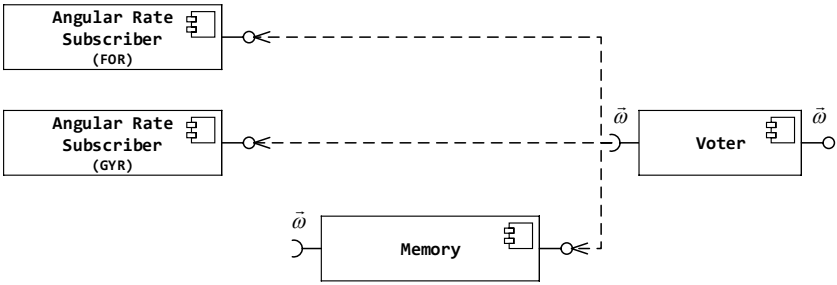

Figure 5.5: Angular Rate Measurements from Multiple Sensors (UML)

There are two sensors which measure the satellite's angular rate: the fiber optic rate sensor (FOR) system and the MEMS gyroscopes (GYRs). Therefore, when compared to the magnetic field estimation and the Sun vector estimation, there are two different subscribers, as shown in Figure 5.5. Due to the distributed ADCS architecture, the measurement data is transmitted from two different hardware nodes. However, this does not have any implications on this level and is in fact not known to the core application. In case the fiber optic rate sensor system is switched off during operations, e. g. to save energy, the voter falls back on the gyroscopes automatically. If the whole node is removed from

the ADCS within a different mission, only the subscriber needs to be removed from the assembly. On the other hand, additional modules for angular rate estimation may be added. For example, such a module may compare the vector measurements with the values from the previous control cycle and thus calculate its rotation angle and axis. This is not necessary for TechnoSat, since even for a loss or malfunction of the fiber optic rate sensor system, 12 redundant gyroscopes still remain. For missions with different sensor sets, however, this might be a useful extension.

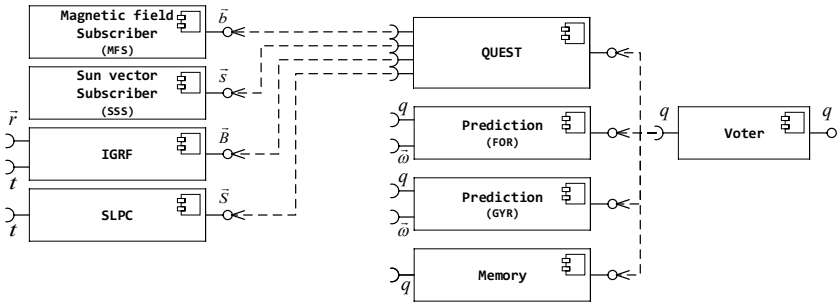

Figure 5.6: Absolute and Relative Attitude Estimation (UML)

The last state quantity addressed for TechnoSat is the attitude, primarily estimated from vector observations. Obviously, there is no Sun vector available during an eclipse and hence this method is not continuously applicable. A fallback solution is the prediction of the attitude from the angular rate based on the satellite kinematics. The mathematical description for this technique is given in Section 4.4.9. Since the prediction provides a relative estimate, its accuracy highly depends on the quality of the initial value. Due to the sensor noise, the integration of the angular rate measurements results in a drift of the attitude, which is generally referred to as angular random walk (ARW). Since there are two different sensor types, there are also two prediction modules. Both are instances of the same module, hence there is no additional implementation effort. Once again, the two stages for fiber optic rate sensor system and MEMS gyroscopes are independent from each other. Figure 5.6 shows all attitude estimation modules used for TechnoSat. The QUEST method is presented in Section 4.4.6.

Due to the unified interfaces, replacing or re-arranging the individual modules is straightforward. The benefits of this flexibility will be discussed for the incremental transformation of the basic configurations into a high-accuracy transformation. Section 5.2 describes how attitude determination and control with a star tracker and reaction wheels is already achievable for TechnoSat once these payloads have been incorporated into the ADCS. The complete TUBIN configuration is addressed in Section 5.3.

5.1.3 Attitude Control Modes

Based on the requirements and the concept of mode derivation presented in Section 3.3.4, the attitude control modes are derived from different characteristics. As mentioned before, only two TechnoSat payloads demand active attitude control. Firstly, the S-band transmitter shall be pointed towards nadir with at least 34.1°. For the star tracker, either inertial pointing is required or nadir pointing with an accuracy of at least 27.9°. Moreover, a damping mode is needed for initial angular rate damping after separation as well as a passive mode to not interfere with experiments using the fluid-dynamic actuator and the reaction wheel system.

Consequently, four independent modes are defined. In the following, the mode description is formulated according to the five abstraction criteria from Section 3.3.4.

- The Suspend Mode (SPM) fulfills TE-ACS-09 and TE-ACS-10, as *no active* control is performed.
- The Detumbling Mode (DTM) performs *active* control of the spacecraft's *angular rate* to *detumble* the satellite, using a *cross-product control law* and *magnetic torquers* and hence fulfills TE-ACS-05.
- To fulfill TE-ACS-07 and TE-ACS-11, the Nadir Pointing Mode (NPM) performs *active* control of the spacecraft's *attitude* to achieve *nadir pointing* using *a cross-product control law* and *magnetic torquers*.
- As an alternative for TE-ACS-07, the Inertial Pointing Mode (IPM) performs *active* control of the spacecraft's *attitude* to achieve *inertial pointing* using *a cross-product control law* and *magnetic torquers*.

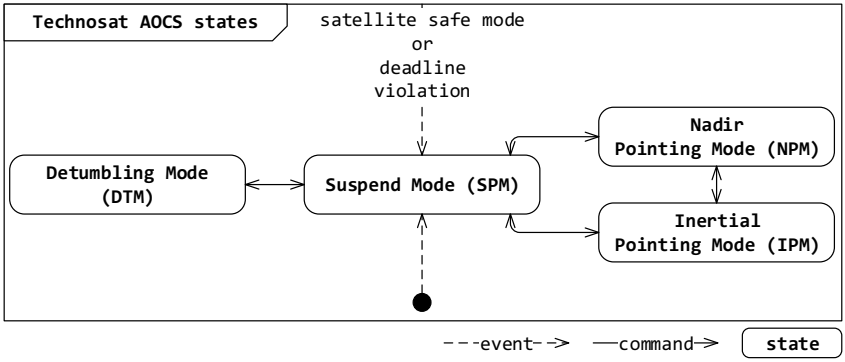

Figure 5.7: ADCS Modes for TechnoSat (UML)

Figure 5.7 shows the TechnoSat control modes as UML state chart.

To conclude the description of the basic attitude control configuration for the TechnoSat Mission, the core module assembly to perform state control is described. While the idea has already been discussed in detail in Section 3.3.4 and an example for the (in-orbit) configuration was given, Figure 5.8 shows the concrete realization. Since the only actuators for TechnoSat are the magnetic torquers, the magnetic dipole is the only controller setting and therefore only one middleware topic is necessary. The core assembly publishes the settings and they are received by the MTS manager application. For the Suspend Mode, no settings need to be sent. The remaining three modes are implemented by an assembly of four library modules. The Detumbling Mode uses a damping algorithm based on a cross-product control law, which is described in Section 4.5.5. The pointing modes, Nadir Pointing Mode and Inertial Pointing Mode, both rely on the same control law and only differ in the target orientation. Therefore, reconnecting the input for the *pointing cross product* module facilitates a mode transition.

Similar to the state estimation module assembly, the extension of the attitude control modes will be shown in the subsequent sections.

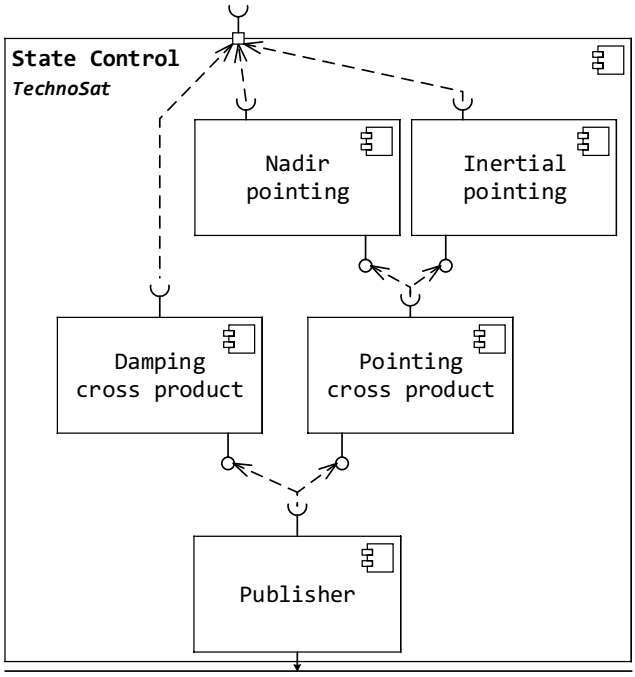

Figure 5.8: State Control Assembly for TechnoSat (UML)

5.2 Enhanced Attitude Control with TechnoSat Payloads

The second mission objective for TechnoSat is the development and in-orbit demonstration of the adaptive nanosatellite platform TUBiX20 (cf. Section 5.1). This reflects that TechnoSat is designed to be a pathfinder mission for the subsequent TUBIN mission. Apart from the design and verification of the modular platform architecture in general, the TechnoSat ADCS design can go one step further here: two of the components for the TUBIN attitude determination and control system are qualified within the TechnoSat mission, a star tracker and a reaction wheel system. Incorporating these components into the control loop allows the realization of the ADCS for TUBIN to a large extent. However, since they have the status of a payload at first, the ADCS configuration presented in the previous section does not rely on them. In fact, they even define their own requirements regarding the attitude control performance of the platform (cf. TE-ACS-06, TE-ACS-07 and TE-ACS-10). After the star tracker and the reaction wheel system have been successfully qualified, however, they may be used in the ADCS. Therefore, high-accuracy attitude control is already achievable within the extended operational phase of TechnoSat. It is therefore desirable to consider both the star tracker and the reaction wheel system in the design from an early stage, while not complicating the realization of the basic functionality. This is one example for the scalability of the concept investigated in this thesis, which allows the seamless transition from basic to high-accuracy attitude control.

In this section, the individual extensions regarding the distributed system hardware, the state estimation module assembly and the attitude control modes are discussed. In all diagrams, the newly introduced elements will be depicted in red to highlight the modification. In turn, all elements depicted in black remain unchanged, showing that the changes to the flight-proven software are kept at a minimum.

5.2.1 Distributed Attitude Control System

The distributed hardware for TechnoSat from Figure 5.1 is extended by the star tracker and the reaction wheel system, which is shown in the bottom half of Figure 5.9. All components are compatible with the system's redundant CAN bus. Since the star tracker does not fully comply with the redundant

power bus, a small adapter circuitry based on basic ICs is necessary. However, no extra microcontroller is needed here.

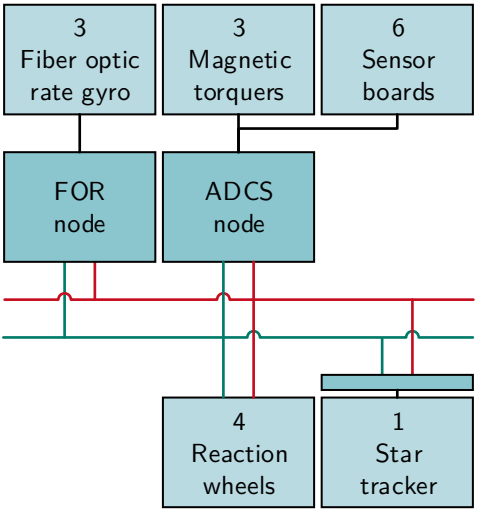

Figure 5.9: Distributed ADCS Hardware for TechnoSat incl. Payloads

In the software domain, each of the new devices is represented by an individual application, following the principle of functional separation. Since star tracker and reaction wheels both comply with the central data bus, the may be run on any node of the system. The ADCS node was chosen as host here, since the FOR node is dedicated to the fiber optic rate sensors and may be switched off during operation if these are not required.

