Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Formal Methods by Stealth: The INSPEX Experience†

Richard Banach*[1] | Joseph Razavi[1] | Olivier Debicki[2] | Suzanne Lesecq[2]

[1]Department of Computer Science, University of Manchester, Oxford Road, Manchester, M13 9PL, UK
[2]MINATEC, Commissariat à l'Énergie Atomique et aux Énergies Alternatives, 17 Rue des Martyrs, F-38054 Grenoble Cedex, France

Correspondence
*Richard Banach, Department of Computer Science, University of Manchester. Email: richard.banach@manchester.ac.uk

Abstract

INSPEX is an *INtegrated Smart sPatial EXploration* system. It relies on a family of sensors, like automated vehicles do, to provide enough information to a digital system for it to make reliable inferences about the location of obstacles and other impediments in its environment. Unlike the automated vehicle case, INSPEX is minaturised, because it is intended for lightweight applications and for portable use by humans, e.g. visually impaired persons navigating outdoors (among many similar use cases). The complexity of this hardware-focused system merited the introduction of formal methods during its (essentially conventionally structured) development. The aim was to improve the dependability of parts of the implemented system, and to estimate system characteristics via modelling and calculation that could not be obtained experimentally within the scope of the project. The paper overviews the experience of the very much human-in-the-loop use of formal techniques in the INSPEX project, and focuses particularly on the human issues that impacted the cooperation between the conventional techniques and formal methods.

KEYWORDS:
Obstacle detection systems, INSPEX, Visually impaired or blind, Formal methods, Dependability, Event-B, Rodin, ProB, PRISM.

1 | INTRODUCTION

Recent history has been punctuated by moments in which progress in information technology has made such an impact on everyday life, that everybody feels that things have changed. In this generation, innovations in sensing technologies, machine learning, and the wide availability of portable computing in the form of smartphones are making the kind of revolutionary changes that personal computing and the internet did a generation ago. Increasingly, the copious deployment of sensors into a myriad different environments enables computation to influence the behaviour of both systems and people in ways unthinkable only a few short years ago. One particular instance of this is in the application of sensors in the automotive industry, in contexts such as assisted parking and the automatic management of headlights, not to mention autonomous driving assistance in the form of adaptive cruise control and similar systems.

In this atmosphere of rapid innovation in the automotive field, obstacle avoidance systems have, of necessity, been a hot research topic in the recent past. As vehicle automation has become increasingly sophisticated, and autonomous vehicles (up to autonomy level 4 say[1,2]) are anticipated to appear on our streets within a couple of years, so the need for increasingly detailed information in the vehicles' intelligent control systems has grown too. Vehicles are quite bulky and robust, so can afford the deficit of large, heavy, complex, power hungry avoidance systems. As such systems have grown more capable, so a wider variety of use cases can be envisaged for them, not all of which can tolerate the size, weight and power costs of the vehicle systems. Fortunately, the trend towards minaturisation of the relevant technologies has led to the increasing feasibility of much more lightweight implementations and a wider range of use cases.

The INSPEX Project[3,4,5,6,7] aimed to leverage these possibilities to create devices at wearable scale, in order that they could assist humans, such as visually impaired or blind (VIB) persons — many other use cases besides VIB can be imagined too. Like the bulkier vehicle systems, the INSPEX

[†] The INSPEX project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730953. The work was also supported in part by the Swiss Secretariat for Education, Research and Innovation (SERI) under Grant 16.0136 730953. We thank them for their support.

technology relies on a family of sensors, and making the best use of these significantly increases the complexity of the system compared with existing single sensor VIB assistive equipment. This complexity merited the judicious deployment of formal techniques during the development, and this is the subject matter reported in this paper. Formal techniques were used in the verification of the data pathway and in the *a priori* modelling of power management and battery lifetime — areas that emerged during the course of the project as potentially benefiting from formal scrutiny for the benefit of the project as a whole. What makes our experience novel is that the formal techniques were applied in a way that was quite different from the well understood top-down methodologies favoured in the formal methods textbook (and research) literature. We explore all this in more detail in the body of the paper.

The rest of this paper is as follows. It starts with background and context for INSPEX as a whole. In Section 2 we briefly review the autonomous vehicle navigation landscape. In Section 3 we examine the INSPEX idea in more detail, and review some of the use cases its capabilities offer. This leads to Section 4, in which we focus on the VIB use case, the primary concern of the INSPEX work, in more detail. Section 5 is the bridge to the involvement of formal methods in the project, in which we introduce the INSPEX consortium, showing the (im)balance between resources devoted to hardware issues and those devoted to formal methods. In Section 6 we describe the preparatory work that was done before the main engagement between the practical system development on the one hand, and the formal modelling and verification work on the other, was carried out. In Section 7 we discuss a number of contextual factors that proved necessary to be taken into account to generate a fruitful collaboration between disciplines with such diverse methods of working. Reporting on these methodological considerations and on their impact, often falling outside the main technical focus and outside typical formal methods deployment methodologies, constitutes the major contribution of this paper. We take some space to explore the differences between typical approaches and what we did in INSPEX, all in the context of the resource imbalance noted already. Section 8 introduces the technical elements of the INSPEX Project where formal techniques were applied. The next sections examine these in more detail: Section 9 looks at the data acquisition pathway in fair detail, explaining the pragmatic elements of the work as well as exploring the simpler formal models involved. Section 10 looks at the issues around power management, and the way that these impacted what could be done at the formal modelling and analysis levels, in particular with the goal of battery lifetime estimation in mind. The positive outcomes for the INSPEX Project of the engagement with formal techniques in these two areas are described, noting that the battery lifetime estimation work constituted a significant advance in this area. Section 11 concludes.

Earlier descriptions of some specific technical aspects of the use of formal techniques in INSPEX have been explored in existing conference publications. In [6], there is a broad overview of the INSPEX Project from a medical instruments perspective. In [8] and [9], preliminary experience of the modelling of the data pathway was reported; this was subsequently extended to the lower level models included in the present paper. The paper [10] presents an overview of the project, and of early experience in the application of formal techniques, for a mixed commercial/technical audience. In [11], the role of formal techniques as experienced in the INSPEX Project, is discussed as an element of a formal development review strategy.

The account in the present paper benefits from the wisdom of hindsight, and so is more complete regarding technical detail. Moreover, the emphasis on the human dimension of the cooperation between formal and conventional development threads —the ‘human in the loop’ aspect of the work, that is, in the authors’ opinion, no less important than the purely technical one— is exclusive to this paper.

2 | AUTONOMOUS VEHICLE NAVIGATION

The inspiration for INSPEX comes from the growing capabilities of autonomous vehicles. Conceived in fiction over three decades ago, autonomous vehicles have become a serious reality within the last decade, especially within the last five years. These days, it is no novelty to see in the news stories of autonomous cars navigating with apparent effectiveness through the streets of some city. It is said that such vehicles will be performing real tasks in urban environments ‘in the wild’, relatively soon. Such vehicles get their navigational capabilities by deploying a large number of sensors, and processing all the data that they produce via sophisticated algorithms whose goal is to produce a navigation strategy for the vehicle that human beings would regard as being appropriate for the circumstances.

There are a large number of societal issues surrounding vehicles which are truly autonomous, from attribution of liability when something really bad happens,[1] to massive job losses as the need for drivers diminishes. Our interest, though, is in the technological aspects.

The left part of Fig. 1 shows a schematic of an autonomous vehicle system. The car is equipped with LiDAR, with RADAR, with ultrasound (US) as well as a camera and vision system. There is also an inertial measurement unit (IMU) not included in the figure. There will also be a GPS system together with a mapping application.

In a real driving context, the data from the sensors is gathered, timestamped, and sent to a data fusion application [13,14,15,16], that calculates, using Bayesian estimation techniques, an *occupancy grid* [14,15]. This divides the 3D space in front of the vehicle (and the sides and back too) into cubical pieces, for each of which, an estimate is calculated of the probability that there is an obstacle of some sort there. The right part of Fig. 1 gives an idea of how this works. The car in front of the instrumented car is detected by the long range RADAR, and the person standing at the side of the

---

[1] And really bad things *really are* starting to happen, e.g. the tragic UBER fatality story [12], which is by no means unique these days.

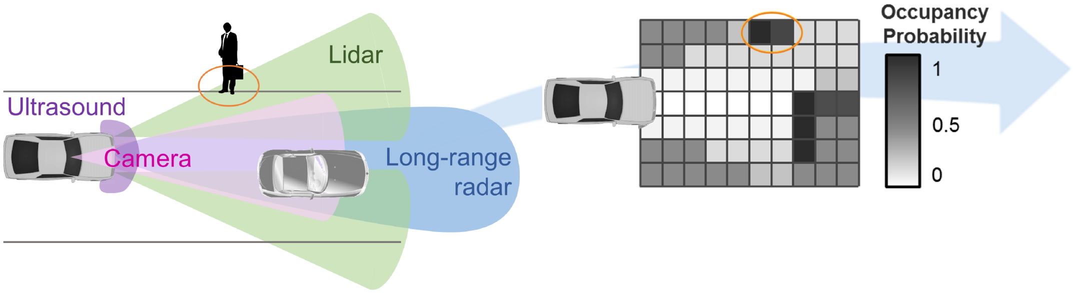


FIGURE 1 Sensor data mapped into an occupation grid in automotive obstacle detection.

road is detected by the LiDAR sweeping to the side. The occupancy grid in the figure indicates the higher probability of there being an obstacle in these positions by the denser shading in the relevant squares.

As noted earlier, the large size, weight and power of a vehicle permits a full suite of useful sensors to be carried. In a minaturised scenario, this is no longer possible, and the main casualty is the vision system. Cameras in themselves are not the problem — these days they can be tiny, but the computations needed in a vision system capable of analysing a real world image in real time would far exceed what is possible in a lightweight portable device (assumed not connected to the cloud). This frames the way that the INSPEX concept was developed.

3 | THE INSPEX CONCEPT

Given that the contemporary trend in sensors of all kinds towards ever greater minaturisation, the driving concept behind INSPEX was to intercept this trend to re-evaluate the autonomous vehicle obstacle detection and navigation architecture. The result is the idea of a small and lightweight device, one which minimises the resources needed for efficient occupation grid estimation. The last point is not a trivial one. The standard calculations for deriving occupation grids[17] are expensive and thus energy intensive. INSPEX benefits from a particularly efficient implementation suitable for low power systems[18]. Moreover, the fusion algorithm employed has to be very flexible and forgiving. Contributing sensors may fail; or they may work at different rates; or they may perform poorly under various weather conditions. The fusion system makes the best estimate it can on the basis of the information available. INSPEX thus offers the potential to enhance the way that navigational and positioning challenges are addressed in a wide variety of application areas.

Fig. 2 gives an indication of a selection of potential use cases for a technology like INSPEX. The figure is divided into several sections.

On the right we see some human centred applications. We see the VIB use case which forms the focus of the initial INSPEX development. Accompanying this are other use cases. The first responder use case covers situations like that of firefighters in smoke-filled (and thus opaque) enclosures, who struggle to gain an appreciation of their environment and of the obstacles that it might contain. Evidently, they can benefit from non-visual cues about obstacles around them. The severely disabled who have impediments to the processing of the visual information from their surroundings can also benefit from INSPEX. There are also use cases for the non-disabled, such as cyclists and pedestrians — in the highly polluted mega-cities of the world, it is regularly the case that smog levels reach a level that visibility is seriously restricted.

At the top of the figure we have some use cases connected with instrumentation. Modern distance measuring tools typically make use of a laser beam whose reflection is processed to come up with the numerical result. This works well when the target is a hard, flat surface. For more uneven, textured surfaces, the reading given may be subject to errors, so an INSPEX enhancement to such instruments could perform better in a wider variety of situations. INSPEX could also improve the working of mobile mapping applications. Modern 3D environment capture instruments already use LiDARs to accurately measure the characteristics of their environment. So, integrating more sensors into such instruments has the potential to improve their performance by better recording of elements of the environment that are transparent to LiDAR. Autofocus applications in high end camera systems are similar, especially when we contemplate their combination with the sophisticated image processing algorithms that are increasingly seen these days.

At the bottom of Fig. 2 we have some use cases mainly concerned with navigation inside large, enclosed environments. Currently, highly automated factories can feature assembly lines consisting of hundreds of robots, each performing a specific task, within tightly constrained parameters. Such facilities have to be planned and laid out with a great deal of precision because the robots within them have limited intelligence. Adding INSPEX capabilities to them enhances the ability of the robots to accurately orient themselves relative to other significant elements of their working environment, easing the planning needed. Things are even more challenging if the robots in question are mobile. Mobile robots need to accurately

