The Formal Derivation of Mode Logic for Autonomous Satellite Flight Formation

Anton Tarasyuk[1], Inna Pereverzeva[1,2](✉), Elena Troubitsyna[1], and Timo Latvala[3]

[1] Åbo Akademi University, Turku, Finland
{anton.tarasyuk,inna.pereverzeva,elena.troubitsyna}@abo.fi
[2] Turku Centre for Computer Science, Turku, Finland
[3] Space Systems Finland, Espoo, Finland
timo.latvala@ssf.fi

Abstract. Satellite formation flying is an example of an autonomous distributed system that relies on complex coordinated mode transitions to accomplish its mission. While the technology promises significant economical and scientific benefits, it also poses a major verification challenge since testing the system on the ground is impossible. In this paper, we experiment with formal modelling and proof-based verification to derive mode logic for autonomous flight formation. We rely on refinement in Event-B and proof-based verification to create a detailed specification of the autonomic actions implementing the coordinated mode transitions. By decomposing system-level model, we derive the interfaces of the satellites and guarantee that their communication supports correct mode transitions despite unreliability of the communication channel. We argue that a formal systems approach advocated in this paper constitutes a solid basis for designing complex autonomic systems.

Keywords: Autonomous flight formation · Formal modelling · Event-B · Refinement · Formal verification

1 Introduction

Nowadays the space industry puts increasingly strong emphasis on novel distributed satellite technology – formation flying. The most recent development in the field – autonomic formation flying – allows multiple satellites autonomously position themselves into a formation, efficiently maintain the formation and change it according to the mission and system requirements [4]. The satellites should function in a coordinated manner to guarantee safety (collision avoidance) and integrity (maintaining proximity) of the formation. Currently, the space industry is experimenting with the novel development and verification technologies that guarantee correctness and safety of autonomic flight formation.

The dynamic behaviour of the formation is defined in terms of modes – mutually exclusive sets of system behaviour [12]. The mode transition logic is complex because the mode transition conditions are defined by a variety of

F. Koornneef and C. van Gulijk (Eds.): SAFECOMP 2015, LNCS 9337, pp. 29–43, 2015.
DOI: 10.1007/978-3-319-24255-2_4

unpredictable factors – the relative position of the satellites, their health and environmental disturbances. The main challenge is to guarantee that, despite highly non-deterministic environment and the absence of the centralised control, the satellites perform mode transitions in a coordinated manner.

To address this challenge, we undertake a formal development of autonomous formation flying in Event-B. Event-B [1] is a state-based approach to correct-by-construction system development. It supports system-level reasoning about properties and behaviour. In particular, it allows us to define system invariants and verify them over all execution scenarios. The main development technique – refinement – supports stepwise construction and verification of complex specifications. We start the development by creating a high-level abstract specification, which is incrementally augmented with the detailed representation of requirements. Each refinement step is accompanied by proofs. When a sufficient level of details is reached, by relying on the modularisation extension [8,16], we can decompose the obtained specification into a number of independent components, i.e., arrive at the distributed architecture.

Event-B has an industrial strength automated tool support – Rodin platform [17]. The platform provides the developers with an integrated engineering environment, which supports modelling in Event-B as well as verification by proofs and model checking. Reliance on a common Event-B model to perform two types of verification helps us to address different aspects of system behaviour: model checking facilitates verification of dynamic properties of the inter-satellite communication, while proofs support reasoning about invariant properties of mode transition logic.

A combination of abstraction, refinement, proofs and decomposition as well as mature tool support makes Event-B a powerful framework for reasoning about behaviour of distributed systems. Development of Event-B and Rodin platform have been significantly advanced in the Deploy project [6]. Space Systems Finland has participated in the project and built a strong in-house expertise in formal modelling. Moreover, it has been encouraging the development of new features of Event-B, such as modularisation support, and validating them in practice. The company continues experimenting with the use of formal techniques in development of high assurance systems. In particular, this work builds on our previous experience in modelling of a reconfigurable on-board satellite system [21] as well as an attitude and orbit control system [10].

In this paper, we define the general patterns for ensuring coordinated mode transitions in the distributed autonomous systems. We analyse the inter-satellite communication, impact of failures and the mechanisms of losing and regaining the coordination. These patterns are integrated into the development of the overall formation flying specification. We start from an abstract model of the entire system, gradually introduce the details of the mode transitions, describe communication between the satellites, and finally decompose the system into independent sub-systems (i.e., satellites) and the communication link between them. Such an approach allows us to define precisely the properties that should hold at the different stages of mode transitions. We believe that the proposed approach offers a powerful technique for formal development and verification of autonomous distributed systems.