Due to the aligned modular architecture in both domains, no modifications of existing hardware or device manager software are required. Evidently, the ADCS core application needs to be updated, since the presence of a star tracker offers new possibilities for the estimation of several state quantities. Moreover, state control modes using the reaction wheels need to be inserted.

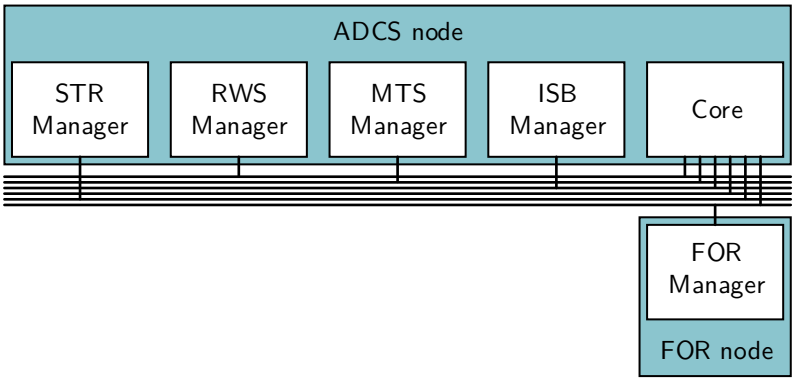

Figure 5.10: Distributed ADCS Software for TechnoSat incl. Payloads

5.2.2 State Estimation Module Assembly

The availability of star tracker measurements does not only benefit the estimation of the attitude directly. Apart from the fact that star trackers usually also provide drift-free measurements for the angular rate, the high-precision attitude data may be used to derive or filter other state quantities.

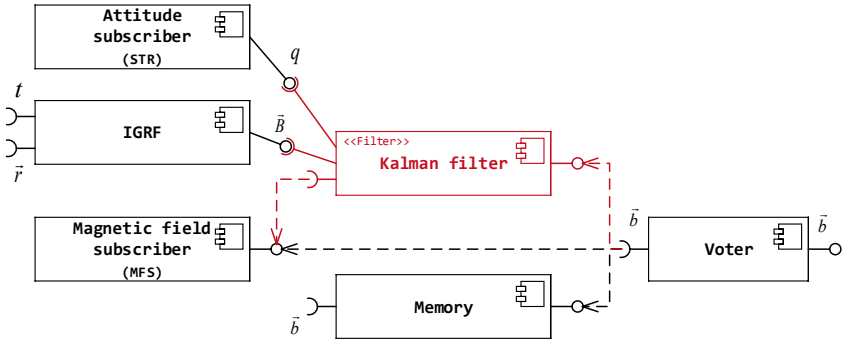

Figure 5.11: Kalman Filter for Magnetic Field Measurements (UML)

A first example for the application of high-accuracy attitude measurements to estimate other state quantities is given in Figure 5.11. Transforming the

inertial magnetic field data estimated by a model such as IGRF into body-fixed coordinates provides a direct reference for the magnetic field sensors, which may then be used in the prediction step of an extended Kalman filter. As has been pointed out in Section 5.1.2, this was not feasible before. The new Kalman filter module is depicted in red in Figure 5.11. However, due to local perturbations and the satellite's residual dipole, the IGRF estimates may deviate from the actual magnetic field. The new filter module, depicted in red, is connected as first input of the voter and hence has the highest priority. Implementing a general *filter* interface, it may be activated or deactivated at runtime.

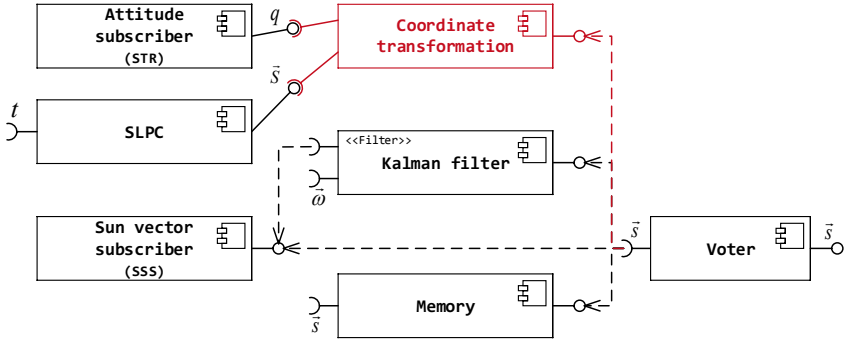

Figure 5.12: Sun Vector Estimation from Attitude and Sun Model (UML)

Similar to the magnetic field, a high-precision attitude is also beneficial for the estimation of the direction to the Sun. Here, the advantage is even more direct: unlike the magnetic field, the direction of the sunlight is not subject to any perturbations, and therefore transforming the inertial Sun position into the body-fixed coordinate frame provides an unobstructed, highly accurate Sun vector. The Sun model used for TUBiX20 is presented in [5] and has an accuracy of 0.1–1 %. As can be seen from Figure 5.12, this extension is simply achieved by adding a coordinate transformation (red) and connecting it as first input of the priority voter.

As has been mentioned before, most star trackers also provide angular rate measurements. These are integrated into the module assembly similar to all other angular rate sensors. In fact, the same source code subscriber classes may be used for all the different sensors here, since their data types are

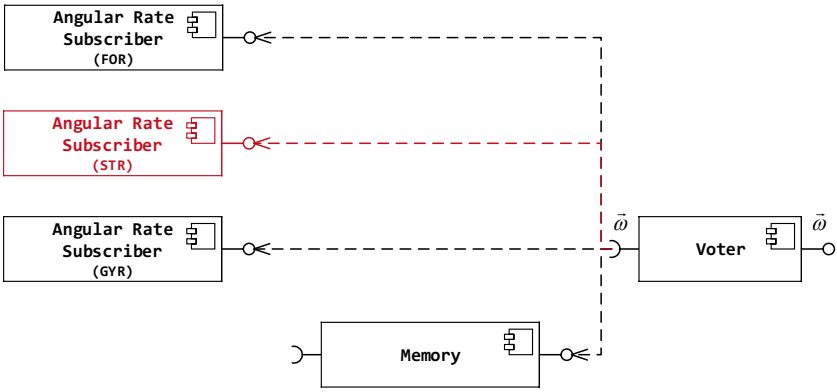

Figure 5.13: Inserting Measurement Sources (UML)

identical and all require the same interface, $\vec{\omega}$. Regarding the priorities, the star tracker is placed between the highly accurate fiber optic rate sensors and the MEMS gyroscopes, as shown in Figure 5.13. As a further point of extension, the drift-free star tracker measurements may be used to calibrate the bias of the other angular rate sensors. Due to the modular architecture, an additional calibration module may be inserted easily.

Finally, the primary state quantity to be measured by the star tracker is addressed: the attitude. Obviously, its measurements are considered to be the most accurate attitude source in the system and hence have the highest priority. Once again, a subscriber module buffers the measurement data sent from the device manager application via the network. It is put in front of the voter input list according to its priority for the selection. However, the presence of the star tracker yields another benefit for the attitude estimation: the attitude prediction via angular rate measurements from the fiber optic rate sensors is also more accurate when the star tracker measurements are taken as an initial value, since the error of this prediction module combines the absolute star tracker error and the relative error of the prediction due to the angular random walk and bias of the fiber optic rate sensors. Given this new situation, the FOR prediction module swaps positions with the QUEST module. While the latter provides an absolute solution, it is however based on the coarse vector observations from the Sun vector and magnetic field.

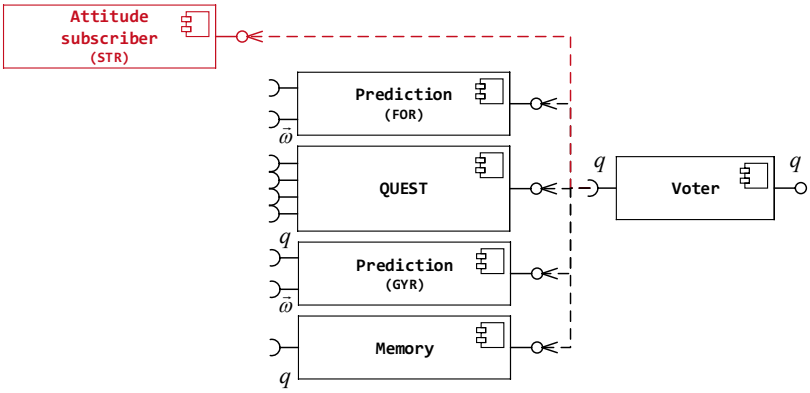

Figure 5.14: Rearranging Priorities of State Quantity Providers (UML)

Figure 5.14 shows the rearranging of the attitude providers resulting from the incorporation of the star tracker. Rearranging the order of the voter inputs does not imply any changes in the voter implementation or the control flow of the core application, but is simply a slight modification within the core assembly.

5.2.3 Attitude Control Modes

Incorporating the reaction wheels as the second TechnoSat payload into the platform ADCS operations, the attitude control modes may be upgraded as well. Once again it is the objective here to pave the way for the TUBIN mission. Here, the reaction wheel system allows more accuracy and agility than the purely magnetic actuation of the basic TechnoSat platform, and hence the pointing modes from the TechnoSat basic ADCS described in Section 5.1.3, Inertial Pointing Mode and Nadir Pointing Mode, may be complemented with high-accuracy modes using the reaction wheels. Unlike the magnetic torquers, the reaction wheel system may apply a torque in any spacial direction required, and hence the cross product control law is replaced by a more suitable technique (cf. Section 4.5.3).

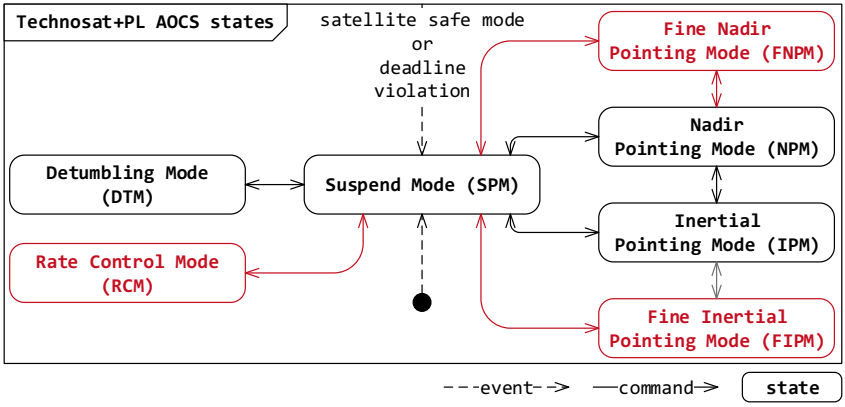

Figure 5.15: ADCS Modes for TechnoSat incl. Payloads (UML)

The updated state chart for TechnoSat and integrated payloads is shown in Figure 5.15. The descriptions for these new modes are, according to the general criteria for mode derivation, as follows:

- The Fine Accuracy Nadir Pointing Mode (FNPM) performs *active* control of the spacecraft's *attitude* to achieve *nadir pointing* using *a state space control law* and *reaction wheels*.
- The Fine Accuracy Inertial Pointing Mode (FIPM) performs *active* control of the spacecraft's *attitude* to achieve *inertial pointing* using *a state space control law* and *reaction wheels*.
- As a means for a gradual and straightforward verification of the reaction wheels, the Rate Control Mode (RCM) performs *active* control of the spacecraft's *angular rate* using *a state space control law* and *reaction wheels*.

