Assurance of Model-Based Autonomy for Robotic Space Missions

Martin S. Feather, PhD, Jet Propulsion Laboratory, California Institute of Technology
Steven L. Cornford, PhD, Jet Propulsion Laboratory, California Institute of Technology
Klaus Havelund, PhD, Jet Propulsion Laboratory, California Institute of Technology

Key Words: assurance, autonomy, models, space missions

SUMMARY & CONCLUSIONS

NASA's robotic space missions must make use of on-board automation and autonomy to control themselves when communication with Earth is slow (due to light-time delays across the solar system) or unavailable. As mission concepts become more ambitious, increasingly capable forms of autonomy are required - ones that go well beyond executing a single preformulated script. Assurance of autonomy - the means by which would-be mission proposers, reviewers and managers will be provided the confidence to trust their expensive assets to autonomous control - is a challenging necessity.

In this paper we look at assurance of some more capable forms of autonomy that use models - models of the environment in which system is to operate, and models of the asset (spacecraft, rover, helicopter, etc.) itself. The latter models must cover the different factors crucial to the asset's health and operation. For example, managing electrical power - its source, storage and consumption - is needed, and models are used to predict the power balance over time. Likewise important is managing the thermal conditions - temperature of the asset, its sensitive instruments, its mechanisms, etc. These models are used by the on-board controller to plan and schedule the asset's activities and to monitor the health of the hardware to take remedial action if need be when things go wrong (due to temporary faults, or permanent degradations and failures).

However, the computational resources on-board are typically but a fraction of those available on Earth. As a result, the on-board models are deliberate simplifications of those created for ground operations, simplified for tractable execution within those limited computational resources. Several key assurance challenges that stem from this are the following:

- What process should we follow to derive simplified models so that the simplifications won't mislead the asset into performing a dangerous activity?
- How will we know the combination of simplified models remains sufficiently accurate (e.g., the power model's simplifications when coupled with the thermal model's simplifications don't compound one another)?
- What can we have the asset do to detect if things are straying too far from model predictions in time to do something about it?

This paper describes approaches to answering these questions, through a combination and adaptation of techniques for analysis, testing and monitoring drawn from other (non-space) domains.

1 INTRODUCTION

On-board autonomy is recognized across NASA as a key enabler for future generations of robotic missions. This point has been cogently made in, for example, [1] and [2]. Such autonomy goes beyond straightforward automation. As described in [3], "*Automation … is the automatically-controlled operation of an apparatus, process, or system using a pre-planned set of instructions (e.g., a command sequence)*." whereas "*Autonomy is the capacity of a system to achieve goals while operating independently from external control*."

A wide range of future missions and programs would be beneficiaries of more autonomy. These include those with short dynamic timescales (e.g., a short-lived Mercury Lander), ambitious operations (e.g., an advanced rover's goal to traverse 1,800 km on the moon without continuous direction from Earth), opportunistic science (e.g., gathering data on transient phenomena), activities requiring rapid reaction to potentially dangerous unpredictable conditions (e.g., proximity operations near oddly-shaped asteroids), and in locations where communication is slow (e.g., explorations of objects beyond the inner planets) or unavailable (e.g., during solar conjunctions, or in subterranean settings).

1.1 *Autonomy's need for models*

In almost all these cases, the autonomy involved would require knowledge of the capabilities and needs of the asset it controls. Typically, such information is provided via behavioral models of the spacecraft and its subsystems, which are then used by on-board planners to schedule activities, by controllers to monitor the performance of those activities, and by fault management capabilities to recognize and respond to anomalies. For a description of command execution systems, see for example [4], and for a survey of model-based approaches, [5]. Fundamental models that would be needed by on-board autonomy in space assets include power models (electrical power is almost always a scarce resource), and thermal models (temperatures have to be carefully managed in the extreme conditions of space).

