Formal Verification for Advanced Sensing Applications: Data Pre-processing in the INSPEX System

Joe Razavi[1], Richard Banach[1], Suzanne Lesecq[2], Olivier Debicki[2], Nicolas Mareau[2], Julie Foucault[2], Marc Correvon[3] and Gabriela Dudnik[3]

[1]*School of Computer Science, University of Manchester, Oxford Road, Manchester, M13 9PL, U.K.*

[2]*CEA, LETI, Minatec Campus, 17 Rue des Martyrs, F-38054 Grenoble Cedex, France.*

[3]*CSEM SA, 2002 Neuchatel, Switzerland.*

{*joseph.razavi,richard.banach*}*@manchester.ac.uk,*
{*suzanne.lesecq,olivier.debicki,nicolas.mareau,julie.foucault*}*@cea.fr,*
{*marc.correvon,gabriela.dudnik*}*@csem.ch*

Keywords: Formal Modelling, Sensor Processing, Embedded Devices, Wearable Technology, Assistive Technology.

Abstract: The INSPEX project aims to miniaturize state-of-the-art obstacle detection technology comprising heterogeneous sensors and advanced processing, so that it can be used for wearable devices. The project focuses on enhancing the white cane used by some visually impaired and blind people. Due to high demand for reliability and performance, the project is a good candidate for the use of formal methods. In this paper, we report lessons we have learned from formal modelling exercises related to the pre-processing of sensor information in INSPEX.

1 INTRODUCTION

At rare and exciting moments, progress in information technology makes an impact on everyday life so great 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.

One such prominent change is in the application of sensors in the automotive industry, in such contexts as assisted parking and automatic management of headlights, not to mention autonomous driving.

Another is in the realm of assistive technology. In this paper, we describe some aspects of work on the INSPEX project, which aims to bring state-of-the-art obstacle detection capabilities to domains requiring small, light, power efficient hardware. The project's particular focus is on assistive technology for visually impaired and blind people, and thus it also incorporates new techniques for the presentation of obstacle information suited to this use. The goal is to produce a prototype system which can be mounted on a white cane as traditionally used by many visually impaired people. The specific aim of this paper is to discuss the application of formal methods in this area. We attend principally to those aspects concerning the pre-processing of sensor data.

We begin by motivating this use case as one for which the application of advances in sensing, data processing, and information presentation are germane. We then briefly describe the details of the project, and describe the relevance of formal methods to the project's goals. We go on to describe lessons learned about the application of formal techniques in this domain, gleaned from our experience so far, finishing by using a simplified example from INSPEX to highlight one aspect of this.

1.1 Augmenting the Capabilities of the White Cane

A number of attempts by students and researchers to attach sensors to the white canes which are an established tool for visually impaired and blind people are visible in the literature (Connoly, 2012; Wang and Kuchenbecker, 2012). Indeed, there are already commercially available white canes enhanced by the addition of sensors (UlraCane, 2012; SmartCane, 2018). One might wonder why a piece of equipment with such a long history of successful use needs modification, but in fact the argument is compelling. While visually impaired people are experts at using the tactile feedback from the cane to understand their surroundings at ground level, a white cane, by its nature, can not 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. This represents an unfortunate constraint on personal independence, since it means proceeding with great caution, and possibly making use of a guide dog, a sighted friend or assistant, or otherwise unwanted hats or similar headwear.

The addition of a range sensor to the cane which points upwards can help with this problem. As the user sweeps the cane, receiving tactile feedback from the ground, they also receive feedback from the sensor about the distance to the nearest obstacle at a higher level.

To advance the state of the art we should therefore ask: how can the next generation of this technology improve on what currently exists? There are lots of possibilities. A crucial one is that what exists today typically relies on a single sensor, say one based on sonar, radar, or a laser. Each type of sensor has its own strengths and weaknesses. Sensors may perform poorly for surfaces with certain reflectivities, curvatures or other properties, or in conditions of varying ambient light and humidity, whereas other sensors are likely to have complementary capabilities. Similarly, some sensors are well suited to resolving small or distant objects, but have a narrow field of view, whereas others may resolve detail less well, but cover a broad area.