Following the functional abstraction for the mode definitions from Section 3.3.4, implementing the new control modes into the core module assembly is realized by adding only the required modules into the state control assembly, which is shown in Figure 5.16. The state space control law and the distribution of the control torque to the reaction wheels are integrated connecting to the interfaces of the existing modules (depicted in red), and hence the algorithms to calculate

the target attitude are now used for both the coarse and the fine pointing modes. Therefore, not only are the same algorithms to determine the control error used, but also the same instances of their realization. This minimizes the effort to integrate the new control modes and also saves resources in terms of the microcontroller's memory usage and software image size.

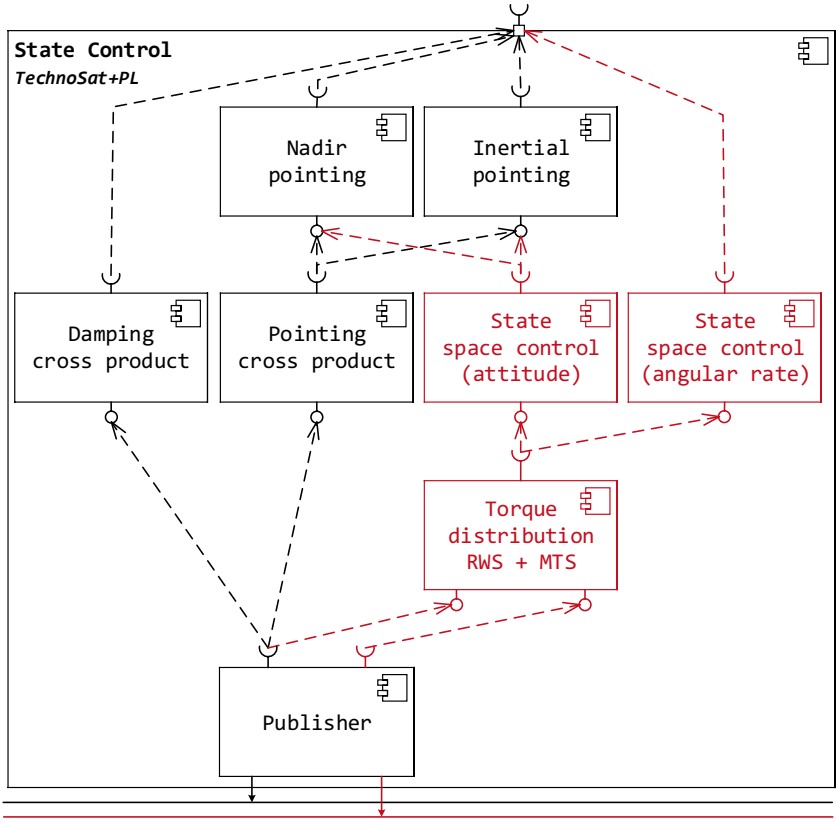

Figure 5.16: State Control Assembly for TechnoSat incl. Payloads (UML)

To achieve single failure tolerance, the reaction wheel system consists of four reaction wheels in a tetrahedron configuration. Hence the three-dimensional control torque needs to be distributed to the four actuators. Furthermore, the

accumulating angular momentum due to disturbance torques must be dumped via the magnetic torquers. Both techniques are implemented as features of the module for torque distribution (cf. Figure 5.16). Their mathematical background is described in detail in Section 4.5.7. To submit the settings for the reaction wheels to their associated device manager application, a new topic is added.

5.3 High-Accuracy Attitude Control for the TUBIN Mission

The TU Berlin Infrared Nanosatellite (TUBIN) will be the first mission to implement a high-accuracy configuration of the TUBiX20 platform ADCS. The mission's objectives are [54]

- the development and in-orbit demonstration of the use of a commercial infrared mirobolometer for wildfire remote sensing on a nanosatellite;
- the demonstration of the capabilities of the TUBiX20 platform to support demanding Earth remote sensing missions.

Similar to the description of the ADCS realization for the TechnoSat mission in Section 5.1, the relevant requirements for the TUBIN are listed below:

TB-ACS-05 For payload calibration, a slew rate of at least 1.0 °/s must be possible.

TB-ACS-06 During payload experiments, the attitude knowledge must be at least 11.5 arcmin.

TB-ACS-07 During payload experiments, the pointing stability must be at least 35.3 arcmin/s.

TB-ACS-08 During payload experiments, the position knowledge must be at least 2 km.

TB-ACS-09 During data downlink, the pointing accuracy to the ground station must be at least 20°.

TB-ACS-10 During payload experiments, the nadir pointing accuracy must be at least 200.3 arcmin.

TB-ACS-15 The ADCS must damp an initial angular rate of at least 8 °/s.

TB-ACS-16 For payload calibration, the payload must be pointed into free space.

The numerical values for attitude knowledge, pointing accuracy and jitter were derived from the requirements formulated by the payload and the mission characteristics such as the orbit. A detailed exposition is given in Appendix C. In the following sections, all modifications regarding additional components, software modules and control modes are discussed in detail.

5.3.1 Distributed Attitude Control System

As stated before, the star tracker and the reaction wheel system for TUBIN have been space qualified with the TechnoSat mission and their incorporation into the TUBiX20 ADCS was discussed previously in Section 5.2. This section describes the integration of three new components which are introduced with TUBIN: a GPS receiver, a fluxgate magnetometer and a second star tracker. The second star tracker was added to increase the reliability of the overall system and furthermore increase the availability of star tracker measurements: if one unit is blinded by straylight from the Earth or Sun, the other unit is still available. Furthermore, their mounting orientation with a 90° angle between their boreside axes is ideal to average the measurement for increased accuracy. The model selected is, however, provided by a different manufacturer and has space heritage from several different missions.

Figure 5.17 shows the distributed ADCS hardware for TUBIN. Following the TUBiX20 architecture, the GPS receiver is connected via an interface node, since its electrical interface does not comply with the redundant TUBiX20 power and data bus. The fluxgate magnetometer (FLX) is an analog sensor, and hence an ADC is required, which in turn requires data sampling and processing. As described in Section 5.2.1, the star tracker already carried with TechnoSat needs a small adapter circuitry to connect to the redundant power

bus. The second star tracker, however, does not comply with neither the power nor the data bus, and hence another TUBiX20 interface node is used here. Consequently, the circuitry for both star trackers is then realized on the same node and hence the measurements from both may be preprocessed on this star tracker node. As pointed out in Section 2.3, the TUBiX20 hardware architecture is based on reference ICs and circuitry. This keeps the design and verification effort needed to introduce the three new interface nodes at a minimum.

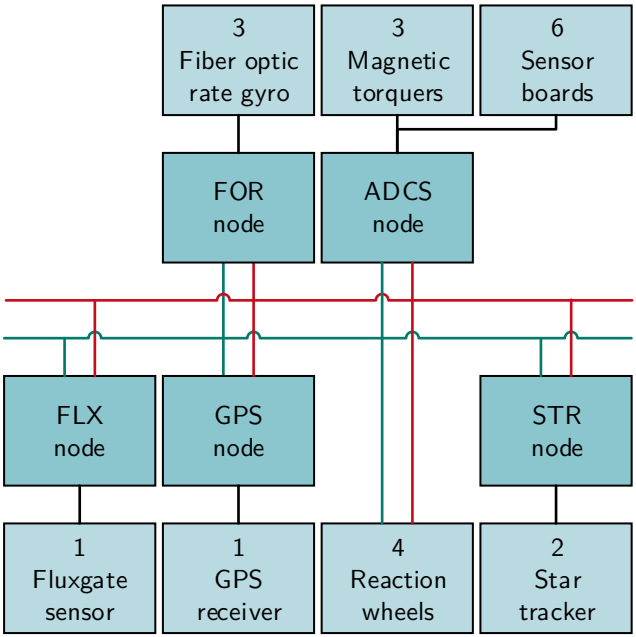

Figure 5.17: Distributed ADCS Hardware for TUBIN

To integrate the additional sensors in the TUBiX20 ADCS software, the network of building blocks is extended as shown in Figure 5.18. Since the GPS and FLX are connected via interface nodes, an associated device manager runs on these nodes' microcontroller, which facilitates the communication with the hardware to obtain the measurements. Since the GPS receiver also

provides highly accurate time information, the GPS node may also serve as a clock reference for the complete satellite by emitting a pulse per second signal via the TUBiX20 hardware interface.

As mentioned before, both star trackers are connected to the same node and their measurements are combined there to increase accuracy. From the perspective of the attitude determination and control system core application, the star trackers are then treated as one integrated unit and it is irrelevant that both components are in fact from different manufacturers. As opposed to the TechnoSat configuration with the integrated payloads shown in Figure 5.9, the star tracker application is now run on the dedicated star tracker (STR) node. Due to the building block approach, moving the application from one node to another does not interfere with the rest of the software. Figure 5.18 shows the distributed software network for the TUBIN ADCS.

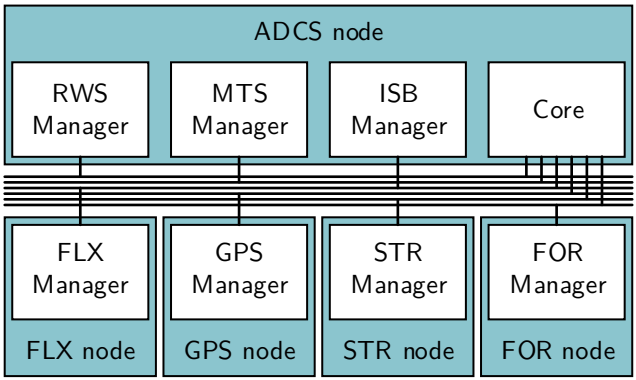

Figure 5.18: Distributed ADCS Software for TUBIN

The averaging of the two different star tracker solutions is shown in Figure 5.19. This extension is part of the star tracker device manager and is simply performed by adding a sink for the additional star tracker and a subsequent module to average the star trackers' attitude quaternions (cf. Section 4.4.8). Although both units are from different manufacturers, the driver layer already abstracts from all hardware dependencies and the same interface is used for both providers.

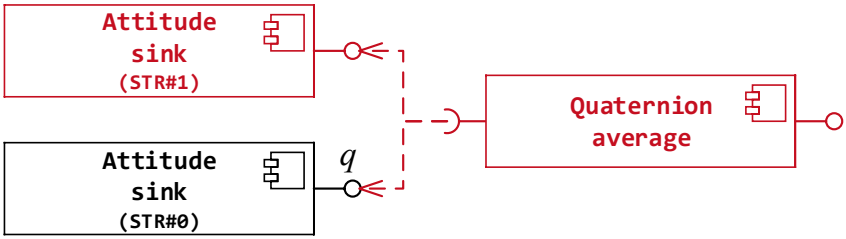

Figure 5.19: Attitude Quaternion Averaging (UML)

5.3.2 State Estimation Module Assembly

With the GPS receiver introduced during the TUBIN mission, highly accurate position, velocity and time information is available. Once again, this also affects the quality of other state estimation modules due to the increased quality of their inputs. Figure 5.20 shows how the GPS position data is added to the state estimation module assembly.

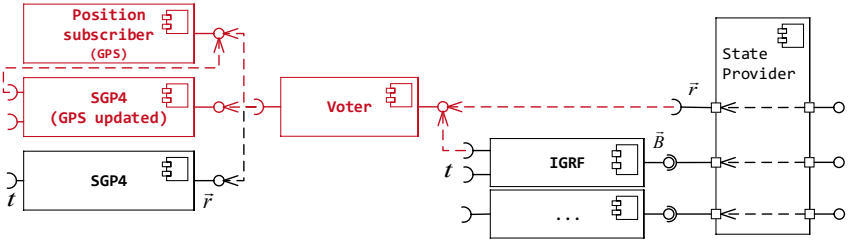

Figure 5.20: High-Accuracy Position Estimation (UML)

Similar to all other data sent from a device manager application, a subscriber buffers the incoming data and implements the quantity provider interface according to Table 3.1, in this case with the symbol $\vec{r}$ for the position in the TOD coordinate system. Apart from using the GPS measurements directly, the data may also be used to update the orbit parameters for the SGP4 orbit propagation model. Since continuously operating the GPS requires a lot of power, this is an energy-efficient and yet sufficiently precise solution. A technique to increase the accuracy of the SGP4 propagation using intermittent

GPS readings is presented in [98]. As there are now several choices for position information, a newly introduced priority voter provides the most accurate solution available. Connecting all modules which have the spacecraft position as input parameter to the voter, they automatically benefit from the increased accuracy. In Figure 5.20, this is indicated by the IGRF geomagnetic field model, but also the state control performance is increased, e. g. due to the more accurate determination of the required attitude for nadir pointing. Since the position voter now serves as input for the state facade, this does not come with any required modifications.

