CORRECTNESS ANALYSIS OF AN EMBEDDED CONTROLLER

Project OHB Abrixas PTC Test

Dr. Holger Schlingloff*, Oliver Meyer*, Thomas Hülsing†

Abstract

In this paper we report on the use of a new method for quality assurance of safety-critical software in aerospace applications. The power and thermal control unit (PTC) of the X-ray satellite "Abrixas", developed by OHB System GmbH, Bremen, was analysed with this new formal method. On the basis of formal specifications we automatically generated and executed test sequences for the reactive behaviour of the controller. We checked the full functionality of the PTC, including real-time and hybrid properties, user interaction, stress- and long-term behaviour, and fault tolerance. Our tool VVT-RT generated an extensive coverage of test sequences for these properties. The tests were then automatically performed and evaluated, using the original hardware in a simulation environment. Due to the universality of VVT-RT, only little effort was necessary to set up the interfaces. Thus, even though we achieved a higher test coverage, the overall costs of the validation were much lower than with comparable methods.

1 Software Quality Assurance

In most high-tech products software quality is the decisive factor for success. Especially in safety critical applications software bugs can cause immense costs. Therefore, it is important to develop methods and tools to ensure the correctness of control software. In recent years, new formal methods have been developed to increase dependability and reduce the overall software design costs. We applied these methods successfully to several non-trivial industrial projects such as train control systems [AD 97, HP98], fault-tolerant aerospace computers [But 97, But 99], cellular phones [Sch 98] and airplane communication systems [HP 95]. In this paper we report on the automatic testing of an embedded satellite controller based on formal specifications.

Conventional software quality assurance comprises version and configuration management, requirements coverage and code inspections, metrics and coding standards, static and dynamic analysis, capture-replay testing, structural testing and simulation. Although these methods can help to reduce the number of program bugs, they are not sufficient to ensure correctness. Using conventional methods, a certain amount of software errors is inevitable. Usually, software is thus delivered as "beta-version", with frequent updates and bugfixes.

For safety-critical systems such a procedure is not acceptable. In most aerospace applications it is not feasible to produce a "beta-version": after takeoff all subsystems must function immediately and correctly. Analysis and correction of program errors during flight are impossible or induce a high risk. Even "small bugs" can cause experiment faults or even a loss of mission. Therefore, correctness is an important design criterion.

In addition to the above standards a further increase in dependability can be achieved by formal methods in software design. For some time, mathematical calculi have been proposed for the logical analysis of programs. Until recently, use of these calculi had been limited due to their inherent complexity. By improvement of algorithms and data structures, however, it is now possible to apply formal methods to programs of industrial size. For embedded systems and control software *automated testing* turned out to be a promising approach.

In this approach logical specifications of the behaviour of the system and its environment are used to automatically generate test sequences. The complete embedded system (hard- and software) is tested with these sequences. A test driver creates inputs for the system, while a test oracle checks the correctness of the system's outputs. In contrast to the usual test-script approach, further continuation of each test is determined by these outputs and by previously tested sequences. A test monitor guarantees that all relevant test sequences are covered. Since the method is completely automatic, a test cover-

*University of Bremen, TZI-BISS, Pf. 330440, D-28334 HB; {hs,emm}@tzi.de; ++49-421-218-{7291, 3337}, FAX: ++49-421-218-3054

†OHB System GmbH, Universitätsallee 27-29, D-28359 HB; huelsing@ohb-system.de; ++49-421-2020-639, FAX: ++49-421-2020-610

age can be achieved which is much higher than with conventional testing.

In this paper we report on the use of this new method in an aerospace application. A satellite control unit was analyzed. The full functionality of the controller was checked, including real-time and hybrid properties, user interaction, stress- and long-term behaviour, as well as fault tolerance. To our knowledge, there are no documented applications of methods covering a similar extensive range of properties. This article is the first report on the application of our testing method in aerospace projects.

The paper is organized as follows: First, we give a short description of the ABRIXAS satellite power and thermal control (PTC), and of the power system check out equipment (Power-SCOE). Then, we describe the testing tool VVT-RT (**V**alidation, **V**erification and **T**est of **R**eal **T**ime Systems) and its connection to the Power-SCOE. In the main part, the development of formal specifications from the requirements document is demonstrated with several examples. Finally, the testing results and experiences during the project are presented.

2 Project Description

The ABRIXAS X-ray satellite is built by OHB-System GmbH, Bremen, in cooperation with various scientific laboratories in Germany. Its mission is the first complete scan of the sky in the medium energy X-ray range up to 10 KEV. The system consists of several modules: bus control, attitude control, camera control, and power and thermal control. The design is highly redundant, each hardware component having at least one backup. Even the software is multiply loaded into different storage areas. However, this gives only protection from hardware-faults; correctness of the software is of equal importance. For example, a bug in the battery charge algorithm within the power and thermal controller could gradually reduce the capacity of the battery and thus eventually lead to a loss of the satellite. Therefore, extensive application of quality assurance methods was necessary in the construction of the software.