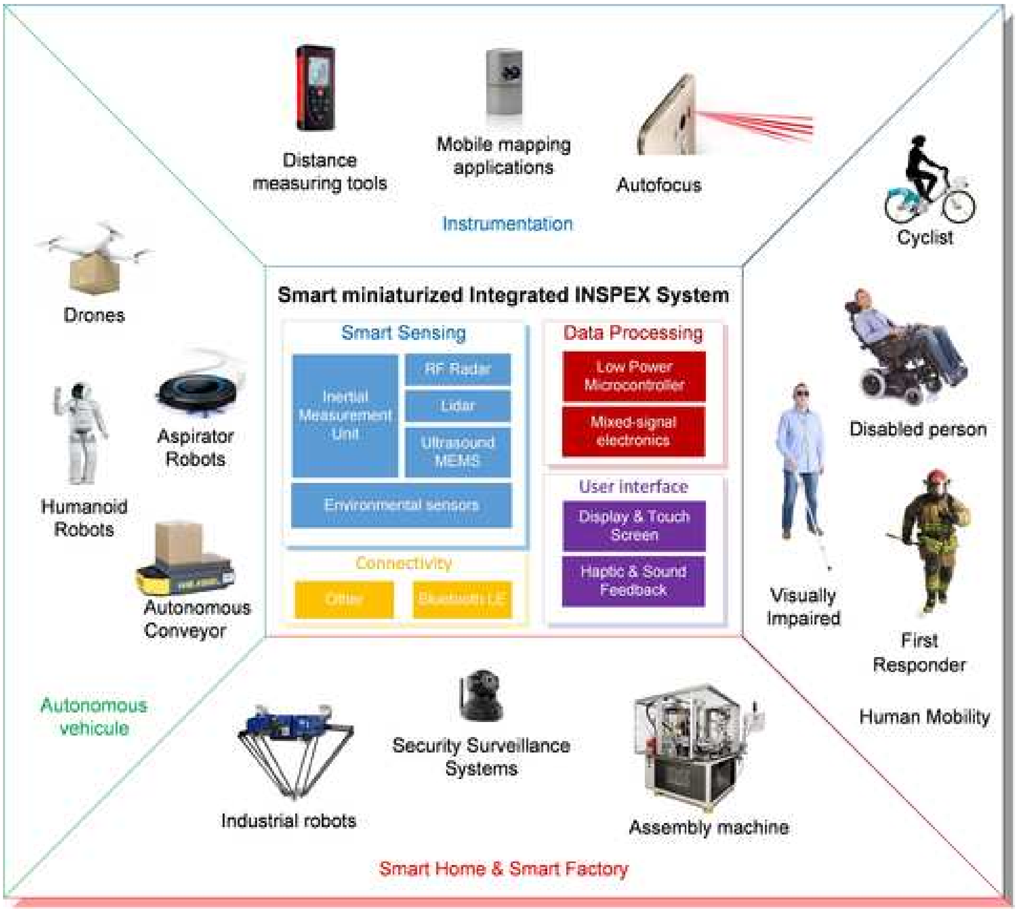

Distance measuring tools
Mobile mapping applications
Autofocus
Instrumentation
Cyclist
Drones
Smart miniaturized Integrated INSPEX System
Smart Sensing
Inertial Measurement Unit
RF Radar
Lidar
Ultrasound MEMS
Environmental sensors
Connectivity
Other
Bluetooth LE
Data Processing
Low Power Microcontroller
Mixed-signal electronics
User interface
Display & Touch Screen
Haptic & Sound Feedback
Aspirator Robots
Humanoid Robots
Autonomous Conveyor
Disabled person
Visually Impaired
First Responder
Human Mobility
Autonomous vehicule
Industrial robots
Security Surveillance Systems
Assembly machine
Smart Home & Smart Factory

FIGURE 2 Potential INSPEX use cases.

orient themselves within their environments, in particular so that they can avoid doing harm to humans who may be in their vicinity. Security surveillance systems, shown in the figure, traditionally relying on infra-red sensors, can also benefit from the enhanced detection of INSPEX.

On the left of the figure we see some use cases in the autonomous *small* vehicle domain. When a vehicle is small, the ability to tolerate the high weight, size and power consumption that large road vehicles can easily cope with, disappears. Small airborne drones are a prime case in point, and have demands that are every bit as exacting as those for the VIB use case. The use of drones is increasing, especially within urban environments, and so they will increasingly need to manoeuvre within quite tightly constrained spaces, without colliding either with the fixed environment, or with each other. INSPEX capabilities can clearly address the sensing requirements of this use case. Humanoid robots and autonomous domestic robotic assistants of various kinds also have technical needs of this kind, and are restricted in their activities by battery longevity. A typical floor cleaning robot will, these days, use a fairly hit and miss approach to finding parts of the floor to clean. Gradually, it learns where obstacles are, and builds an approximate map to improve its navigation subsequently. More accurate navigation capabilities acquired from equipment such as INSPEX can contribute to shortening the learning curve in these circumstances.

Among this plethora of possibilities, the INSPEX Project itself focused on creating a device for the VIB community, and this is described in more detail in the next section. The project's goal was to create an advanced prototype at Technology Readiness Level 4 (TRL4). This was to show the viability of the concept, without attracting the significant additional costs associated with producing a fully commercial product.

4 | THE VIB USE CASE

According to the World Health Organisation, the number of VIB persons numbers 285 million and, due to an aging global population along with the debilitating health conditions that greater age brings, this number will double in twenty years or so[19]. The normal assistive device used by VIB people is the white cane, and the primary use case addressed by the INSPEX project was the creation of a clip-on enhancement to the white

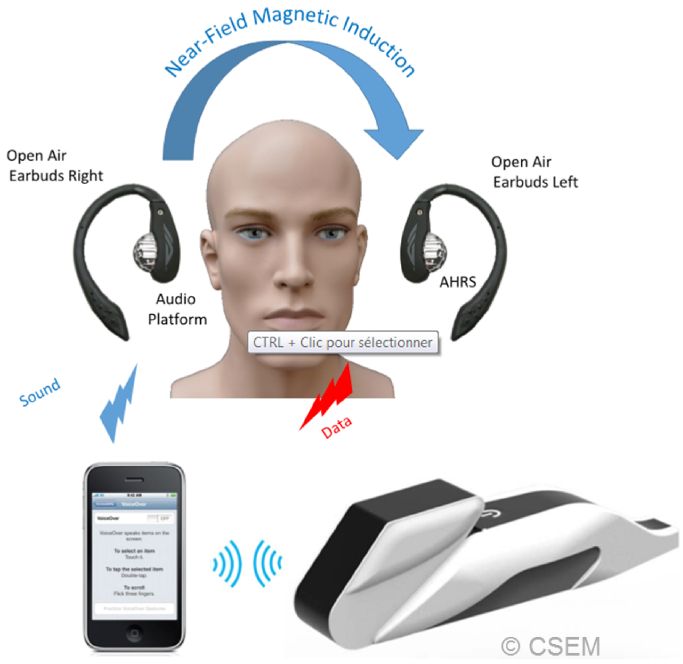


FIGURE 3 The complete INSPEX system for the VIB use case.

cane that would significantly increase its utility. The wide range of conditions that someone may encounter in the outdoor environment makes the design of a system that gives really useful feedback to users into a highly nontrivial challenge.

The argument for improvement of this simple piece of equipment, the white cane, its many years of successful use notwithstanding, is compelling. While visually impaired people are experts at using the tactile feedback from the cane to understand their surroundings at ground level, especially along familiar routes, a white cane, by its nature, cannot give any warning about obstacles at chest or head level. Whether they are low branches in natural settings, or signs, cordons, or temporary fences in an urban environment, such obstacles are extremely common, and an unexpected collision with them has the potential to cause serious injury. In fact, such injuries occur frequently[20]. VIB persons may wear headgear, even if for sartorial reasons they might prefer not to, to offer at least some cushioning in case of unanticipated collisions.

Other hazards need to be avoided by VIB persons too. Danger lurks in the approach to holes in the surface, whether empty, or filled with water (or other nonsupporting material). Staircases are frequently found in the urban landscape, and these, particularly descending sets, constitute a significant risk factor for the VIB. (It is worth remarking that escalators, although similar to stairs, are much less risky for VIB persons, since they emit a typical 'escalator rumble' which the VIB person can hear and take into account.)

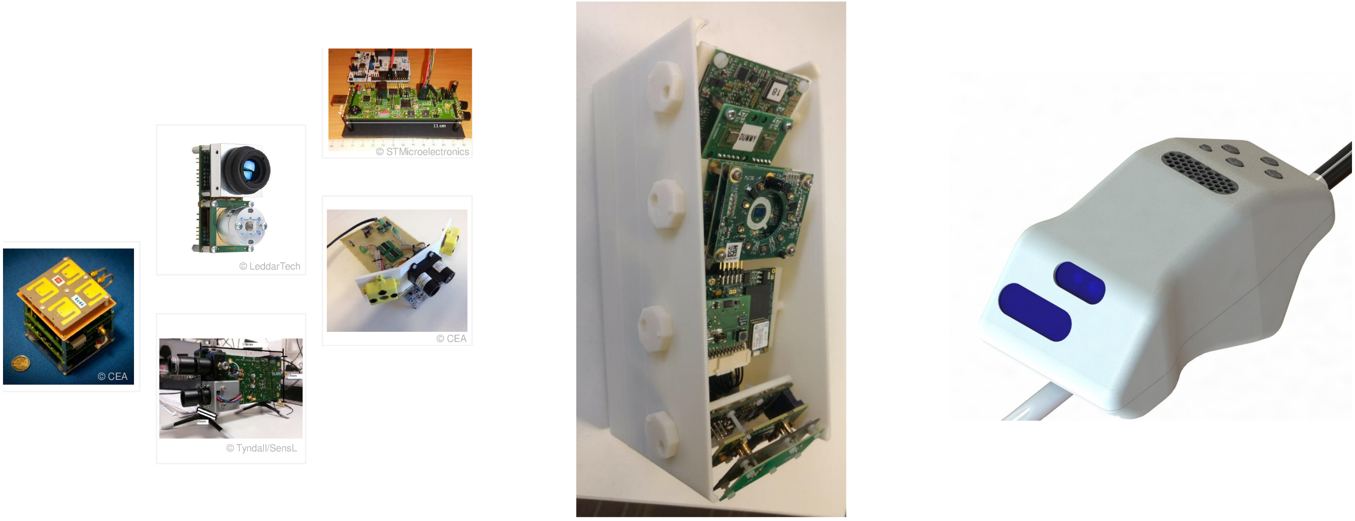
In the face of this range of challenges, the aim of INSPEX was to supplement the ground level sensory information with information about the environment higher up, provided by minaturised sensors inspired from the automotive use case. Specifically, INSPEX employs short and long range LiDAR sensors, an ultrawideband radar (UWB) and a MEMS ultrasound sensor. For all of these sensors, the hardware development strands of INSPEX were occupied with integrating all of the physics processing on-chip, and the needed computation on supporting ASICs. The wide range of sensors is intended to address, as well as is possible, the effect of varying lighting and weather conditions, and to overcome the deficiencies of individual sensing technologies which impede their effectiveness under particular conditions.

The information from these sensors is fused in the add-on device and is combined with orientation data provided by an Inertial Measurement Unit (IMU). The integrated information is communicated to the user's smartphone by Bluetooth, and the smartphone computes a binaural sound image, which is fed back to the user through earbud headphones. These themselves contain another IMU to account for the user's head movements (these being independent of the movement of the white cane), and this IMU data is taken into account in computing the sound image so that the binaural image conveyed to the user is stable in 3D space.[2] Fig. 3 shows the components of the complete INSPEX VIB system.

Fig. 4 shows the progress of the hardware development strand of the INSPEX Project towards its TRL4 goal. P0 is the initial prototype, consisting of early versions of individual sensors, developed individually and tested in isolation. P1 is the first version to integrate everything into a single

---

[2] Recent research has made clear that the perception of binaural sound is significantly modulated also by the relative orientation between the head and the torso, so this additional information was also taken into account in the latter stages of the INSPEX Project.



P0: initial prototype P1: first integrated prototype P2: final integrated prototype

FIGURE 4 Hardware prototypes for the INSPEX VIB system.

package, in which partly minaturised versions of the sensors are packaged into a rather large box, for early integrated testing. P2 shows the ultimate goal, where everything has been fully minaturised and integrated to the greatest extent possible, and is packaged into the production quality enclosure shown.

The idea of creating an assistive device to help VIB persons to navigate safely, that functioned by translating information gained via wave physics based sensing, into aural or tactile information communicable to a VIB person, did not originate with INSPEX. Early prototypes that embark on this road, and which are visible in the literature, include[21,22], and among existing systems present in the market there are Smartcane[23], Ultracane[24], Bawa[25], Rango[26], etc.

Aside from these initiatives which are focused on making white canes smart, there are many devices that are deployed via mechanisms such as headwear, devices that clip on to the user's belt, or devices that can be strapped on to the chest or hung round the neck. However, the reaction from the VIB community to these proposals has been resoundingly negative, as they are perceived to be stigmatising to a significant degree. And besides these products that originate in the SME sector, the increasing societal assistance available for citizens that suffer from various disabilities, spurs an increasing interest also from large corporations, e.g.[27].

The key observation about earlier systems is that they typically use a single sensing channel to probe the 3D environment in front of the user. This simplification avoids the complexities and costs of data fusion, at the price of forcing the toleration of any weak elements or 'blind spots' inherent in the physics of the given sensor. INSPEX is one of the earliest VIB multi-channel assistive devices, and more and more such devices are being proposed nowadays.

5 | THE INSPEX PROJECT AND FORMAL METHODS

INSPEX was an EU Horizon 2020 ICT *Smart Systems Integration* (SSI) project. It ran for the three years from the start of 2017 till the end of 2019. The project consortium members are listed in Table 1, along with their main responsibilities. The table makes clear that the overwhelming weight of effort in the project was devoted to hardware oriented issues. In particular, far more people were devoted to these hardware issues than the two Manchester authors of the present paper who were the sole people working on formal techniques.

The emphasis in SSI projects is on achieving significant increases in the functionality of increasingly complex hardware configurations. So the overwhelming emphasis in the project was on overcoming the technical challenges in bringing the equipment to life. Everything, from the detailed properties of the sensors and their physical signals, to the minaturisation of the constituent devices to the extent that that is possible, to the movement of these signals and their isolation from one another, to the significant software challenges inherent in a design of such complexity, to the properties of the main INSPEX device container, with its need for robustness and durability under a variety of weather conditions and handling

TABLE 1 The INSPEX Consortium

Participant	Acronym	Main focus
Commis. Énerg. Atom. Énerg. Alt. (FR)	CEA	RADAR; Firmware; Coordinator
University of Manchester (UK)	UNIMAN	Formal methods
Cork Institute of Technology (IR)	CIT	Long range LiDAR
STMicroelectronics SRL (IT)	STI	Ultrasound
Centre Suisse d'Electr. Microélectr. (CH)	CSEM	Short range LiDAR; General Processors
University College Cork, Tyndall (IR)	Tyndall	Enclosure
Université de Namur (BE)	UNamur	Ethics and legal
GoSense (FR)	GoSense	Headset; Audio
SENSL Technologies Limited (IR)	SENSL	Long range LiDAR

regimes while at the same time permitting each sensor to transmit its signal and receive the corresponding reflection, all had to take priority in the project.

The increasing complexity of these integrated systems inevitably magnifies the challenge of ensuring correct operation (by comparison with the situation for earlier, structurally simpler systems). The recognition of this fact very early in the preparation stages of the project proposal, was the direct cause of the involvement of formal techniques in the INSPEX Project. Nevertheless, the priority given to hardware integration in SSI projects meant that formal verification aspects, though recognised as being helpful for achieving the project's ultimate goals, received relatively little attention in the proposal preparation activity, and in the early stages of the project.