The integration of the fluxgate magnetometer has in fact already been described earlier, although in a more general form. Figure 3.5 shows the abstraction from a diverse hardware setup with the example of two magnetic field sensors which are connected via different hardware nodes, which is exactly the situation here. Hence, the fluxgate magnetometer measurements are processed using the same data flow as the magnetometer ICs on the integrated sensor boards: while the driver abstracts from the hardware-specific functionality, the modules for calibration and data transmission to the ADCS core application are re-used.

5.3.3 Attitude Control Modes

Reviewing the TUBIN requirements listed earlier in terms of attitude control, all required modes are already present in the extended TechnoSat implementation from Figure 5.15: the Fine Accuracy Nadir Pointing Mode supports the operation of all remote sensing payloads as well as data downlink via the S-band transmitter, while the Fine Accuracy Inertial Pointing Mode may be used to face the infrared mirobolometer for calibration. However, to increase the contact time for the data downlink and facilitate the payloads to take multiple images of the same spot from different perspectives during one flyover, a Target Pointing Mode (TPM) is added.

Figure 5.21 shows the attitude control state chart for the TUBIN mission. For simplification, the Target Pointing Mode is only accessible when the satellite is pre-aligned towards nadir. Expressed through the mode derivation criteria, "the Target Pointing Mode performs *active* control of the spacecraft's *attitude* to achieve *target pointing* using *a state space control law* and *reaction wheels*".

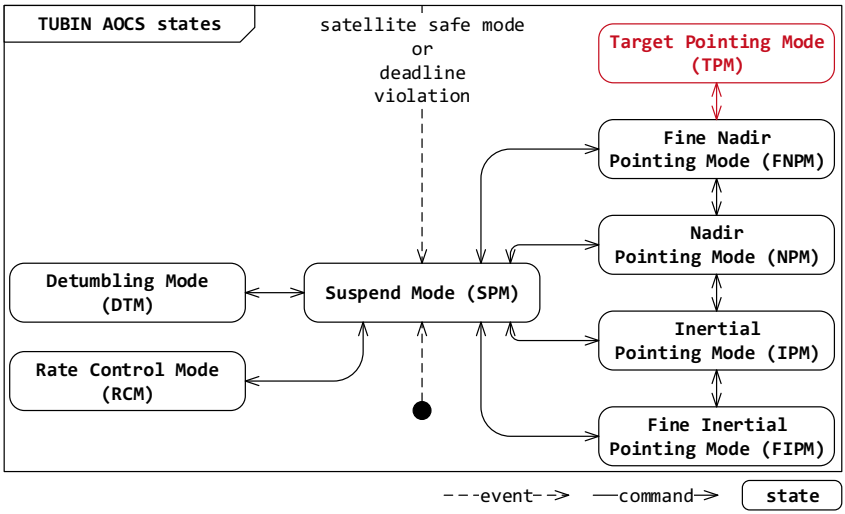

Figure 5.21: ADCS Modes for TUBIN (UML)

When compared to the definition of the Fine Accuracy Nadir Pointing Mode from Section 5.2.3, only the required orientation differs. Therefore, the integration of the Target Pointing Mode into the TUBiX20 state control module assembly is straightforward, as shown in Figure 5.22. Only the module to determine the control error is added. For the calculation of the control torque and its distribution to the actuators, the same modules as for the Fine Accuracy Nadir Pointing Mode and Fine Accuracy Inertial Pointing Mode are used. The core application framework then re-routes the connections according to the current attitude control mode at runtime.

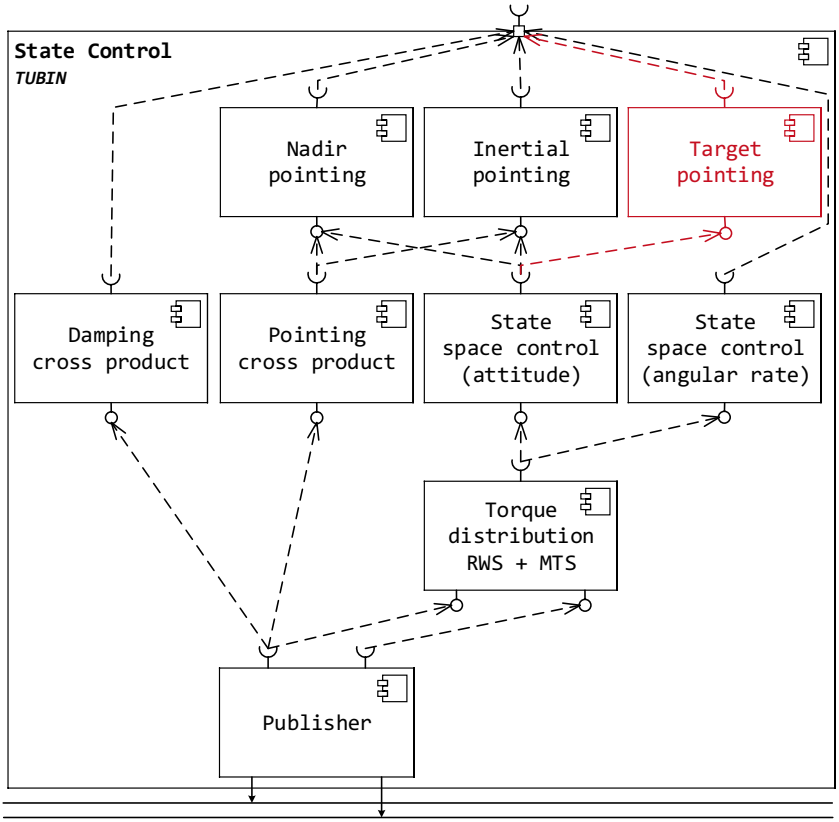

Figure 5.22: State Control Assembly for TUBIN (UML)

6 Summary and Conclusion

The objective of this thesis is to investigate a new concept for the flexible design and verification of an attitude determination and control system (ADCS) for a nanosatellite platform. In this context, the term "flexible" characterizes the ADCS as "capable of adapting to new, different or changing requirements" [31].

The research starts with a brief retrospective of the history of nanosatellites, a term which is in fact not generally defined in the small satellite community, as the mass boundaries for categorization vary between different organizations. This thesis follows the categories defined by Brieß [12] and hence regards satellites with a mass of 4–20 kg as nanosatellites. The retrospective of the history of nanosatellites shows that some of the first satellites may be assigned to this category. Launched between 1958 and 1962, these pioneers of spaceflight were, however, rather demonstration missions and no nanosatellites were launched thereafter until the mid-nineties [8], when the technology miniaturization promised the realization of more complex mission scenarios. Accelerated by the advances in the consumer electronics and automotive sector after 2010, nanosatellites soon carried out scientific missions [21], and Earth observation constellations for commercial use were realized for the first time [23].

To meet the demand for satellites supporting a wide range of possible applications, their developers intend to reuse one flexible design for several missions and therefore develop platform concepts, which are then tailored to a specific scenario. Nevertheless, these missions' requirements regarding accuracy, dependability, but also economic efficiency and time to market are very demanding. Therefore, it is crucial to investigate new solutions for adaptable and yet high-performing satellites, as carried out in this thesis for the attitude determination and control system.

As a basis to investigate guidelines for the design of a flexible ADCS, observations of the satellite market and missions are recorded. The research carried

out by the author shows that most nanosatellite use a similar set of sensors and actuators, but due to the active COTS market, new products are released frequently while others are discontinued. The miniaturization of high-accuracy sensors and actuators further extends the potential of nanosatellites in general and attitude determination and control systems in particular, and an evolution of mission objectives towards constellations for science and Earth observation missions is clearly visible. With the growing interest of private companies, the requirements regarding reliability and availability increase, since a failure of the spacecraft comes with financial loss when either a service can no longer be provided or a data product cannot be generated. Universities have recently carried out missions of equally high aspiration, however mostly as technology demonstrations. The challenge here is to develop a complex spacecraft with only limited resources and the transfer of knowledge from one mission to another. Enabled by the advances in COTS microcontrollers, software has become a substantial part of the spacecraft development process. A flexible design for reliable and capable software is needed, since the complexity of on-board tasks is increasing drastically. Design techniques such as code generation and object-oriented design have been introduced into space science, however there is still a backlog when compared to other sectors like automotive or consumer electronics.

Following these observations, the author formulates design criteria to serve as a reference for the conceptual design of the flexible ADCS. Firstly, the design must allow the integration of updated or novel technology to respond to the active market and further allow to extend the platform's performance. To be able to support a diverging range of missions scenarios, scalability is required for optimization in regards to complexity, development effort, performance and resources by deriving a configuration which is reduced to only components and software modules which are really necessary. It is also suggested that the ADCS is developed and verified gradually, which means that putting the hardware into operation and integrating the software should be performed in small steps, e. g. by using simulation models to decouple its development from delays in other subsystems or procurements. Moreover, the feasibility of the concept may be proven at an early stage of the project and the progress of the development may be evaluated more precisely. To minimize cost for qualification and reduce the overall risk, modifications to verified hard and software should be kept at a minimum. As the life cycles of different missions realized with the same platform may overlap, it is important that the

configuration management supports concurrent mission design, i. e. different realizations need to be developed at the same time. Finally, the ability to reconfigure the software in-orbit allows the introduction of new or updated functionality, either to enhance performance or as a response to the loss of a hardware component.

The research of this thesis was carried out in the context of the development of TU Berlin's nanosatellite platform TUBiX20 and its first two missions, TechnoSat and TUBIN. To support diverging mission scenarios for future missions of the university, TUBiX20 targets modularity, reuse and dependability as main design goals. As a result, a generic, single-failure tolerant systems architecture has been elaborated. The development followed the hardware/software co-design approach which lead to a modular architecture with well-coordinated hardware and software and mutually matched interfaces. In addition to the classical breakdown of the satellite into subsystems, TUBiX20 is partitioned into four hierarchical levels, which enables extensive reuse and benefits a comprehensive and yet flexible FDIR strategy. The system is composed of a network of cold redundant computational nodes which perform the individual tasks of a device or subsystem. The nodes connect via a redundant CAN bus and communicate through the middleware of the operating system. To maximize flexibility and reuse, all nodes are based on the same generic architecture, comprising a set of reference components and circuitry for the hardware domain as well as a library of global building blocks. The author acted in the role of TUBiX20's systems engineer for software, and is therefore responsible for meeting the design goals (cf. Section 2.3) throughout the project.

Based on the analysis of design criteria for a flexible attitude determination and control system, the key design considerations for the TUBiX20 platform were continued for the investigations carried out in this thesis. The resulting concept implements the ADCS as a distributed system of devices complemented by a hardware-independent core application for state determination and control. In the distributed ADCS, each sensor or actuator type, such as a triaxial angular rate sensor system or a group of three magnetic torquers, is represented by a cold redundant computational node communicating via the middleware. By decoupling the devices from other nodes and using standardized hard- and software interfaces providing an abstraction from a sensor's individual output format on the lowest level possible, devices may be added, updated and removed without any interference to the rest of the system. This proved

to benefit the flexibility by fulfilling several of the elaborated criteria. Novel technology may be integrated at low design effort by introducing a new device node which then implements the standard interfaces. Moreover, this enables the scalability of the ADCS regarding diverging mission requirements while not implying any modifications to qualified hardware. By reducing the complexity of the system to the required minimum without adaptions except removing devices, the overall costs as well as development and verification time can be reduced. The approach comes, however, with the drawback that eventually more microcontrollers than the required minimum are used and hence the power consumption and requirements regarding volume are slightly increased.