Another important potential improvement is in the cognitive load placed on the user. When the feedback presented to the user is in the form of relatively direct range readings from the sensor, the user's brain must do the work of interpreting this information and reconstructing the three dimensional environment. This may involve getting used to the quirks and idiosyncrasies of the sensor. In a system with multiple sensors, this problem would be compounded. The user must then combine this with their model of the ground-level situation, their plans and route, and their knowledge of contextual issues such as the location of pedestrian crossings and amenities.

By combining the inputs of multiple sensors, and processing them to make the most useful information more salient to the user, the usefulness of cane-mounted sensors could be greatly improved: these are the prospects identified by the INSPEX project; its aim is to realize them. This entails that significant technical challenges be met.

In INSPEX, the feedback given to the visually impaired user is presented via 3D immersive sound, transmitted to the user via binaural headphones. In order to be convincing for the user, the sound picture must be stable with respect to a 3D inertial frame, so as well as the issues of cognitive load etc., discussed above, the INSPEX system must be aware of the user's head movements, in order to achieve the needed spatial stability for the sound image. This adds another technical challenge.

1.2 The INSPEX Project

The INSPEX project (INSPEX Homepage, 2017) is an international collaboration with the goal of developing a small, lightweight system which combines the inputs of multiple sensors and integrates their readings into a three dimensional model of the obstacles in its surroundings. Such a system would have multiple potential applications, including autonomous drones and fire-fighters working in low-visibility conditions. The main focus, however, is on the use case of assistive technology which could be attached to a white cane, as described above.

Achieving this ambition means facing several fundamental obstacles. To be a useful tool to a visually impaired person, INSPEX must provide reliable, high performance functionality over the course of many hours. If the user is required to stop several times per day to re-charge batteries, then their independence and quality of life will not have been significantly improved. At the same time, the system must be held in the hand and moved continuously by the muscles of the wrist for long stretches of time. This implies that the system must be lightweight; otherwise, rather than enhancing their day-to-day life, the system could cause injury to the user. These requirements translate into a number of technical challenges.

First, the sensors themselves need to be significantly miniaturized, and their weight and power consumption must be reduced. Concomitantly, the computation power needed for processing must be tightly controlled, so that lighter, more efficient processors can be used. For this reason, the standard Occupancy Grid algorithm used for obstacle detection in the automotive domain has had to be significantly optimized for architectures with fewer facilities than its usual implementations (Dia et al., 2017). This constraint also implies that clever optimizations will be required throughout the basic utility systems which underpin the advanced data processing. These will have to function with minimal memory, and will have to share hardware resources with other parts of the system, requiring them to be robust against competition for resources.

Second, power must be managed in a sophisticated way. The needs of the data processing algorithms for a supply of frequent, timely data must be balanced against the overall requirement to conserve power. Sensors and communication facilities cannot be allowed to operate when they are not needed, yet they must function at the proper time when their functionality is required. This means that power management cannot be treated as a simple matter, and some of the system's scarce computing power must be used to determine when to supply power to which subsystem.

Finally, the system must be robust, and guarantee a certain level of performance. The project will not be a success if it is possible, even under rare conditions, for the user to rely on the system only to be stranded by crashing, deadlock, or because of incorrect power consumption estimates. This must be true for long-term use every day, in which one should assume that all possible conditions will eventually be encountered. To play a role in the user's life which is as fundamental as that played by a traditional white cane, an obstacle detection system must be as dependable as that simple, effective piece of technology. This requirement is in tension with the need for highly optimized code with advanced functionality in a highly concurrent environment. Experts know well that it is easy to introduce subtle bugs under such conditions. These circumstances indicate the possibility for formal methods to have a useful impact.