6 | FORMAL ASPECTS: PREPARATORY WORK

Given the enormous focus on the hardware side of the project, it was felt that the hardware elements needed to gain some momentum before engagement with verification technologies could proceed. Accordingly, and since it was not clear, early on, what style of modelling and verification would prove most beneficial in the long run, the initial period of the verification strand of the project was spent in trialing a number of verification tools of different kinds, in order to have a variety of capabilities available to meet the demands of the project as it evolved.

The tools that were chosen to trial had to meet a number of criteria.

- They had to be well established, with a significant track record that could support 'production' use.
- They had to be free. The available budget for the project was almost entirely taken up with salaries and with supporting the hardware side, so there was no funding available for purchasing commercial tools.

Invariably, besides the above, familiarity also guided the choice. In the end, ten tools were trialed, summarised in Table 2. As is clear from Table 2, the tools chosen were mainly model checking tools (Pro-B, FDR, Uppaal, PRISM, NuSMV), and model based state machine tools (Rodin, Atelier B, TLA+, Perfect Developer). The final tool, for checking C code (BLAST), was added later, for reasons that will be explained below.

In the end, the tool that received the most use during the project was Rodin, particularly when combined with its Pro-B plugin to handle the model checking end of things. This enabled the same source code to be used for both activities. A major reason for Rodin's usefulness was its incorporation of modern SMT solvers for prover support[38,39,40]. These often reduced the proof task for a model to close to push button simplicity, which speeded up the work significantly. Another bonus for Rodin was its visually appealing user interface, which proved very convenient for communicating with the rest of the project and for presenting the results to project reviewers unfamiliar with formal methods.

Besides Rodin, PRISM also proved useful in the latter phases of the project when numerical estimates of stochastic properties of white cane use were needed. The various formalisms supported by PRISM offer a range of possible approaches for this, including ones that aligned well with the Rodin work.

TABLE 2 Tools Trialed for INSPEX Modelling and Verification

Tool	Chief Characteristics
ProB [28]	Event-B model checking
FDR [29]	CSP deadlock checking
Uppaal [30]	Timed finite automata
PRISM [31]	Probabilistic model checking
NuSMV [32]	Model checking
Rodin [33]	Model based
Atelier B [34]	Model based
TLA+ [35]	Model based
Perfect [36]	Model based
Blast [37]	C source code checking

7 | CONTEXTUAL CONSIDERATIONS FOR FORMAL MODELLING AND VERIFICATION (M&V)

At the outset, the hardware side of the project and the formal side had little conception of how the two activities would fit together — it was the nicest kind of blind date. As already mentioned, the impetus lay with the hardware development, and clearly, trying to alter the already well established hardware *modus operandi* to better align with formal methods views about working practices, would have been hugely counterproductive. This was especially the case when the hardware goals of the project were already very demanding (and were intentionally so). In any case, the formal methods side was massively outnumbered, as we noted already. The result of all this was that the usual recommendations about bringing formal methods into a 'conventionally structured' project environment, were rendered impotent. Hence our title.

7.1 | Specific Issues Concerning Formal/Conventional Interplay

In this section we give an account of a number of issues that affected the way that formal techniques were integrated with the critical path of the INSPEX Project, the impact that they had, and the way that they were dealt with.

[A] (Top-down) An optimistic formal methods practitioner, habituated to typical textbook top-down formal techniques, would be forgiven for supposing that the whole INSPEX development process would be characterised by a generally top-down approach. In reality, this was impossible for INSPEX and always would be so for a project of this type. The overwhelming brunt of the work needed for INSPEX concerned the characteristics and behaviour of the hardware, and the consequences for the whole system that these imply. This is an essentially bottom-up way of working. Physics and engineering are uncompromising. If the fundamental behaviour of a device is not compatible with some desirable prospective properties conjectured for it at a user requirements level, there is no use for a top level specification to insist on it. So INSPEX was driven by unavoidable bottom-up considerations, and higher levels of design had to be adapted to inescapable low level constraints.

[B] (Documentation) Following on from **[A]**, an optimistic formal methods practitioner, habituated to typical top-down formal approaches, would be forgiven for supposing that the INSPEX development process would be characterised by a reasonably full suite of design documentation. Especially so as the VIB use case makes it *de facto* a class 1 (assistive) medical device[3] — given that the European standard for software for medical devices[41,42] demands that copious documentation be maintained on all aspects of the product lifecycle, from initial requirements capture, to rollout of fault fixes for devices in the field. Such documentation, at the design level, would considerably ease the formal modelling and verification (M&V) activity, since it would make the essence of the design more immediately perspicuous, and initial models derived from it could then be more easily reconciled with the implementation. In fact in INSPEX the situation was quite patchy. One reason for this was the aggressive timescales and tight resources of the project. There simply was no time to document everything in an ideal waterfall manner. So parts of the implementation that reused previously developed components were left largely undocumented. By contrast, parts requiring cooperative design from multiple partners did generate enough documentation to enable separately developed components to be designed with sufficient consistency that they worked together properly. The latter possibility evidently made the M&V activity easier than the former. This uneven level of documentation is rendered more understandable given the resource limitations of the project, and the fact that INSPEX was intended to achieve only an advanced (TRL4) prototype, and and not a commercial level product, making the demands of the software standard[41] less pressing.

[C] (Pragmatics) Following on from **[A]** and **[B]**, the way that formal M&V activity became deployed in INSPEX was a rather *ad hoc* affair, guided (with feedback from the formal side) by intuition from the hardware and firmware teams regarding where in the project formal M&V might be the most productive. In the event, this proved to be a very appropriate approach. The *ad hoc* nature of the process entailed a mixture of top-down and bottom-up approaches, depending on context. The combination of approaches varied very widely among the different parts of the project. Top level desiderata could be enunciated, but with far too little detail to allow useful design to proceed therefrom. Complementary contributing low level facts could be brought out to inform the design process, and with this greater focus, greater precision could augment the top-down view. All being well, top-down and bottom-up approaches could be brought together consistently in the middle.

---

[3] Currently, there is some debate about whether such assistive devices ought to be classified thus, taking into account the burden that certification imposes on development, and the desire to not unduly impede innovation in this space.

[D] (Lateness) An extreme incarnation of the preceding occurs when, towards the end of a project like INSPEX, the development of a component is completed only just in time to meet the final deadline (or perhaps worse, even after the formal deadline). In such a case, there is little or no time for a formal modelling process to reflect adequately on the properties of that component, and to contribute positively to the robustness of the overall design (as regards that component). Obviously such a state of affairs weakens the overall contribution that a formal approach can make to design dependability, but does not at all nullify it.

[E] (IP issues) As with all EU projects, there was a comprehensive grant agreement and a detailed consortium agreement put in place in INSPEX to adequately protect partners' background IP and to properly organise the foreground IP generated. Such measures are not, though, guaranteed to cut any ice with partners' individual managements. If local management is not prepared to sanction access to local IP to other partners, it can stop cooperation in its tracks and significantly impact progress. For a while, this issue impeded progress in modelling and verification when access to actual code was needed, but it was resolved in due course.

[F] (Language) Commonality of language is vital. The same words mean different things to different specialisms,[4] and lack of awareness of subtle differences in interpretation can have, *in extremis*, disastrous consequences.[5] To avoid this, there needs to be enough overlap between the expertise of different consortium members, and they need to be sufficiently alert to this problem, that such subtle differences can be spotted early, and addressed. In the case of INSPEX, the fact that the first author of the present paper used to be a physicist, enabled such potential problems to be nipped in the bud, given that he had had copious acquaintance with all the relevant lexicons. There is no substitute for extensive experience with multiple disciplines.

[G] (Clues) It was vital to participate in as many of the discussions as the geographically distributed nature of the project consortium allowed. Even when we were just listening to other participants, people would, without being aware of it, say things that provided clues for developing a well structured formal modelling approach. Of course, commonality of language, noted in **[F]**, and flexibility of approach noted in **[A]** and **[B]**, were vital in successfully picking up such clues.

[H] (Cooperation) As many comments above have shown, a flexible and pragmatic approach was crucial in order that the formal M&V activity in INSPEX added value to the project rather than acting as an impediment. Insisting on rigid textbook application of top-down methods would have yielded little in the reality of the constant interplay between low level and high level considerations in INSPEX. Equally important was a preparedness by all participants to take on board others' concerns and to work towards common goals, this being perhaps the most significant ingredient in the success of the project.

[I] (Optimisation) The natural way to manipulate data at an abstract formal modelling level is a metaphor for data copying. However, the extremely optimised nature of low level embedded device code means that data is often manipulated via pointers, and is moved between subsystems by pointer passing and not by copying. Moreover, navigating through data structures is frequently accomplished via address arithmetic and not by referencing or dereferencing the pointers themselves. Modelling such code faithfully (let alone actually generating it automatically) implies, in effect, building a formal model of the memory and of the way that data is mapped onto it, *as part of the formal model of the application* (absent a highly tuned formal toolchain that is specifically adapted for the task). Building a bespoke instance of such a thing within the constraints of INSPEX would have been prohibitively costly, so could not be pursued, despite the obvious appeal in terms of code coverage in principle, of doing so.

[J] (Optimism) The grass is always greener on the other side: lack of familiarity with a discipline can easily lead to over-optimism about what it can achieve. It is by now a truism that top-down formal techniques can lead to generated code that is of critical system quality (e.g.[34]). Generating code automatically had some appeal for the application side, but doing it properly would have demanded changes in development practices, and would have led to impacts on cost and on time, that the resources of INSPEX could not sustain, so this avenue was not pursued in any depth in the project.

[K] (Abstraction) A remedy for **[J]** that naturally suggests itself is to use source code verification tools. Unfortunately, when this was tried, it proved difficult to connect the level of abstraction at which the tool naturally worked with the level of abstraction needed to capture the high level properties of interest for verification, so this line of investigation did not yield the results that were hoped for.

[L] (Demoability) As noted, we were working in a hardware oriented environment, in which expected outputs are tangible objects that can be picked up, manipulated in the fingers, and hooked up to a display rig of some kind for demonstration purposes. Formal methods are not naturally adapted to such modes of appraisal, so the relatively highly visual UI of Rodin proved to be a real boon in this regard. Of course, on a fairly recent machine, Rodin's response is too instantaneous to show off the dynamics of the tool well, so we 'slugged' Rodin by the simple expedient of running several instances concurrently. This provided enough contention for CPU and memory, that the performance of the instance being used for the demo, was sufficiently slowed that (for instance) the colour changes of the flags against individual POs in the PO pane were discernable individually as the machine was committed after each edit during the demo.

---

[4] A good example is the word 'variable', which means different things to physicists and engineers on the one hand and to computer scientists and logicians on the other. The word 'model' also comes to mind, for which the same observations apply. Even in different technical subfields of conventional engineering, phrases such as 'response time' can carry totally incompatible interpretations.

[5] As a fairly extreme example we may cite the Mars Climate Orbiter story[43,44], in which the spacecraft crashed onto Mars because a software module was created using Imperial units whereas its output was interpreted as being in SI units. Fortunately, no humans were harmed during this incident.

7.2 | Pragmatism vs. Existing Approaches

As indicated above, the conventional way of embedding formal methods into systems development presumes an *a priori* strategy for the cooperation between formal and non-formal techniques. There are a number of works in the literature that discuss alternative approaches, and recommend possibilities. A large part of the debate has traditionally focused on the perceived costs of using formal methods compared with the costs of not doing so (and the widely accepted by now view that judicious deployment for formal methods does not incur additional costs, while nevertheless increasing overall quality). Early works covering this kind of focus include[45,46,47,48,49,50]. As the formal methods field expanded, so did the range of approaches that were made available. So survey papers that appeared later had more scope for comparing the pros and cons of diverse techniques. We can cite[51,52,53,54,55,56,57].

Among the approaches that are perhaps closest to ours here is the so-called 'lightweight formal methods' approach, daing back to the mid 1990s[58,59,60,61]. In these, the 'standard' way of deploying formal techniques is replaced by a less comprehensive approach. Thus, the standard methodology is characterised by: first deciding on a technology approach, then focusing on writing a formal specification that completely captures all requirements of the desired system, and subsequently, employing formal techniques such as refinement to rigorously and verifiably develop the formal specification into executable code. A lighter version, instead focuses on a selection of the potentially very many characteristics of the whole systems, and judiciously applies formal techniques to generate higher confidence in these selected areas. This lighter touch nevertheless has, in common with the standard approach, the presumption that the decisions about how to proceed with formal methods is taken right at the outset, before detailed work begins.

However, the issues highlighted above indicate that the use of formal M&V in INSPEX did not follow even the lightweight pattern, let alone the standard one. One contrast with conventionally conceived lightweight formal methods comes in their discouragement of highly expressive formalisms, e.g. Z or B, to make the process indeed more lightweight; whereas we used such tools because they proved to be the most helpful, but in a lightweight manner, without pursuing the formality that they are capable of to the extreme.

Another contrast with formal methods of any style, arose from the context of the working methods of the hardware teams, which might be sloganised as: conceive it, design it, build it, test it, try to find out why it didn't work, and repeat. This is iterative in a sense, but with uniquely hardware characteristics. The formal M&V work had to adapt to that, so there was no predetermined methodological position that drove it at the outset, and the usual recommendations about bringing formal methods into a 'conventionally structured' project environment were rendered impotent. Thus, it was discussions with the other project partners as the project ran, that evolved the shape of the formal work that was done in an organic manner. The point that emerges from this, is that despite the very different ways of working of different parts of the project, and despite the lack of integration of these practices (which there was simply no time to explore), formal M&V could still make an undeniable contribution (as reported below).

It is a major aim of this paper to disseminate the idea that *ad hoc* and pragmatic working methods such as arose within INSPEX between hardware and formal M&V teams, could nevertheless produce tangible benefits for the project as a whole, when the two sides were willing to engage significantly with one another.

8 | MODELLING AND VERIFICATION (M&V) IN INSPEX

In the following sections we overview the parts of the INSPEX Project where formal methods were utilised, the outcomes obtained, and with a particular emphasis on how the issues discussed above interacted with the technical issues. It is clear that with the wide technical scope of INSPEX, the formal methods technical manpower (which consisted of the two Manchester based authors of the present paper) had to be judiciously deployed. In discussions with the wider consortium, it was decided to focus the formal methods effort in two areas. One was the data acquisition pathway, in which the complexity of the code merited the independent scrutiny that formal methods could bring to bear. The other was power management modelling, where the anticipated late finalisation of the sensors being developed within the project, implied that a technique based on rigorous modelling and prediction could compensate for the non-availability of the devices in the earlier phases of the project.