For the second part of the concept, the author carries the approach of abstracting functionality from its realization into the state determination and control algorithms. Based on the technique of component-based software engineering, the system is partitioned into self-contained modules which implement unified interfaces. These interfaces specify the state quantity of an input or output, such as angular rate or control torque, but also its unit and coordinate system, complemented by a mathematical symbol for unambiguous documentation. Since only matching interfaces may be connected, errors in unit or coordinate transformations are prevented automatically, which proved to decrease the verification effort. Each encapsulating the implementation of an algorithm, while outwardly acting as a *black box*, the modules are collected in a library which is independent from all concrete missions realized.

The ADCS core application is structured into three parts: library, framework and assembly. While the library provides a collection of implementations, but not the context these are used in, the framework manages the control flow (e. g. TM/TC or mode management) of the application independently from the implementation. Library and framework are connected via the assembly, which defines which modules from the library are selected for a specific ADCS realization. Hence only the assembly is mission-specific, but library and framework are not. This in turn means that large parts of the application may be reused directly from one mission to another without modifications, which reduces the time for development and verification drastically. However, insights from one mission, e. g. required updates for implemented algorithms in the library or functional extensions in the core framework flow directly back into the code base for all missions. Expressed in terms of the flexibility criteria, concurrent mission design is supported while modifications to verified code are kept at a minimum. Moreover, re-routing the connections of module

inputs and outputs enables re-configuration in-orbit without a software upload required.

The concept investigated in this thesis allows the implementation of different ways of determining a state quantity in parallel without any dependencies among each other. This leads to a clear code structure and control flow, which improved the comprehension of the code and reduced the development and verification time significantly, as recognized during the realization of the TechnoSat and TUBIN projects. Moreover, the approach offers the comparison of different solutions for a state quantity during operations. Transferring techniques such as object-oriented design to the definition of attitude control modes resulted in a clear abstraction of general properties from specialized features and facilitated to reuse or even share common functionality, hence reducing code size but also development and verification time, and yet enabling flexible and fast evaluation of new techniques.

The design and verification process for the TUBiX20 ADCS has also been elaborated within this research. The approach targets the gradual development of the subsystem from a purely virtual satellite within a closed-loop simulation to the verification of the fully integrated system on an air-bearing testbed. It is inspired by the software development process of continuous integration (see glossary), and aligns with typical spacecraft development process models [69]. However it has been optimized in regards to synergy effects and cost efficiency, hence being especially suitable for the university environment. While all TUBiX20 projects generally follow the phases of the ECSS spacecraft life cycle, these phases were further partitioned into four consecutive steps for the ADCS: analysis, design, implementation and simulation. Following this approach for the realization of the TechnoSat and TUBIN ADCS resulted in a structured workflow and enabled the author to evaluate the development progress, which was especially challenging due to the missions' concurrent time schedule. In this context, the closed-loop simulations provided valuable data for the PDR and CDR reports. The intermediate integration step of software in the loop simulation model, which embedded the completely implemented flight software in a still virtual simulation environment, proved especially useful, since the ground support software could be used from a very early stage. Moreover, the development of the ADCS could be decoupled from the procurement or manufacturing of the hardware to a large extent, which assured that the ADCS was never a hindrance to the progress of the project. However, the additional integration steps came with a very low development

overhead and low extra costs for equipment and software tools due to the exploitation of synergy effects.

The theoretical background for the state estimation and control techniques is provided in a separate chapter. The description, however, follows the quantity definitions from the conceptual part including all mathematical symbols, coordinate systems and physical units. Therefore, this part provides the implementation details for all models previously treated as *black boxes*, and yet serves the reader as a general reference which is independent from the software concept investigated.

As the last part of this thesis, the concurrent realization of the investigated concept within the TechnoSat and TUBIN missions is discussed. Starting with the individual ADCS requirements, the scalability of the approach is demonstrated in three stages: from a coarse, but cost- and energy-efficient configuration to realize a technology demonstration mission with moderate requirements (TechnoSat) to a high-performance configuration to support Earth observation missions (TUBIN). For each configuration, the composition of the subsystem as a selection of hardware nodes and software modules is presented, highlighting the possibilities for extension, update or removal of functionality according to the individual needs.

At the time of the completion of this thesis, TechnoSat had just been launched into orbit and the TUBIN CDR had been prepared, hence one spacecraft had already passed the complete ADCS development process successfully, while the second one had already been designed in detail. In conclusion, transferring process models (continuous integration, test-driven development) and design techniques (object-oriented design, component-based software engineering, frameworks) from the IT sector enabled a gradual and concurrent development of two ADCS configurations with diverging mission objectives and requirements. Following the flexibility criteria formulated by the author, the concept supports rapid technology upgrades, which has been shown within the evolution of the platform from TechnoSat to TUBIN. The modularized design based on unified interfaces could be applied consistently for hardware and software, and hence the addition, upgrade or removal of devices as well as state determination and control strategies imply only minimum modifications to a verified system. Therefore, insights from the TechnoSat operations in-orbit flow back into the development of the TUBIN ADCS and hence enable an ongoing advancement of the platform. Synergy effects were successfully exploited between the two

missions, but also the different phases of the spacecrafts' life cycles, and the continuous integration of the ADCS from purely virtual to fully integrated lead to a large verification coverage from an early stage.

During the course of the development of the TUBiX20 platform, different aspects of its architecture, but also the design process have been published with the contribution of the author. Apart from general descriptions of the TechnoSat and TUBIN missions ([55] and [54], respectively), the generic systems architecture was first presented in [49], where its main design considerations including modularity, reuse and dependability, but also concurrent development of hardware and software were discussed. The capabilities of the platform to perform rapid technology updates by combining a modular architecture with agile development processes is addressed in [40], and [2] expands on the software development process. Special focus on the distributed software based on building blocks was placed in [53]. The FDIR concept of TUBiX20 was first presented in [50], while [51] targets the flexible integration of different payloads. Complementing the modular space segment of the platform, the ground support software enables "full access to the platform through all development stages of the project", which was presented in [44].

The concept for the flexible design and verification of the attitude determination and control system (ADCS) was presented by the author in two stages. Firstly, [63] addresses its design as a distributed system and the composition of the state determination and control library, framework, and core assembly. Subsequently, the flexible design process from a virtual satellite to a fully integrated and verified system was addressed by the author in [68].

Bibliography

[1] International Earth Rotation Service, *Explanatory Supplement to IERS Bulletins A and B*, 2012. [Online]. Available: `http://hpiers.obspm.fr/eoppc/bul/bulb/explanatory.html`.

[2] K. Gordon, A. Graf, and M. F. Barschke, "Practical Experience in using Continuous Integration within the Development of Nanosatellite Software", presented at the 10th IAA Symposium on Small Satellites for Earth Observation, Berlin, Germany, 2015.

[3] P. Duvall, S. Matyas, and A. Glover, *Continuous Integration – Improving Software Quality and Reducing Risk*, 1st edn. Addison-Wesley Professional, 2007.

[4] T. Noergaard, *Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers*, 2nd ed. Elsevier Science & Technology, 2012.

[5] O. Montenbruck and E. Gill, *Satellite Orbits: Models, Methods, and Applications*. Springer, 2000.

[6] E. Thébault, C. C. Finlay, C. D. Beggan, P. Alken, J. Aubert, O. Barrois, F. Bertrand, T. Bondar, A. Boness, L. Brocco, E. Canet, A. Chambodut, A. Chulliat, P. Coïsson, F. Civet, A. Du, A. Fournier, I. Fratter, N. Gillet, B. Hamilton, M. Hamoudi, G. Hulot, T. Jager, M. Korte, W. Kuang, X. Lalanne, B. Langlais, J.-M. Léger, V. Lesur, F. J. Lowes, S. Macmillan, M. Mandea, C. Manoj, S. Maus, N. Olsen, V. Petrov, V. Ridley, M. Rother, T. J. Sabaka, D. Saturnino, R. Schachtschneider, O. Sirol, A. Tangborn, A. Thomson, L. Tøffner-Clausen, P. Vigneron, I. Wardinski, and T. Zvereva, "International Geomagnetic Reference Field: the 12th generation", *Earth, Planets and Space*, vol. 67, no. 1, p. 79, 2015.

[7] F. R. Hoots and R. L. Roehrich, "Spacetrack Report No. 3", Department of Defense, Tech. Rep., 1988.

[8] J. Bouwmeester and J. Guo, "Survey of worldwide pico- and nanosatellite missions, distributions and subsystem technology", *Acta Astronautica*, vol. 67, pp. 854–862, 2010.

[9] M. Swartwout, "Twenty (plus) Years of University-Class Spacecraft: A Review of What was, An Understanding of What Is and a Look at What Should Be Next", in *Proceedings of the AIAA/USU Conference on Small Satellites*, Logan, USA, 2006.

[10] M. Cho, M. Hirokazu, and F. Graziani, "Introduction to Lean Satellite and ISO Standard for Lean Satellite", presented at the Recent Advances in Space Technologies (RAST), Istanbul, Turkey, 2015, pp. 789–792.

[11] E. Kulu, *The NANOSAT FP7 Project – Nanosatellites database*, Accessed: 2017-07-04, 2017. [Online]. Available: `http://www.nanosats.eu/`.

[12] K. Brieß, "Small Satellite Programmatics of the Technische Universität Berlin", presented at the 11th IAA Symposium on Small Satellites for Earth Observation, Berlin, Germany, 2017.

[13] National Space Science Data Center, *NSSDC Master Catalog*, 2016. [Online]. Available: `http://nssdc.gsfc.nasa.gov`.

[14] M. Davidoff, *The Radio Amateur's Satellite Handbook*. American Radio Relay League, 1998.

[15] R. Schulte, "TUBSAT-N, A Global Communication Satellite System, based on Nanosatellites", presented at the Small Satellites Systems and Services Symposium, Antibes, France, 1998.

[16] U. Renner, "Flight Results of TUBSAT-A", presented at the 42nd Congress of the International Astronautical Federation, Paris, France, 1991.

[17] S. Müncheberg, M. Krischke, and N. Lemke, "Nanosatellites and Micro Systems Technology – Capabilities, Limitations and Applications", *Acta Astronautica*, vol. 39, no. 9-12, pp. 799–808, 1996.

[18] *CubeSat Design Specification*, The CubeSat Program, 2015.

[19] C. I. Underwood, G. Richardson, and J. Savignol, "In-orbit results from the SNAP1 nanosatellite and its future potential", *Philosophical Transactions of The Royal Society A Mathematical Physical and Engineering Sciences*, pp. 199–203, 2003.

[20] *6U CubeSat Design Specification*, The CubeSat Program, 2016.

[21] W. Weiss, S. Rucinski, A. Moffat, A. Schwarzenberg-Czerny, O. Koudelka, C. Grant, R. Zee, R. Kuschnig, S. Mochnacki, J. Matthews, P. Orleanski, A. Pamyatnykh, A. Pigulski, J. Alves, M. Guedel, G. Handler, G. Wade, K. Zwintz, CCD, and P. T. Teams, "BRITE-Constellation: nanosatellites for precision photometry of bright stars", *Publications of the Astronomical Society of the Pacific*, vol. 126, no. 940, pp. 573–585, 2014.

[22] G. Bonin, N. Orr, S. Armitage, N. Roth, B. Risi, and R. E. Zee, "The CAN-X-4-5 Mission: Achieving Precise Formation Flight at the Nanosatellite Scale", presented at the 64th International Astronautical Congress, Beijing, China, 2013.