2 FORMAL METHODS

2.1 Applying Formal Methods in Innovative Sensing Applications

Formal methods are a class of techniques which allow software to be compared against mathematical models, providing an improved ability to detect subtle functional and logical bugs, and, under good conditions, prove that a system is free from certain classes of problems. Formal methods are nowadays well-established for safety-critical systems and mission-critical infrastructure in industrial applications. These techniques are becoming mature in settings wherein the function of the system is well-defined. In the INSPEX system, and in similar sensing applications, the requirement for highly reliable, but sophisticated software is a motivation to pursue the use of formal methods. However, the nature of an innovative project focussing on sensing means that it is more difficult to give a fixed, precise description of correct system functioning *a priori*. This creates opportunities to learn interesting methodological lessons in the application of formal techniques.

2.1.1 Functioning Under Conditions of Uncertainty

One difficulty in attempting formal verification of such systems is that usually their main function is to perform advanced statistical inference on incoming data, or to apply novel heuristics to improve performance in a complex, ill-defined domain. While one can certainly employ static analysis to catch low-level bugs, and one could attempt to show that the implementation code is a faithful version of the intended algorithms, it may appear that beyond this, innovative sensing applications are rather sterile from the point of view of applied formal methods. After all, one can as little hope to prove that INSPEX correctly identifies all obstacles in an environment as to prove that a spam filter catches all and only spam email.

However, even in the example of a spam filter, one can still hope to prove that, for example, email from whitelisted or blacklisted addresses is allowed or blocked as appropriate. In the context of a sensing application, there is a lot of scope to prove that the data being supplied to the statistical analysis is of a good quality and freshness, and satisfies any pertinent constraints.

For example, many sensing systems require incoming data to be pre-processed into a form suitable for further computation. One could imagine that this pre-processing consists of computing a function $f(x,y)$ of heterogeneous input data as frequently as possible, but under certain constraints. We might imagine that y values represent crucial readings, while x values give contextual information. Thus, while x values may be re-used, y values must be used at most once. Moreover, a timing constraint must hold between the times when x and y values which are to be combined were received, to ensure that the contextual information x is really relevant to the reading y. Of course, in real applications, these values x and y are likely themselves to be composed of data from various heterogeneous sources under constraints of their own, and so on.

One easily foresees that under strict memory limitations, and in the presence of autonomous sensors providing information asynchronously, these types of constraints can lead to difficult problems. One wants to show that as long as the system receives enough input information of the correct types, it outputs the pre-processed data for statistical analysis with a reasonable frequency. This could fail to happen if, for example, small internal buffers become full, preventing the storage of new input data. This blocked data might be precisely what is required to produce output satisfying the operating constraints and thus discharge some of the held data. The result of this situation would be deadlock. Conversely, if contextual data is overwritten by incoming readings which rely on it in order to be processed, this too will cause no output ever to be produced. Finally, it is possible that fresh data are discarded in order to finish processing old data. This can impact the frequency and freshness of the data used for analysis, and consequently the quality of information ultimately delivered to the user.

Intermediate between simple low-level bug detection and issues of data frequency and quality are problems related to basic underpinning functionality. For example, wherever there is communication, it is likely that there are systems which must maintain a degree of synchrony with each other. They may have to do this under conditions in which they may be interrupted, since they compete for scarce processor time with other aspects of the system. One must show that the algorithms used to maintain the required level of synchrony can withstand these interruptions and continue to operate with reasonable performance. The presence of communication also implies issues of encoding and decoding. For example, transmitted data is frequently processed such that certain special sequences never occur. These sequences may be used as delimiters or framing markers in packet-oriented communication. Alternatively, they may be command characters, which are common for hardware devices such as low-power bluetooth modules: these often have only one input line for both control commands and data, using a special data sequence to switch from one mode to another. This implies, at least, the application of bit-stuffing, byte-stuffing, or string escaping algorithms (Cheshire and Baker, 1999) which must be correctly applied at one end of a communication, and correctly inverted at the other.

