Towards Assurance Cases for Resilient Control Systems

(Invited Paper)

James Weimer, Oleg Sokolsky, Nicola Bezzo, and Insup Lee
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA, USA
Email: weimerj,sokolsky,nicbezzo,lee@seas.upenn.edu

***Abstract*—The paper studies the problem of constructing assurance cases for embedded control systems developed using a model-based approach. Assurance cases aim to provide a convincing argument that the system delivers certain guarantees, based on the evidence obtained during the design and evaluation of the system. We suggest an argument strategy centered around properties of models used in the development and properties of tools that manipulate these models. The paper presents the case study of a resilient speed estimator for an autonomous ground vehicle and takes the reader through a detailed assurance case arguing that the estimator computes speed estimates with bounded error.**

***Keywords*-Assurance case, cyber-physical systems, resilient control systems**

I. Introduction

Cyber-physical systems (CPS) are often deployed in critical environments, where human life and safety, as well as success of expensive missions, depend on the system being able to perform its functions in adverse conditions that are hard to predict in advance. These adverse conditions may include faults, unpredictable environments, or malicious activity. To succeed, the system must be designed to be resilient to these conditions. A substantial fraction of CPS design efforts are spent of establishing and guaranteeing resilience.

As CPS become more and more complex, providing such resilience guarantees is more and more difficult. Rigorous model-based design techniques, extensive verification and validation (V&V) are all necessary to ensure resilience. However, these activities need to be performed in a concerted fashion to make sure that all efforts are consistent and nothing important is missed. Design and V&V activities yield large amount of artifacts – such as requirement specifications, test and verification results, design reviews, etc., – that can serve as evidence that the system achieves the desired goals. However, evaluating consistency between different evidence items and any potential gaps is a daunting task that requires a deep understanding of the system requirements, its intended requirements, design approaches, etc. Assumptions made in the process of design and V&V are critical for proper understanding of the available evidence.

In a large CPS design project, when a large team is engaged in design and V&V activities it can be difficult to maintain a centralized, coherent view of the system and its associated evidence in all its detail. It can be even more difficult to communicate this view to regulators who need to evaluate the system for safety and grant permission for its use. Assurance cases have been proposed as means to organize the evidence into a coherent argument that captures what evidence is available, what assumptions have been made in the design process, how each piece of evidence contributes to the overall assurance, etc.

There is no clear understanding yet, however, how to build an assurance case for a given resilience property, combining together arguments performed at different levels of abstraction and using different reasoning techniques. In this paper, we consider a case study of one component in a resilient control system, namely a resilient speed estimator (RSE) for an autonomous ground vehicle. We construct a detailed assurance case for the component that covers both a mathematical model of the state estimator and its physical environment, as well as a software implementation of the state estimation algorithm. The purpose of the case study is to gain understanding of what levels of modeling are involved in the design and implementation of a control system, what reasoning techniques are used at each level, and what assumptions are likely to be made at each level, as well as how these assumptions can be justified by guarantees established in a lower-level model. While the models considered in the case study are specific to the control system and its intended deployment platform, we believe that the modeling levels and assumptions encountered on each level in this case study are typical of many other CPS control problems.

The paper is organized as follows. Section II introduces the concept of assurance cases and discusses the main strategy employed in the development of our assurance case. Section III describes the robotic platform and the problem of resilient state estimation. Section IV presents the assurance case for the RSE. We conclude with a discussion of our approach to the assurance case construction, and the role of the assurance case in the context of the whole vehicle.