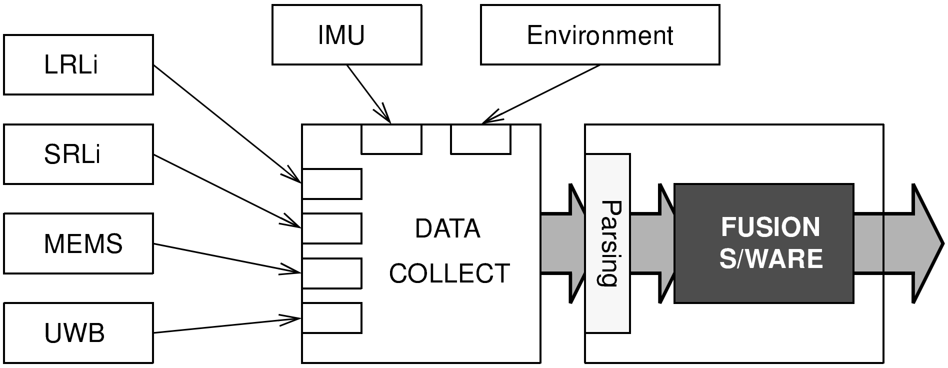


FIGURE 5 The INSPEX data management pathway.

9 | THE DATA PATHWAY

In Fig. 5, we illustrate the main data pathway for the INSPEX white cane addon device. The device has long range and short range LiDAR sensors (LRLi, SRLi), an ultrawideband radar (UWB) and a MEMS ultrasound sensor. These devices work in very different ways. The LiDARs produce large amounts of data at each sample, and can sample frequently because the essential time constant for their operation is determined by the speed of light. By comparison, ultrasound is extremely sluggish, because the speed of sound is orders of magnitude less than the speed of light. The ultrawideband radar is much closer to the LiDAR case. There is also an inertial measurement unit (IMU) containing accelerometers that keeps track of (the second derivative of) the relative displacement of the device in 3D space. There are further environmental sensors that monitor light, temperature, humidity, etc., for the benefit of performance optimisation algorithms.

Each of the four main sensors (LRLi, SRLi, UWB, MEMS) was a key development objective within the INSPEX Project, and was produced in a small standalone package so that it can ultimately be marketed separately, as well as being combined within the INSPEX VIB assistive device. The other sensors are off-the-shelf (OTS).

From each sensor package there is a data link to a one message buffer on the Generic Embedded Platform (GEP, labelled DATA COLLECT in Fig. 5), which is an OTS embedded processor. In outline, the contents of the buffers are collected and sent to a second, identical embedded processor for presentation to the fusion software which resides on it. This computes the occupation grid, which is transmitted to the smartphone to compute the 3D sound image to be presented to the user.

9.1 | M&V of the Data Pathway — General Observations

The activities on the data collection GEP constitute a highly concurrent set of data management processes, managed by the FreeRTOS[62] instance running on the GEP. The code is highly optimised, as is usually the case in embedded software, and after the preliminary work described in Section 6, it was agreed that formal M&V would be deployed here to increase assurance about its dependability.

At this point we ran into issues **[A] (Top-down)** and **[B] (Documentation)**. It became clear that the only documentation was the code itself, evolved from applications developed earlier, and written at a fairly low level of abstraction. This has pros and cons. The biggest con was the challenge of understanding the code from cold, a job that had to be tackled head on. A small pro, but hardly compensating for the magnitude of the con, was the demonstrable absence of any inconsistency between the code and any documentation about it.

Once the above was realised, issue **[E] (IP issues)** came to the fore, and there was some delay before M&V could gain access to the code. The prior work convinced us that Event-B and Rodin[33] would yield the greatest productivity, so M&V was based round its metaphors. One consequence of dealing only with code, is that the high level machine invariants that are the *sine qua non* of a top-down process like Event-B, had to be invented on the job, and doing so was a conscious obligation of the M&V work.

The highly concurrent code already referred to, consists of tasks that manage the individual sensor buffers, and aggregate the messages received into a large central buffer. The concurrency is managed by a mix of *bona fide* mutual exclusion facilities provided by FreeRTOS, and more *ad hoc* approaches based on timing heuristics. The latter are not logically watertight in the same way that genuine mutual exclusion primitives are intended to be, but while taking such an approach may seem horrifying to a verification specialist, we have to remember the inescapable fact that the input data is imperfect and noisy for a whole host of possible hardware reasons. Adding a little more noise due to the occasional mishandling of a message is tolerable provided it happens rarely enough that in the overall statistical context, it leads to a negligible degradation in overall performance, and adds value in terms of code efficiency and simplicity.

It will also not have escaped the reader that if the preceding is regarded as acceptable and normal, the writing of high level invariants will require a degree of inventiveness beyond the everyday. In fact the invariants one comes up with are closer to fairness statements. These can be expressed as invariants when the required properties are pegged to a reliable underlying progress property. Such a property is readily available, as all activities on the GEP march to the beat of the embedded hardware clock. Beyond that, creating sensible invariants of the kind just described requires the introduction of additional history variables, so these featured in the data pathway M&V activity, as required.

The messages coming in from the various sensors are not aggregated arbitrarily, but are segregated into batches by the receipt of IMU messages. The point is the following. An individual sensor reading simply yields one (or more in the case of the LiDARs) distance measurement. However the sensor itself has no idea in which direction it is pointing. Sensor distance readings thus have to be associated with a direction before they can contribute to the creation of an occupancy grid and the data from the IMU is used to compute the direction.[6] So the sensor messages arriving between successive IMU messages are batched for forwarding to the fusion software GEP, the IMU messages thus acting as brackets for such batches.[7]

Since two GEPs are used for processing the sensor data,[8] data communication hardware and software are present on both GEPs to handle the date transfer from the first to the second. Part of the M&V work was to check that this was well integrated with the rest of the data pathway code.

Once the batches of sensor data, along with their bracketing IMU messages, arrive on the fusion software GEP, they are presented to the parsing task. In this, the list of sensor readings in each batch is sorted by timestamp, which is a requirement for the input data for the fusion software itself. This uses interpolation of a distance reading timestamp between the timestamps of the IMU brackets, to work out how to weight a particular distance reading in the occupation grid calculation.

The fusion software is a key piece of IP belonging to project partner CEA. It computes the occupation grid using only integer arithmetic, a huge performance leap beyond the usual floating point approaches, and indispensable for minaturised portable applications like INSPEX. It is fiercely protected by patent and nondisclosure, so M&V of anything involving this software encounters issue **[E] (IP issues)** to an extreme degree, since the code is never released outside CEA itself, and only CEA employees can have access to it. Any M&V activity involving the fusion software thus sees only a black box. This specifically affects the handover of data from the parsing task, since it is the fusion software that releases the buffer space occupied by the parsing task data. This had to be modelled simply as a set of assumptions about the black box, none of which could be checked directly in the M&V.

In the work described, most of the issues discussed in Section 7 made themselves felt at one time or another. We have mentioned **[A] (Top-down)**, **[B] (Documentation)** and **[E] (IP issues)** already. The question of how deeply to model **[I] (Optimisation)**, was uppermost in mind at all times, and the pragmatism espoused in **[C] (Pragmatics)** and **[H] (Cooperation)** was key to making the right decisions. The high degree of visual accessibility of the Rodin tool and of its ProB companion **[L] (Demoability)**, was key in the evaluation of the M&V work.

In retrospect, the M&V of the data pathway came to be viewed as a very rigorous flavour of code inspection. In conventional code inspections, which originated with the work of Fagan[63,64], structured readthroughs and discussion are supposed to reveal flaws in the code. But the process can only be as good as the vigilance of the readers, and there is no independent measure of that in the conventional process. However, the effort to build a formal model of the code and to verify properties of that model, yields a very exacting scrutiny of at least the consistency of the understanding of the code (even if it admits a logically consistent mismatch between the code and the model).

9.2 | M&V of the Data Pathway — Overview of the Models

In this section, we make some of the preceding discussion more concrete by exploring some of the models involved. As already mentioned, the difficulty encountered in modelling the data pathway was the lack of a clear abstract specification. The formal modelling could not proceed in a strictly top-down manner, but had to work bidirectionally. The tension between the actual behaviour found in the code and various conjectured high-level descriptions was a fruitful one. It produced a suitable abstract description, but the details of this continued to be in flux until the end of the modelling process. A notable feature of this abstract system is that it places only very weak constraints on behaviour, while still conveying some useful information.

The main function of the data pathway is to take distance readings from various range sensors, arriving asynchronously, and position readings from the IMU sensor which detects the orientation of the system, and to associate to each range reading a position which is an interpolation of the positions recorded before and after it arrived. From a logical point of view, the details of the interpolation function do not matter (and indeed may be subject to change); what matters is that an acceptable *triple of arguments* (a range message and two position messages) is produced for interpolation.

---

[6] In fact, because two Euler integrations are needed to derive positional information from the raw accelerometer readings from the IMU, the positional information thus derived rapidly destabilises. Some fusion with readings from the UWB radar were needed in order to get reliable positional information.

[7] Each IMU message thus acts as a closing bracket for earlier sensor readings and as an opening bracket for later ones.

[8] A future iteration of INSPEX may well integrate the work of both onto a single processor.

Regarding the Event-B models themselves, the state variables of the abstract model are three sets: 'allReceived', 'unprocessed', 'allOutputs'. The first two are sets of messages, a notion defined in the Event-B CONTEXT seen by the Event-B MACHINEs containing the dynamics. Messages have a 'msg_type' which is either 'dist_msg' or 'pos_msg', and a 'msg_time' when they were received. The variable allReceived contains all messages ever received by the system, even if they have been deleted, but ignoring those which a sensor transmitted but were not recorded by the system, while unprocessed contains messages actually available for interpolation. The triples generated as arguments to the interpolation process constitute allOutputs.

Initialization is straightforward since all variables must be the empty set. There are events 'getRangeReading' and 'getPosReading' representing the receipt of messages. For example getPosReading is as follows.

```
Event getPosReading ⟨ordinary⟩ $\widehat{=}$
      any m
      where
            grd1: $\mathsf{m} \in \mathsf{messages}$
            grd2: $\forall \mathsf{n} \cdot \mathsf{n} \in \mathsf{allReadings} \Longrightarrow \mathsf{m} \neq \mathsf{n}$
            grd3: $\mathsf{msg\_type}(\mathsf{m}) = \mathsf{pos\_msg}$
      then
            act1: $\mathsf{allReceived} := \mathsf{allReceived} \cup \{\mathsf{m}\}$
            act2: $\mathsf{unprocessed} := \mathsf{unprocessed} \cup \{\mathsf{m}\}$
      end
```

Because of the high level of abstraction, the interpolation event is somewhat complicated to describe. We first give it with some guards omitted, then present the omitted guards. The effect of interpolation is to add some new interpolated events, modelled as a parameter 'imsgs' to the set allOutputs, and removes a set 'rmsgs' from unprocessed. This is possible when there are position messages 'm0' and 'm1' with range messages temporally between them.

```
Event interpolate ⟨ordinary⟩ $\widehat{=}$
      any m0 m1 imsgs rmsgs
      where
            grd1: $\mathsf{m0} \in \mathsf{unprocessed}$
            grd2: $\mathsf{m1} \in \mathsf{unprocessed}$
            grd3: $\mathsf{msg\_type}(\mathsf{m0}) = \mathsf{pos\_msg}$
            grd4: $\mathsf{msg\_type}(\mathsf{m1}) = \mathsf{pos\_msg}$
            grd5: $\mathsf{msg\_time}(\mathsf{m0}) < \mathsf{msg\_time}(\mathsf{m1})$
            grd6: $\exists \mathsf{m} \cdot \mathsf{m} \in \mathsf{unprocessed}$
                  $\wedge\ \mathsf{msg\_time}(\mathsf{m0}) \leq \mathsf{msg\_time}(\mathsf{m}) \wedge \mathsf{msg\_time}(\mathsf{m}) < \mathsf{msg\_time}(\mathsf{m1}) \wedge \mathsf{msg\_type}(\mathsf{m}) = \mathsf{range\_msg}$
      then
            act1: $\mathsf{unprocessed} := \mathsf{unprocessed} \setminus \mathsf{rmsgs}$
            act2: $\mathsf{allOutputs} := \mathsf{allOutputs} \cup \mathsf{imsgs}$
      end
```

It remains to specify the contents of imsgs and rmsgs. Any position message which arrived between m0 and m1 can be interpolated, by associating those position messages to it. We include in this those messages which arrived at the same time as m0, but not those which arrived at the same time as m1 which will have to be dealt with once a later position message arrives. Finally, all messages which arrived strictly before m1 must be deleted, since they have either been used for interpolation, or else are considered too old.

```
            grd7: $\mathsf{imsgs} \subseteq \mathsf{interpolatedMessages}$
            grd8: $\mathsf{imsgs} \neq \emptyset$
            grd9: $\forall \mathsf{i} \cdot (\mathsf{i} \in \mathsf{imsgs}) \Longrightarrow (\exists \mathsf{m} \cdot \mathsf{m} \in \mathsf{unprocessed} \wedge \mathsf{msg\_type}(\mathsf{m}) = \mathsf{dist\_msg}$
                        $\wedge\ \mathsf{msg\_time}(\mathsf{m0}) \leq \mathsf{msg\_time}(\mathsf{m}) \wedge \mathsf{msg\_time}(\mathsf{m}) < \mathsf{msg\_time}(\mathsf{m1})$
                        $\wedge\ \mathsf{imsg\_time}(\mathsf{i}) = \mathsf{msg\_time}(\mathsf{m}) \wedge \mathsf{imsg\_range}(\mathsf{i}) = \mathsf{m}$
                        $\wedge\ \mathsf{imsg\_pre}(\mathsf{i}) = \mathsf{m0} \wedge \mathsf{imsg\_post}(\mathsf{i}) = \mathsf{m1})$
            grd10: $\forall \mathsf{i}, \mathsf{j} \cdot \mathsf{i} \in \mathsf{imsgs} \wedge \mathsf{j} \in \mathsf{imsgs} \wedge \mathsf{imsg\_range\_id}(\mathsf{i}) = \mathsf{imsg\_range\_id}(\mathsf{j}) \Longrightarrow \mathsf{i} = \mathsf{j}$
            grd11: $\mathsf{rmsgs} \subseteq \{\mathsf{m} \mid \mathsf{m} \in \mathsf{unprocessed} \wedge \mathsf{msg\_time}(\mathsf{m}) < \mathsf{msg\_time}(\mathsf{m1})\}$
```

Note that one might have wished for various optimality properties, for example that a range message is associated with the position messages which are *closest* to it in time, or that every range message which occurs after a position message is eventually given an interpolated position once a subsequent position message arrives. However, while the concrete implementation strives to achieve these properties, it does not *guarantee* to do so. This is ultimately because of the memory constraints it operates under. To state the sense in which these desirable properties are achieved, it is necessary to describe some details of the structure of the internal buffers. Thus we have a natural example of a functional correctness property which must be added at a more concrete stage of refinement.