The automated testing tool VVT-RT was developed by Verified Systems International GmbH, Bremen, in cooperation with the Bremen Institute of Safe Systems (BISS) within the Center for Computing Technology (TZI) at Bremen University. This paper reports on the use of VVT-RT in aerospace projects. It describes the analysis of the ABRIXAS **P**ower and **T**hermal **C**ontrol unit (PTC) with this formal tool.

2.1 The ABRIXAS–PTC

The main task of the ABRIXAS–PTC is to guarantee power supply of all active consumers. During the sun phase the energy is generated by the solar strings; during the shade phase the battery is used as a power source. The PTC has to control discharge and recharge of the battery. It further influences power supply and temperature of various components (e.g., battery, mirror system, camera, attitude control, antennas). Another task is the central acquisition of a significant number of electrical and thermal data, and their transmission to the tracking station at GSOC. An overview on the components controlled by the PTC is given in Fig. 1.

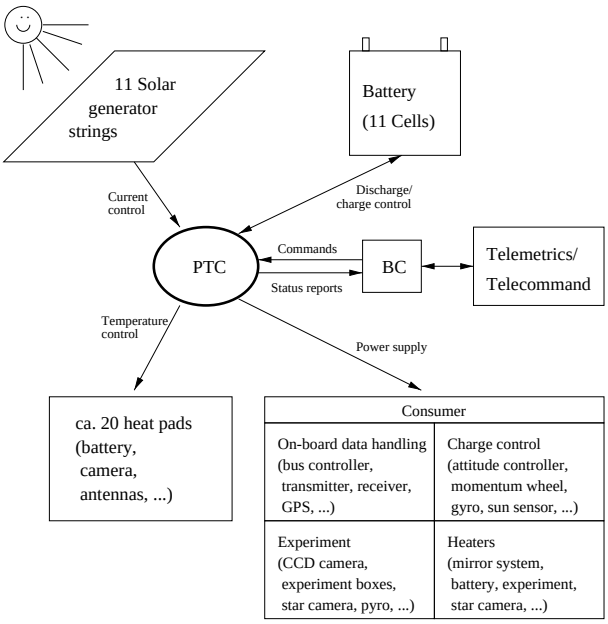

Figure 1: Environment of the ABRIXAS-PTC

The PTC is realized as a separate box consisting of a motherboard (PTC controller, PTCC) with several additional cards for data acquisition, energy distribution etc. The PTCC holds a standard (space approved) 80C31 processor, which can access data from all other cards via a digital bus. The additional cards consist of special circuitry to record voltage, current, pressure and temperature of various parts of the satellite.

The PTC software is coded in C and Assembler. It is configured with a set of tables written into the ROM. Input of the PTC are approximately 260 signals, i.e. analog measured data. The PTC controls approximately 100 switches via the bus, and 70 open collector wires. It can receive approximately 20 different types of telecommands from the ground station, and transmit error and diagnostic reports. Furthermore,

it has the possibility of latch-up control via the bus, and can communicate with the second (redundant) PTC. Fig. 2 gives an overview of the PTC interfaces.

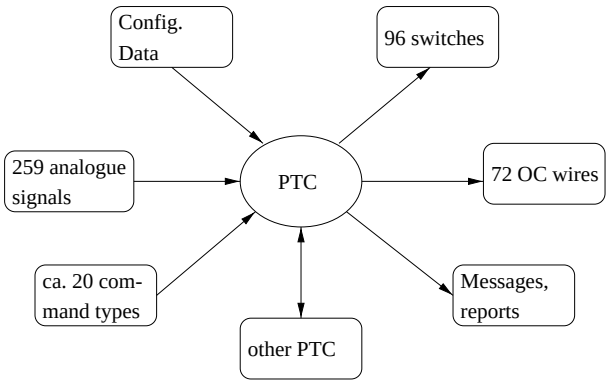

Figure 2: Interfaces of the ABRIXAS-PTC

The PTC software consists of approximately 36 modules. A typical control task is to keep the temperature of a mirror system within a constant interval around 20°C. Another task is to control the charging of the battery during the sun phase, until the amount of energy discharged during the preceding shade phase is recharged and a certain battery pressure is reached. Since the hardware is highly redundant, this specification has to be satisfied even in case of certain hardware faults.

2.2 The Power-SCOE

For the purpose of testing the PTC the so called Power-SCOE (**S**ubsystem **C**heck-**O**ut **E**quipment) was built by OHB. The Power-SCOE consists of the following components:

- Battery simulator (adjustable power supply)
- Consumer simulators (adjustable power sinks)
- Solar generator simulator, produced by Hewlett Packard
- COSMI-PC to compute the simulation values
- Extenderboard between PTCC and data acquisition cards
- ADMEG-PC for generating and measuring data

A block diagram of the Power-SCOE is given in Fig. 3.