Figure 1. Argument node types

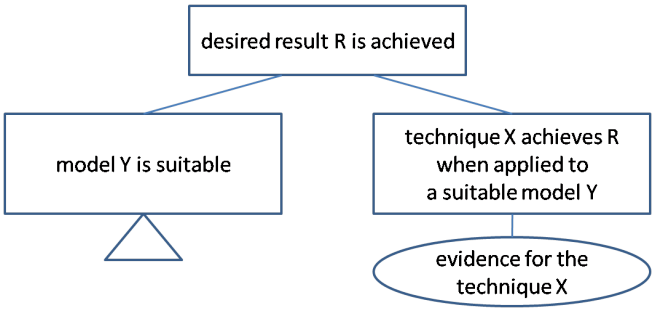

Figure 2. Model-manipulation strategy

II. Assurance cases

In a straightforward generalization from [1], we define an assurance case as *a documented body of evidence that provides a convincing and valid argument that a system has desired critical properties for a given application in a given environment*. A common example of such a critical property is system safety, in which case the argument is known as a safety case.

A commonly used notation for expressing assurance cases is Goal-Structuring Notation (GSN) [2]. In GSN, the argument is represented graphically. A *goal* node states the claim in an argument, a *strategy* node decomposes the further argument into sub-claims. Alternatively, an *evidence* node can refer to a direct support for the claim. There are also special nodes to express *assumptions* and *context* for the argument. In this paper, we use a similar notation. However, in our case study, all claims (except where noted) are using the same strategy, which is described below. To avoid duplication and simplify visual representation of the argument, we therefore do not use strategy nodes and connect the claim nodes directly to their sub-claims. Where needed, the strategy is described in the text. Nodes used in argument fragments in this paper are summarized in Figure 1.

Model-manipulation strategy: Throughout the argument, we rely on what we call a model-manipulation strategy. The structure of the argument is visually illustrated in Figure 2. This strategy is related to the *from-to* assurance case pattern, described in [3], that targets generative model-based development methods and is a simplified, one-step application of the same idea. In our case, we make a claim about the application of an analysis algorithm or some other transformation to a given model. This application may be subject to additional assumptions. For example, we apply discretization to a continuous-time model. In order to claim that the discrete-time model accurately describes the real system, we need to show that discretization is correctly performed, under the assumption about sampling rates of sensors on the platform, and that the continuous-time model was accurate, in the first place. We thus obtain two sub-claims. The first one, which we refer to as the technique sub-claim, is about the application of the technique. It does not need further argument and appeals to the evidence about the technique, such as proofs of the algorithm or tool qualifications. The second sub-claim, called the model sub-claim, is about the model itself. We may have to further extend the argument that the claim about the model is justified. The model-manipulation strategy can be applied iteratively, where the model sub-claim is again argued using the same strategy. In Section IV, we will present an assurance case constructed in this way.

Figure 3. LandShark vehicle

III. Case study

A. Problem definition and design approach

We study the construction of assurance cases in the context of a *resilient cruise control system* of LandShark [1], a fully electric Unmanned Ground Vehicle (UGV) shown in Figure 3. In our scenario, the operator specifies the desired vehicle speed, while the on-board control has to ensure that the desired speed is maintained, even in the presence of malicious activity aimed to disrupt the operation of the vehicle. A crucial part of the control system is a *state estimator*, which receives inputs from sensors and fuses multiple streams to derive an estimate for the system state. In our case study, the only state variable is vehicle speed. Speed readings can be obtained from wheel odometry and GPS sensors.

In this work, we consider attacks on sensor data, which result in wrong values being delivered to the state estimator. These attacks may be external to the vehicle, resulting from sensor spoofing, or internal, when the attacker can manipulate messages on the vehicle bus.

[1]See http://www.blackirobotics.com/LandShark_UGV_UC0M.html.

B. Resilient speed estimation

The estimation of the vehicle speed is performed following the technique presented in [4], which follows closely the seminal work in [5] and [6], where recent results on error correction over the reals and compressed sensing are used to derive secure state estimators when system sensors or actuators are under attack [7].