2 Satellite Flight Formation

Formation flying is a novel technology for a future generation of space missions. It offers benefits from both economical and technological perspectives. To perform a specific mission it is usually more cost-effective to compose a system from a number of simpler satellites rather than to develop a single dedicated spacecraft. Moreover, formation flying is easier to manage from the fault tolerance point of view – a failed satellite can be replaces by a similar one without the need to abort the entire mission. Finally, relatively small and simple satellites with specific functionalities are faster to develop and easier to maintain.

A few European missions have already exercised formation flying on low earth orbit, e.g., PRISMA [15] and TANDEM [3] missions. However, due to high gravity on lower orbits, it is unfeasible to establish precise relative positioning of satellites in the formation and therefore to perform sophisticated scientific observations. To overcome these limitations, the European Space Agency is currently developing PROBA-3 [4,14] mission (to be launched in 2017) that should demonstrate autonomous formation flying on the highly elliptical orbits. In this paper, we use the currently adopted configuration of the PROBA-3 mission.

The purpose of the PROBA-3 mission is to obtain the pictures of the inner solar corona. The mission consists of two satellites – the Coronagraph Spacecraft and Occulter Spacecraft. In simple non-technical terms, the Coronagraph is responsible for detecting sun position and taking pictures, while the Occulter should provide the shadow. To actually take the pictures, the spacecraft should maintain close proximity to each other (with relative distance from 25 to 250 m), which is achievable only in the low gravity region of the elliptical orbit.

Traditionally, in the space sector, the global system behaviour is specified in terms of modes – the mutually exclusive sets of system behaviour [12] – defined according to the mission and system requirements. The PROBA-3 mission has the following modes: STACK, MANUAL, OPERATIONAL and PARKING. The scheme of possible mode transitions is given in Fig. 1. The STACK mode is the initial mode where spacecraft are not yet separated. MANUAL is the safest mode that used for formation commissioning and during error recovery. OPERATIONAL and PARKING are "active" modes, at which the formation flying is performed. They are highly autonomous modes.

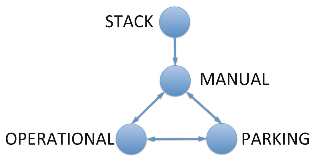

Fig. 1. System modes

The mode transition scheme given in Fig. 1 looks fairly simple. However, system autonomy and unreliability of the communication channel posses a significant challenge to coordination of the mode transitions. In the next section, we will investigate this problem in detail and derive the requirements guaranteeing correct implementation of the coordination.

3 Coordinated Mode Transitions in Autonomous Systems

Mode-rich distributed systems typically have the hierarchical architecture [9]. There is a dedicated component, called a Leader, that triggers mode transitions by broadcasting the id of the next target mode to all components in the system. The other spacecraft(s) is called the Follower. Essentially, after receiving the mode transition command from the Leader, the Follower performs the actions required to execute mode transition.

To make the decisions about mode transitions, the Leader should have the knowledge of the global state, i.e., its own state, the state of the Follower and the environment. In the case of the centralised or ground-coordinated distributed systems, such knowledge is always available. However, in case of the autonomous systems, due to the communication failure, the Leader might lack the knowledge of the Follower state. The active autonomous modes OPERATIONAL and PARKING require the maintenance of the close spacecraft proximity. Therefore, in the off-nominal situations of communication failure or relative position failure, the Leader has an incomplete knowledge of the global state and cannot make safe decision about the active mode transition, i.e., collision avoidance cannot be guaranteed under such uncertainty. This leads us to the first requirement imposed on the mode logic:

R1: The Leader can trigger transition to OPERATIONAL and PARKING modes only in the nominal situations.

To guarantee safety, in the off-nominal situations, we should merely take care of avoiding a collision, i.e., put the spacecraft at the safe distance from each other. In other words:

R2: In case of relative positioning failure, the Leader should trigger the MANUAL mode.
R3: In case of communication failure, the Leader and the Follower should enter the MANUAL mode. This is a self-triggered mode transition, i.e., the Leader and the Follower perform it independently upon detection of failure.

The requirement R3 leads to an important safety design constraint R4:

R4. Communication failures should always be detectable within the predefined time bound.

Now we have to address two problems: how to restore the coordination, and how to ensure that the communication failures are always detected.

Essentially, in the autonomous flight formation, failures result in the loss of the situation awareness that should be regained to restore coordination. Since the formation has the elliptical trajectory, each round includes the perigee phase (low gravity region of the orbit), where GPS measurement and mission ground control are available. The intervention of the ground control is required to restore the coordination between the spacecraft. Since there are no other means to restore the coordination, failure to restore coordination by the ground control implies failure of the entire mission. This observation implies our next requirement:

R5. In the MANUAL mode in the perigee phase the ground control should control both the Leader and the Follower to restore coordination between them. Upon successful completion of this, the control is passed to the Leader and a transition to an active mode PARKING or OPERATIONAL becomes enabled.