These examples show that the main function of an advanced sensing system is usually underpinned by many subsystems which do have clearly defined roles and correctness conditions. Verification of conditions of this type can build upon existing work which has been done to verify operating-system level components. For example, there is a body of work related to the verification of the real-time operating system FreeRTOS (FreeRTOS, 2011; Divakaran et al., 2015), which is widely used in this type of application. From our point of view, it is interesting that this work uses a multi-tool approach, starting with refinement based tools to relate overall system requirements to those of specific subsystems, and then transitioning to tools designed to verify C code in order to check that these subsystems correctly perform their function.

2.1.2 Formal Methods and Design-Rich Projects

Another source of difficulty in applying formal methods to cutting-edge applications relates to projects which have a significant element of ongoing design activity. Such projects seek to discover the extent to which innovative techniques can improve the state of the art in their domain. At the start of the project, it is not clear what level of performance is possible, and what level of performance is necessary in order to achieve meaningful success. Indeed, the whole point of such exploratory development is that this is to be discovered while the project is ongoing. This implies that many aspects of the design will be open to modification throughout the project, as lessons are learned from the implementation effort. This gives rise to several challenges.

First, there may be requirements which appear to be related to optimization, but which ultimately turn out to be necessary for proper functioning. For example, suppose one is working with a statistical algorithm which performs best when its input is a sparse matrix. During pre-processing, various heuristics may be applied to try to find a representation in which the data has this sparse form. At the outset, this appears to be a non-functional optimization. However, it may be discovered that it is critical that this step always attains some minimal level of performance; otherwise, the later processing stages may not perform well enough for the system to provide the functionality needed. A similar situation occurs for power management. A wide variety of tricks may be employed to make power consumption as small as possible, but it may not be known what level of power consumption is acceptable. This can pose a challenge not only for formal modelling, but also for requirements gathering.

Engineers will have invested a significant amount of ingenuity in trying to make performance as good as possible, and the properties of the output which are thereby aimed for may be the focus of their explanation of certain subsystems. It may very well be unclear which of these properties are necessary to avoid crashes or deadlocks, which are required for a usable level of functionality, and which are enhancements which stretch the frontier of the possible. While it is important to try to elicit this distinction from domain experts, it is equally important to recognize that when the success of a project consists in improving on the state of the art, the issue is not as clear cut as we might like.

To deal with challenges of this type, two strategies can be employed. Thus, one can try from the outset to prove that these optimizations achieve a bare minimum of performance, such as universally improving over the un-optimized situation. Furthermore, one can again make use of a multi-tool approach, using a tool such as PRISM (Kwiatkowska et al., 2011) to find bounds on performance, possibly using conservatively simplified models. If engineers are happy with these bounds, other tools could be used to confirm that the detailed design and implemented code adhere to them.

A second set of challenges which proceed from the innovative nature of cutting-edge projects relates to the fact that these projects will often not start from a blank slate, but instead proceed by modification of previous work. One ought not to think of this as an error; instead, it arises from the nature of invention. Out of the infinity of possible human wants, the productive innovator must choose to pursue those which are realistically achievable. The best guide to this is a cultivated sense of the directions in which existing work admits modification. This precludes an idealized version of formal development, in which code is derived from abstract models. It necessitates a pragmatic approach, where models are derived partly from existing code, as well as feedback from engineers about the intended functioning of the system.

A crucial observation is that these systems are also often developed in an iterative fashion, via a series of prototypes. This is unavoidable in work where it is unknown whether the proposed method will really work, and lessons may have to be drawn from the initial attempts which guide the final development. This means that the effort put into discovering abstract models of the initial version can be put to use to guide the development of subsequent iterations. Indeed, abstract models can provide a valuable alternative view of the situation to that revealed by code. This is particularly useful because the low-level details may have to be radically different in different prototypes of a system, once constraints coming from fundamental hardware issues are discovered. For efficiency reasons, logically separate concerns may not be well separated in the original code, and these may need to be teased apart if the underlying structure of the system changes. In addition, elements which were necessary in one version may become obsolete in another, but this may not be visible in highly-optimized code. On the other hand, an abstract view of the system can make this clear. A simplified example of this type of phenomenon is discussed in the next section.