In [4] it is shown that the state (speed) can be estimated using N sensors measured at M time steps as the solution to the following minimization problem

$$\begin{aligned} \arg\min_{\mathbf{E},\mathbf{x}} \ & \|\mathbf{E}\|_{l_0} \\ s.t. \ & |\mathbf{Y} - \phi(\mathbf{x},\mathbf{U}) - \mathbf{E}| \leq \mathbf{\Delta} \end{aligned} \tag{1}$$

where $\mathbf{x}$ is the state to be estimated, $\mathbf{U}$ are the applied actuator inputs, $\mathbf{Y} \in \Re^{N\times M}$ represents a matrix of measurements, ϕ maps $\mathbf{x}$ and $\mathbf{U}$ onto $\Re^{M\times N}$, $\mathbf{\Delta} \in \Re_{+}^{N\times M}$ has elements corresponding to the worst-case sensor uncertainty bounds, and $\mathbf{E} \in \Re^{N\times M}$ denotes a matrix with non-zero entries corresponding to the estimated sensor attack values. Each row of $\mathbf{Y}$ (and likewise $\mathbf{E}$) corresponds to one of the N sensors, such that the objective in (1) is equivalent to minimizing the number of rows of $\mathbf{E}$ with a non-zero entry (i.e. minimize the number of sensors which are estimated to be attacked). It is assumed in [4] that the (possibly altered) sensor measurements $\mathbf{Y}$, applied actuation $\mathbf{U}$, mapping ϕ, and worst-case uncertainty $\mathbf{\Delta}$ are provided.

C. Implementation strategy

We employ a model-centric approach to develop and implement the state estimator. In the case study, the LandShark vehicle is running the ROS middleware [8]. In ROS, a control system is built as a collection of periodic or aperiodic nodes that communicate via a publish/subscribe mechanism. We use the tool ROSLab [9] to describe the architecture of the control system. We model the resilient state estimator as a single periodic node that publishes speed estimates and subscribes to individually published sensor streams. The node invokes the platform-independent step function that solves the optimization problem in (1). The solver is generated using the CVXGEN tool [10]. ROSLab generates a ROS wrapper for the step function that introduces subscribers and publishers according to the model of the state estimator node, and invokes the state estimator periodically at the rate specified in the model.

IV. Assurance Case for the LansShark Resilient Speed Estimator

A. Overall assurance case structure

The top-level claims of the assurance case are shown in Figure 4. The argument is partitioned into two parts. One part is concerned with the *algorithmic* correctness of the state estimator. We refer to this part of the assurance case as

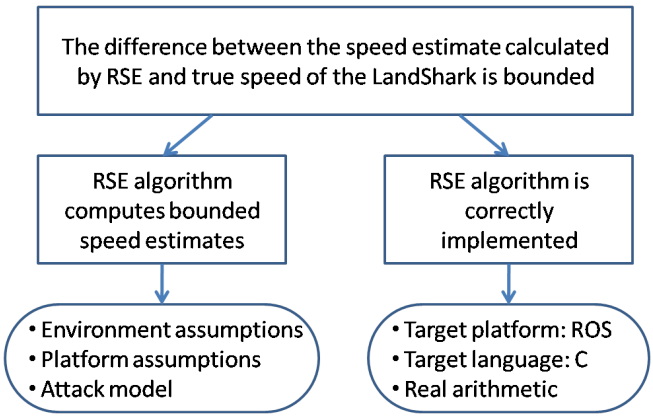

Figure 4. Top level claims of the assurance case

the control-level argument, since it deals with mathematical models of the estimator and relies on control-theoretic reasoning about these models. The other part addresses the implementation of the state estimator algorithm and the way it is deployed on the LandShark platform.