However, *some* account must be made of the fact that messages may be lost due to memory constraints. This is done by adding an event which removes messages in a highly nondeterministic way. The only constraint is that this is only happens because there are too many messages, that is, when the cardinality of unprocessed exceeds a constant 'max_buffer_size'.

```
Event  removeMessages ⟨ordinary⟩ ≙
       any  msgs
       where
              grd1: msgs ⊆ messages
              grd2: card(unprocessed) ≥ max_buffer_size
              grd3: card(msgs) ≤ card(unprocessed) − max_buffer_size
       then
              act1: unprocessed := unprocessed ∖ msgs
       end
```

The meaning of max_buffer_size is somewhat flexible. It should not be thought of as being set *a priori* in the abstract model. Instead its presence in the abstract model represents the constraint that there *must exist* such a bound, but its value can be set once the concrete details are known. In this way, it expresses a property of the concrete system (the existence of a bound) as well as documenting the degree to which the system approaches the ideal (the value of the bound). One would like to do this for other aspects of the system where low-level challenges make it unreasonable to demand complete adherence to ideal abstract behaviour. However, this was not possible for the other cases mentioned above (like temporal proximity of data used for interpolation), for which it seems there is no such bound in general.

An intermediate-level model of the system, which elaborated the system description given in the abstract model, was produced. This will now be outlined. The structure of the model is that various sensors of different kinds receive readings and post them to a 'message pool' shared data structure, modelled by a queue datatype in the Event-B context. These readings are then copied to an ordered 'list' data structure which at this level of abstraction is modelled simply as a set. From here, they can be used for interpolation, since events which process messages in the 'list' may safely refer to the temporal order of messages.

Sensors have two possible patterns of interaction with the message pool, which we call 'direct' and 'buffered'. The buffered sensors first reserve a position in the message pool, then wait for a reading to fill it with. The direct sensors wait until they have a reading, then allocate a position in the message pool and immediately post their data to the pool. Which pattern of interaction is appropriate is a reflection of low-level details of the sensor. The events of interest are those which deal with the message pool.

For buffered sensors, there are two events, the allocation of a message pool location, and the posting of a filled message to the pool.

```
Event  bufferedSensorAllocateMsg ⟨ordinary⟩ ≙
       any  s
       where
              grd1: s ∈ bufferedSensors
              grd2: sensorTaskState(s) = wait_msg_pool
              grd3: card(filledMsgQueue) + card(sensorsHoldingMsgs) < msg_pool_max
       then
              act1: sensorsHoldingMsgs := sensorsHoldingMsgs ∪ {s}
              act2: sensorTaskState(s) := wait_reading
       end
```

```
Event  bufferedSensorGetReading ⟨ordinary⟩ ≙
       any  s
       where
              grd1: s ∈ bufferedSensors
              grd2: sensorTaskState(s) = wait_reading
              grd3: sensorBufferFull(s) = TRUE
              grd4: s ∈ sensorsHoldingMsgs
       then
              act1: sensorsHoldingMsgs := sensorsHoldingMsgs ∖ {s}
              act2: sensorBufferFull(s) := FALSE
              act3: sensorTaskState(s) := wait_msg_pool
              act4: filledMsgQueue := pushMsg(filledMsgQueue ↦ sensorBuffer(s))
       end
```

For direct sensors there is a single event as follows.

```
Event  directDistSensorAllocateMsg ⟨ordinary⟩ ≙
       any  s
       where
              grd1: s ∈ directSensors
              grd2: s ≠ Imu
              grd3: sensorTaskState(s) = wait_msg_pool
              grd4: card(filledMsgQueue) + card(sensorsHoldingMsgs) < msg_pool_max
       then
              act1: filledMsgQueue := pushMsg(filledMsgQueue ↦ directReading(s))
              act2: sensorTaskState(s) := wait_reading
       end
```

Next, messages are copied to the list. In the real system, this involves being sent from one physical board to another, as well as a sorting procedure. In the present model, it is represented simply as copy operation, albeit one which can cause data to be overwritten.

```
Event msgGoesToList ⟨ordinary⟩ $\widehat{=}$
    when
        grd1: $\mathrm{card}(\mathrm{filledMsgQueue}) \geq 1$
        grd2: $\mathrm{card}(\mathrm{list}) + 1 \leq \mathrm{list\_max}$
    then
        act1: $\mathrm{filledMsgQueue} := \mathrm{popMsg}(\mathrm{filledMsgQueue})$
        act2: $\mathrm{list} := \mathrm{list} \cup \{\mathrm{headMsg}(\mathrm{filledMsgQueue})\}$
    end
```

```
Event msgOverwritesOldListItem ⟨ordinary⟩ $\widehat{=}$
    any m
    where
        grd1: $\mathrm{card}(\mathrm{filledMsgQueue}) \geq 1$
        grd2: $\mathrm{card}(\mathrm{list}) = \mathrm{list\_max}$
        grd3: $\mathrm{m} \in \mathrm{list}$
        grd4: $\forall \mathrm{n} \cdot \mathrm{n} \in \mathrm{list} \implies \mathrm{msg\_time}(\mathrm{n}) \geq \mathrm{msg\_time}(\mathrm{m})$
    then
        act1: $\mathrm{filledMsgQueue} := \mathrm{popMsg}(\mathrm{filledMsgQueue})$
        act2: $\mathrm{list} := (\mathrm{list} \setminus \{\mathrm{m}\}) \cup \{\mathrm{headMsg}(\mathrm{filledMsgQueue})\}$
    end
```

Now the interpolation event is similar to, but more explicit than the one for the abstract system. Omitting the guards, it is as follows.

```
Event interpolate ⟨ordinary⟩ $\widehat{=}$
    any m0 m1 imsgs
    then
        act1: $\mathrm{list} := \mathrm{list} \setminus \{\mathrm{m} \mid \mathrm{m} \in \mathrm{list} \wedge \mathrm{msg\_time}(\mathrm{m}) < \mathrm{msg\_time}(\mathrm{m1})\}$
        act2: $\mathrm{allOutputs} := \mathrm{allOutputs} \cup \mathrm{imsgs}$
    end
```

It is at this level of abstraction that one can specify that the position messages actually used for interpolation have no position messages between them in the list, and that all suitable messages in the list are used, which is the stronger constraint alluded to above which the system satisfies. It cannot be described without modelling the two different internal buffers, since there could be relevant messages in the message pool.

Although a formal refinement was not carried out, the intermediate model presented can be seen to be a refinement of the abstract one. The crucial point is that unprocessed is represented by the union of list and filledMsgQueue, and hence max_buffer_size must be set equal to list_max. One modification to the code suggested by our analysis would be that direct sensors which are waiting to receive a position in the message pool should continually overwrite the reading they are storing, so that it is the newest messages, rather than older ones, which are passed to the system. In this case, we may wish to consider the range of directReading to be part of the representation of unprocessed too. In this case, the abstract event removeMessages would be refined by two concrete events, representing data loss by the list and by the direct sensors. The event modelling data loss by the sensors would have to have an assumption that it can not occur when there is a sufficient amount of space in the queue. This assumption would then amount to a liveness condition, implying that one sensor's request for a queue position can not be deferred for an arbitrarily long time in favour of the other sensors. This flexibility of refinement is an advantage in modelling exercises where information needs to percolate upwards from lower lever to higher level models.

9.3 | Outcomes of the Data Pathway M&V

Formal modelling of the sensor preprocessing was done based on analysis of the code and discussion of the algorithm with engineers. This resulted in four Event-B models, the top level model, the intermediate level model, and two low-level models. The second refines the first, and the final two can be seen as refinements of it, albeit only refining some of its events.

The abstract and intermediate level models are described in the previous section in some detail. The abstract model is small with few proof obligations and was trivially proved correct by automatic provers, but the model at the intermediate level posed more of a challenge. Ignoring the context, this model has 134 proof obligations, of which 104 were proved automatically, and 30 were proved interactively. The latter involved adding some helper 'theorems' to the context, which themselves could be proved automatically. There are 20 invariants in total. These are mostly typing invariants, but there are interesting invariants about the cardinality of the queue and of the message pool. The automated prover generally had a hard time with the cardinality constraints, though for the most part these were quickly dealt with by SMT solvers after some work in the interactive prover to simplify the problem into a suitable form. One of the typing constraints proved to be problematic for the automated prover, namely that the message pool only contains messages which have been received by the system. The message pool is a queue structure which was modelled as is standard by using partial functions whose domain is a finite prefix of the natural numbers. The problem is essentially to prove that

$\mathsf{ran}(f) \subseteq \mathsf{allRecieved}$ given the assumptions that for all $1 \leq x \leq n$ we have $f(x) \in \mathsf{allRecieved}$, $f(n+1) \in \mathsf{allReceived}$, and $\mathsf{dom}(f) = \{1..n+1\}$. The 'creative' step of rewriting $\{1..n+1\} = \{1..n\} \cup \{n+1\}$ was too much for the automated prover in this context, even when provided with this fact as a theorem. In addition, much trouble seems to be caused by the low-level approach taken to define basic operations on partial functions. For instance, when rewriting $x \in \mathsf{ran}(f)$ for a partial function f, the result is $\exists y \cdot (y \mapsto x) \in f$ rather than the more informative $\exists y \in \mathsf{dom}(f) \cdot f(y) = x$. To unpack this information, one must rewrite the definition of $\mathsf{dom}(f)$ at some point, generally after manually selecting the relevant hypothesis which defines $\mathsf{dom}(f)$. But seeing how to do this, or that the sequence of rules involved is likely to help is not trivial, and sometimes beyond the abilities of the auto-provers. Finally, a certain amount of nuisance is caused to users of the interactive prover by the hypothesis hiding features. In this particular instance, it would have been an improvement if hypotheses which have been explicitly selected by the user are not automatically removed.

As mentioned above, two much lower level models were produced. This division into two models reflects the separation of the real system into two hardware boards, one 'acquisition subsystem' which collects the asynchronous messages from sensors, and one 'fusion subsystem' which performs interpolation and passes the result to the statistical algorithms. These models take into account fine details of the data structures. For instance, in the acquisition system, the message pool is not a single queue but has other supporting structures which determine which free position on the queue is allocated to a sensor which has requested one. In the fusion subsystem, although we do not model the process of data transmission between the systems, we model the implications that the format of the data received has on memory management, namely that memory is allocated and deallocated in blocks. The model captures the possibility of memory errors, for instance, blocks being deallocated while partially in use. In addition, the precise details of the ordered data structure are modelled, including the sorting procedure.

While the memory of the C program is not modelled explicitly, the pointer structure of the list is faithfully captured by sets of abstract 'memory locations' with functions between them. Important sections of the code were modelled with a single event per straight-line section of code, so that the control flow of the those parts of program was completely captured. The model of the acquisition system has 127 proof obligations, of which 68 were proved automatically. Of the remaining 59, 4 were proved by hand. The model of the interpolation system has 630 proof obligations, of which 247 were proved automatically, and 10 were proved by hand. As can be seen, the complexity involved in these models was too much for the automated prover, and too great for one person to fully verify in the time available. However, the exercise of trying to prove selected invariants interactively was valuable. Use of the interactive prover implied that one could try to restate the proof problem in a more conceptual form, even if it could not be proved in this form. Then, reflection might reveal that the system possesses the property in question. More often, it revealed a flaw in the model. In cases where this was a missing invariant, the experience was valuable, as the correctness of the system was shown to depend on subtle properties which were possessed by the code, but not known explicitly to the engineers. Documenting these properties was important for a system subject to change.

In other cases an error in the code was found; the form of the proof problem could then be used to suggest a simple change to the code. In general, the activity of attempting formal proofs of selected properties aided model development, which in turn suggested questions about specific parts of the code. The result was a kind of model-directed code review, in which a few thousand lines of code could be productively analysed by a single person, finding problems with small details missed by the existing code review processes of the engineers. The lower-level models fed back into higher level models, and the properties these suggested, or the attempts to prove these properties, guided the addition of details to the low level models. These details necessitated the detailed study of the code, but crucially only those parts which impacted difficult-to-prove properties. It was observed that these parts might not be in the obvious places. Instead the model might show that a property of one part of the system might depend on the precise behaviour of another part. This demonstrated the advantages of using an expressive formalism, but of taking a 'light weight' approach to the verification task — in contrast to methodologies which focus on weak formalisms and plug-and-play code analysis when formal modelling resources are in short supply.

Overall, the problems revealed by this analysis were of three kinds:

- A single issue of functional correctness was identified, in which a function did not satisfy its specification.
- Several issues were discovered for which correctness could not clearly be established, indicating potential problems with the code, such as a reliance on certain events happening sufficiently quickly.
- In some cases, while the code was correct, the formal analysis suggested issues of robustness, for example, invariants which may be violated if the system were subject to greater parallelism.

In all of these cases, the analysis did not just detect the problem, but instead it was possible to verify the correctness of alternatives at the high level, which led to concrete suggestions about the code.