[23] C. Boshuizen, J. Mason, P. Klupar, and S. Spanhake, "Results from the Planet Labs Flock Constellation", in *Proceedings of the AIAA/USU Conference on Small Satellites*, Logan, USA, 2014.

[24] S. P. Neeck, "Small Satellites and NASA Earth Science", presented at the 11th IAA Symposium on Small Satellites for Earth Observation, Berlin, Germany, 2017.

[25] M. Fritz, J. Shoer, L. Singh, T. Henderson, J. McGee, R. Rose, and C. Ruf, "Attitude determination and control system design for the CYGNSS microsatellite", in *2015 IEEE Aerospace Conference*, Mar. 2015, pp. 1–12.

[26] W. Blackwell, G. Allen, C. Galbraith, T. Hancock, R. Leslie, I. Osaretin, L. Retherford, M. Scarito, C. Semisch, M. Shields, M. Silver, D. Toher, K. Wight, D. Miller, K. Cahoy, and N. Erickson, "Nanosatellites for Earth environmental monitoring: The MicroMAS project", in *2012 12th Specialist Meeting on Microwave Radiometry and Remote Sensing of the Environment (MicroRad)*, Mar. 2012, pp. 1–4.

[27] A. da Silva Curiel, A. Cawthorne, L. Sills, N. Navarathinam, L. Gomes, and S. M. Sweeting, "What can we expect from very high resolution small satellites", presented at the 11th IAA Symposium on Small Satellites for Earth Observation, Berlin, Germany, 2017.

[28] M. Steckling, "Lageregelung im Bogensekundenbereich am Beispiel des Mikrosatelliten DLR-TUBSAT", German, PhD thesis, Technische Universität Berlin, 1998.

[29] S. Schulz, "Interaktive Lageregelung zur Erdbeobachtung mit Mikrosatelliten am Beispiel DLR-TUBSAT", German, PhD thesis, Technische Universität Berlin, 2001.

[30] Z. Yoon, "Robust Three-Axis Attitude Control System for Micro Satellites", PhD thesis, Technische Universität Berlin, 2011.

[31] Merriam-Webster Online Dictionary, *Definition of flexible*, Accessed: 2017-04-21. [Online]. Available: `http://www.merriam-webster.com/dictionary/flexible`.

[32] J. Mason, M. Baumgart, T. Woods, D. Hegel, B. Rogler, G. Stafford, S. Solomon, and P. Chamberlin, "MinXSS CubeSat On-Orbit Performance and the First Flight of the Blue Canyon Technologies XACT 3-axis ADCS", in *Proceedings of the AIAA/USU Conference on Small Satellites*, Logan, USA, 2016.

[33] D. Hegel, "FlexCore: Low-Cost Attitude Determination and Control Enabling High-Performance Small Spacecraft", in *Proceedings of the AIAA/USU Conference on Small Satellites*, Logan, USA, 2016.

[34] F. Tubb, R. McEwen, J. Farazian, and A. Waddell, "MSTI-3 Spacecraft Attitude Control Software Development using Automatic Code Generation", in *Proceedings of the AIAA/USU Conference on Small Satellites*, Logan, USA, 1994.

[35] M. Reid, W. Hansell, and T. Phillips, "The Implementation of Satellite Attitude Control System Software using Object Oriented Design", in *Proceedings of the AIAA/USU Conference on Small Satellites*, Logan, USA, 1998.

[36] E. Birrane, K. Bechtold, C. Krupiarz, A. Harris, A. Mick, and S. Williams, "Linux and the Spacecraft Flight Software Environment", in *Proceedings of the AIAA/USU Conference on Small Satellites*, Logan, USA, 2007.

[37] S. Fitzsimmons and A. Tsuda, "Rapid Development using Tyvak's Open Source Software Model", in *Proceedings of the AIAA/USU Conference on Small Satellites*, Logan, USA, 2013.

[38] J. Hansen and L. Hansen, "Development of attitude control systems for modular spacecraft", presented at the IEEE Aerospace Conference, Big Sky, USA, 2014.

[39] S. Grocott, "Attitude Control System for Microsatellites with Stringent Pointing Requirements", in *Proceedings of the AIAA/USU Conference on Small Satellites*, Logan, USA, 2000.

[40] M. F. Barschke, K. Gordon, and S. Junk, "Modular Architecture and Rapid Technology Update for a Flexible Nanosatellite Platform", presented at the 11th IAA Symposium on Small Satellites for Earth Observation, Berlin, Germany, 2017.

[41] ECSS, *ECSS-M-ST-10C: Space Project Management - Project Planning and Implementation*, ESA, ESA Publications Division, 2009.

[42] Jenkins Continuous Integration Server, *project homepage*, Accessed: 2016-03-22. [Online]. Available: `https://jenkins.io/`.

[43] ECSS, *E-ST-10-03A: Space Engineering - Testing*, ESA, 2009.

[44] P. Werner, M. Starke, K. Gordon, A. Graf, and M. F. Barschke, "Modular Ground Support Software for Nanosatellites", presented at the 10th IAA Symposium on Small Satellites for Earth Observation, Berlin, Germany, 2015.

[45] B. A. Roberts, J. W. Kruk, T. B. Ake, T. S. Englar, B. F. Class, and D. M. Rovner, "Three-axis Attitude Control with Two Reaction Wheels and Magnetic Torquer Bars", presented at the AIAA Guidance, Navigation, and Control Conference and Exhibit, 2004.

[46] M. F. Barschke, K. Brieß, and U. Renner, "Twenty-five Years of Satellite Development at Technische Universität Berlin", presented at the Small Satellites Systems and Services Symposium, Valetta, Malta, 2016.

[47] Z. Yoon, W. Frese, A. Bukmaier, and K. Briess, "System Design of an S-Band Network of Distributed Nanosatellites", *CEAS Space Journal*, vol. 6, no. 1, Mar. 2013.

[48] M. F. Barschke, F. Baumann, W. Ballheimer, K. Großekatthöfer, C. Nitzschke, and K. Brieß, "TUBiX20 – The novel Nanosatellite Bus of TU Berlin", in *Small Satellites for Earth Observation*, Berlin, Germany: Ed. Paris: International Academy of Astronautics, 2014, pp. 121–128.

[49] M. F. Barschke and K. Gordon, "A Generic System Architecture for a Single Failure Tolerant Nanosatellite Platform", presented at the 65th International Astronautical Congress, Toronto, Canada, 2014.

[50] M. F. Barschke, K. Gordon, P. von Keiser, and M. Starke, "FDIR Approach of a Modular Satellite Platform Architecture", presented at the 67th International Astronautical Congress, Guadalajara, Mexico, 2016.

[51] M. F. Barschke and K. Gordon, "Enabling Flexible Payload Management through Modularity", presented at the 66th International Astronautical Congress, Jerusalem, Israel, 2015.

[52] S. Montenegro and F. Dannemann, "RODOS: Real Time Kernel Design for Dependability", presented at the Data Systems in Aerospace (DASIA), Istanbul, Turkey, 2009.

[53] M. F. Barschke, K. Großekatthöfer, and S. Montenegro, "Implementation of a Nanosatellite On-Board Software based on Building-Blocks", presented at the Small Satellites Systems and Services Symposium, Porto Pedro, Spain, 2014.

[54] M. F. Barschke, J. Bartholomäus, K. Gordon, M. Lehmann, and K. Brieß, "The TUBIN nanosatellite mission for wildfire detection in thermal infrared", *CEAS Space Journal*, pp. 1–12, 2016. [Online]. Available: `http://dx.doi.org/10.1007/s12567-016-0140-6`.

[55] M. F. Barschke, K. Gordon, M. Lehmann, and K. Brieß, "The TechnoSat Mission for On-Orbit Technology Demonstration", presented at the Deutscher Luft- und Raumfahrtkongress, Braunschweig, Germany, 2016.

[56] O. Balagurin, H. Kayal, and H. Wojtkowiak, "Validation and qualification of a CMOS based miniature star tracker for small satellites", presented at the Small Satellite Systems and Services Symposium, Portoroz, Slovenia, 2013.

[57] W. Bauer, O. Romberg, and R. Putzar, "Experimental verification of an innovative debris detector", *Acta Astronautica*, vol. 117, pp. 49–54, 2015.

[58] G. Kirchner, L. Grunwaldt, R. Neubert, F. Koidl, M. Barschke, Z. Yoon, and H. Fiedler, "Laser ranging to nano-satellites in LEO orbits: plans, issues, simulations", presented at the 18th International Workshop on Laser Ranging, Fujiyoshida, Japan, 2013.

[59] R. Alavi, K. Briess, H. Podolski, J. Riesselmann, A. Weiland, and W. Frese, "In Space Verification of the Pico-Satellite S-Band Transmitter HISPICO on a Sounding Rocket", presented at the 60th International Astronautical Congress, Daejeon, South Korea, 2009.

[60] D. Noack, J. Ludwig, and K. Brieß, "Fluid-Dynamic Attitude Control Experiment for TechnoSat", presented at the Small Satellites Systems and Services Symposium, Majorca, Spain, 2014.

[61] G. T. Heinemann and W. T. Councill, *Component-Based Software Engineering*. Addison-Wesley, 2001.

[62] *OMG Unified Modeling Language*, Version 2.5, OMG Document Number: formal/2015-03-01, Object Management Group, 2015. [Online]. Available: `http://www.omg.org/`.

[63] K. Gordon and M. F. Barschke, "A New Concept of Software Architecture for a Flexible Attitude Determination and Control of Nanosatellites", presented at the 66th International Astronautical Congress, Jerusalem, Israel, 2015.

[64] J. C. Springmann, J. W. Cutler, and H. Bahcivan, "Magnetic Sensor Calibration and Residual Dipole Characterization for Application to Nanosatellites", presented at the AIAA/AAS Astrodynamics Specialist Conference, 2010.

[65] J. C. Springmann and J. W. Cutler, "Attitude-Independent Magnetometer Calibration with Time-Varying Bias", *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 4, pp. 1080–1088, 2012.

[66] M. D. Shuster and S. D. Oh, "Three-Axis Attitude Determination from Vector Observations", *Journal of Guidance and Control*, vol. 4, pp. 70–77, 1981.

[67] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Person, 1995.

[68] K. Gordon, M. Lehmann, and M. F. Barschke, "Flexible Low-Cost Verification of Attitude Determination and Control Systems", presented at the 11th IAA Symposium on Small Satellites for Earth Observation, Berlin, Germany, 2017.

[69] J. Eickhoff, *Simulating Spacecraft Systems*. Springer, 2009.

[70] B. P. Douglass, *Real-Time Design Patterns: robust scalable architecture for Real-time systems*. Addison-Wesley, 2003.

[71] K. Großekatthöfer and Z. Yoon, "Introduction into Quaternions for Spacecraft Attitude Representation", Chair of Space Technology, Technische Universität Berlin, Tech. Rep., 2012.

[72] J. R. Wertz, *Spacecraft Attitude Determination and Control*. Springer Netherlands, 1978.

[73] M. J. Sidi, *Spacecraft Dynamics And Control*. Cambridge University Press, 1997.

[74] W. Ley, K. Wittmann, and W. Hallmann, Eds., *Handbook of Space Technology*. John Wiley & Sons, Ltd., 2009.

[75] R. Wisniewski, "Fully Magnetic Attitude Control for Spacecraft Subject to Gravity Gradient", *Automatica*, vol. 35, pp. 1201–1214, 1999.

[76] W. J. Larson and J. R. Wertz, *Space Mission Analysis and Design*. Microcosm, 1999.

[77] M. Ding, D. Chen, K. Xing, and X. Cheng, "Localized fault-tolerant event boundary detection in sensor networks", in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, IEEE, vol. 2, 2005, pp. 902–913.

[78] A. Munir, A. Gordon-Ross, and S. Ranka, *Modeling and Optimization of Parallel and Distributed Embedded Systems*, ser. Wiley - IEEE. Wiley, 2016.