Communication is a critical aspect in ensuring coordination and safety of the autonomous formation flying. The spacecraft communicate with each other to coordinate not only mode transitions but also orbital manoeuvring. The orbital manoeuvring is structured by four phases: the perigee phase, the apogee phase (high gravity region of the orbit) and the intermediate preparation phases. The phase transitions are performed according to the predefined logic.

According to *R4,* we should ensure that communication failures can be promptly detected by each spacecraft. Eventual but slow detection might cause a collision. This rules out the asynchronous communication and implies the following requirements:

R6: Communication should be periodic.
R7. Communication timeouts are set for sending and receiving messages during each communication period. No communication during the timeout is treated (by a spacecraft) as a failure of the communication link.

The analysis presented above shows that ensuring correctness of coordinated mode transitions in autonomous formation flying is a challenging engineering task. To approach it in a systematic rigorous way, in the next section we present our modelling framework – Event-B.

4 Modelling and Refinement in Event-B

Event-B is a state-based formal approach that promotes the correct-by-construction development paradigm and formal verification by theorem proving. In Event-B, a system model is specified using the notion of an *abstract state machine* [1]. An abstract state machine encapsulates the model state, represented as a collection of variables, and defines operations on the state, i.e., it describes the dynamic behaviour of a modelled system. The important system properties to be preserved are defined as model invariants. A machine usually has the accompanying component, called context. A context may include user-defined carrier sets, constants and their properties (defined as model axioms).

The dynamic behaviour of the system is defined by a collection of atomic *events*. Generally, an event has the following form:

$$\mathsf{e} \;\widehat{=}\; \textbf{any}\ a\ \textbf{where}\ G_e\ \textbf{then}\ R_e\ \textbf{end},$$

where e is the event's name, a is the list of local variables, and (the event *guard*) G_e is a predicate over the model state. The body of an event is defined by a *multiple* (possibly non deterministic) assignment to the system variables. In Event-B, this assignment is semantically defined as the next-state relation R_e. The event guard defines the conditions under which the event is *enabled*, i.e., its body can be executed. If several events are enabled at the same time, any of them can be chosen for execution nondeterministically.

Event-B employs a top-down refinement-based approach to system development. A development starts from an abstract specification thatnondeterministically models the most essential functional system behaviour. In a sequence of refinement steps, we gradually reduce non determinism and introduce detailed design decisions. In particular, we can add new events, refine old events as well as replace abstract variables by their concrete counterparts.

The consistency of Event-B models – verification of model well-formedness, invariant preservation as well as correctness of refinement steps – is demonstrated by discharging the relevant proof obligations. The Rodin platform [17] provides tool support for modelling and verification. In particular, it automatically generates all required proof obligations and attempts to discharge them. When the proof obligations cannot be discharged automatically, the user can attempt to prove them interactively using a collection of available proof tactics. Moreover, a user can also rely on verification by model checking supported by ProB plug-in.

Recently the Event-B language and the tool support have been extended with a possibility to define *modules*. Modules are components containing groups of callable atomic operations [8,16]. Modules can have their own (external and internal) state and invariant properties. An important characteristic of modules is that they can be developed separately and, when needed, composed with the main system. Since decomposition is a special kind of refinement, such a model transformation is also a correctness-preserving step that has to be proven.

A module description consists of two parts – *module interface* and *module body*. A module interface is a separate Event-B component that consists of the external module variables, the module invariants, and a collection of module operations, characterised by their pre- and postconditions. In addition, a module interface may contain a group of standard Event-B events. These events model autonomous module thread of control, expressed in terms of their effect on the external module variables. In other words, they describe how the module external variables may change between operation calls. A formal development of a module starts with the deciding on its interface. Once an interface is defined, it cannot be changed in any manner during the development. This ensures that a module body may be constructed independently from a system model that relies on the module interface. A *module body* is an Event-B machine. It implements the interface by providing a concrete behaviour for each of the interface operations. To guarantee that each interface operation has a suitable implementation, a set of additional proof obligations is generated.

A general strategy of a distributed system development in Event-B is to start from an abstract centralised specification and incrementally augment it with design-specific details. When a suitable level of details is achieved, certain events of the specification are replaced by the calls of the interface operations. The variables become distributed across the modules. As a result, a monolithic specification is decomposed into the separate modules and communication mechanisms are introduced explicitly. In the next section, we demonstrate how to use such a general refinement strategy to model the mode logic of the autonomous flight formation.

5 Modelling Satellite Flight Formation in Event-B

In this section, we outline the overall Event-B development of formation flying and discuss the most challenging aspects. The full Event-B development can be found in [20].

Abstract Specification. The outline of the initial model FFS_abs is shown in Fig. 2. It abstractly represents the Leader's mode transitions as well as failure occurrence and handling. The variable $cur\_mode\_leader \in$ MODES represents the current mode of the leader spacecraft, while the variable $prev\_mode\_leader \in$ MODES stores its previous mode. Here the set MODES = {STACK, MANUAL, OPERATIONAL, PARKING} contains all system modes that can be entered by the satellites.