1.2 *Computational capabilities to execute models*

Models are used extensively during the *development* of space missions to evaluate and assess design alternatives, and to plan how missions will be operated. Models are also used by ground control as missions take place, for planning the commands to send to the assets. Because these models are being run on computers on Earth, they can take advantage of powerful computational resources to execute. However, when shifting to autonomy *on-board the assets themselves*, the available computational resources are much more limited in terms of both speed and memory. This is because space assets are constrained in the power and volume they can provide to processors, and furthermore, processor designs tolerant of the radiation environments in space are computationally inferior to their terrestrial counterparts, typically a decade behind the state of the art.

For example, NASA's Perseverance rover on Mars uses a BAE Systems RAD750 radiation-hardened single board computer, which has a couple of orders of magnitude less computing power than the Snapdragon 801 on the accompanying Ingenuity helicopter. This rover must fulfill the primary science mission, so extremely high assurance of the successful and continued operation of its computer was demanded, whereas the helicopter's mission is experimental in nature [6] and so open to use of more cutting-edge technology.

Although there are efforts underway to advance the capabilities of highly reliable spaceflight computing (e.g., [7]), such capabilities will continue to lag behind ground-based resources, especially when compared to supercomputing facilities.

1.3 *Simplified models*

As a result, it is infeasible to execute on-board the complex and detailed models as would be used on the ground. Instead, the on-board autonomy must make do with deliberately simplified models. The need for such model simplification is widespread, not just for execution on space assets [8]. Such simplified models may also be referred to as "reduced order models" [9] or "low fidelity models".

1.4 *Concerns with use of simplified models*

Adoption of autonomy by flight projects hinges on their willingness to have trust in its correct operation – i.e., confidence that it will not harm the space asset(s) it controls, and that it will support the accomplishment of mission objectives. Assurance of spacecraft autonomy has long been a recognized as a concern, e.g., as expressed almost two decades ago in [10].

If autonomy is to manage fundamental space asset domains such as power and temperature, and yet can only afford to execute simplified models of these factors, this raises the obvious question of how to trust the result. Might the simplification of a thermal model, say, mislead the autonomy into directing the asset into a dangerous state of overheating? On-board autonomy does have the advantage of access to actual conditions, as provided by the asset's sensors. In contrast, when mission control plans future operations, it must extrapolate what the conditions will be at the time of those operations. Despite this advantage, there is still the concern that autonomy, misled by a simplified model, might set the asset on an irreversible course of action that will lead to failure. This concern is exacerbated when it is not just one model that is simplified, but a set of interconnected models. The thermal model must take into account the heat generated by power-consuming devices. But the amount of power that will be needed by a device will itself be determined from a power model – again, a *simplified* model.

1.5 *Uncertainty, Unpredictability and Model adaptation*

Further complicating the situations of many space missions are the inevitable factors of uncertainty and unpredictability. The environments in which space missions take place are often fraught with uncertainty. For example, the gravity field around a previously unvisited small body (e.g., an asteroid) will be uncertain in advance. It is important to approach with care to avoid the risk of unintended impact. Operations on the surface of a body will often face uncertainty in properties of that surface, as seen when OSIRIX-REx's sample acquisition from the surface of asteroid Bennu gathered a surprising excess of material, including pebbles, propping open the flap that was intended to prevent escaping material [11]. The characteristics of the space asset itself can be uncertain. For example, spacecraft instrument or equipment performance uncertainty is typically resolved by calibration before use, e.g., calibration of thrusters [12]. Over time, and due to the harsh conditions of space, equipment can degrade, and can develop failures that affect its performance. These may be expected, especially for long-duration missions, but exactly when they will occur is not predictable.

For all these cases, models of the asset and its environment may have some degree of uncertainty, and may need to be updated to reflect changed understanding and changed situations. If there is not time to relay information to ground control to have them determine the model updates, then the autonomy must update its own models. This of course raises the concern of how to trust the updated models to not set the asset on an irreversible course of action that will lead to failure.