Simulation of the battery was done by an adjustable power supply, since the original battery was not

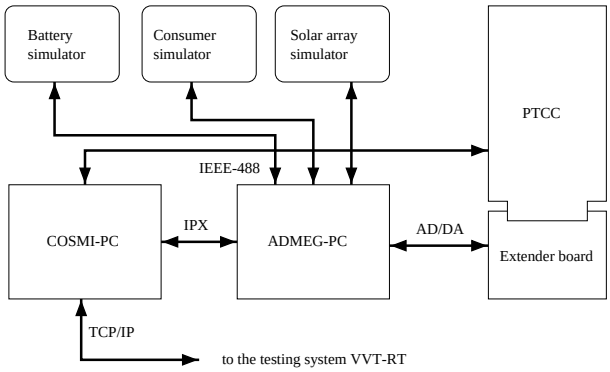

Figure 3: The ABRIXAS Power-Subsystem Check Out Equipment

available for testing. Furthermore, this allowed us to simulate arbitrary charge states without actually charging the battery. In this simulation it was possible to mark single battery cells as being defective. The charge state is detected by the PTC via two pressure sensors attached to the battery, and by its temperature. The respective pressure- and temperature values were generated by the COSMI-PC in the simulation.

Within the satellite there are approximately 35 power consumers, which generate a variable load. These include heaters, transmitters, receivers, bus controller, etc. In a first simulation the ABRIXAS consumer were implemented in software; later these were replaced by a hardware implementation with resistors and capacitors.

The solar array simulator is an adjustable power supply manufactured by Hewlett Packard for simulation purposes. It emulates the exact behaviour of a real solar generator. In our project it simulated the 11 solar strings of the satellite. Parameters were the percentage of current and voltage in dependency of solar radiation, summer or winter time, age of the strings and possible string faults.

The COSMI PC calculated the battery simulation values and replaced the bus controller of the satellite, which is responsible for switching of switches and for telemetrics and telecommunication. The current position of all switches, the analog measured values and the PTC configuration data could be displayed and manipulated via this PC. It communicated with the ADMEG by a IPX data channel and with VVT-RT by 10BaseT TCP/IP connection.

The ADMEG PC sampled the analogue signals in real time and transmitted them to the PTC via the extender board. The test system was able to manipulate every single of these values. Data which were

not available from hardware- or software simulations were generated by VVT-RT on the basis of a formal specification of the environment. Therefore, the reaction of the PTC to unusual signals could be tested by substitution of regular data with faulty values. This way, a complete simulation of normal and exceptional behaviour of all environment components could be realised.

3 The Tool VVT-RT

The automated testing tool VVT-RT is based on the testing theory by M. Hennessy [Hen 88] and the formal spezification language CSP (Communicating Sequential Processes) of C.A.R. Hoare [Hoa 85]. The current version uses the verification tool FDR (Failure-Divergence-Refinement) of Formal Systems, Ltd. [FDR 94]. A modern introduction to CSP and FDR can be found in [Ros 98].

VVT-RT realises a fully automatic execution of the following activities:

- test case generation
- test execution and monitoring
- test logging and evaluation

For the tests, the requirements document was translated into a logical description of target system and environment. From this formal specification VVT-RT generates a large number of test cases, which are executed by providing a stream of inputs and monitoring the corresponding outputs of the target. The architecture and structure of the target is completely hidden for VVT-RT, thus the complete system (hardware, software and their connection) is tested. The testing theory guarantees a coherent test coverage: since all possible execution sequences are considered, the testing procedure converges to a proof. Specifying the properties to be tested is the only human activity, the rest of the testing process is completely automatic. Therefore, a test coverage can be achieved which is far beyond conventional possibilities. From the generated protocols it is easy to locate and correct errors in the system; the additional effort for regression testing is minimal. Application areas for this new method are process control systems, firmware, programmable logic controllers, communication protocols, fault tolerant computing systems and embedded real time systems.

VVT-RT is available on all UNIX-platforms and can communicate with the system under test via all standard interfaces (serial, parallel, ethernet etc.). If the target does not provide a standard interface, usually it is not much effort to build a driver realising the communication. For time critical applications VVT-RT can be distributed onto several machines; this way, a potentially arbitrary execution speed can be achieved. It is planned to integrate a Java module for the internet, which allows to execute long distance tests on remote targets.

The principal components of VVT-RT are shown in Fig. 4 [Pel 96a]. Subsequently, we explain each component with the present application.