The events StackSeparation, ModeTransitionManual and ModeTransition-Autonomous model all possible mode transitions (i.e., entering a target mode) of the Leader. The satellite might also maintain the current mode as modelled by the event RemainCurrentMode. This event will be further refined by the events modelling the manoeuvres within the currently active mode, i.e., by the phase transitions. The mode transition rules (defined in Fig. 1) are specified as the model invariant properties, where the auxiliary function $nextMode$ defines all possible successor modes for any system mode.

The failure detection is modelled by the event FormationFailureDetection that assigns value TRUE to the flag *failure*. This will enable the event ModeTransition-Manual, i.e., according to the requirements *R2* and *R3* trigger transition to the Manual mode. In this paper, we keep an abstract representation of the failure detection and recovery procedures.

First Refinement: Modelling Follower Behaviour. Our first refinement step introduces the similar data structures and events for the Follower spacecraft. At this stage, we do not constrain mode transitions, i.e., there is no coordination

```
Machine FFS_abs
Variables $cur\_mode\_leader,\ prev\_mode\_leader, failure$
Invariants $cur\_mode\_leader \in \text{MODES} \wedge prev\_mode\_leader \in \text{MODES} \wedge failure \in \text{BOOL} \wedge$
$cur\_mode\_leader \neq \text{STACK} \Rightarrow cur\_mode\_leader \in nextMode(prev\_mode\_leader) \ldots$
Events
StackSeparation ...                  // spacecraft separation
ModeTransitionManual ...             // transition to MANUAL mode
ModeTransitionAutonomous $\widehat{=}$         // autonomous mode transition
  any $mode$
  when $cur\_mode\_leader \in \{\text{OPERATIONAL}, \text{PARKING}\} \wedge$
       $mode \in nextMode(cur\_mode\_leader) \wedge failure = \text{TRUE} \Rightarrow mode = \text{MANUAL}$
  then $cur\_mode\_leader, prev\_mode\_leader := mode, cur\_mode\_leader$   end
RemainCurrentMode ...               // spacecraft remains in the current mode
FormationFailureDetection $\widehat{=}$       // failure detection
  when $cur\_mode\_leader \in \{\text{OPERATIONAL}, \text{PARKING}\} \wedge failure = \text{FALSE}$
  then $failure := \text{TRUE}$   end
FormationFailureHandling ...  // failure handling in MANUAL mode
```

Fig. 2. Satellite flight formation: initial model

between the satellites. It will be introduced upon modelling the inter-satellite communication link in the next refinement.

Modelling Mode-level Communication. Modelling and verification of the communication mechanism is the central part of our development. In our second refinement step, we focus on modelling the *mode-level* communication – the high level communication between satellites – used to coordinate the transitions between the modes of the formation.

To trigger mode transitions in the nominal conditions (according to *R1*) the Leader sends the unique id of the target mode to the Follower via the inter-satellite communication link. Upon delivery of the message, the Follower performs transition into the requested mode and sends the acknowledgement to the Leader. Upon receiving the acknowledgement, the Leader also makes the transition to the new target mode. Essentially, our communication is a simple version of the sliding window protocol with the one-place buffers. However, our communication is *two-layered* – it is split into the higher-level mode communication and the lower-level phase communication – which makes proof-based verification of correctness challenging.

According to the *R6* and *R7* the communication between the satellites is periodic and has the predefined maximal delay. This requirements allow us to fulfil the *R4*, i.e., ensure detection of failures. In our models, we could have represented the communication failure as a non-deterministic change of the inter-satellite link status as it is typically done in Event-B modelling. However, we have chosen another approach and decided to model the link failure as an explicit notification that is delivered to both satellites. This approach closely resembles the timeout mechanism used to detect communication failures. Moreover, it facilitates specification of invariant properties of the inter-satellite communication.

To model communication according to the defined requirements, we introduce the models of one-place incoming and outgoing buffers for each of the satellites:

– *modeOutgoing*: leader's outgoing buffer for target mode

```
LeaveOperationalMode $\widehat{=}$
 any   $mode\_id$
 where $cur\_mode\_leader = \mathrm{OPERATIONAL} \wedge cur\_mode\_follower = \mathrm{OPERATIONAL} \wedge$
       $modeOutgoing = \varnothing \wedge modeDeliveryReport = \varnothing$
       $mode\_id \in \{\mathrm{PK}, \mathrm{MAN}\} \wedge failure = \mathrm{FALSE} \wedge \ \ldots$
 then  $modeOutgoing := \{mode\_id\}$  end
ModeCommunicationLink $\widehat{=}$
 any   $msg$
 where $modeOutgoing \neq \varnothing \ \wedge \ msg \in modeOutgoing \cup \{\mathrm{LOST}\}$
 then  $modeOutgoing, modeDeliveryReport, modeIncoming := \varnothing, \{msg\}, \{msg\}$  end
EnterParkingModeLeader $\widehat{=}$
  refines ModeTransitionManualLeader, ModeTransitionAutonomousLeader
  when  $(cur\_mode\_leader = \mathrm{MANUAL} \ \vee \ cur\_mode\_leader = \mathrm{OPERATIONAL}) \wedge$
        $modeDeliveryReport = \{\mathrm{PK}\}$
  with  $mode = \mathrm{PARKING}$
  then  $cur\_mode\_leader, prev\_mode\_leader := \mathrm{PARKING}, cur\_mode\_leader$
        $modeDeliveryReport := \varnothing$   end
EnterParkingModeFollower $\widehat{=}$
  refines ModeTransitionManualFollower, ModeTransitionAutonomousFollower
  when  $(cur\_mode\_follower = \mathrm{MANUAL} \ \vee \ cur\_mode\_follower = \mathrm{OPERATIONAL}) \wedge$
        $modeIncoming = \{\mathrm{PK}\}$
  with  $mode = \mathrm{PARKING}$
  then  $cur\_mode\_follower, prev\_mode\_follower := \mathrm{PARKING}, cur\_mode\_follower$
        $modeIncoming := \varnothing$    end
```

Fig. 3. Flight formation: second refinement

– *modeIncoming*: follower's incoming buffer for target mode
– *modeDeliveryReport*: leader's acknowledgment delivery buffer

To demonstrate communication during the mode transition, in Fig. 3 we show several events representing the transition to the PARKING mode. The event LeaveOperationalMode models initiating a mode transition by the Leader while being in the OPERATIONAL mode. In this case, two transitions are enabled – either to the PARKING or to the MANUAL mode. At this stage the choice between them is modelled nondeterministically ($mode\_id \in \{\mathrm{PK}, \mathrm{MAN}\}$). The mode-level communication link is modelled by the event ModeCommunicationLink that can either deliver the issued instructions, i.e., the next mode id stored in the *modeOutgoing* buffer, or "deliver" the LOST message, that abstractly models detection of a communication failure by the timeout. Finally, two events EnterParkingModeLeader and EnterParkingModeFollower represent transition into the PARKING mode by the leading and the following satellites correspondingly. Transitions to other modes are modelled in the similar way.

In case a satellite receives LOST message, it independently initiates the transition into the MANUAL mode (according to the requirement *R3*). The events modelling off-nominal conditions can be found in the full development [20].

Note that at this stage we still rely on the availability of the global knowledge, i.e., we are not yet ready to decompose the system into a distributed model. Indeed, to guarantee correctness of the coordination, we explicitly allow the Leader to assess the current mode of the Follower before it schedules the next mode transition (see the second guard of LeaveOperationalMode). This modelling trick helps us to postulate important properties of the coordination. In the later refinement step, we abandon this abstraction and refine the global state representation into the model of a distributed state space. The Leader will rely solely on the communication to assess state of the Follower.

Formal development in Event-B allows us to formulate and verify a number of the *mode consistency* properties. They are defined and proved as system invariants. For example,

$$\begin{aligned} &cur\_mode\_leader \neq cur\_mode\_follower \Rightarrow \\ &\qquad cur\_mode\_leader = prev\_mode\_follower \vee \\ &\qquad\qquad cur\_mode\_follower = prev\_mode\_leader \qquad (1) \end{aligned}$$

$$modeOutgoing \neq \varnothing \Rightarrow cur\_mode\_leader = cur\_mode\_follower \qquad (2)$$

Property (1) stipulates that the satellites can be *at most one* mode transition ahead (or behind) each other. Moreover, the divergence can be only in the case, when the satellites are actually performing the transition to the next mode (one satellite has already made a transition and another is still in process of doing it). In case, when the Leader's outgoing buffer is not empty (i.e., it is ready to initiate a mode transition), the satellites are in the *same mode* (as defined by property (2)).