1.6 *Addressing the concerns*

The following four approaches combine as means to develop trust in autonomy's use of on-board models:

1. Codify best-practice guidance to developing simplified models from their sophisticated ground-based models.
2. Develop an analysis technique with which to assess the safety of simplified models when they are coupled together.
3. Develop "point-of-no-return" monitors to recognize when to interrupt the ongoing execution to avoid leading to an unsafe state.
4. Show that the mechanism for autonomously updating models will operate correctly and retain safety.

The sections that follow expand upon each of these.

2 BEST-PRACTICE GUIDANCE FOR DEVELOPMENT OF SIMPLIFIED MODELS

As illustration, consider a model of a spacecraft's thermal system. This model will include sources of heat, e.g., sunlight impinging on external surfaces; thermal inertia of the components within the asset; conduction of heat between components abutting one another; and losses of heat, e.g., heat radiating into space on the shadowed side of the spacecraft.

A simplification of this model might abstract from a pair of components and the thermal conductivity between them (across a shared interface), to treat them as a single, homogeneous block. This would mean that nuances of gradual heat transfer across an interface from one of them to the other would be lost in the simplified model. Assurance is needed that the spacecraft's autonomy will keep the spacecraft safe when it relies upon the thermal predictions from this simplified model. Work germane to this area is found in [13], which presents a method for quantifying uncertainty in conceptual-level design via a computationally efficient probabilistic method, showing an application to estimate the maximum-expected temperature of several critical components on a spacecraft.

2.1 *Models and control systems*

A useful way of viewing this is from the perspective of the interplay between the control system and the models it uses to keep state estimates. This is the style of thinking espoused in Leveson's "Systems-Theoretic Accident Model and Processes" (STAMP) [14]. Part of this approach hinges on recognizing that a controller must contain a model of the process being controlled. It promotes consideration of how inaccuracies in the model could potentially mislead the controller and whether the results could be dangerous, and if so leads to designing protections against those situations.

In the case of autonomy using deliberately simplified models, the nature of the simplifications inform the reasoning of how the controller could possibly be misled, and therefore how to protect against that. Consider the example of a thermal model having simplified a pair of connected components into a single homogeneous block. The heat flow across that block predicted by the simplified model may be faster than that in reality if, for example, there was thermal insulation at the interface between those two components. The question to ask is whether this misprediction could pose a danger. Perhaps the source of the incoming heat would not cool as quickly as it was predicted, or perhaps the first of those two components would experience a dangerous increase in heat. If recognized as possible dangers, these might be addressed by having the controller assume more pessimistic margins on allowable temperatures.

2.2 *NASA's Modeling and Simulation Standard*

This combination of estimating the simplification-induced inaccuracies together with the criticality of those inaccuracies is akin to the thinking behind NASA's Modeling and Simulation (M&S) standard [15], which states "*This NASA Technical Standard establishes a minimum set of requirements and recommendations for M&S influencing or supporting decisions, particularly critical decisions."* One of the relevant requirements imposed by this standard is shown here:

> [M&S 6] Shall perform and document the criticality assessment for the M&S.
> *[Rationale: The criticality assessment ensures communication of the amount of influence an M&S has on a particular situation relative to the consequences of that situation.]*

The standard defines key factors with which to assess the *credibility* of the results produced by models and simulations. These factors are grouped into *Development:* Data Pedigree, Verification, Validation; *Operations:* Input Pedigree, Uncertainty Characterization, Results Robustness; and *Supporting Evidence:* M&S History, and M&S Process/Product Management. The standard's factors are intended for when a model or simulation is first constructed. To instead assess the credibility of a model formed as a simplification of an already constructed model, some reinterpretation of the key factors is warranted. The following lists several examples of such, giving the original phrasing, prefaced by (Standard), and the revised phrasing, prefaced by (Reinterpretation):