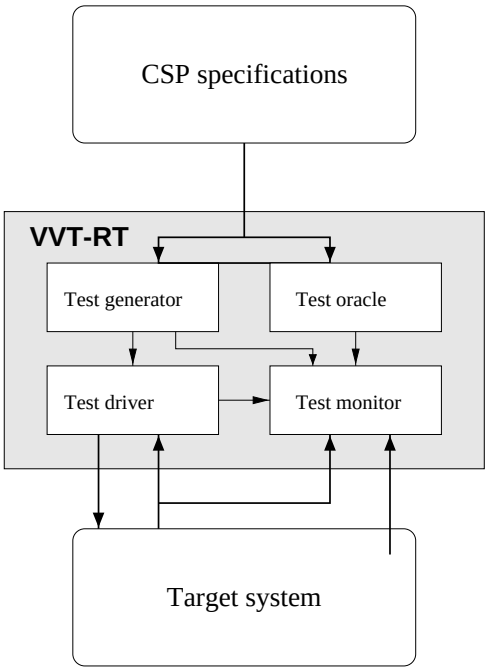

Figure 4: The testing system VVT-RT

The use of VVT-RT in other areas is described in [Pel 96b] and [AD 97]. In our project, a K6-200 PC was connected via standard ethernet with the COSMI-PC of the Power-SCOE. Licence terms forced us to generate the FDR-dependent parts of the tests on a Solaris workstation. We then transferred the data to the VVT-RT PC, which executed and evaluated the tests under Linux.

From the formal specification of system requirements VVT-RT's **test generator** uses FDR to generate a *transition graph*. This graph represents admissible steps of the target system. Each transition is associated with an abstract *event*. In contrast to CSP input- and output-events are distinguished. Example events are the sending of a message on the serial port, a switch command issued on the bus, or the change of the voltage on an open collector wire. From these the *event mapping library* generates abstract output events which are sent to VVT-RT. In the other di-

rection, certain abstract input events in the specification influence the behaviour of the target system. Examples are the modification of analog sensor readings, the sending of telecommands to the PTC and the control of the hardware simulators. Each of these input events is mapped into corresponding actions at the input interfaces of the target system.

In order to execute the test, the **test driver** component of VVT-RT selects abstract events according to the current state and possible transitions in the transition graph. The corresponding action is passed on to the target system and may lead to a reaction. VVT-RT changes the current node in the transition graph accordingly and checks whether the observable reactions are admitted by the specification at that time. If this is not the case, or if an unexpected event occurs, the **test oracle** reports a failure. In contrast to conventional tools for functional testing we are testing *reactive* real time behaviour: it might be possible that a single input causes a chain of outputs, or that some output requires several inputs at specific times.

The **test monitor** guarantees that the coverage is strictly increasing. In any given state it selects an appropriate input according to the coverage strategy. Therefore the degree of reliability increases with each test: theoretically, each combination of events in the transition graph is tested at some time.

In reactive systems, which in general are nonterminating, each admissible execution is infinite. Therefore, in the present application the tests were initiated and executed "on the fly", during continuous operation of the PTC. Thus, several properties and stress and long term behaviour could be tested in the same test run.

4 Requirement Specification

In order to formally specify the system requirements, we first identified and classified properties describing the complete capacity of the system. Each of these properties corresponds to a class of functionalities to be tested. There were three groups of requirements:

- requirements for switching functions,
- requirements for the energy control, and
- requirements for the thermal control.

As an example for a switching requirement we mention the telecommand to turn a consumer on or off. This command is delivered to the PTC via the bus controller. The PTC checks the command for syntactical and semantical consistency. Usually, in order to turn a consumer on, several switches have to be toggled. The PTC executes the command by putting a sequence of appropriate signals onto the bus. It then checks via voltage and current sensors whether the operation was correctly performed, and sends an acknowledgement or error message to ground control. A simple switching requirement is given in Fig. 5.

> At any given moment, it is possible to send telecommands for turning any consumer on or off. All switching operations neccessary to activate or deactivate this consumer must be performed within a given time constant `T_Swtch_Consumer`.

Figure 5: Requirement for Switching Functionality

An example requirement for the energy control is the central part of the battery charge control. The satellite battery is highly delicate and must be charged and decharged according to certain manufacturer instructions. Charging is done in two phases: main charge and supplementary charge. During main charge it is important to quickly recharge the battery with a high current, while during supplementary charge the average pressure and temperature of the battery influence the charging. The requirement for the main charge phase is given in Fig. 6.

> During the sun phase the battery is charged with a charge current of `I_Charge`, until the amount of energy consumed during the last shade phase is recharged, and the minimal absolute pressure `P_minabs` is reached or the sun phase ends.