The argument also specifies assumptions and the implementation context. We rely on three categories of assumptions. *Attack assumptions* represent our model of the attacker capabilities. We consider attacks on sensor data and do not restrict the attacker's capability to manipulate a stream of sensor data. However, we assume that less than half of the redundant sensors are attacked. We have three sensors that provide speed data and thus assume that no more than one is attacked at any time. Given that the LandShark platform has three speed sensors, we assume that at most one sensor can be compromised at any time. There is no direct way to prove that this assumption holds, since it describes the limitation on the capability of the attacker. Indirect justification for the attack model can be derived from the implementation of the control system. In particular, sensors are implemented as different ROS nodes and publish their readings on separate ROS topics, making it more difficult for an attacker to compromise multiple sensor streams. *Environmental assumptions* describe the intended operating environment of the vehicle. These assumptions are used in evaluating the accuracy of the model of LandShark dynamics. We generally assume that the robot is operated on dry, almost level surface and is driving in an almost straight line. These assumptions can be validated in deployment, when deciding whether the robot is fit for a given mission. Finally, *platform assumptions* and the implementation context deal with the properties of the LandShark platform. Here, we assume a certain sampling frequency, expected latency of sensing and actuation, maximum actuation jitter (that is, deviations from periodic application of the control output to actuators), etc. These assumptions need to be validated on the platform and, if an assurance case for the

whole vehicle is constructed, should correspond to claims made in other parts of the assurance case.

B. Control-level arguments

The structure of the control-level argument is shown in Figure 5.

Main control-level claim: The first control-level claim, immediately derived from the top-level claim of the assurance case, is that the resilient state estimation algorithm achieves bounded state estimation. The algorithm operates on a relation between measurements, inputs, and the state of the system. The algorithm requires that less than half of the sensors are compromised, thus our attack assumptions match the expectations of the algorithm. The evidence for algorithm correctness is the proof published in [4]. The proof is constructed under the condition that the system model has bounded parametric uncertainty. The remainder of the argument concentrates on the system model, targeting the uncertainty of the model and its accuracy with respect to the real LandShark vehicle. For this, we move to the next claim in the argument.

Claims about optimization constraint: We claim that (a) the mapping ϕ and uncertainty $\mathbf{\Delta}$ in (1) describes the sampled dynamics of the LandShark platform with acceptable accuracy and (b) the uncertainty $\mathbf{\Delta}$ is bounded. For sub-claim (a), we demonstrate how the mapping ϕ and $\mathbf{\Delta}$ can be derived from parameters of a discrete-time model of the LandShark. The evidence used in this step is the analysis provided in [4], which explicitly states the linear mappings and transformations required to generate ϕ and the elements of $\mathbf{\Delta}$ from the discrete-time model,

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + \mathbf{v}_k. \end{aligned} \tag{2}$$

where $\mathbf{y}_k$ and $\mathbf{u}_k$ denote the sensor measurements and actuator inputs at time step k, $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are the state gain, input gain, and measurement gain, respectively, and $\mathbf{w}_k$ and $\mathbf{v}_k$ are the process and measurement uncertainty. Furthermore, for sub-claim (b) we demonstrate that when $\mathbf{w}_k$ and $\mathbf{v}_k$ are bounded, then $\mathbf{\Delta}$ is also bounded. The evidence used in this sub-claim is also provided in the analysis in [4].

Claims about discrete-time model: Here, we claim that the discrete-time model in (2) describes the dynamics of the LandShark platform with acceptable accuracy. For the claim we demonstrate how the parameters of the model, $\mathbf{A}$ and $\mathbf{B}$, as well as process and discretization disturbance terms $\mathbf{w}_k$ and $\mathbf{v}_k$ are derived from parameters of a continuous-time model of the LandShark. The evidence used in this step is the analysis provided in [4], which follows closely the mathematics in [11] and [12]. This analysis states that, with proper initialization, the outputs of the discrete-time model in each step, are identical to the sampled outputs of the following continuous-time model

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}_c\mathbf{x}(t) + \mathbf{B}_c\mathbf{u}(t) + \mathbf{w}_c(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{v}_c(t) \end{aligned} \tag{3}$$

In the remainder of this document, we will refer to the model in (3) as the reduced-order continuous-time model.