[79] P. Zarchan and H. Musoff, *Fundamentals of Kalman Filtering*. Progress in Austronautics and Aeronautics Volume 190, 2000.

[80] S. Theil, P. Appel, and A. Schleicher, "Low Cost, Good Accuracy – Attitude Determination Using Magnetometer and Simple Sun Sensor", presented at the 17th Annual AIAA/USU Conference on Small Satellites, Istanbul, Turkey, 2003, pp. 789–792.

[81] C. W. de Boom, J. A. P. Leijtens, L. M. H. v. Duivenbode, and N. van der Heiden, "Micro Digital Sun Sensor: System in a Package", in *MEMS, NANO and Smart Systems, 2004. ICMENS 2004. Proceedings. 2004 International Conference on*, 2004, pp. 322–328.

[82] Z. Yoon, T. Terzibaschian, C. Raschke, and O. Maibaum, "Robust and fault tolerant AOCS of the TET satellite", presented at the 7th IAA Symposium, 2009.

[83] G. Wahba, "A Least Squares Estimate of Satellite Attitude", *SIAM Review*, vol. 7, Issue 3, pp. 323–461, 1965.

[84] F. L. Markley, "Attitude Determination Using Vector Observations: A Fast Optimal Matrix Algorithm", *The Journal of the Astronautical Sciences*, vol. 41, No. 2, pp. 261–280, 1993.

[85] D. Mortari, "Second Estimator of the Optimal Quaternion", *Journal of Guidance, Control and Dynamics*, vol. 23, No. 5, pp. 885–888, 2000.

[86] F. L. Markley and D. Mortari, "How to Estimate Attitude from Vector Observations", presented at the AAS/AIAA Astrodynamics Conference, Girdwood, USA, 1999.

[87] E. Lefferts, F. Markley, and M. D. Shuster, "Kalman Filtering for Spacecraft Attitude Estimation", *Journal of Guidance, Control, and Dynamics*, vol. 5, pp. 417–429, 1982.

[88] M. D. Shuster, "Kalman Filtering of Spacecraft Attitude and the QUEST Model", *The Journal of the Astronautical Sciences*, vol. 38, No. 3, pp. 377–393, 1990.

[89] P. Singla, J. L. Crassidis, and J. L. Junkins, "Spacecraft Angular Rate Estimation Algorithms for a Star Tracker Mission", presented at the 13th Annual AAS/AIAA Space Flight Mechanics Meeting, 2003.

[90] J. L. Crassidis, F. L. Markley, and Y. Cheng, "Survey of Nonlinear Attitude Estimation Methods", *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 1, pp. 12–28, 2007.

[91] M. L. Psiaki, F. Martel, and P. K. Pal, "Three-Axis Attitude Determination via Kalman Filtering of Magnetometer Data", *Journal of Guidance, Control, and Dynamics*, vol. 13, No. 3, pp. 506–514, 1990.

[92] J. Treurnicht and W. H. Steyn, "A Robust Attitude Measuring System for Agile Satellites", presented at the IFAC First African Control Conference, 2003.

[93] P. Joergensen, J. Joergensen, and T. Denver, "On-the Fly Merging of Attitude Solutions", in *Small Satellites for Earth Observation*. Springer, 2008, pp. 175–183.

[94] Y. Cheng, F. L. Markley, J. L. Crassidis, and Y. Oshman, "Averaging Quaternions", presented at the 17th AAS/AIAA Space Flight Mechanics Meeting, 2007.

[95] L. Romans, *Optimal Combination of Quaternions from Multiple Star Cameras*, JPL, 2003.

[96] T. Holzhüter, "Zustandsregelung", German, Fachhochschule Hamburg, Tech. Rep., 2009.

[97] Z. Tudor, "Design and Implementation of Attitude Control for 3-axes Magnetic Coil Stabilization of a Spacecraft", Master's thesis, Norwegian School of Science and Technology, 2011.

[98] M. R. Greene and R. E. Zee, "Increasing the Accuracy of Orbital Position Information from NORAD/SGP4 Using Intermittent GPS Readings", presented at the 23rd Annual Small Satellite Conference, 2009.

[99] O. Montenbruck, E. Gill, and T. Terzibaschian, "Note on the BIRD ACS Reference Frames", DLC-GSOC, Tech. Rep. TN 00-01, 2000.

[100] D. A. Vallado, J. H. Seago, and P. K. Seidelmann, "Implementation Issues Surrounding the New IAU Reference Systems for Astrodynamics", presented at the 16th AAS/AIAA Space Flight Mechanics Conference, 2006.

[101] K. Gordon, M. F. Barschke, and M. Lehmann, "Phase C Bericht des TechnoSat - Lageregelung", German, Chair of Space Technology, Technische Universität Berlin, Tech. Rep. TE-3500-TR001-01, 2014.

Appendices

A TU Berlin's Satellite Missions

The following Table A.1 gives an overview of all TU Berlin satellite missions to date (September 14th, 2017). All future launch dates are yet to be confirmed.

Table A.1: TU Berlin's Satellite Missions (adapted from [46])

Name	Mass [kg]	Launch	Objective	Operational [months]
TUBSAT-A	35	1991	Communications	188
TUBSAT-B	45	1994	Earth observation	1
TUBSAT-N/N1	3 8	1998	Communications	23 11
DLR-TUBSAT	45	1999	Earth observation	118
MAROC-TUBSAT	47	2001	Earth observation	> 84
LAPAN-TUBSAT	56	2007	Earth observation	128 (to date)
BEESAT-1	1	2009	Tech demo	40
BEESAT-2	1	2013	Tech demo	52 (to date)
BEESAT-3	1	2013	Tech demo/ education	-
BEESAT-4	1	2016	Tech demo	12 (to date)
TechnoSat	20	2017	Tech demo	2 (to date)
S-Net-1/2/3/4	8	2017	Communications	-
TUBIN	20	2018	Earth observation	-
BEESAT-5/6/7/8	0.33	2018	Communications	-

B Coordinate Systems

Every attitude control system uses different coordinate systems for the representation of state quantities. Here, it is distinguished between inertial, Earth-fixed and body-fixed coordinate systems. The coordinate systems used in this thesis are presented in the following sections. Figure B.1 gives an overview and shows the relation of the different coordinate systems to each other.

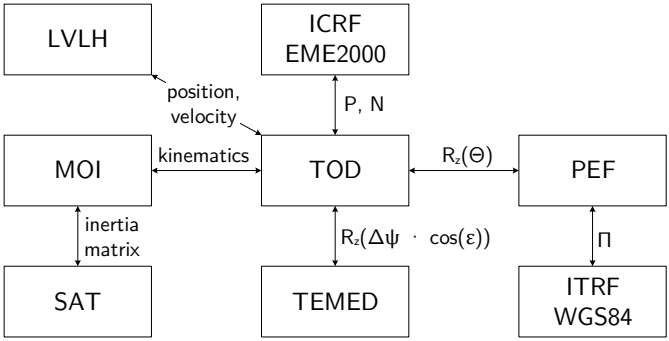

Figure B.1: Coordinate Systems

International Celestial Reference Frame

The International Celestial Reference Frame (ICRF) is a practical realization of the International Celestial Reference System (ICRS) based on VLBI measurements of extragalactic radio sources. The origin is the barycenter of the solar system.

The ICRF corresponds to Earth Mean Equator and Equinox of Epoch J2000 (EME2000) to an accuracy of several milli arcseconds. Its z-axis is the mean

rotational axis of the Earth and its x-axis points towards the mean equinox at the epoch J2000 [5].

True Of Date

The True Of Date (TOD) coordinate system is an inertial coordinate system referring to the true equator and equinox at the point of consideration. The z-axis is the instantaneous rotational axis (CEP) of the Earth, while the x-axis points towards the true vernal equinox. The origin of the TOD system is the geometrical center of the Earth [99].

The transformation from ICRF to TOD is performed by Equation B.1:

$$\vec{v}_{TOD} = N(t) \cdot P(t) \cdot \vec{v}_{ICRF} \tag{B.1}$$

N is the nutation matrix and P the precession matrix of the Earth's rotation [5].

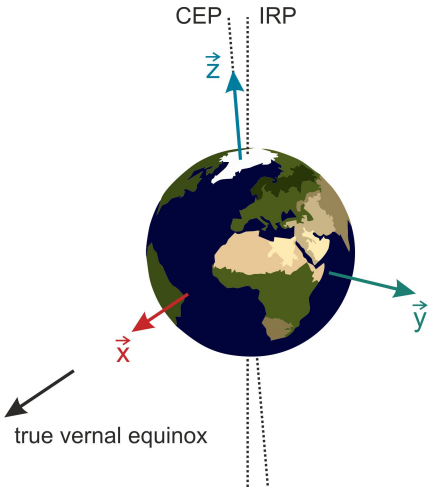

Figure B.2: True Of Date (TOD) System
adapted from: NASA Blue Marble, https://visibleearth.nasa.gov/

True Equator and Mean Equinox of Date

The inertial coordinate system True Equator and Mean Equinox of Date (TEMED) is oriented towards the true equator and the mean equinox. It is used for the NORAD Two Line Elements (TLE) and the SGP4 algorithm for orbit propagation. The z-axis is the instantaneous rotational axis (CEP) of the Earth and the x-axis points towards the mean equinox. The origin of the coordinate system is the geometrical center of the Earth.

To transform a vector from TEMED to TOD, Equation B.2 is applied:

$$\vec{v}_{TOD} = R_z(\Delta\psi \cdot \cos\epsilon) \cdot \vec{v}_{TEMED} \tag{B.2}$$

The term $\Delta\ \psi \cos\epsilon$ is called *equation of the equinox* and states the difference between the true and the mean equator [99].

Pseudo Earth Fixed

The Pseudo Earth Fixed (PEF) coordinate system uses the CEP as z-axis and the x-axis points towards the IERS reference meridian. Its origin is the Earth's geometrical center. Since the rotational axis of the Earth changes continuously, the PEF System is not exactly Earth-fixed, hence the name *Pseudo* Earth Fixed.

The relation between the PEF and TOD coordinate systems is given via the true sidereal time, Θ [99]:

$$\vec{v}_{PEF} = \Theta(t) \cdot \vec{v}_{TOD} \tag{B.3}$$

World Geodetic System 1984

The World Geodetic System 1984 (WGS84) system is a widespread version of an Earth-fixed coordinate system defined by the National Imagery and Mapping Agency (NIMA). Its origin is the geometrical center of the Earth.

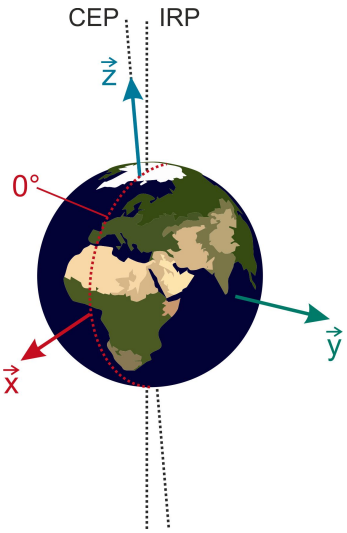

Figure B.3: Pseudo Earth Fixed (PEF) System
adapted from: NASA Blue Marble, https://visibleearth.nasa.gov/

The x-axis points towards the IERS reference meridian, while the z-axis points toward the IERS reference pole [99].