Figure 6: Requirement for the Energy Control

A third example is the requirement for the thermal control of the CCD camera. The PTC has to regulate the temperature with electric heat pads, such that it is as stable as possible; in any case it must remain within a certain interval. In Fig. 7 the normal behaviour is described. This simplified requirement could be implemented with a simple hardware switch. The complete requirement, which includes the desired behaviour in case of hardware faults and insufficient power supply, requires a nontrivial switching logic and is beyond the scope of this paper.

In order to automatically generate a (usually very large) number of test cases and test data from these requirements, we formulated them in the specification language CSP. Although some expert knowledge is required to write such formal specifications, the

The temperature of the CCD camera is always within fixed bounds. If the camera is turned on, its heater must be deactivated. If the camera is turned off, the following rules apply: If the temperature of the CCD is less than `H_CCD_opt-H_CCD_tol`, the corresponding heater is switched on. If the temperature of the CCD is greater than `H_CCD_opt+H_CCD_tol`, the corresponding heater is switched off.

Figure 7: Requirement for the Thermal Control

process is comparable to programming in a high level programming language and could be done by system engineers. First, the interfaces of each requirement were listed in a systematic way. Then dependencies between interfaces were located and the requirements were grouped in several classes. For each analog channel bounds were defined, such that the transgression of these bounds leads to the generation of an event. The event mapping library was programmed and the communication software connecting VVT-RT to the Power-SCOE was implemented.

```
SPEC = ( SWITCHCONS [|{| Tau_nextTC |}|] TCTIM )
       |||  TIMCHK

SWITCHCONS =  Tau_nextTC -> (
     (Com_PYRO_PWR_CONS_ON -> setTimSwt
        -> Swt_BS_ON_MAIN_ON -> Swt_PYRO_PRE_MAIN_ON
        -> Swt_PYRO_PWR_MAIN_ON -> resTimSwt
        -> SWITCHCONS)
 |~| (Com_PYRO_PWR_CONS_OFF  -> setTimSwt
        -> Swt_PYRO_PWR_MAIN_OFF -> resTimSwt
        -> SWITCHCONS)
 |~|  ... )

TIMCHK = elaTimSwt -> errorSwitchTimer -> TIMCHK

TCTIM = Tau_nextTC -> setTimTick -> elaTimTick
          -> TCTIM
```

Figure 8: CSP-code for the requirement in Fig. 5

Then the requirements were formalised in CSP. As examples we give the CSP code for the above requirements. Fig. 8 shows part of the formulation of the requirement in Fig. 5. The formal specification `SPEC` consists of three parallel subprocesses `SWITCHCONS`, `TIMCHK` and `TCTIM`. Process `SWITCHCONS` nondeterministically chooses a consumer to be turned on or off. (In the figure, we only display the consumer `PYRO` which is used to open the cover of the mirror system.) The process then generates an event `Com_PYRO_PWR_CONS_ON` or `Com_PYRO_PWR_CONS_OFF`, which is translated by the event mapping library into an appropriate telecommand for turning the pyro on or off. Then a timer `TimSwt` is activated which supervises the required time bound `T_Swtch_Consumer`. For turning on the pyro, the PTC has to switch the three consecutive switches `Swt_BS_ON_MAIN_ON`, `Swt_PYRO_PRE_MAIN_ON` and `Swt_PYRO_PWR_MAIN_ON`. Process `SWITCHCONS` waits for the corresponding events. If they arrive in time, it resets the timer `TimSwt`. To turn the pyro off it suffices to switch `Swt_PYRO_PWR_MAIN_OFF`. All tests are done in an endless loop, which is realised by a recursive call. Process `TIMCHK` is executed interleaved with process `SWITCHCONS`. If the timer `TimSwt` elapses, VVT-RT generates the event `elaTimSwt` which is delivered to this process. In this case, it generates an appropriate error event `errorSwitchTimer` for the test oracle. This way, it is possible to detect timing errors without stopping or restarting the PTC.

First experiments with this CSP code revealed that the PTC could not satisfy the requirement. The reason was that the test system generated telecommands too quickly. As soon as a switch was toggled by the PTC, the test system without delay asked for the next consumer to be switched. Sometimes the PTC failed to notice these commands. In reality, where commands are emitted from human operators at ground control, it is reasonable to assume that such a frequency of commands can not be reached. Therefore, the informal requirement that "at any given moment" telecommands must be accepted was supplemented. We assumed that at most one telecommand per second is sent. In the CSP specification, this assumption is realised by an additional timing process `TCTIM`, which is synchronized with process `SWITCHCONS` via the internal event `Tau_nextTC`. It delays the sending of new commands by the assumed bound. With this modified specification, the PTC was fast enough to pass the switching tests.

```
CHARGECONTROL = Tau_SUN_ON -> setTimChargeControl
                  -> MAINCHARGE(false, false)

MAINCHARGE(PressureOK, Recharged) =
    (Evt_I_BATT_MAIN_IS_I_Charge
       -> resTimChargeControl
       -> MAINCHARGE(PressureOK, Recharged))
 [] (elaTimChargeControl -> errorChargeControl
       -> setTimChargeControl
       -> MAINCHARGE(PressureOK, Recharged))
 [] (Evt_I_BATT_MAIN_ISNOT_I_Charge
       -> setTimChargeControl
       -> MAINCHARGE(PressureOK, Recharged))
 [] (Evt_P_BATT_PRESS_1_OK ->
       if Recharged then SUPPLEMENTARYCHARGE
       else MAINCHARGE(true, Recharged))
 [] (Evt_Recharged ->
       if PressureOK then SUPPLEMENTARYCHARGE
       else MAINCHARGE(PressureOK, true))
 [] (Tau_SUN_OFF -> DISCHARGECONTROL)
```