We can also formally define the connection between the formation modes and the state of the inter-satellite link, i.e., the values of the satellite buffers as shown by properties (3) and (4):

$$\begin{aligned} &modeIncoming = \{\text{PK}\} \wedge cur\_mode\_leader = \text{PARKING} \Rightarrow \\ &\qquad prev\_mode\_leader = cur\_mode\_follower \qquad (3) \end{aligned}$$

$$\begin{aligned} &modeIncoming = \{\text{PK}\} \wedge cur\_mode\_leader \neq \text{PARKING} \Rightarrow \\ &\qquad cur\_mode\_leader = cur\_mode\_follower \qquad (4) \end{aligned}$$

In the similar way, properties (5) and (6) below describe relationships between the Leader's acknowledgement buffer and the current Follower mode.

$$\begin{aligned} &modeDeliveryReport = \{\text{PK}\} \wedge cur\_mode\_follower = \text{PARKING} \Rightarrow \\ &\qquad prev\_mode\_follower = cur\_mode\_leader \qquad (5) \end{aligned}$$

$$\begin{aligned} &modeDeliveryReport = \{\text{PK}\} \wedge cur\_mode\_follower \neq \text{PARKING} \Rightarrow \\ &\qquad cur\_mode\_leader = cur\_mode\_follower \qquad (6) \end{aligned}$$

$$failure = \text{TRUE} \wedge modeOutgoing \neq \varnothing \Rightarrow modeOutgoing = \{\text{MAN}\} \qquad (7)$$

Property (7) describes dependency between the communication failure and the leader's outgoing buffer. In case, when a failure has been detected, the Leader can only command transition to the MANUAL mode. The properties describing dependency between the communication failure and other buffers are formulated in the similar way. In total, we have formulated and proved more than 40 invariant properties of mode-level communication.

Modelling Phase-level Communication. In the OPERATIONAL and PARKING modes the designed orbital routine for the nominal operation consists of four different phases. The phase-level communication is used to coordinate the transitions between the phases. Our third refinement step described in [20] focuses on modelling it. The phase-level communication is more complex than the mode-level communication. To ensure synchronisation and safety of orbital manoeuvring, not only the Follower but also the Leader sends the acknowledgements of the phase-transition message delivery. The modelling style is similar to the previous step. Modelling the fine-grained phase-level communication resulted in a complex specification that contains a hierarchy of properties ensuring consistency not only at level of mode-logic but also at the level of phases. At this refinement step we formulated and proved more than 70 logical properties described the phase-level communication.

Modelling Communication With the Ground. As described in Sect. 3 (requirement $R5$), the ground control should intervene when the spacecraft have lost the coordination. The ground control generates and sends telecommands to the leading satellite. A telecommand might consist of the next target mode and a number of orbits to be performed (if the target mode is OPERATIONAL).

Decomposition. As a result of the previous refinement steps, we have arrived at a detailed centralised model of the flight formation system. In the final refinement step, we employ decomposition approach provided by the Modularisation plug-in and derive a distributed architecture of the flight formation mode logic. We decompose centralised model into two independent components representing the satellites and the communication link. A graphical representation of the system after decomposition refinement is given in Fig. 4. The previous, more abstract model is refined by a machine CommLink and two modules for satellites – Leader and Follower. To enable two-way communication between the spacecraft, the generic interfaces of the modules Leader and Follower describe a collection of externally callable operations. The machine CommLink, that models all types of communication, invokes these operations in the bodies of its events.

The machine CommLink imports two module interfaces – Leader and Follower. The events ModeCommunicationLink, LeaderFollowerPhaseComm, FollowerLeader-PhaseComm, ManualTC and FailureTC are refined by the communication link events. The rest of the events (e.g., mode transitions) are now becoming a part of the autonomous processes of the Leader and Follower modules. Thus, the events modelling the behaviour of the Leader satellite are now refined by the Leader's interface events, while the events related to the follower satellite are refined by the corresponding events of the Follower's interface. Similarly, the variables of abstract model are now refined by the variables of Leader and Follower modules.

Discussion. The presented development focuses on studying the coordination aspect of mode transitions. The further refinement steps can be performed to

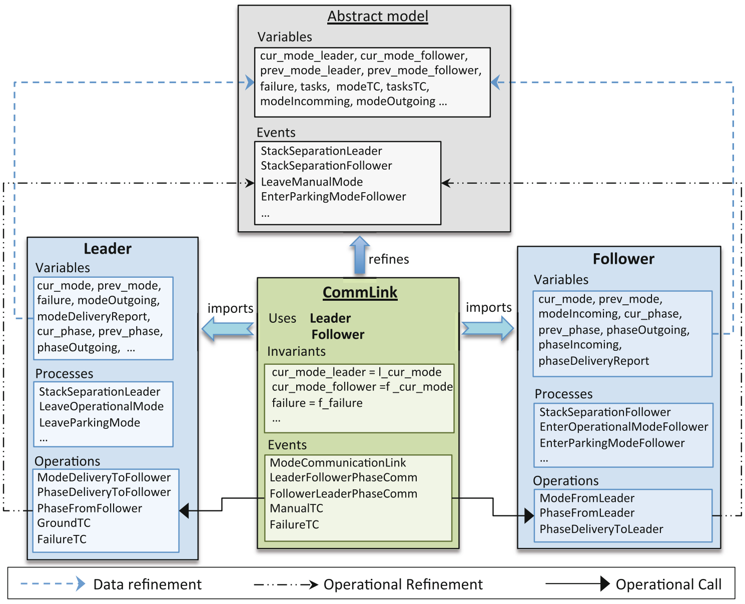

Fig. 4. The decomposition refinement

model the internal architecture and detailed behaviour of each spacecraft. The properties of coordination were formulated as model invariants that are related to the high-level requirements informally defined in Sect. 3.