- Development
  - Verification
    - (Standard) Were the models implemented correctly, per their requirements/specifications?
    - (Reinterpretation) Were the simplifications implemented correctly?
  - Validation
    - (Standard) Did the M&S results compare favorably to the referent data, and how close is the referent to the RWS (Real World System)?
    - (Reinterpretation) Do the simplified M&S results stay sufficiently close to the referent data when used for control purposes?
- Operations
  - Uncertainty Characterization
    - (Standard) Is the uncertainty in the current M&S results appropriately characterized? What are the sources of uncertainty in the results and how are they propagated through to the results of the analysis?
    - (Reinterpretation) Is the increased uncertainty in the simplified model appropriately characterized? What simplifications are the sources of increased uncertainty in the results and how are they propagated through to the control system's use of the simplified model?
- Supporting Evidence
  - M&S Process/Product Management

- (Standard) How well managed were the M&S processes and products?
- (Reinterpretation) How well managed was the process of deriving the simplified model(s)?

The standard goes on to explain each of its key factors in more detail, and indicates how they are to be assessed. Just as the factors themselves require some reinterpretation to be applied to model simplification, so do these details.

3 COUPLED MODELS

As illustration of coupled models, consider combining a simplified model of a spacecraft's thermal system with a simplified model of its power system. Thermal and power can interact. When using a radar, say, it consumes power and generates heat. So when planning to use the radar to take science observations, it may be necessary to consider whether doing so would cause an excessive build-up of heat. Even more coupling between thermal and power occurs in spacecraft designs that do not rely solely on passive distribution of heat, but use active – and hence powered – "heat redistribution systems" [16].

To address the assurance aspects of coupled models, the perspective of Leveson's STAMP approach is again relevant. Much of the rationale behind application of STAMP is based on the observation that in today's complex systems, mishaps may occur from untoward interactions among otherwise correctly operating components. A key step in the STAMP approach (for details, see [17]) is to examine control actions between components, and for each, systematically consider the ways in which a control action might be unsafe. For model simplification, instead of focusing on control actions between components, focus on data exchange between the coupled simplified models. For each data exchange, consider how one model's inaccuracy, when propagated across a data exchange to another model, affects the inaccuracies of that second model. It is particularly important to look for feedback loops and determine whether or not they are reinforcing (compounding the uncertainty). Early work in this direction is to be found in [18].

4 POINT-OF-NO-RETURN MONITORS

Current practice is to use fault protection monitors of current state to recognize a dangerous situation (e.g., overcurrent) and take immediate action to preserve health and safety (e.g., turn off the offending power draw). It is necessary to take this concept further, using runtime monitoring of predictive conditions to recognize in advance when action is necessary to avoid a future health and safety hazard.

4.1 *Fault protection in space systems*

In the space domain, fault protection monitors are standard practice, used as far back as the Voyager spacecraft [19], and play an important role in the more general area of fault management [20]. Fault protection monitors look for symptoms of faults and failures, triggering an emergency response when critical conditions are recognized. A general response to these is to go into "Safe Mode," which configures the asset to a state which is safe and predictable so that diagnosis of more complex faults can be addressed by the Operations Team back on Earth [21].

On NASA's Curiosity rover operating on Mars, fault protection would be triggered if, for example, the rover's autonomous driving led it onto unexpectedly steep terrain – the reaction in this case would be to stop the drive. In addition, there are monitors of the rover's executing software, checking "runtime assertions". These assertions express conditions that the programmer assumed would hold at that point in the execution. The violation of an assertion is an indication that something is amiss – often the first such indication. As recounted in [22]: "*One final departure from earlier practice was that on the MSL mission all assertions remained enabled in flight, whereas before they were disabled after testing. A failing assertion is now tied in with the fault-protection system and by default places the spacecraft into a predefined safe state where the cause of the failure can be diagnosed carefully before normal operation is resumed.*"

4.2 *Runtime monitoring*

Runtime monitoring is a general term for detecting conditions during execution. It has long been an active area of study: an annual workshop on "Runtime Verification" began in 2001, and became a conference in 2010 – see https://runtime-verification.github.io/index.html, and for comprehensive overviews, [23] and [24]. Its use in conditions of uncertainty is described in [25], stating "A software system can successfully operate in multiple dynamic contexts by using runtime models that augment information available at design-time with information monitored at runtime."