The formal models also provided a concrete representation which helped to document the code at a high level. This included making clear what would stay the same if the hardware architecture were changed, for example if a single processor were used rather than having the processing split over two devices. Formal analysis also revealed implicit invariants which explain why assumptions which seem to be made by the code are

TABLE 3 Issues Identified During Data Pathway Modelling and Verification

Description	Model involved	Tool used	Modeling phase	Impact	Modelling effort
Fresh data deleted, keeping older data	Intermediate abstraction Event-B model of sensor preprocessing	Rodin	Model development	Suggestions to improve code provided	Moderate: found while translating code to intermediate-level models
Possible assertion failure in sensor task	Low level Event-B model of acquisition platform	Rodin	Automated theorem proving	Suggestions to improve code provided	Moderate: some examination of automated proof output necessary
Undocumented assumptions in sensor interpolation code	Low level Event-B model of interpolation system	Rodin	Examining automated prover output	Suggestions to improve code provided	High: detailed study of prover output
Incorrect behaviour due to data structure invariant violation	Most detailed Event-B model of interpolation system	Rodin	Examining automated prover output	Instructions to correct code provided	High: detailed study of prover output

justified. This type of analysis can aid not only code clarity and maintainability, but in a project such as INSPEX, can help to satisfy legal and ethical requirements, by documenting design decisions. While the need to justify design decisions in healthcare related domains can be a barrier to the reuse of existing code developed for other application areas, the formal modelling provides a precise justification for its inclusion.

The practical impacts of the M&V effort on the data pathway of the INSPEX Project are summarised in Table 3 below. The table includes only concrete impacts, leaving out, for example, advantages related to documentation, communication, and justifiable design.

The scrutiny that the M&V work brought to bear on the data pathway code brought to light a number of issues, discussed above, that were subsequently communicated to the code's authors for future resolution. It was found that engineers valued the questions and feedback generated by these M&V activities. This eloquently vindicated the decision to deploy formal techniques within the INSPEX Project.

9.4 | Abstraction Level Impedance Mismatches — Other Verification Tasks

During the work described above, a number of other approaches were also tried and were found, within the context we had, to be insufficiently effective to pursue in depth. We mention these briefly now.

In[65] there is an account of a formal model in Z of the FreeRTOS system. The discovery of this work was welcomed with joy, as it held out the promise of being able to integrate our application modelling with an equally rigorous model of the underlying system that the code relied on. In the event, this turned out to be overoptimistic. Essentially, there was an abstraction level impedance mismatch that prevented efficient interworking. We would have had to model much more of the basic memory system than we could afford, in order to bring our models to the level of abstraction of[65].

In point **[J] (Optimisation)** we mentioned the enthusiasm for automatic code generation. However, we could see at a distance the mismatch between the kind of code that we could expect to generate automatically, and the anticipation of highly optimised code that was the day to day reality in the embedded world, so we did not pursue this angle seriously; all this notwithstanding the impact on working practices that would be needed, and the nontrivial cost of production quality tools in this area.

As an approach to greater scrutiny of the code at the level of pointer manipulation etc., and given that we had no higher level documentation to work with, we decided after some time, to consider source code verification, and the BLAST tool in particular (see point **[K] (Abstraction)**). BLAST has the virtue of being very well established, and of having a relatively comprehensive suite of documentation online that permitted a fairly short learning curve into 'production use'. With relatively little adjustment, relevant parts of the mutithreaded code we were working with could be adapted to BLAST's essentially single threaded world view. However, we hit another abstraction level impedance mismatch. The properties we

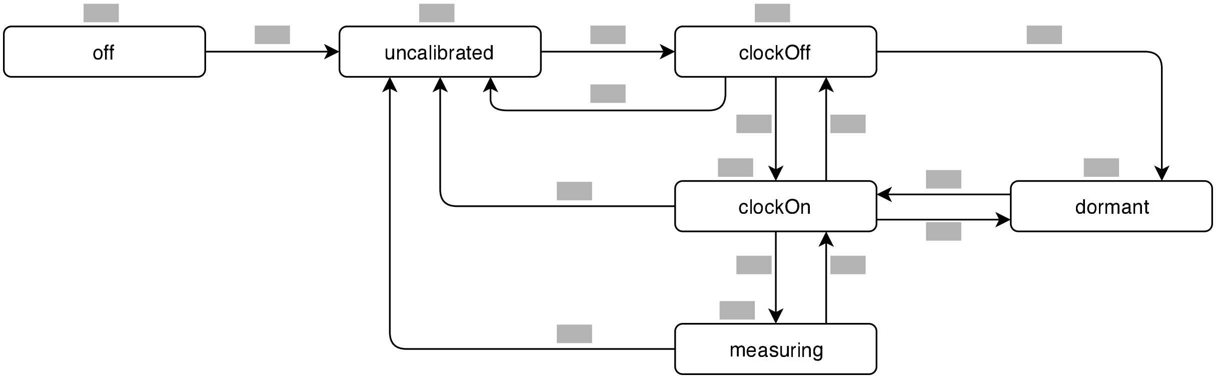


FIGURE 6 A power management state transition diagram for the Long Range LiDAR. Shaded squares conceal numerical power dissipation data, including transient characteristics for the transitions.

were seeking to prove were at too high a level of abstraction for BLAST's reasoner to be able to bridge the gap. Still, if we were attempting source code verification approaches in a similar context in future, we would certainly try other tools, such as ESBMC[66].

10 | POWER MANAGEMENT

As for any portable electronic device, parsimony in the use of power is a prime consideration for INSPEX credibility, and would certainly be so for a fully commercialised version of the device. Thus, the design of a power management strategy was an important objective of the project proposal. The hardware teams were fully occupied in getting their individual sensors working, so the higher level aspects of power management fell to the M&V team, with the view that the M&V tools could yield some useful high level predictions.

10.1 | Modelling and Analysis of the Power Management System

In constructing formal models of the power management system, issue **[D] (Lateness)** hit hard — the power management strategy needed to be designed to a reasonable degree of completeness before the devices were finalised. To address this, two insights were brought to bear. The first was that a certain amount of modelling could be done using uninstantiated constants, whose numerical value could be supplied later. The second was to start the more detailed work with the OTS components, for which data was available.

Thus, the available documentation on the OTS components was scoured for the definitions of each one's states and their power consumption characteristics, as well as any other information in the documentation that was relevant to power consumption. The information gleaned was assembled into a state transition diagram decorated with numerical data about the power characteristics of the device, giving a self contained model for that component.

For the devices being designed within the project itself, information was slow to emerge, as it was simply unknown in the early stages of the project what the laws of physics would permit the required characteristics to be. In this regard, issues **[F] (Language)** and **[G] (Clues)** were very germane. Listening to others' discussions about problems connected with the emerging designs of their devices could yield useful clues in the preliminary model building of the requisite state transition diagrams.

Fig. 6 gives an illustration of a state transition diagram for one of the developed sensors. It is the STD for the Long Range LiDAR device. The small grey rectangles decorating states and transitions are numbers describing power characteristics. The actual numerical values remain commercially confidential at this juncture. A power consumption numeric for a state is a natural concept: it is just the rate that power is consumed when the device is in that state. The reader may wonder though at the veiled numbers decorating the transitions, since in a normal STD, transitions are understood to occur instantaneously and a *rate* of doing anything is meaningless for them. In reality of course, the transition is not *physically* instantaneous, and an STD like Fig. 6 is an abstraction of a more complex structure, modelling a process in which a transition has a nonzero, if short (and generally fixed) duration, which may have nontrivial power repercussions. This can be captured in a refinement of the Fig. 6 STD in which such a transition is refined using a triple: *TransStart*, *TransOngoing*, *TransEnd*, effectively introducing an additional *TransOngoing* state half way along the Fig. 6 arrow, to which power consumption characteristics can be ascribed.

In the low power embedded systems world, there is a natural unit for power, the milliwatt, mW. While most power consumption characteristics were indeed expressed in mW, some were not, giving figures in e.g. milliamps (predicated on a tacit understanding about the relevant voltage

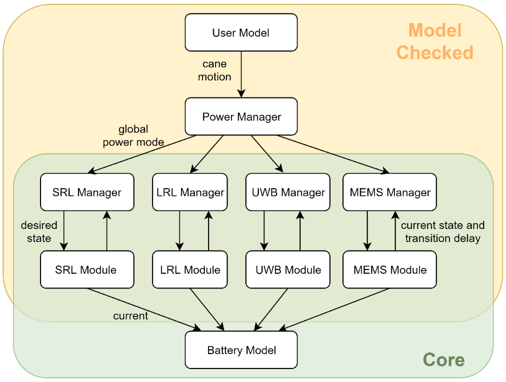


FIGURE 7 Structure of the PRISM power management modules.

levels). This raised further concerns connected with issues **[F] (Language)** and **[G] (Clues)** on commonality of language and familiarity with the other disciplines. For domain experts this kind of discrepancy regarding documentation of power consumption is a triviality, but it is not everyday *lingua franca* in the verification world.

The approach outlined thus far yields the STDs for individual components. These models can be used to calculate the amount of energy consumed for a given parameter setting, or for the amount of time in which a sensor spends in a non-measuring state. PRISM can then be used to compute expected values of these costs. More ambitiously, we can calculate the rate at which sensor readings are received, or the number of dropped messages, though this is significantly affected by the fine details of memory management and internal communication inside the device itself.

One significant source of complexity is that the OTS devices attempt to minimise their power consumption autonomously, modulated by external control. If the mechanisms for this are not visible to the user in the documentation, the precision of an overall power management strategy is compromised.

The individual models are combined into an overall model of power consumption. Power management policies can then be formulated. A simple example strategy is to have more readings when the cane is moving faster. More sophisticated strategies which have evolving goals can be envisaged. For example, if the battery level is low, the priority might switch from data quality to saving power. Environmental measurements such as humidity and light level can also play a role. A power management strategy has to function under uncertainty about environmental conditions, user behaviour, and the relative autonomy of its components. So it must co-ordinate the individual power management behaviours of the subsystems so that their aggregate effect is to use power judiciously.

Fig. 7 shows the overall model structure. Technically, the system is modelled in PRISM using PRISM's Markov Decision Process formalism. The physical submodules of the INSPEX devices and their power supplies are modelled by PRISM 'modules' (processes composed in parallel to form the whole model: 'SRL Module', 'LRL Module', etc.). These each have a variable indicating the current rate of power consumption, which is set by their state transitions. The power management policy for each INSPEX device is represented in another module ('SRL Manager', 'LRL Manager', etc.) whose transitions are synchronised with the transitions of the device submodules, controlling when those transitions can take place.

A further module, 'Power Manager', manages time and the consumption of power, and translates high level power management goals into directives for individual INSPEX device submodule managers. To calculate the energy consumed, it reads the submodule models' power variables, multiplying by the time elapsed. We chose to model time manually, using 'tick' events of various durations, which are associated with 'rewards' (PRISM's terminology for a quantity of interest) such as total time elapsed or total energy consumed. One reason for this choice was to keep the PRISM models as consistent as possible with the existing Rodin models of the submodules and other components.

The goal of analysing various power management strategies using the PRISM tool is to take advantage of the fact that it permits their properties to be studied without costly implementation effort, especially for more complicated solutions. This provides evidence that future commercial development beyond TRL4 could meet stricter power and weight requirements by deploying more sophisticated strategies, and could allow concrete guidance to be given as to which ideas are likely to have meaningful effects.

The modelling of power dissipation can be made more realistic by incorporating a model of cane motion, based on the work done on mechanical properties of the cane by project partner CIT in Cork. This studied how users typically move their cane. At first this was a relatively simple model, but it was further supplemented by modelling more specific aspects of user behaviour, for example, the possibility that the user might simply stop to talk to someone. Such behaviours change the cane dynamics in a readily discernible way. All of these elements were captured in the 'User Model' of Fig. 7. Finally, a model of the battery itself (‘Battery Model’) was incorporated in the overall PRISM system model. This is discussed in greater detail in Section 10.2.

The architecture of a PRISM model, based on parallel composition of processes, facilitates the kind of modelling needed to capture all of the above, in which a high-level transition system can enable or disable lower-level models which capture stylised versions of simpler behaviours. Of course, very simple models of the user cannot be expected to be ‘formally provable’; they have to simply be assumed. However, by analysing a model of the user moving the cane in a ‘standard’ manner, information can be supplied to prospective VIB users of a real cane, which they may find useful.

As was stated at the beginning of this section, the details of the power management model were accumulated gradually. The basic structure of the various STDs came first. These were enhanced with numerical data that became firmer over a period of time. As the complete model grew in size and scope, there emerged a perceptible tension between what could be accomplished via model checking, and what could be accomplished via simulation approaches. For a large model, exploration via simulation is obviously the easiest thing to do. However, the lack of completeness of simulation leaves open the possibility of undetected deadlocks remaining in the model. Confirming the absence of these required the creation and model checking of abstracted models, and monitoring the correspondence between these and the full model became a significant element of the work.

10.2 | Estimating Battery Lifetime

The modelling strategy described above permits the calculation of energy consumption at a given moment in time. The main purpose of this is to allow the battery lifetime to be estimated. This is a highly non-trivial task, because it requires a model of battery behaviour. Since battery behaviour is complex, models of battery performance are necessarily simplifications. However, for lifetime estimation to be meaningful, nonlinear effects, such as the tendency of the battery to appear to recover charge when idle, must be taken into account[67]. Under the guidance of CIT, the Rakhmatov model of battery behaviour[68] was selected. At present it is rather unusual to find discussion of such a realistic battery model in the formal methods literature; but see[69] for an interesting counter-example. This section describes the Rakhmatov model, and goes on to discuss the challenges inherent in representing it in the PRISM formalism. Note that the Rakhmatov model takes as input the current at each moment in time. Since the voltage supplied by the battery is known in the application, the calculation of energy consumption above is sufficient information.

The model uses two constants, α and β to describe the battery. The parameter α is related to the battery capacity, while β is related to its non-linear discharge behaviour. Given a specific battery, the two parameters are determined by standardised experimental measurements performed on it in the laboratory, so for modelling purposes, we regard α and β as known.

The current load on the battery is represented as a series of step functions. For simplicity of presentation we assume that there are n steps, each of duration $1/f$ units of time, and that the battery runs for L time units, which is equal by assumption to n/f. We denote the current at time step k by I_k. Rakhmatov and Vrudhula show that α, β and L are related (to a good approximation) by the following formula.

$$\alpha = \sum_{k=1}^{n} 2I_k \left(X\left(L - \tfrac{k-1}{f}\right) - X\left(L - \tfrac{k}{f}\right)\right)$$

where $X(d)$ is the function

$$X(d) = \sqrt{d}\left(1 + 2\sum_{m=1}^{10} e^{-\frac{\beta^2 m^2}{d}}\left(1 - \frac{\pi}{\pi - 1 + \sqrt{1 + \frac{\pi d}{\beta^2 m^2}}}\right)\right)$$

This can be thought of as a time-dependent nonlinear correction to the contribution to α caused by a change in current at time $L - d$. Note that $X(d)$ is monotonically increasing in d, and $X(0) = 0$.