To verify correctness of the models we discharged more than 2200 proof obligations. Around 86 % of them have been proved automatically by the Rodin platform and the rest have been proved manually in the Rodin interactive proving environment. Most of the manual proofs were related to proving the invariants describing the mode/phase consistency properties as well as logical connections between the formation modes and the state of the inter-satellite link. However, some of the manually proved POs were identical (as some mode/phase transitions are identical up to the variable/constant names) and the corresponding proofs were reused. Moreover, a significant amount of manually proved POs were promptly discharged by calling a single external prover (e.g., predicate prover or SMT-prover). Thus, actual amount of manually proved POs was less that 10% of the total amount, which is a very good result for such a complex model.

The considered formation of two satellites is inspired by a PROBA-3 mission. However, the created model of inter-satellite communication is rather generic and can be used to model formations with similar architecture. For instance, it can be adopted for modelling formations with an arbitrary number of follower satellites by using support for decomposition into an an indexed collection of modules.

6 Conclusion and Related Work

In this paper, we have presented a formal development and verification of autonomous formation flying in Event-B. We have proposed a novel approach to modelling unreliable asynchronous communication in Event-B. We formally define requirements that ensure coordinated mode transitions of autonomic flight formation. The Rodin platform was used to automate modelling and verification efforts. The framework has demonstrated a good scalability and provided us with a suitable basis for designing such a complex distributed system. We believe that the following aspects were critical for the success of the development. The first aspect is support for refinement and decomposition. It allowed us to start from a centralised succinct system model and derive complex and tangled communication mechanism gradually in a correctness preserving way. The second aspect, is a support for highly iterative development provided by the Rodin platform. Proofs provided us with an immediate feedback on our models and helped to spot many intricate interdependencies between modes, phases and effects of faults. Finally, our experience has also shown that it is essential to combine proving with model animation and model checking [11] because they help to validate the model and prevent introducing deadlocks in intertwined communication. For instance, we used ProB model checker to validate invariants, i.e. check that requirements are correctly represented by the logical formulae.

In our modelling we focused on verifying consistency of mode logic in presence of satellite and communication failures. The autonomy of satellites required complex handshaking schemes to ensure proper coordination between them. As a future work, it would be interesting to extend the proposed approach to modelling multiple satellite formation and explore the properties of autonomic coordination in such complex multi-party environment.

Event-B follows top-down refinement strategy that allowed us to formulate system-level properties defining mode consistency conditions. An alternative approach is to rely on high-level programming paradigms to facilitate the design of the decentralized control. In particular, it would be interesting to compare the approach proposed in this paper, with the actor-based design approach [2].

In our previous work [10] we have studied development of centralised mode-rich systems. We have demonstrated how to derive a specification of an Attitude and Orbit Control System – a generic system of satellites by refinement in Event-B. However, due to centralised nature of the system, the model was much simpler with smaller set of properties. The development presented in current paper, is much more complex. The resultant model contains a large set of invariants describing in details the relationships not only between modes and phases but also the effect of failures at different stages of communication.

The application of Event-B formalism to describe safety layered architectures for Unmanned Aerial Vehicle (UAV) control system has been presented in [5]. The authors unfold the system architecture by a refinement, while in our work we rely on refinement technique to model and derive complex and tangled communication mechanism for mode transitions.

Formal validation of the mode logic and, in particular, fault tolerance mechanisms of satellite software has been studied by Rugina et al. [19]. They have investigated different combinations of simulation and model checking. The similar approach that relies on collaborative modelling and simulation was also used to validate the spacecraft's functional behaviour and command/control FDIR [18]. Though the authors combine model checking with simulation while we combine proofs and model checking, they draw the similar conclusions: a combination of different techniques as well as reliance on abstraction are required to model such complex systems as satellites.

The work [7] reports on modelling and analysis effort of a satellite platform using the COMPASS toolset that is based solely on model checking. The work aims at combining different safety analysis techniques and modelling to verify correctness, safety and dependability of a single satellite. A quantitative evaluation of dependability of navigation of satellite systems using probabilistic model checking is presented in [13]. In our work, we aimed at studying logical aspects of the impact of failures on system behaviour.

Webster et al. [22] adopted the agent concept to verify decision-making aspects of Autonomous Unmanned Aircraft control. Specifically, they choose the particular Rules of the Air and verified that the behaviour of the system does not violate them. Verification is performed using agent model checker AJPF. This is an interesting approach for verification and can be seen as complementing our work. In our approach, formal development helps not only verify but also identify the properties of the system.