Figure 9: CSP-code for the requirement in Fig. 6

Fig. 9 shows the formalisation of the requirement for the main charge phase of the power control. Charging starts with the main charge phase as soon as the sun rises. Within the time bounds defined by timer `TimChargeControl` the PTC has to adjust the charge current to the value `I_Charge`. Whenever this value is reached (within a certain precision) VVT-RT generates the event `Evt_I_BATT_MAIN_IS_I_Charge`. If process `MAINCHARGE` receives this event before timer `TimChargeControl` elapses, it resets the timer; if the timer elapses before the required charge current is reached a test error is signalled. If during main charging the charge current leaves the specified precision range around `I_Charge` due to switching operations, the timer is set again.

It is required that main charging continues until the amount of energy discharged during the last shade phase is recharged and the minimal absolute battery pressure is reached, or until the sun phase ends. In our CSP encoding of this requirement, the state variables "recharged" and "pressure O.K." are realised by boolean parameters of the process `MAINCHARGE`. The event `Evt_P_BATT_PRESS_1_OK` is generated by the test system whenever the required pressure range is reached. If this event reaches the process where the state variable `Recharged` is false, then the PTC should still be in the main charge phase. If the process is in state `Recharged`, then the supervision of supplementary charge begins. The test system generates the event `Evt_Recharged` if the integral of the charge current reaches the integral of the discharge current during the last shade phase. This way, arbitrary complex hybrid properties can be tested. The CSP code for the main charge is given in Fig. 9.

For conciseness, the formalisation of the requirement for the CCD camera thermal control is just sketched here. The temperature of the camera is simulated by an algorithm which runs in parallel with the test system. The CSP processes in Fig. 10 describe both the behaviour of the environment upon activation or deactivation of heat pads, and the required behaviour of the CCD heater control algorithm. Internal events (starting with `Evt_`) are used to synchronise the test system with the temperature simulation algorithm. For example, whenever at least one of both heaters (main or redundant) is active, the simulation has to choose a temperature curve which models the warm up of the camera (`Evt_warmer`). Similar to above, VVT-RT checks that certain bounds are respected. For example, if the first upper limit (`H_CCD_opt` + `H_CCD_tol`) is reached, the simulation component generates the event `Evt_warm`. In this case, the specification requires the deactivation of both heat pads within a certain time. Since the temperature should always stay within certain bounds,

```
ED = TC2_CCD(ok,false,false)
-- 1st parameter: temperature state, initially o.k.
-- 2nd parameter: heater MAIN state, initially off
-- 3rd parameter: heater REDU state, initially off


-- Thermal Control Testcase2, Inner1

TC2_CCD(temp,main,redu) =

      Swt_CCD_HEAT_MAIN_ON -> Evt_warmer
        -> TC2_CCD(temp,true,redu)
   [] Swt_CCD_HEAT_REDU_ON -> Evt_warmer
        -> TC2_CCD(temp,main,true)

   [] Swt_CCD_HEAT_MAIN_OFF ->
       (if (not redu) then Evt_colder -> SKIP
                      else SKIP);
       TC2_CCD(temp,false,redu)
   [] Swt_CCD_HEAT_REDU_OFF ->
       (if (not main) then Evt_colder -> SKIP
                      else SKIP);
       TC2_CCD(temp,main,false)

   [] Evt_too_warm -> errorTooWarm -> setTim1
       -> TC2_CCD(temp,main,redu)
   [] Evt_warm -> setTim1 -> TC2_CCD(warm,main,redu)
   [] Evt_ok -> resTim1 -> TC2_CCD(ok,main,redu)
   [] Evt_cold -> setTim1 -> TC2_CCD(cold,main,redu)
   [] Evt_too_cold -> errorTooCold -> setTim1
        -> TC2_CCD(temp,main,redu)

   [] elaTim1
       -> if ( (temp==warm and (main or redu))
            or (temp==cold and (not (main or redu))))
          then (errorTooLate -> setTim1
                -> TC2_CCD(temp,main,redu))
          else TC2_CCD(temp,main,redu)

   [] Swt_EXP_PWR_MAIN_ON
       -> (  (Swt_CCD_HEAT_MAIN_OFF
                -> Swt_CCD_HEAT_REDU_OFF -> SKIP)
          [] (Swt_CCD_HEAT_REDU_OFF
                -> Swt_CCD_HEAT_MAIN_OFF -> SKIP));
       Evt_constOK -> TC2_CCD_ON(main)
   [] Swt_EXP_PWR_REDU_ON
       -> (  (Swt_CCD_HEAT_MAIN_OFF
                -> Swt_CCD_HEAT_REDU_OFF -> SKIP)
          [] (Swt_CCD_HEAT_REDU_OFF
                -> Swt_CCD_HEAT_MAIN_OFF -> SKIP));
             Evt_constOK -> TC2_CCD_ON(redu)



TC2_CCD_ON(MAIN_REDU) =
      Swt_CCD_HEAT_MAIN_ON -> errorHeaterOn
        -> TC2_CCD_ON(MAIN_REDU)
   [] Swt_CCD_HEAT_REDU_ON -> errorHeaterOn
        -> TC2_CCD_ON(MAIN_REDU)

   [] Swt_EXP_PWR_MAIN_OFF ->
       (if (MAIN_REDU==main)
        then (Evt_colder -> TC2_CCD(ok,false,false))
        else TC2_CCD_ON(MAIN_REDU))
   [] Swt_EXP_PWR_REDU_OFF ->
        (if (MAIN_REDU==redu)
         then (Evt_colder -> TC2_CCD(ok,false,false))
         else TC2_CCD_ON(MAIN_REDU))
```