Claims about the reduced-order continuous-time model: We claim that the reduced-order continuous-time model in (3) represents the dynamics of the LandShark platform with sufficient accuracy; however, the model is an approximate representation of the LandShark dynamics and its parameters cannot be directly tied to the parameters of the vehicle. Employing accurate reduced-order models provides two benefits in resilient estimation: elimination of non-observable model modes, and reduction in run-time computational requirements. The argument, therefore, proceeds by establishing an approximate bisimulation between the reduced-order model in (3) and a full-order continuous-time model, denoted as

$$\begin{aligned} \dot{\tilde{\mathbf{x}}}(t) &= \tilde{\mathbf{A}}_c\tilde{\mathbf{x}}(t) + \tilde{\mathbf{B}}_c\mathbf{u}(t) + \tilde{\mathbf{w}}(t) \\ \mathbf{y}(t) &= \tilde{\mathbf{C}}\tilde{\mathbf{x}}(t) + \tilde{\mathbf{v}}(t) \end{aligned} \tag{4}$$

The theory of approximate bisimulation of linear systems is described in [13] and is supported by the tool Matisse.[2]

Claims about the full-order continuous-time model: The full-order model in (4) is derived from first principles and reflects the actual design of the LandShark platform. We obtain the model by noting that the movement of a skid-steering vehicle, such as the LandShark, can be modeled using first-principle physics following the mode-switching dynamics as described in [15], such that the elements of the full-order continuous-time state vector, $\tilde{\mathbf{x}}$, are written as

$$\tilde{\mathbf{x}} = \begin{bmatrix} l & \theta_L & \theta_R & v & F_L & \omega_L & i_L & F_R & \omega_R & i_R \end{bmatrix}^\top \tag{5}$$

where l and v denote the LandShark linear position and speed, F_L and F_R are the left and right tractive forces, and $\theta_L, \omega_L, \imath_L$ ($\theta_R, \omega_R, \imath_R$) are the left (right) DC motor position, angular velocity, and current, respectively. Since there are multiple state-space representations for the LandShark platform, our selection of a first principles representation is based on ensuring that the parameters in the governing differential equations are either known (i.e. constants or provided via datasheets), or can be accurately estimated over the entire operating range. When all the first principle models are derived from known parameters and accurately estimated parameters, we claim that the resulting full-order continuous-time model is acceptably accurate.

The requirement that the unknown parameters for each first-principles differential equation be estimated over the entire operating range is crucial in validating the claim

[2]In the case study, bisimulation analysis has not been performed. Instead, we constructed the reduced-order model by identifying the dominant eigenvalues of (4) [14].

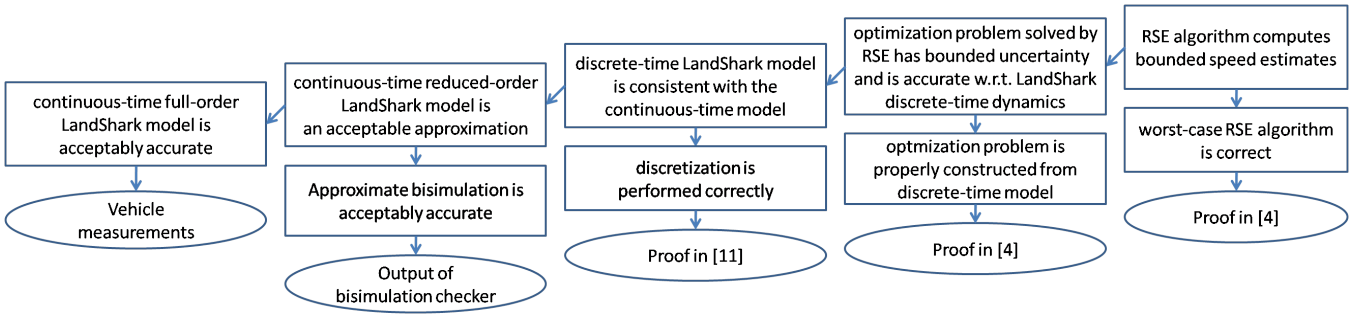

Figure 5. Control-level reasoning

that the model is acceptably accurate. As an example, we consider the first-principles differential equation for the left-side LandShark DC motor[3], namely