One usually wants to compute the battery lifetime L given the sequence of loads I_k. The equation above is hard to solve for L; Rakhmatov and Vrudhula suggest that L should be approximated by calculating, for a series of conjectures l about the value of L, the following quantity.

$$\bar{\alpha}(l, n) = \sum_{k=1}^{n} 2I_k \left(X\left(l - \tfrac{k-1}{f}\right) - X\left(l - \tfrac{k}{f}\right)\right)$$

Then, the best approximation to L is the smallest l such that $\bar{\alpha}(l, n) \geq \alpha$.

To implement the above as a PRISM model requires the use of a state variable for each conjectured value of l. At each step of time, represented by a ‘tick’ event synchronized with the rest of the model, the value of $\bar{\alpha}(l, n)$ is incremented for each of the values of l in the model, according to the formula above. The battery is considered to be exhausted if at any step of time, the value $\bar{\alpha}(l, n)$ turns out to be greater than the model parameter α, for the smallest value of l greater than or equal to the elapsed time. Evidently, when the elapsed time is greater than l, it is meaningless to continue incrementing $\bar{\alpha}(l, n)$. Concretely, this means that (within the limits permitted by the approximation) the expected lifetime can be computed by running the PRISM query:

```
Rmin =? [ F (battery_fail | ti = 0) ]
```

which returns the expected number of time steps before the battery fails or the simulation time ‘ti’ runs out. The formula ‘battery_fail’ is a disjunction of one failure condition for each conjectured lifetime, which is true if the calculation for the conjectured lifetime indicates battery failure while that conjectured lifetime is the best approximation to the true value. For instance, the failure condition for the third conjectured lifetime is:

```
formula l3_batt_fail = (l2_a > ti) & (l4_a < ti) & (l3_a >= scale*alpha)
```

where the state variable ln_a is $\bar{\alpha}(\mathsf{l}, n)$.

This way of modelling the system requires a very large state space, and thus precludes model checking. Therefore, values of interest must be computed by simulation. This means that, ideally, path lengths of computations by the model should be kept to a minimum. Unfortunately, the INSPEX system deals with phenomena at a wide range of timescales, from sensors’ power mode transitions at the shotest scale, through the oscillating cane motions made by the user at an intermediate scale, to the frequency with which the user intersperses intervals of activity and rest, which we can imagine to happen at the longest timescale. The power manager interacts with all these phenomena, and so its interesting properties are disclosed only by long computation paths.

A partial solution to this problem is that many power management strategies keep all components in a low power state while their use is inactive. This permits the use of variations on the ‘tick’ event representing long periods in which the system’s state does not change. This is not a complete solution, however, because it does not cope well when the user is active. Even within a period of activity, there can be large differences between the submodule power mode transition times and the frequency of cane motion. In addition, the use of multiple ‘tick’ events of different lengths complicates the model, adding details which do not correspond to anything in the domain. For example, it is critical that only one such event is possible in any given state, necessitating some synchronization machinery to be added to every module which can often be more complex than the logic of the component modelled.

Some experiments were made in using PRISM ‘rewards’ to model the value of $\bar{\alpha}(\mathsf{l}, \mathsf{n})$ since this is a sum over states in a computation path. However, these quantities in themselves do not interact well with taking expected values, in that the expected value can include contributions from computation paths on which they are physically meaningless, such as when $\bar{\alpha}(\mathsf{l}, \mathsf{n})$ first exceeds and then later falls below α.

An example of the kind of analysis permitted by the model is shown in Figure 8. It shows the behaviour of a simple power management strategy which incorporates two features: sensors are put into a non-measuring ‘ready’ state at the two ends of the cane swing when the cane is moving slowly, and sensors are put into a low power mode if the user is taking a long rest. The system detects a long rest by transitioning to this low power state with some small probability at each step of time when no motion is detected. This probability is a parameter of the strategy which we refer to as ‘eagerness’ in the figure. The user is modelled by a simple Markov model, which exhibits variations in cane sweeping frequency and rest length, combining frequent short pauses with less frequent long breaks. The parameters of the user model are held fixed in the analysis shown. Two data series are depicted in the graph, both calculated using PRISM rewards. The top curve represents the expected total lifetime of the system, while the lower curve represents the expected time spent productively, in the sense that the system is taking measurements, or the user is in a state where measurements are not required. These times have been normalized by dividing by the lifetime of the system without power management. Analysing the data shows that for the user parameters given, it is reasonable to expect a 40% improvement in productive system lifetime, though this comes at a cost of increased unproductive time. In this way the formal models can be used to study the engineering trade-offs involved in a proposed power management strategy without having to perform implementation and testing.

The PRISM models were organized in a modular way, with the model of the power management strategy interacting indirectly with the sensor models, via intermediate ‘submodule manager’ modules which represent low-level drivers. In this way, the high-level power manager can set a goal state, and the submodule managers deal with the interpretation of that global power state in terms of the specific sensor, and crucially the sequence of steps which the sensor must go through to safely transition from its current state to the goal. The high level manager can change its mind while these transition processes are part-way through, and the intermediate level systems ensure that the sensor’s state responds intelligently. For example, the submodule manager for the Long Range Lidar deals with the fact that the power manager may request a global ‘measuring’ state while the sensor is in the ‘dormant’ mode. In this case the sensor cannot transition to ‘measuring’ directly. Hence part of the PRISM module for the Long Range Lidar manager is as follows.

```
module  pm_lr
  r_lr  :  [1..lr_states]  init lr_off ;
  [tick]  (lr_d > 0)  ->  true ;
  //  some code omitted
  [tick]  (pm_mode = pm_meas)  &  (lr_s = lr_off)      &  (lr_d = 0)  ->  (r_lr′ = lr_uncal) ;
  [tick]  (pm_mode = pm_meas)  &  (lr_s = lr_uncal)    &  (lr_d = 0)  ->  (r_lr′ = lr_clkOff) ;
  [tick]  (pm_mode = pm_meas)  &  (lr_s = lr_clkOff)   &  (lr_d = 0)  ->  (r_lr′ = lr_clkOn) ;
  [tick]  (pm_mode = pm_meas)  &  (lr_s = lr_dorm)     &  (lr_d = 0)  ->  (r_lr′ = lr_clkOn) ;
  [tick]  (pm_mode = pm_meas)  &  (lr_s = lr_clkOn)    &  (lr_d = 0)  ->  (r_lr′ = lr_meas) ;
  [tick]  (pm_mode = pm_meas)  &  (lr_s = lr_meas)     &  (lr_d = 0)  ->  true ;
endmodule
```

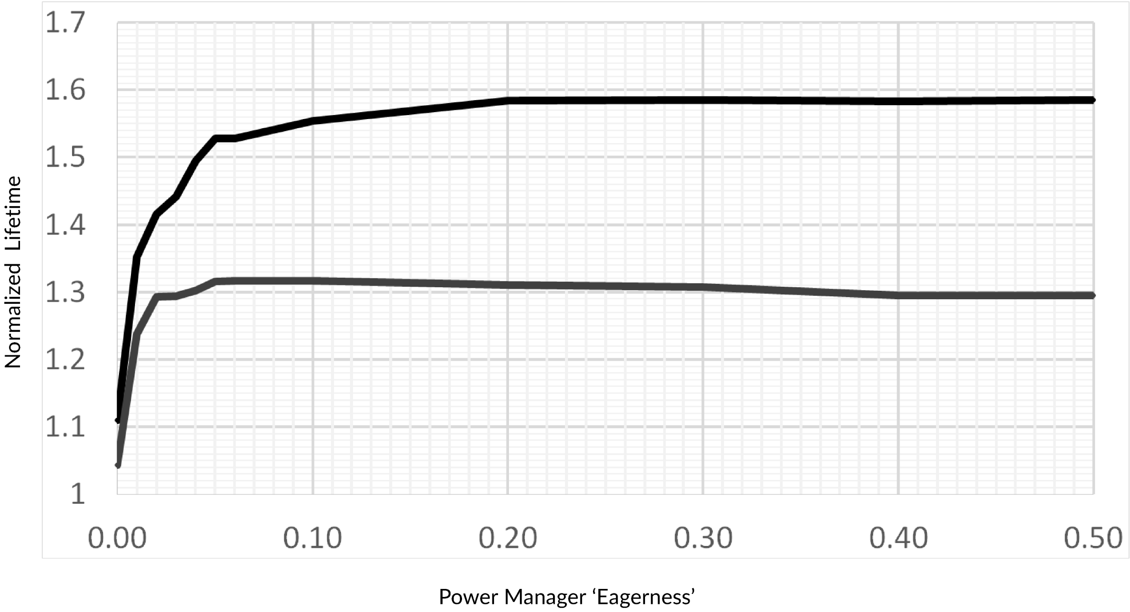


FIGURE 8 A plot of expected battery lifetime against 'eagerness'. Total system lifetime (upper curve) is compared to 'productive time' (lower curve).

The low-lever sensor is instructed to change state by setting the variable r_lr to the desired state. This can only be done when the sensor is not mid-transition, indicated by lr_d = 0. The high level power manager indicates a desired global power mode by setting pm_mode.

In this way there is a separation of concerns between representing allowed and erroneous transitions, which is done by the sensor model, managing multi-step state transitions to achieve global power states, which is done by the submodule manager, and setting global power states, which is done by the power manager.

Together, the sensor and submodule manager modules form a core which permits different user models and power management strategies to be specified at a high level and tested against the core. Recall that the model used to generate Fig 8 is a simple strategy of going into a low power state with some probability for each unit of time in which the system detects no user movements. This probability is a parameter of the strategy which here is called the 'eagerness'. The system is assumed to be able to detect the difference between cane swinging, turning, and rest, but not between short and long rests which are a private psychological state of the user. The power manager is then modelled with the following PRISM code.

```
module  pm
    pm_mode  :  [1..pm_states]  init pm_dorm ;

    [tick]  pm_mode = pm_meas  &  user_s = user_sweep   ->  (pm_mode′ = pm_meas) ;
    [tick]  pm_mode = pm_meas  &  user_s = user_turn    ->  (pm_mode′ = pm_pause) ;
    [tick]  pm_mode = pm_meas  &  user_resting          ->  (pm_mode′ = pm_pause) ;

    [tick]  pm_mode = pm_pause  &  user_s = user_sweep  ->  (pm_mode′ = pm_meas) ;
    [tick]  pm_mode = pm_pause  &  user_s = user_turn   ->  (pm_mode = pm_pause) ;
    [tick]  pm_mode = pm_pause  &  user_resting         ->  (pm_eagerness) : (pm_mode′ = pm_dorm) ;
                                                              + (1 - pm_eagerness) : (pm_mode′ = pm_pause) ;

    [tick]  pm_mode = pm_dorm  &  user_resting          ->  (pm_mode′ = pm_dorm) ;
    [tick]  pm_mode = pm_dorm  &  !(user_resting)       ->  (pm_mode′ = pm_pause) ;

endmodule
```

The battery model is more cumbersome than the other parts of the system, and has a simple structure but a huge state space. Therefore, all sensor models interact with the battery only by setting variables which indicate the current they draw on their power supplies. A formula combines these currents into the current experienced by the battery, which is the data needed for battery load modelling with the Rakhmatov model. This separation means that the system without the battery model can be model checked, for example to show the absence of deadlocks, by running the standard CTL query:

```
E[F "deadlock"]
```

Then the combined system, including the battery model, must be simulated to gain insight into the battery behaviour implied by the power management strategy.

TABLE 4 Issues Identified During Power Management Modelling and Analysis

Description	Model involved	Tool used	Modeling phase	Impact	Modelling effort
Undefined behaviour of Bluetooth module	Intermediate abstraction Event-B model	Rodin	Model development	Affected states avoided in power management design	Moderate: found while translating documentation into formal model
Deadlock detection while modelling power management	PRISM models of low-level power managers	PRISM	Model checking during model development	Ensures modelled strategy is deadlock free	Low: early use of model checking simplified model development
Power saving trade-offs analysed	PRISM models of integrated power manager	PRISM	Simulation	Allows battery life estimation and optimal parameter setting	Low: most effort in battery modelling which is reusable

10.3 | Outcomes of the Power Management Modelling and Analysis

The formal modelling activity that took place regarding the issue of power management applied formal methods according to a more traditional 'top down' approach, consisting of requirements gathering, high level policy design, and lower level elaboration, guided at each step by tool-supported formalisms. In the initial stages of the work, the submodules were represented by models in Rodin. As noted earlier, there were two of families of these, a top level family of models, and an intermediate level family of models. The models also provided a framework for discussing more detailed quantitative information, as it became available, undertaken with the PRISM tool.

The models of submodules, augmented by a model of battery behaviour, provided a foundation for the modelling of a simple power management strategy, with assumptions about user behaviour encoded by another PRISM model. The higher level power management strategy was combined with details about submodules' power consumption characteristics, yielding an intermediate level of description. The PRISM models embodying this level of detail could then be used to compute an approximation to the expected battery lifetime relative to a probabilistic model of the user.

The important contributions of this work to the INSPEX Project were of two kinds.

- Power management strategies were designed, optimized and evaluated at a high level, without detailed implementation having to be done.
- The correctness and practicality of the strategies designed were ensured by model checking, avoiding in advance issues such as undefined behaviour, deadlock and violations of sensor operating conditions.

Experience was gained regarding the limitations of the PRISM tool when implementing realistic battery models, as the model could only be represented at a moderate level of detail, due to combinatorial explosion phenomena of a familiar kind. Battery modelling is an increasingly important application area these days, so the fact that real-world systems were modelled, and that lessons were drawn from the results, constituted a significant advance.

The practical impacts of the modelling and analysis of power consumption and management in INSPEX are summarized in Table 4. As in the previous section, collateral issues, such as advantages related to documentation, communication, and justifiable design, are left out.