2.2 Formal Modelling in INSPEX

In the INSPEX system, we find many opportunities for the fruitful application of formal techniques. Examples abound in the power management firmware relating, for example, to the need for circuits which operate a high-power subsystem to always be switched on when the system they manage is. In this section, we focus on an example based on the sensor pre-processing stage.

We imagine a system with multiple sensors, each of which may transmit a reading to an acquisition module at any time. The acquisition module then performs a computation along the lines of that described above, integrating the heterogeneous data received into a form suitable for further processing. This data is then encoded and transmitted to the main processing platform which decodes it and works through the items one by one.

Since memory usage is a critical issue, it is freed as soon as possible. In the acquisition system, this occurs once the data it contains has successfully been transmitted. In the main processor, this occurs once it has been used by the analysis algorithm.

It would be tempting to model these two subsystems individually, as separate verification projects — but this would be a mistake. One reason is visible from the point of view of modelling in itself. The inputs or outputs of these systems, corresponding to the transmitted information, are quite complex, even if one abstracts from the precise details of the encoding. Furthermore, if it is ultimately decided that these models should be combined to reason about global behaviour, then one would have to grapple with ensuring that refinements adding in these low-level details interact well with shared events. This suggests considering the encoding, transmission, and decoding together as a conceptual unit. It operates on much simpler data and, conceptually, implements the fetching of pre-computed data by the main algorithm. This simple subsystem could be decomposed, if required, at a much later stage of refinement.

A more application oriented reason to leave out this aspect in the abstract models of the system is that the underlying connectivity of the hardware may well change in a more advanced prototype. Since we would like our abstract models to guide the transition from one prototype to another, at an abstract level they should express what such systems must have in common.

Therefore, in our abstract system, we have three main kinds of events: sensors transmitting raw data, pre-processing to compute combined values, and the main algorithm consuming these pre-processed values. In addition, we have the action of freeing the memory used to store these resources. In this paper, we ignore how the memory related to raw values is managed, focussing on when combined values may be freed. In the abstract system, this can be done only after they are used by the main algorithm.

In the INSPEX project, the majority of the formal modelling is done using Event-B (Abrial, 2010) via the Rodin tool (Abrial et al., 2010), though other techniques also contribute. In Event-B, the above could be modelled as follows. We assume the state of the abstract system is given by a set *messages* representing the set of all integrated messages ever computed by the pre-processing stage. We model the memory status of messages and whether they have been consumed by the main algorithm using two variable functions, *inMemory* and *used*. These map elements of *messages* to booleans. All three of these variables are initialized to the empty set. We ignore the other aspects of the state, which relate to the collection and integration of raw messages received, although one would include them in a complete model. Focussing on the consumption and deallocation of processed messages, there are two corresponding events in the abstract system, which we call *consume* and *free*. We write these events in the Event-B formalism as follows.

```
EVENTS
  consume
    ANY  $m$
    WHEN
      $m \in messages$
      $inMemory(m) = TRUE$
      $used(m) = FALSE$
    THEN
      $used(m) := TRUE$
    END
  free
    ANY  $m$
    WHEN
      $m \in messages$
      $inMemory(m) = TRUE$
      $used(m) = TRUE$
    THEN
      $inMemory(m) := FALSE$
    END
```

The 'ANY' clause of an event specifies its parameters: in the case of both events above this is just the message m which is to be consumed by the main algorithm or deallocated. The 'WHEN' clause specifies *guards*, conditions which must hold for the event to take place; these are simply boolean conditions on the parameters and the state variables of the model. The 'THEN' clause determines what the effect of the event on the variables describing the system's state is. For both events above, this takes the form of changing the value of a variable function on one of its arguments to a new value. For example, in the event *consume* we set $used(m)$ to $TRUE$.