Figure 10: CSP-code for the requirement in Fig. 7

the event (`Evt_too_warm`) is generated whenever the upper bound is reached and leads to a test error.

The temperature control should only be active when the camera is off. If the camera is turned on (`Swt_EXP_PWR_xxx_ON`), then the waste heat suffices to warm it. Process `TC2_CCD_ON` checks that in this case the heat pads are never on. Event `Evt_constOK` causes the simulation to assume that the temperature of the operating camera is constant.

5 Results obtained with Automated Testing

The CSP test procedures were automatically executed by the test driver with the above configuration. The test results could both be interactively compared to the expected results and automatically evaluated. Since the whole testing process is completely automatic, a test coverage and cost efficiency could be achieved which vastly improves conventional methods. The main efforts in our approach were the adaptation of the TCP/IP based communication protocol to the specific formats used in the ABRIXAS project, and the exact formal coding of the system requirements.

A specific problem in the latter respect was that in some cases the required behaviour of the PTC was not yet defined. For example, the reaction to some combination of hardware faults was unspecified. Many of the problems we found in the initial phase were due to such incomplete specifications and, resulting from that, unpredictable control behaviour of the PTC. A feedback process with the system designers led to the documentation of requirements and an overall improvement of the implementation.

Small bugs were found in table entries and conflicting versions of the configuration data. An example is a wrong entry in a table which concerned the number of switches to be activated for opening the cover of the camera with the pyro. Thus, when executing the command to open the cover, only one of two necessary switches was toggled; therefore, additional user interaction would have been necessary to start the experiment.

We also found problems in the C code of the PTC software. For example, a wrong sign in the statement calculating the necessary charge duration had as consequence that the difference between discharge and recharge amount was always negative. Thus, in contrast to the above specification, the minimal absolut battery pressure was the only criterion for the charging. In case of a fault of the pressure sensors there was no redundant way to determine the necessary charge duration.

Furthermore, an incompatibility between hardware and software was revelead: The software to store updated parameter tables or new software in the E^2proms during the mission was too fast for the hardware. Although the software timinig was correct with respect to the specificiation of the E^2proms, the actually written bytes where sometimes different from the source. It turned out that the employed E^2proms did not meet their specification; the necessary delays between consecutive write commands were longer than expected. This error demonstrates the importance of hardware-in-the-loop testing at system level even if the software is proven to work correctly: The communication between hardware and software might introduce new problems.

Systematic testing thus mainly discovered problems in the exceptional behaviour, which "normally" never occurs. Such problems can not be detected by more conventional testing and debugging methods. For safety critical applications it is required that the systems is reliable even in unforeseen circumstances. Here, our method can be succesfully applied.

Since there was a close cooperation between development and testing team, most problems could be immediately solved. Incomplete specifications were updated in the requirements document and immediately changed in the formal specification. Since the testing was completely automatic, regression tests could be performed for the improved software after compilation and loading without further efforts.