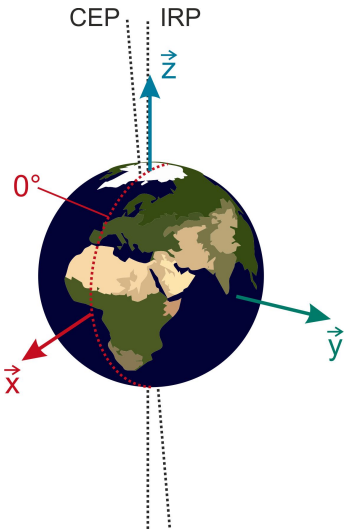

Figure B.4: World Geodetic System 1984 (WGS84) System
adapted from: NASA Blue Marble, https://visibleearth.nasa.gov/

The transformation between PEF and WGS84 is achieved via the pole displacement matrix Π, which expresses the rotation between CEP with respect to IERS reference pole (IRP) [5]:

$$\vec{v}_{WGS84} = \Pi(t) \cdot \vec{v}_{PEF} \tag{B.4}$$

Applying Equation B.3, the transformation from TOD to WGS84 follows as:

$$\vec{v}_{WGS84} = \Pi(t) \cdot \Theta(t) \cdot \vec{v}_{TOD} \tag{B.5}$$

The WGS84 corresponds to the International Terrestrial Reference Frame (ITRF) with an accuracy of several milli arcseconds [100]. The ITRF is a practical realization of the International Terrestrial Reference Systems (ITRS)

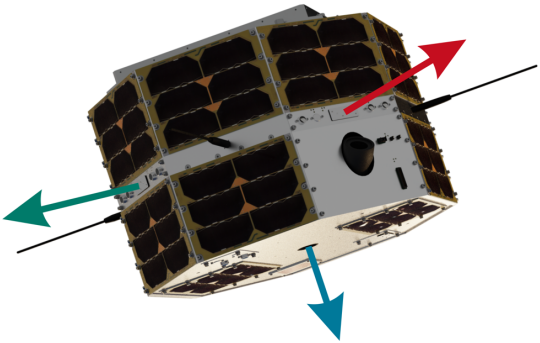

Figure B.5: Satellite-fixed geometrical System (SAT) for TechnoSat
Image credit: Marc Lehmann

based on measurements of selected observation stations. It is updated frequently by the IERS.

From Equation B.1 and Equation B.5, the transformation between ICRF and ITRF follows as [5]:

$$\vec{v}_{ITRF} = \Pi(t) \cdot \Theta(t) \cdot N(t) \cdot P(t) \cdot \vec{v}_{ICRF} \tag{B.6}$$

Satellite-Fixed Geometrical System

The Satellite-fixed Coordinate System (SAT) is defined according to the geometry of a satellite. For TechnoSat, it is shown in Figure B.5. Its origin is the geometrical center of the satellite. The z-axis points into the direction of the camera payload, while the x-axis points towards the star tracker.

Moment Of Inertia System

The Moment of Inertia Coordinate System (MOI) system is aligned to the principal axes of inertia of the satellite. Its origin is the satellite's center of gravity. The x-axis is parallel to the principal axis of inertia which is closest

to the x-axis of the SAT coordinate system. The y- and z-axis are defined in a similar way.

Local Vertical Local Horizontal

The Local Vertical Local Horizontal (LVLH) system is aligned at the orbit of the satellite. The z-axis points towards nadir, while the x-axis points in flight direction. The origin of the LVLH-Systems is the satellite's center of gravity.

Figure B.6: Local Vertical Local Horinzontal (LVLH) System
adapted from: NASA Blue Marble, https://visibleearth.nasa.gov/

The coordinate axes of the LVLH system with respect to the TOD system depend on the satellite's position and velocity:

$$\vec{\tilde{x}}_{LVLH} = \|\vec{v}_{TOD}\| \tag{B.7}$$

$$\vec{z}_{LVLH} = -\|\vec{r}_{TOD}\| \tag{B.8}$$

$$\vec{y}_{LVLH} = \vec{z}_{LVLH} \times \vec{\tilde{x}}_{LVLH} \tag{B.9}$$

$$\vec{x}_{LVLH} = \vec{y}_{LVLH} \times \vec{z}_{LVLH} \tag{B.10}$$

Here, $\vec{r}_{TOD}$ is the satellite's position and $\vec{v}_{TOD}$ its velocity in the TOD system. Due to the elliptic orbit of the satellite, the x-axis is first calculated based on the true flight direction and later corrected to receive a rectangular coordinate system.

The transformation matrix from LVLH to TOD, $T_{TOD \leftarrow LVLH}$, is according to Equation 4.11:

$$T_{TOD \leftarrow LVLH} = \begin{bmatrix} \vec{x}_{LVLH} & \vec{y}_{LVLH} & \vec{z}_{LVLH} \end{bmatrix} \tag{B.11}$$

C Derivation of ADCS Requirements for TUBiX20 Missions

This appendix documents the derivation of the requirements for the two TUBiX20 missions TechnoSat and TUBIN.

TechnoSat

To perform experiments with the star tracker payload, the sensor must not be blinded by straylight from the Sun or Earth. The easiest way to operate the star tracker would be a dedicated pointing mode with an inertially fixed target orientation. To operate the star tracker while pointing towards nadir, e. g. for simultaneous data downlink or imaging, the required pointing accuracy depends on three influencing factors:

- the mounting orientation of the sensor within the satellite
- the exclusion angle of the baffle
- the orientation of the satellite in relation to the Sun and Earth.

The mounting orientation is 45° off-nadir, while the sensor baffle has an exclusion angle of approximately 40°, both shown in Figure C.1.

As can be seen here, the mounting orientation allows a rotation of the satellite's z axis to avoid sunlight incidence. Regarding straylight from Earth, however, this is not possible. The required pointing accuracy, β, is the angle between the edge of the atmosphere and the sensor's boreside axis, subtracted by the baffle exclusion angle:

$$\begin{aligned} \beta &= 90° - \alpha + 45° - 40° \\ &= 27.9° \end{aligned}$$

A complete derivation of this result is given in [101].

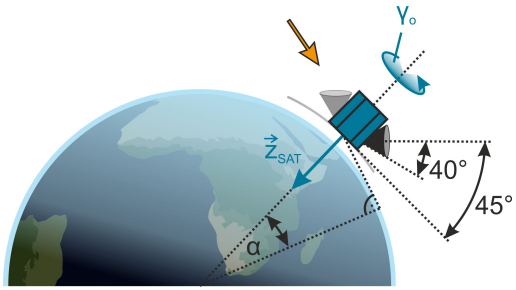

Figure C.1: Star Tracker Experiments while Nadir Pointing [101]
adapted from: NASA Blue Marble, https://visibleearth.nasa.gov/

A second payload which requires active attitude control is the S-band transmitter. To perform the experiments, the downlink capacity does not need to be maximized, but a contact time of at least 90 s is required. This is achievable even if the patch antenna is not pointing directly to the ground station. A detailed calculation has been carried out in [101] and resulted in a maximum tilt angle, β, of 34.1° (cf. Figure C.2).

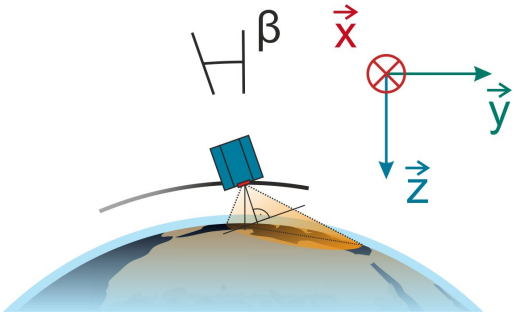

Figure C.2: S-band Transmitter Experiments while Nadir Pointing [101]
adapted from: NASA Blue Marble, https://visibleearth.nasa.gov/

TUBIN

The following analysis transforms the requirements formulated by the payload in terms of ground pixel knowledge, pointing accuracy and stability to direct numerical values for the ADCS.

The pointing knowledge of the spacecraft in-orbit, α_k, results from the knowledge of a camera pixel on ground, l_k, and the altitude, h.

$$\alpha_k = \arctan\left(\frac{l_k}{h}\right) \tag{C.1}$$

The TUBIN payload requires the knowledge of a ground pixel of at least 2 km. The altitude of the orbit is 600 km. Therefore, the required pointing knowledge is

$$\begin{aligned} \alpha_k &= 3.3\,\text{mrad} \\ &= 11.5\,\text{arcmin} \end{aligned}$$

Similar to the pointing knowledge, the pointing accuracy, α_a, refers to the accuracy of a camera pixel on ground, l_a:

$$\alpha_a = \arctan\left(\frac{l_a}{h}\right) \tag{C.2}$$

With a required accuracy of 35 km, the required pointing accuracy is

$$\begin{aligned} \alpha_a &= 58.3\,\text{mrad} \\ &= 200.3\,\text{arcmin} \end{aligned}$$

The pointing stability (jitter) of the satellite in-orbit, j, is derived from the size of a ground pixel, p and the time to take a single image, t_i (cf. Figure C.3).

$$j = \frac{1}{t_i}\arctan(\frac{c}{h}) \tag{C.3}$$

with

$$c = \sqrt{2} \cdot b \tag{C.4}$$
$$b = p - a \tag{C.5}$$

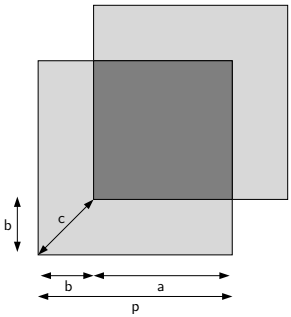

Figure C.3: Ground Pixel Stability

The TUBIN payload tolerates a maximum of 10 % of p, and hence

$$a = p \cdot \sqrt{0.9} \tag{C.6}$$

Finally,

$$t_i = 1\,\text{ms}$$
$$p = 85\,\text{m}$$

results in

$$j = 10.3\,\text{mrad/s} = 35.3\,\text{arcmin/s}$$

The minimum slew rate, ω_{sl}, is calculated as follows:

$$\omega_{sl} = \frac{\beta}{t_s} \tag{C.7}$$

Here, β is the slew angle and t_s is the slew duration. For the required slew of 180° in 3 min, the required slew rate is

$$\omega_{sl} = 1\,^\circ/\mathrm{s}$$

Schriftenreihe **Institute of Aeronautics and Astronautics: Scientific Series**
Hrsg.: Prof. Dr.-Ing. Dieter Peitsch, Prof. Dr.-Ing. Andreas Bardenhagen,
Prof. Dr.-Ing. Klaus Brieß, Prof. Dr.-Ing. Robert Luckner,
Prof. Dr.-Ing. Oliver Lehmann, Prof. Dr.-Ing. Thomas Grund

ISSN 2512-5141 (print)
ISSN 2512-515X (online)

01: Behrend, Ferdinand: Advanced Approach Light System. Der Einfluss eines zusätzlichen visuellen Assistenzsystems zur Steigerung des Situationsbewusstseins bei kritischen Wetterbedingungen hinsichtlich vertikaler Fehler im Endanflug. - 2017. - XXIV, 206 S.
ISBN **978-3-7983-2904-1** (print) EUR **16,50**
ISBN **978-3-7983-2905-8** (online)

Universitätsverlag der TU Berlin

A flexible attitude control system for three-axis stabilized nanosatellites

This thesis investigates a new concept for the flexible design and verification of an attitude determination and control system (ADCS) for a nanosatellite platform. The research was carried out during the development of TU Berlin's nanosatellite platform TUBiX20 and its first two missions, TechnoSat and TUBIN. The concept implements the ADCS as a distributed system of devices complemented by a hardware-independent core application for state determination and control, while the software is partitioned into self-contained modules which implement unified interfaces. These interfaces specify the state quantity of an input or output but also its unit and coordinate system. The design and verification process for the TUBiX20 ADCS was also elaborated during the course of this research. The approach targets the gradual development of the subsystem from a purely virtual satellite within a closed-loop simulation to the verification of the fully integrated system on an air-bearing testbed. Finally, the concurrent realization of the investigated concept within the TechnoSat and TUBIN missions is discussed. Starting with the individual ADCS requirements, the scalability of the approach is demonstrated in three stages.

ISBN 978-3-7983-2968-3 (print)
ISBN 978-3-7983-2969-0 (online)

ISBN 978-3-7983-2968-3

http://verlag.tu-berlin.de