In the case of space assets whose models may both be deliberate simplifications, and include uncertainty in their knowledge, runtime monitoring is a way to recognize when conditions diverge from expectations. For example, the experience of operating rovers on Mars has shown that driving over highly sloped and sandy terrains can lead to wheel slippage. Had the rover's estimate of its position been limited to wheel odometry alone (how much the wheels have turned), its inaccuracy would have forced the mission planners to conservatively command short drives in such circumstances, checking each day where the rover had actually moved to. Fortunately in this case a technique called Visual Odometry allowed the rover to keep accurate track of its position during driving [26], eliminating most of the uncertainty, and thus allowing mission planners to command longer daily drives.

Furthermore, it may be necessary to not just recognize and react to dangerous conditions as and when then occur, but also to recognize when a course of action *could* lead to a dangerous condition from which recovery would be impossible. Thus monitoring must be of predictive conditions.

There are two challenges to this approach: derivation of crucial correctness conditions from understanding of the coupled reduced-order models, and casting those conditions as computations that are tractable for continuous evaluation. The monitored conditions must encompass both the model simplifications, and their explicit treatment of uncertainties

(e.g., placing bounds on the latter to serve as triggers). As long as these monitors are in place, it allows the autonomous short-to-medium term planning to be less conservative in its scheduling. Instead of being forced to be overly conservative (based on pessimistic assumptions about deviations from expectations), it can set the asset on course of action confident that monitoring will recognize if and when deviations are accumulating. As long as activity is halted before reaching a point of no return, safety will be preserved.

5 FUTURE WORK - AUTONOMOUSLY UPDATED MODELS

During operation, information about the space asset's environment, and about the asset's own performance, will become available. This information could be used to decrease uncertainty in its models, either by communicating the information back to mission control to have them decide how to update the asset's on-board models, or, ambitiously, by the space asset itself. Similar model updates would be needed if the asset's performance changes due to degradation or failure of its hardware components. This raises two concerns if it is to be done autonomously:

- Will the update be performed correctly?
- Will the control system's use the updated model(s) remain correct, i.e., never set the asset on an irreversible course of action that would lead to failure?

Addressing these concerns is an area for future work, in parallel with an effort looking into the approaches that will perform the autonomous updating. Together with this effort, we are in the process of identifying related work in other domains that may be transferrable to space asset autonomy. We mention the following:

Updating models as information is gained during operation is the topic of [27], where it is referred to as "calibration" and shows use of a Bayesian technique specifically suited to addressing initial and remaining uncertainty. Particularly relevant is the approach in [28], to allow for rapid model adaptation. The intended purpose is to allow an aircraft to dynamically replan a safe mission in response to structural damage or degradation. Its approach combines a library of component-based reduced-order models. In the software engineering domain, [29] considers the need to assure continued compliance with requirements as autonomous adaptation takes place, a process they term "perpetual assurances for self-adaptive systems".

An even more ambitious avenue of development would be space missions' use of machine learning derived algorithms (e.g., deep neural nets). The culture of space exploration, driven by the need to succeed on the first and only try, currently has strong aversion to these, in part because it is challenging to identify an attribution for the prediction/decision made by these algorithms. Nevertheless, how to develop trust in these kinds of algorithms is an area of active investigation [30].

A parallel effort to this work is one exploring test automation, to make feasible the large amounts of verification testing (mostly in simulation) needed to cover the multivariate nature of the environment (illumination, gravity, surfaces, etc.).

6 ACKNOWLEDGEMENTS

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004) and funded through the internal Research and Technology Development program.