The relationship between the M&V work and the rest of the INSPEX Project in the context of power management was of a qualitatively different kind than for the data pathway case. There was much more interaction involved in agreeing the structure of the various STDs for the individual devices, and in estimating the relevant numerical values needed. The formal approach ensured that a consistent picture was maintained throughout the design process, again vindicating the introduction of formal techniques into the project.

10.4 | Other PRISM Formalisms

PRISM offers a number of different formalisms that can be used for verification purposes.[9] Although these were considered in the course of our work, in the end we did not use any of the alternatives to the Markov Decision Process formalism. Continuous Time Markov Chains were not used since nondeterminism and probability are necessary features for our application. For Probabilistic Timed Automata, the choice was less clear. Indeed, some features of these (e.g. the clocks) would have simplified our task somewhat. However, then modelling expenditure of energy would have required extra clocks to measure idle time etc., and the proper integration of all these features would have necessitated a repetition of a lot of what was required in the more manual approach via Markov Decision Processes, which thus nullified some of the advantages. So this, and the desire for consistent models mentioned above, decided in favour of Markov Decision Processes.

---

[9] As well as Markov Decision Processes, PRISM offers Continuous Time Markov Chains, and Probabilistic Timed Automata.

11 | SUMMARY AND CONCLUSIONS

The INSPEX Project provided a unique opportunity to deploy formal methods in a context that was far from the kinds of application area in which such techniques are usually used. As a *Smart Systems Integration* project, the main impetus was to develop sensor prototypes to a close to marketable level, and to integrate their working. The resource constraints that this imposed meant that formal techniques could only be applied in a selective way in the project. Guided by the project's lead partners, the M&V activity focused on the data acquisition pathway and on the power management strategy.

To allow the project to gather momentum, in the early stages we evaluated a number of potentially useful tools, summarised in Section 6. As noted there, Rodin and Pro-B turned out to be the tools that were found to be the most helpful over the majority of the project, allowing high level models of various aspects of the system to be constructed. For power management, relevant Rodin models could be translated to PRISM, and numerical values could then be added for evaluation of quantitative measures.

Crucial to the successful deployment of formal methods in INSPEX was the appreciation of many issues that were far from the usual technical focus of formal methods work. Section 7 summarised these, the majority of which concerned establishing a strong enough connection between formal methods concerns and human and interdisciplinary concerns. The way that these impinged on the technical M&V work was highlighted in the remainder of the paper.

The following two sections examined the work of the two main focus areas for formal methods in INSPEX in detail. In Section 9 we examined the work done on the data pathway. This was, perhaps, closest to the conventional way that formal methods are normally used, albeit it necessitated a very bespoke combination of top-down and bottom-up approaches, eliciting the high level properties from low level code, rather than mandating them at the outset. In Section 10 we examined the work done on power management modelling. This was, perhaps, a more unusual deployment of formal methods in that the numerically decorated transition diagrams were used to calculate expectation values for power consumption in a principled way, whereas the usual approach to the eliciting of such estimates is via relatively informal and *ad hoc* simulation.

As the account of the earlier sections showed, the use of formal techniques within INSPEX was very much human-in-the-loop. This constrasts markedly with most standard, textbook advocated approaches. In textbook approaches, one either starts in a top-down manner and refines towards code, or one starts with code and verifies properties of it — although reality usually entails a departure from the ideal textbook story. The M&V activities in INSPEX did not fit either pattern well. Instead formal techniques were interjected into the development path in a very pragmatically led way. Formal work maintained a constant dialogue with the output of conventional approaches, acting as a fresh pair of eyes, and utilising the tools to confirm rigour and correctness at the points of application. In some respects, this cooperation could be viewed as a sophisticated kind of design or code inspection. The human-in-the-loop cooperation with conventional techniques turned out to be the most novel aspect of the formal work in INSPEX, and was the most vital in ensuring its successful contribution to the project. It is hoped that the account in this paper will encourage a wider adoption of formal techniques to situations beyond the confines of the usual textbook advocated approaches, and that this will, in turn, help to convince system creators of the merits of the cooperative use of formal methods, even when the critical path of the development methodology is entirely conventional.

References

1. Rödel C, Stadler S, Meschtscherjakov A, Tscheligi M. Towards Autonomous Cars: The Effect of Autonomy Levels on Acceptance and User Experience. In: ACM Proc. AutomotiveUI-14, (2014): 11:1-8.
2. Wikipedia: Self-driving car https://en.wikipedia.org/wiki/Self-driving_car.
3. INSPEX Homepage http://www.inspex-ssi.eu/.
4. Lesecq S, Foucault J, Correvon M, et al. INSPEX: Design and Integration of a Portable/Wearable Smart Spatial Exploration System. In: Proc. DATE-17, (2017): 746-751.
5. Lesecq S, Ouvry L, Correvon M, et al. INSPEX: Integrated Smart Spatial Exploration System. In: Proc. SSI-17, (2017).

6. Lesecq S, Debicki O, Ouvry L, et al. Assistive Smart, Structured 3D Environmental Information for the Visually Impaired and Blind: Leveraging the INSPEX Concept. In: Proc. FedCSIS Communications, BEDA-18, (2018): 73-82.
7. Foucault J, Lesecq S, Dudnik G, et al. INSPEX: Optimize Range Sensors for Environment Perception as a Portable System. *Sensors* 2019; 19: 4350.
8. Razavi J, Banach R, Lesecq S, Debicki O, Mareau N, Foucault J. Formal Verification for Advanced Sensing Applications: Data Pre-processing in the INSPEX System. In: INSTICC Proc. ICSOFT-18, (2018): 664-671.
9. Banach R, Razavi J, Debicki O, Mareau N, Lesecq S, Foucault J. Exploring Applications of Formal Methods in the INSPEX Project. In: Springer LNCS Vol. 11176, Proc. STAFF-18 Workshops, (2018): 205-215.
10. Banach R, Razavi J, Lesecq S, et al. Formal Methods in Systems Integration: Deployment of Formal Techniques in INSPEX. In: Springer Proc. CSDM-18, (2018): 3-15.
11. Banach R, Razavi O, Lesecq S. Formal Modelling and Verification as Rigorous Review Technology: An Inspiration from INSPEX. In: Springer LNCS Vol. 12232, Proc. AFFORD-19, (2019): 77-91.
12. UBER Self Driving Car Fatality https://www.theguardian.com/technology/2018/mar/19/uber-self-driving-car-kills-woman-arizona-tempe.
13. Kedem B, De Oliveira V, Sverchkov M. *Statistical Data Fusion*. World Scientific 2017.
14. Thrun S, Burgard W, Fox D. *Probabilistic Robotics*. MIT Press 2005.
15. Moravec, H. and Elfes, A. High Resolution Maps from Wide Angle Sonar. In: Proc. IEEE ICRA, (1985).
16. Hall D. *Mathematical Techniques in Multisensor Data Fusion*. Artech House 2004.
17. Scalise L, Primiani V, Russo P. Experimental Investigation of Electromagnetic Obstacle Detection for Visually Impaired Users: A Comparison with Ultrasonic Sensing. *IEEE Trans. on Inst. and Meas.* 2012; 61: 3047-3057.
18. Dia R, Mottin J, Rakotavao T, Puschini D, Lesecq S. Evaluation of Occupancy Grid Resolution through a Novel Approach for Inverse Sensor Modeling. In: Proc. IFAC World Congress, Vol. 50, FAC-PapersOnLine, (2017): 13841-13847.
19. WHO. Visual Impairment and Blindness http://www.who.int/mediacentre/factsheets/fs282/en/.
20. Mandruchi R, Kurniavan S. Mobility-Related Accidents Experienced by People with Visual Impairment. *Insight: Research and Practice in Visual Impairment and Blindness* 2011.
21. Connoly D. Grade 9 Science Fair Wunderkind Creates a Smarter White Cane. 2012. Engineering.com, http://www.engineering.com/DesignerEdge/DesignerEdgeArticles/ArticleID/16419/Grade-9-Science-Fair-Wunderkind-Creates-a-Smarter-White-Cane.aspx.
22. Wang Y, Kuchenbecker K. HALO: Haptic Alerts for Low-hanging Obstacles in white cane navigation. In: IEEE Haptics Symposium (HAPTICS), (2012): 527-532.
23. Smartcane https://www.phoenixmedicalsystems.com/assistive-technology/smartcane/.
24. Ultracane https://www.ultracane.com/.
25. Bawa https://www.bawa.tech/.
26. Rango http://www.gosense.com/rango/.
27. Apple VIB Navigation Announcement https://appleinsider.com/articles/18/06/28/apple-mulls-system-for-helping-visually-impaired-navigate-environment.
28. ProB Tool https://www3.hhu.de/stups/prob/.
29. FDR Tool https://www.cs.ox.ac.uk/projects/fdr/.
30. UPPAAL Tool http://www.uppaal.org/.

31. PRISM Tool https://www.prismmodelchecker.org/.
32. NuSMV Tool nusmv.fbk.eu/.
33. RODIN Tool http://www.event-b.org/.
34. Atelier B Tool http://www.atelierb.eu/en/atelier-b-tools/.
35. TLA Toolbox https://lamport.azurewebsites.net/tla/toolbox.html.
36. Perfect Tool https://www.eschertech.com/.
37. BLAST Tool https://forge.ispras.ru/projects/blast/.
38. Déharbe D, Fontaine P, Guyot Y, Voisin L. SMT Solvers for Rodin. In: Springer LNCS Vol. 7316, Proc. ABZ-12, (2012): 194-207.
39. Déharbe D, Fontaine P, Guyot Y, Voisin L. Integrating SMT solvers in Rodin. *Sci. Comp. Prog.* 2014; 94: 130-143.
40. Rodin SMT Solvers https://wiki.event-b.org/index.php/SMT_Solvers_Plug-in.
41. IEC 62304 https://webstore.iec.ch/publication/22794.
42. Theisz V. *Medical Device Regulatory Practices: An International Perspective.* Pan Stanford Publishing 2015.
43. NASA: Mars Climate Orbiter https://solarsystem.nasa.gov/missions/mars-climate-orbiter/in-depth.
44. Wikipedia: Mars Climate Orbiter https://en.wikipedia.org/wiki/Mars_Climate_Orbiter.
45. Abrial JR. Formal Methods in Industry: Achievements, Problems Future. In: 2006: 761-768.
46. Bowen J, Hinchey M. Seven More Myths of Formal Methods. *IEEE Software* 1995; 12: 34-41.
47. Hall A. Seven Myths of Formal Methods. *IEEE Software* 1990; 7: 11-19.
48. Bowen J, Hinchey M. *High-Integrity System Specification and Design.* Springer, 1999.
49. Bowen J, Hinchey M. *Industrial-Strength Formal Methods in Practice.* Springer, 1999.
50. Clarke E, Wing J. Formal Methods: State of the Art and Future Directions. *ACM Comput. Surv.* 1996; 28: 626-643.
51. Hall J. Realising the Benefits of Formal Methods. *JUCS* 2007; 13: 669-678.
52. Banach R., ed., *Special Issue on the State of the Art in Formal Methods*. Vol. 13(5) of *Journal of Universal Computer Science*; 2007.
53. Woodcock J, Gorm Larsen P, Bicarregui J, Fitzgerald J. Formal Methods: Practice and Experience. *A.C.M. Computing Surveys* 2009; 41(4): Art. 19.
54. Kossak F, Mashkoor A. How to Select the Suitable Formal Method for an Industrial Application: A Survey. In: Springer LNCS Vol. 9675, Proc. ABZ-16, (2016): 213-228.
55. Mashkoor A, Kossak F, Egyed F. Evaluating the Suitability of State-Based Formal Methods for Industrial Deployment. *Soft. Prac. Exp.* 2018; 48: 2350-2379.
56. Andronick J, Jeffery R, Klein G, et al. Large-Scale Formal Verification in Practice: A Process Perspective. In: Proc. ACM/IEEE ICSE, (2012): 374-393.
57. Gleirscher M, Marmsoler D. Formal Methods in Dependable Systems Engineering: A Survey of Professionals from Europe and North America. *Empirical Software Engineering* 2020; 25: 4473-4546.
58. Jackson M, Wing J. Lightweight Formal Methods. *IEEE Computer* 1996; 29(4): 22-23.
59. Jones C. A Rigorous Approach to Formal Methods. *IEEE Computer* 1996; 29(4): 20-21.
60. Agerholm S, Larsen P. A Lightweight Approach to Formal Methods. In: Springer LNCS Vol. 1641, FM-Trends-98, (1999): 94-113.

61. Zamansky A, Spichkova M, Rodriguez-Navas G, Herrmann P, Blech J. Towards Classification of Lightweight Formal Methods. In: SCITEPRESS Proc. ENASE-18, (2018): 305-313.

62. FreeRTOS https://www.freertos.org/.

63. Fagan M. Design and Code Inspections to Reduce Errors in Program Development. *IBM Systems Journal* 1976; 15: 182-211.

64. Fagan M. Advances in Software Inspections. *IEEE Trans. Soft. Eng.* 1986; SE-12: 744-751.

65. Divakaran S, D'Souza D, Kushwah A, Sampath P, Sridhar N, Woodcock J. Refinement-Based Verification of the FreeRTOS Scheduler in VCC. In: Springer LNCS Vol 9407, Proc. ICFEM-15, (2015): 170-186.

66. Gadelha M, Monteiro F, Morse J, Cordeiro L, Fischer B, Nicole D. ESBMC 5.0: An Industrial-Strength C Model Checker. In: ACM Proc. ACM/IEEE ASE-18, (2018): 888-891.

67. Jongerden M, Haverkort B. Which Battery Model to Use?. *IET Software* 2009; 3: 445-457.

68. Rakhmatov D, Vrudhula S. An Analytical High-Level Battery Model for Use in Energy Management of Portable Electronic Systems. In: IEEE Proc. IEEE/ACM Int. Conf. CAD, (2001): 488-493.

69. Ahmad W, Pol v. dJ. Synthesizing Energy-Optimal Controllers for Multiprocessor Dataflow Applications with Uppaal Stratego. In: Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques, Springer (2016): 94-113.

How to cite this article: R. Banach, J. Razavi, O. Debicki, S. Lesecq (9999), Formal Methods by Stealth: The INSPEX Experience, *Journal of Software: Evolution and Process*, *9999;12:345-678*.