This abstract model might seem wrong, because we know that in our concrete system, it is safe for the pre-processing subsystem to free the data once it has been transmitted to the main processor. However, the 'memory' of the abstract system is, in reality, the combined memories of the two components of the concrete system. To be freed in the abstract system is, in some sense, no longer to exist in the memory of either module. This guides the way we understand the relationship between concrete and abstract systems in refinement.

For this reason, it makes sense for the model to remember every piece of data which is ever received, even those which have been deallocated. Those which are still in memory are marked as such, and a precondition of any event which makes use of them must then be that they are present in memory.

Performing one step of refinement, we would refine the action by which the pre-processor makes data available to the main algorithm so that it now includes the step of transmission. This could be subsequently refined to model the encoding and decoding, issues of synchronization, and finally even bit-level modelling of the sequence of transmitted signals.

In the concrete system, the notion of being present in memory is elaborated so as to specify that an item is present in the pre-processing subsystem or the main processor (or both). The abstract notion of being present in memory is simply the disjunction of these conditions. The guards of various actions in the concrete system must specify which subsystem they require their data to be present on, thus strengthening the abstract guards. At this intermediate level of abstraction, the transmission action essentially copies data from one subsystem to another. This makes it available for the second subsystem to use, but also means that the first is free to deallocate it without disturbing the abstract invariant that it be present somewhere in the system. It is this which explains why the logic of deallocating transmitted information makes sense.

In Event-B, we model the concrete system by replacing the *inMemory* function with two functions *inMainMemory* and *inPreMemory* to represent being in the memory of the main module and the pre-processing module respectively. We also add a new function *transmitted* from *messages* to booleans to model whether a message has been sent from the pre-processing module to the main module.

Since we have removed the function *inMemory*, we have to describe how this abstract variable can be recovered from the concrete data. Following the reasoning above, we add an invariant, a property of the state variables which must be true in every state, indicating the intended relationship.

```
INVARIANTS
  $\forall m \cdot m \in messages \implies$
    $inMemory(m) = TRUE \iff$
      $(inPreMemory(m) = TRUE \lor$
       $inMainMemory(m) = TRUE)$
```

Unlike a normal invariant, which is a boolean expression involving only the state variables of the model it is part of, this invariant involves variables from both the concrete system and the abstract system. This special type of invariant is called a *glueing invariant*. The fact that we intend the concrete system to refine the abstract system must be made explicit, and this is done by using the 'REFINES' keyword at the appropriate place.

The *consume* and *free* events must now be modified so as to refer to the variables of the concrete system. The *free* event is to be refined by multiple events, which involves some subtlety. However, the concrete version of the *consume* event is straightforward:

```
EVENTS
  consume
    REFINES consume
    ANY $m$
    WHEN
      $m \in messages$
      $inMainMemory(m) = TRUE$
      $used(m)         = FALSE$
    THEN
      $used(m) := TRUE$
    END
```

We also add a new event *transmit* which models sending a message from the pre-processing module to the main module.

```
  transmit
    ANY $m$
    WHEN
      $m \in messages$
      $inPreMemory(m) = TRUE$
      $transmitted(m) = FALSE$
    THEN
      $transmitted(m)  := TRUE$
      $inMainMemory(m) := TRUE$
    END
```

Note that as *transmit* does not refine any events of the abstract system it must not have any effect on the variables of the abstract system. This is true because $inMemory(m)$ is true whenever the event is triggered, (because of the guard $inPreMemory(m) = TRUE$), and it remains true afterwards.

We now turn to refining the *free* event. Conceptually, we replace this by two events, one to free transmitted messages in the pre-processing module, and one to free consumed messages in the main module. However, a concrete event must either refine an abstract event or else it must leave the abstract state unchanged. To refine a abstract event, the guards of the corresponding concrete event must imply those of the abstract event, and the glueing invariant must be preserved by the concrete event if the abstract state is changed simultaneously by the abstract event it refines.