REFERENCES

1. D. Miranda, "2020 NASA technology taxonomy," available from NTRS - NASA Technical Reports Server at: https://ntrs.nasa.gov/citations/20190032038, 2019.
2. R. Amini, A. Azari, S. Bhaskaran, P. Beauchamp, J. Castillo-Rogez, R. Castano, S. Chung, J. Day, R. Doyle, M. Feather, L. Fesq, "Advancing the Scientific Frontier with Increasingly Autonomous Systems," *arXiv preprint arXiv:2009.07363*.
3. NASA, "2015 NASA Technology Roadmaps," available from https://www.nasa.gov/sites/default/files/atoms/files/2015_nasa_technology_roadmaps_ta_0_introduction_crosscutting_index_final_0.pdf
4. V. Verma, A. Jónsson, R. Simmons, T. Estlin, R. Levinson, "Survey of command execution systems for NASA spacecraft and robots," *Plan Execution: A Reality Check," Workshop at The Int. Conf. on Automated Planning & Scheduling, ICAPS*, 2005, (pp 92-99.
5. M. Tipaldi, L. Glielmo, "A survey on model-based mission planning and execution for autonomous spacecraft," *IEEE Systems Journal*, *12*(4), 2017, pp 3893-3905.
6. "Mars Helicopter/Ingenuity" https://mars.nasa.gov/files/mars2020/MarsHelicopterIngenuity_FactSheet.pdf
7. W.A. Powell, "High-performance spaceflight computing (hpsc) project overview," *Radiation Hardened Electronics Technology Conference (RHET)*, 2018.
8. B. Peherstorfer, K. Willcox, M. Gunzburger, "Survey of multifidelity methods in uncertainty propagation, inference, and optimization," *Siam Review*, *60*(3), 2018, pp.550-591.
9. W.H. Schilders, H.A. Van der Vorst, J. Rommes, "Model order reduction: theory, research aspects and applications," (Vol. 13, p. 13). Berlin: springer, 2008.
10. J.P. Blanquart, S. Fleury, M. Hernek, C. Honvault, F. Ingrand, J.C. Poncet, D. Powell, N. Strady-Lécubin, P. Thévenod, "Software product assurance for autonomy on-board spacecraft," *DASIA 2003-Data Systems In Aerospace* (Vol. 532), 2003.
11. D.S. Lauretta, O.R.T. Team, "The OSIRIS-REx Touch-and-Go Sample Acquisition Event and Implications for the Nature of the Returned Sample," *Lunar and Planetary Science Conference* (No. 2548, p. 2097), March, 2021.
12. P.J. Wiktor, "On-orbit thruster calibration," *Journal of guidance, control, and dynamics*, *19*(4), pp.934-940, 1996.
13. D.P. Thunnissen, S.K. Au, G.T. Tsuyuki, "Uncertainty quantification in estimating critical spacecraft component temperatures," *Journal of thermophysics and heat transfer*, *21*(2), 2007, pp 422-430.