In summary, the use of the tool VVT-RT in the ABRIXAS project was very successful. In the meantime, we applied our method to other systems: the DASA fault-tolerant computer FTC in the DMS-R project for the international space station[UKP 98], the AIRBUS cabin communication system director[PZ 99], a fail safe data transceiver for South African Railways[Sch 99], and others. Our experiences are that an early use of formal methods, already during the design and analysis phase of a project, is most promising. Besides improvement of overall design quality, the formalisation of system properties allows to use various development and analysis tools. Amongst these, the automatic testing method presented in this paper has the potential to become a standard method for quality assurance in aerospace applications.

References

[AD 97] P. Amthor, S. Dick: Test eines Bordcomputers für ein dezentrales Zugsteuerungssystem unter Verwendung des Werkzeuges VVT-RT. In: *7. Kolloquium Software-Entwicklung - Methoden, Werkzeuge, Erfahrungen: Mächtigkeit der Software und ihre Beherrschung* (Sep. 1997)

[But 97] B. Buth, M. Kouvaras, J. Peleska, H. Shi: Deadlock Analysis for a Fault-Tolerant System In Michael Johnson (Ed.): Algebraic Methodology and Software Technology. Proceedings of the AMAST'97, Sidney, Australia, December 1997, Springer LNCS 1349 (1997), 60-75.

[But 99] B. Buth, J. Peleska and H. Shi: Combining Methods for the Livelock Analysis of a Fault-Tolerant System. In A. M. Haeberer (Ed.): Algebraic Methodology and Software Technology. Proceedings of the 7th International Conference, AMAST 98, Amazonia, Brazil, January 1999. Springer LNCS 1548, pp. 124-139, 1998.

[FDR 94] Formal Systems (Europe), Ltd: Failures-Divergence-Refinement FDR 1.4, User Manual and Tutorial. (1994)

[HP98] A. Haxthausen and J. Peleska: Formal Development and Verification of a Distributed Railway Control System. In Proceedings of the 1st FMERail Workshop, Utrecht, The Netherlands, June 8th-9th 1998.

[HP 95] U. Hamer and J. Peleska: The CIDS A330/340 Cabin Communication System – A Z Application. Extended version of an article in J. P. Bowen and M. G. Hinchey (Eds.): "Applications of Formal Methods", Prentice Hall International Series in Computer Science (1995).

[Hen 88] M.C. Hennessy: Algebraic Theory of Processes. MIT Press (1988).

[Hoa 85] C.A.R. Hoare: Communicating Sequential Processes. Prentice Hall International (1985).

[Pel 96a] J. Peleska: *Formal Methods and the Development of Dependable Systems.* Bericht 9612, Christian-Albrechts-Universität Kiel, Institut für Informatik und Praktische Mathematik (1996)

[Pel 96b] J. Peleska: Test Automation for Safety-Critical Systems: Industrial Application and Future Developments. In: M.-C. Gaudel, J. Woodcock (eds.): *FME'96: Industrial Benefit and Advances in Formal Methods.* Springer LNCS 1051 (1996), pp. 39–59.

[PZ 99] J. Peleska and C. Zahlten: Test Automation for Avionic Systems and Space Technology (Extended Abstract). To appear in Softwaretechnik-Trends, Proceedings of the GI Working Group Test, Analysis and Verification of Software, Munich, February 4th - 5th 1999.

[Ros 98] A.W. Roscoe: The Theory and Practice of Concurrency. Prentice Hall International (1998).

[TSS 98] L. Twele, H. Schlingloff, H. Szczerbicka: Performability Analysis of an Avionics-Interface; Proc. IEEE Conf. on Systems, Man and Cybernetics; San Diego, N.J., pp. 499-504, (Oct. 1998)

[Sch 98] H. Schlingloff: Modelling Message Buffers with Binary Decision Diagrams; In: Proc. 3rd RelMiCS '97, Hammamet, Tunesia; Reappeared in: Using Relations in Computer Science, A. Jaoua, P. Kempf, G. Schmidt (eds.), Technical report 98/03, Univ. d. Bundeswehr, Munich (1998)

[Sch 99] Michael Schrönen: Methodology for the Development of Microprocessor Based Safety Critical Systems. Dissertation, Universität Bremen, BISS Monographs, Shaker Verlag, 1999

[SMH 98] H. Schlingloff, O. Meyer, Th. Hülsing: Korrektheitsanalyse eingebetteter Systeme; (this is the german version of this paper). Report, Univ. Bremen, (Juni 1998); http://www.tzi.de/~hs/Publikationen/

[Sne 88] H.M. Sneed: Software-Testen. Informatik-Spektrum 11:303-311 (1988).

[UKP 98] G. Urban, H.-J. Kolinowitz, J. Peleska: A Survivable Avionics System for Space Applications. Proc. FTCS-28, 28th Annual Symposium on Fault-Tolerant Computing, Munich, June 23-25, 1998, 372-381.