However, deallocating a message on just one of the modules sometimes has no effect on the abstract state (if the message is present in the memory of the other module) or else it changes the state of *inMemory* (if the other module has already deallocated the message). For this reason, each of the two *free* operations has two cases, one refining the abstract *free* and one which does not affect the abstract state. Since the two cases are very similar, it suffices to consider the operation of deallocation on the pre-processing module.

```
  pre_only_free
    ANY $m$
    WHEN
      $m \in messages$
      $inPreMemory(m)  = TRUE$
      $transmitted(m)  = TRUE$
      $inMainMemory(m) = TRUE$
    THEN
      $inPreMemory(m) := FALSE$
    END
  pre_last_free
    REFINES free
    ANY $m$
    WHEN
      $m \in messages$
      $inPreMemory(m)  = TRUE$
      $transmitted(m)  = TRUE$
      $inMainMemory(m) = FALSE$
    THEN
      $inPreMemory(m) := FALSE$
    END
```

The fact that the guards of these events refer to the state of the main module means that some care must be taken when modelling each subsystem in isolation. However, in practice the problems are not too severe, since the effect on the local state of the preprocessing module is the same in either case.

3 CONCLUSIONS

In this paper, we describe the application of formal methods in the INSPEX project. This project aims to develop a small, light and power efficient obstacle detection system which incorporates heterogeneous sensors and provides feedback to the user in the form of three dimensional immersive sound.

We have set this work in the context of the problem of applying formal methods to projects featuring continuously ongoing design, particularly in the sensing domain. We hope that this report of our experiences will be useful for others facing the need for the increased dependability that formal techniques can bring, in contexts where the design is volatile to a significant degree. As applications of this type become increasingly common, it is important to emphasize the benefits of formal methods in such settings.

ACKNOWLEDGEMENTS

This 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.

REFERENCES

Abrial, J.-R. (2010). *Modeling in Event-B: System and Software Engineering*. CUP.

Abrial, J.-R., Butler, M., Hallerstede, S., Hoang, T. S., Mehta, F., and Voisin, L. (2010). Rodin: an open toolset for modelling and reasoning in Event-B. *International Journal on Software Tools for Technology Transfer*, 12(6):447–466.

Cheshire, S. and Baker, M. (1999). Consistent overhead byte stuffing. *IEEE/ACM Transactions on Networking*, 7(2):159–172.

Connoly, D. (2012). Grade 9 science fair wunderkind creates a smarter white cane. Engineering.com, http://www.engineering.com/DesignerEdge/DesignerEdgeArticles/ArticleID/16419/Grade-9-Science-Fair-Wunderkind-Creates-a-Smarter-White-Cane.aspx.

Dia, R., Mottin, J., Rakotavao, T., Puschini, D., and Lesecq, S. (2017). Evaluation of Occupancy Grid Resolution through a Novel Approach for Inverse Sensor Modeling. In *Proc. IFAC World Congress, FAC-PapersOnLine*, volume 50, pages 13841–13847.

Divakaran, S., D'Souza, D., Kushwah, A., Sampath, P., Sridhar, N., and Woodcock, J. (2015). Refinement-Based Verification of the FreeRTOS Scheduler in VCC. In Butler, Conchon, and Zaidi, editors, *Proc. ICFEM-15*, volume 9407 of *LNCS*, pages 170–186. Springer.

FreeRTOS (2011). https://www.freertos.org/.

INSPEX Homepage (2017). http://www.inspex-ssi.eu/.

Kwiatkowska, M., Norman, G., and Parker, D. (2011). PRISM 4.0: Verification of probabilistic real-time systems. In Gopalakrishnan, G. and Qadeer, S., editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer.

SmartCane (2018). http://smartcane.saksham.org/.

UlraCane (2012). https://www.ultracane.com/.

Wang, Y. and Kuchenbecker, K. (2012). Halo: Haptic alerts for low-hanging obstacles in white cane navigation. In *2012 IEEE Haptics Symposium (HAPTICS)*, pages 527–532.