$$\dot{\omega}_L = \frac{1}{J}\left(\alpha i_L - \omega_L B_L - \frac{r}{g_r}F_L + \epsilon_L\right) \tag{6}$$

where α, J, r, g_r are the current-to-torque ratio, angular moment of inertia, the tire radius, and the drivetrain gear ratio and all are available via the LandShark datasheets[4]; however, the drivetrain rotational resistance, B_L, and model noise ϵ_L are not provided through the datasheet and must be accurately estimated. Estimation of B_L and ϵ_L is achieved in a laboratory setting by lifting the LandShark off its wheels (such that $F_L = 0$), and applying the entire operating range of current, i_L, to the motor, and measuring, for each current setting, the angular velocity, ω_L, using a tachometer at steady state (such that $\dot{\omega}_L = 0$). The resulting equation relating the current, i_L, steady-state angular velocity, ω_L, rotational resistance, B_L, and model noise, ϵ_L, is written as

$$\omega_L = \frac{\alpha}{B_L} i_L + \frac{1}{B_L}\epsilon_L \tag{7}$$

Observing that (7) is a linear equation, we can accurately identify unknown parameters B_L and ϵ_L from the applied current and measured rotational velocity, by choosing B_L such that the slope is a best fit, and ϵ_L as the worst-case error bias.

C. Implementation-level arguments

In addition to claiming that the RSE algorithm achieves the desired goal, we also need to argue that the algorithm is correctly implemented and deployed on the LandShark platform. This part of the argument is given in Figure 6. The strategy is to separate the argument into two sub-claims. The first one covers the platform-independent implementation of the RSE algorithm, implemented as a *step function* periodically invoked by the platform. The second

[3]A similar differential equation governs the right-side LandShark DC motor, just with (potentially) different parameter values.

[4]See http://www.blackirobotics.com/LandShark_UGV_UC0M.html and http://www.thunderstruck-ev.com/Manuals/PMG132curve.pdf.

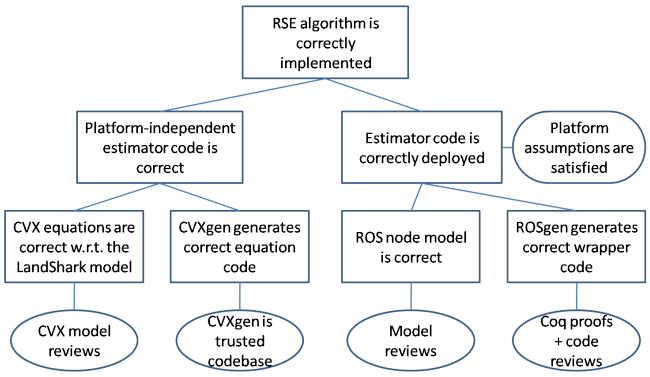

Figure 6. Argument for the code-level claims

sub-claim considers the deployment of the step function within a platform-specific wrapper, which handles periodic invocation of the step function, its connection to the streams of sensor data, and makes speed estimates available to other modules in the system. Arguments for both sub-claims are instances of the model-manipulation strategy. The step function is obtained using the CVXGEN tool, which generates embedded solvers for optimization problems. From our perspective, CVXGEN is trusted code base, and we use its widespread use as evidence. For the model sub-claim, we show that the model used by CVXGEN represents the optimization problem in (1), which can be determined by model reviews. The wrapper for the step function is produced from the architectural model of the LandShark platform, which captures ROS topics and their respective publishers and subscribers. The wrapper generator has been implemented in Coq and supplies a proof that (a) the wrapper subscribes to the sensor topics as specified in the architectural model, and that subscribed values are passed to the parameters of the step function, and also that (b) the step function is invoked with the period specified in the architectural model. We use this proof as evidence for the technique sub-claim, and perform review of the architectural model as evidence for the model sub-claim.