Acknowledgements. This work is supported by Cyber Trust program http://www.digile.fi/cybertrust.

References

1. Abrial, J.R.: Modeling in Event-B. Cambridge University Press, Cambridge (2010)
2. Agha, G.A., Kim, W.: Actors: a unifying model for parallel and distributed computing. J. Syst. Archit. **45**(15), 1263–1277 (1999)
3. Buckreuss, S., Werninghaus, R., Pitz, W.: German satellite mission TerraSAR-X. In: Radar Conference 2008, pp. 1–5. IEEE (2008)
4. Castellanic, L.T., Llorente, S., Fernandez, J.M., Ruiz, M., Mestreau-Garreau, A., Cropp, A., Santovincenzo, A.: PROBA-3 mission. In: SFFMT 2013 (2013)
5. Chaudemar, J.C., Bensana, E., Seguin, C.: Model based safety analysis for an unmanned aerial system. In: Dependable Robots in Human Environments (2010)
6. DEPLOY: IST FP7 IP Project. http://www.deploy-project.eu/
7. Esteve, M., Katoen, J., Nguyen, V.Y., Postma, B., Yushtein, Y.: Formal correctness, safety, dependability, and performance analysis of a satellite. In: ICSE 2012, pp. 1022–1031. IEEE (2012)
8. Iliasov, A., Troubitsyna, E., Laibinis, L., Romanovsky, A., Varpaaniemi, K., Ilic, D., Latvala, T.: Supporting reuse in Event B development: modularisation approach. In: Frappier, M., Glässer, U., Khurshid, S., Laleau, R., Reeves, S. (eds.) ABZ 2010. LNCS, vol. 5977, pp. 174–188. Springer, Heidelberg (2010)

9. Iliasov, A., Troubitsyna, E., Laibinis, L., Romanovsky, A., Varpaaniemi, K., Väisänen, P., Ilic, D., Latvala, T.: Verifying mode consistency for on-board satellite software. In: Schoitsch, E. (ed.) SAFECOMP 2010. LNCS, vol. 6351, pp. 126–141. Springer, Heidelberg (2010)
10. Iliasov, A., Troubitsyna, E., Laibinis, L., Romanovsky, A., Varpaaniemi, K., Ilic, D., Latvala, T.: Developing mode-rich satellite software by refinement in Event-B. Sci. Comput. Program. **78**(7), 884–905 (2013)
11. Leuschel, M., Butler, M.: ProB: an automated analysis toolset for the B method. Int. J. Softw. Tools Technol. Transf. **10**(2), 185–203 (2008)
12. Leveson, N., Pinnel, L.D., Sandys, S.D., Koga, S., Reese, J.D.: Analyzing software specifications for mode confusion potential. In: Human Error and System Development, pp. 132–146 (1997)
13. Peng, Z., Lu, Y., Miller, A., Zhao, T., Johnson, C.: Formal specification and quantitative analysis of a constellation of navigation satellites. CoRR abs/1402.5599 (2014)
14. Peters, T.V., Brancob, J., Escorial, D., Castellani, L.T., Cropp, A.: Mission analysis for PROBA-3 nominal operations. Acta Astronautica **102**, 296–310 (2014)
15. Prisma Satellites. http://www.prismasatellites.se/
16. Rodin: Modularisation plug-in. http://wiki.event-b.org/index.php/Modularisation_Plug-in
17. Rodin: Event-B platform. http://www.event-b.org/
18. Rugina, A., Leorato, C., Tremolizzo, E.: Advanced validation of overall spacecraft behaviour concept using a collaborative modelling and simulation approach. In: WETICE 2012, pp. 262–267. IEEE Computer Society (2012)
19. Rugina, A.E., Blanquart, J.P., Soumagne, R.: Validating failure detection isolation and recovery strategies using timed automata. In: EWDC 2009 (2009)
20. Tarasyuk, A., Pereverzeva, I., Troubitsyna, E., Latvala, T.: The formal derivation of mode logic for autonomous satellite flight formation. Technical Report 1137, Turku Centre for Computer Science (2015)
21. Tarasyuk, A., Pereverzeva, I., Troubitsyna, E., Latvala, T., Nummila, L.: Formal development and assessment of a reconfigurable on-board satellite system. In: Ortmeier, F., Lipaczewski, M. (eds.) SAFECOMP 2012. LNCS, vol. 7612, pp. 210–222. Springer, Heidelberg (2012)
22. Webster, M., Fisher, M., Cameron, N., Jump, M.: Formal methods for the certification of autonomous unmanned aircraft systems. In: Flammini, F., Bologna, S., Vittorini, V. (eds.) SAFECOMP 2011. LNCS, vol. 6894, pp. 228–242. Springer, Heidelberg (2011)