14. N.G. Leveson, “Engineering a safer world: Systems thinking applied to safety,” The MIT Press, 2016.
15. NASA, “STANDARD FOR MODELS AND SIMULATIONS” https://standards.nasa.gov/standard/nasa/nasa-std-7009
16. H. Ochoa, J. Hua, A.J. Mastropietro, R. Lee, A. Paris, N. Emis, B. Williams, “Design and Development of the Heat Redistribution System for the Europa Clipper Spacecraft,” *47th International Conference on Environmental Systems*, 2017.
17. N.G. Leveson. J.P. Thomas. “STPA Handbook” 2018, https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf
18. J.C., Doyle, J.E. Wall. G. Stein, “Performance and robustness analysis for structured uncertainty,” *21st IEEE conference on decision and control,* IEEE., 1982, pp 629-636.
19. R.L. Heacock, “The voyager spacecraft,” *Proceedings of the Institution of Mechanical Engineers*, *194*(1), 1980, pp 211-224.
20. NASA “Fault Management Handbook – Draft 2” NASA-HDBK-1002, available from https://www.nasa.gov/pdf/636372main_NASA-HDBK-1002_Draft.pdf
21. P.S. Morgan, “Fault Protection Techniques in JPL Spacecraft,” *First International Forum on Integrated System Health Engineering and Management in Aerospace*, NASA, 2005.
22. G.J. Holzmann, “Mars code” *Communications of the ACM*, 57(2), 2014. Pp 64-73.
23. M. Leucker, C. Schallhart, “A brief account of runtime verification,” *The Journal of Logic and Algebraic Programming*, 78(5), 2009, pp.293-303.
24. E. Bartocci, Y. Falcone, A. Francalanza, G. Reger, “Introduction to runtime verification,” *Lectures on Runtime Verification* (pp. 1-33). Springer, Cham. 2018.
25. H. Giese, N. Bencomo, L. Pasquale, A.J. Ramirez, P. Inverardi, S. Wätzoldt, S. Clarke, “Living with uncertainty in the age of runtime models,” *Models@ run. Time*, Springer, Cham. 2014, pp 47-100.
26. M. Maimone, Y. Cheng, L. Matthies, “Two years of visual odometry on the mars exploration rovers,” *Journal of Field Robotics*, 24(3), 2007, pp 169-186.
27. M.C. Kennedy, A. O'Hagan, “Bayesian calibration of computer models,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3), 2001, pp 425-464.
28. M.G. Kapteyn, D.J, Knezevic, D.B.P. Huynh, M. Tran, K.E. Willcox, “Data-driven physics-based digital twins via a library of component-based reduced-order models,” *International Journal for Numerical Methods in Engineering*. 2020
29. D. Weyns, N. Bencomo, R. Calinescu, J. Camara, C. Ghezzi, V. Grassi, L. Grunske, P. Inverardi, J.M. Jezequel, S. Malek, R. Mirandola, “Perpetual assurances for self-adaptive systems,” *Software Engineering for Self-Adaptive Systems III. Assurances*, Springer, Cham., 2017, pp 31-63.
30. L. Mandrake et al, “Space Applications of a Trusted AI Framework: Experiences and Lessons Learned,” (in submission).

BIOGRAPHIES

Martin S. Feather, PhD
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr., Pasadena, California 91109, USA

e-mail: martin.s.feather@jpl.nasa.gov

Martin Feather is a Principal Software Assurance Engineer in JPL’s Office of Safety and Mission Success, and has been with JPL for over 25 years. His activities are focused on identifying, developing and infusing research results to improve assurance of space missions, with a particular interest in software. He earned his PhD in Artificial Intelligence from the University of Edinburgh, UK. He has authored over 170 papers.

Steven L. Cornford, PhD
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA, 91109, USA

e-mail: steven.l.cornford@jpl.nasa.gov

Steven Cornford is a Principal Engineer in the System Modeling and Analysis Program Office at JPL. He has a double major in Mathematics and Physics from UC Berkeley, a MS in Quantum Gravity and a PhD in Experimental Atomic Physics from Texas A&M University. He has performed a variety of line management and project management functions. He has been a system engineer, reliability engineer, test engineer and a DARPA Principal Investigator. He is currently particularly interested in cross-cutting problems and efforts to increase the breadth of early-phase modeling and trade space exploration. He has authored over 100 papers.

Klaus Havelund, PhD
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA, 91109, USA

e-mail: klaus.havelund@jpl.nasa.gov

Klaus Havelund is a Principal and Senior Research Scientist in the Flight Software and Avionics Systems Section at JPL. He has a Masters and PhD in Computer Science from the University of Copenhagen. He has worked at JPL since 2006 and before that at NASA Ames Research Center since 1997. His research is focused on formal methods. He has in particular over the last decades contributed to the runtime verification field. He has authored 154 papers