V. Discussion and conclusions

We have considered an approach to construct an assurance case for a specific property of the resilient state estimation module in a control system of an autonomous vehicle. The assurance case is intended to be used as a part of a larger assurance case for the whole vehicle. This overall assurance case is the subject of an on-going multi-institutional project funded by the DARPA HACMS program. Some of the platform assumptions made in our argument will eventually be claims delivered by other parts of the overall assurance case.

Our approach to the construction of the assurance case is motivated by the understanding that the outcome of model-based development of a system is only as good as the model used in the process. Therefore, in each step of the argument we argue that not only we apply sound model analysis and correct model transformations, but also that models we operate on are adequate representations of the reality. To this end, we established a chain of reasoning from the first-principles model of the vehicle, which is directly tied to the measurements on the device, all the way to the model used in the state estimation algorithm, demonstrating that the accuracy of the model is preserved in each transformation step. In practice, some of these steps, along with the associated argument, are left implicit. For example, it may be possible to start with a discrete-time model of the vehicle, which would be obtained by system identification. In this case, we rely on the expertise of control engineers and common practices of control design to ensure the accuracy of the model. However, we believe that first-principles analysis and reasoning is adding confidence in the argument and makes evaluation of the argument easier.

Acknowledgements

Research has been supported in part by DARPA under agreement number FA8750-12-2-0247. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

References

[1] *ASCAD – The Adelard Safety Case Development (ASCAD) Manual*, Adelard, 1998.

[2] T. Kelly and R. Weaver, "The goal structuring notation - a safety argument notation," in *Proceedings of Workshop on Assurance Cases (Satellite workshop of Dependable Systems and Networks)*, 2004.

[3] A. Ayoub, B. Kim, I. Lee, and O. Sokolsky, "A safety case pattern for model-based development approach," in *Proceedings of the 4^{th} NASA Formal Methods Symposium*, Apr. 2012, pp. 223–243.

[4] M. Pajic, J. Weimer, N. Bezzo, P. Tabuada, O. Sokolsky, I. Lee, and G. Pappas, "Robustness of attack-resilient state estimators," in *Proceedings of the 5^{th} ACM/IEEE International Congerence on Cyber-Physical Systems (ICCPS)*, Apr. 2014, pp. 163–174.

[5] H. Fawzi, P. Tabuada, and S. Diggavi, "Secure state-estimation for dynamical systems under active adversaries," in *Proceedings of the 2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2011, pp. 337–344.

[6] ——, "Security for control systems under sensor and actuator attacks," in *Proceedings of the 51st IEEE Conference on Decision and Control*, 2012.

[7] R. Smith, "A decoupled feedback structure for covertly appropriating networked control systems," *Proc. IFAC World Congress*, pp. 90–95, 2011.

[8] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *Proceedings of the Open-Source Software workshop at the International Conference on Robotics and Automation (ICRA)*, 2009.

[9] N. Bezzo, J. Park, A. King, P. Geghard, R. Ivanov, and I. Lee, "Demo abstract: Roslab a modular programming environment for robotic applications," in *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2014, p. 214.

[10] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, 2012.

[11] J. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 138–162, Jan 2007.

[12] W. Zhang, M. Branicky, and S. Phillips, "Stability of networked control systems," *Control Systems, IEEE*, vol. 21, no. 1, pp. 84–99, Feb 2001.

[13] A. Girard and G. J. Pappas, "Approximate bisimulation: A bridge between computer science and control theory," *European Journal of Control*, vol. 17, no. 5-6, pp. 568–578, 2011.

[14] L. Fortuna, G. Nunnari, and A. Gallo, *Model order reduction techniques with applications in electrical engineering*. Springer-Verlag London, 1992, vol. 257.

[15] J. Nutaro, *Building Software for Simulation: Theory and Algorithms, with Applications in C++*, ser. Wiley Online Library: Books. Wiley, 2011. [Online]. Available: http://books.google.com/books?id=tEd0wC0QfCUC