2019

Autonomous Systems with Reuse: A Survey on the State-of-the-Practice

Denny LaVerne Hood III
WVU, dlh0038@mix.wvu.edu

Follow this and additional works at: <https://researchrepository.wvu.edu/etd>

Part of the [Artificial Intelligence and Robotics Commons](#)

Recommended Citation

Hood, Denny LaVerne III, "Autonomous Systems with Reuse: A Survey on the State-of-the-Practice" (2019). *Graduate Theses, Dissertations, and Problem Reports*. 7391.
<https://researchrepository.wvu.edu/etd/7391>

This Problem/Project Report is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Problem/Project Report in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/ or on the work itself. This Problem/Project Report has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.

2019

Autonomous Systems with Reuse: A Survey on the State-of-the-Practice

Denny LaVerne Hood III

Follow this and additional works at: <https://researchrepository.wvu.edu/etd>

Part of the [Artificial Intelligence and Robotics Commons](#)

AUTONOMOUS SYSTEMS WITH REUSE: A SURVEY ON THE STATE-OF-THE-PRACTICE

Denny Hood

Problem Report submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements for the degree of
Master of Science
in
Computer Science

Katerina Goseva-Popstojanova, Ph.D., Committee Chairperson
Roy Nutter, Ph.D.
Yanfang Ye, Ph.D.
Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2019

Keywords: autonomous systems, model-based software engineering, reuse, software
assurance, survey

Copyright © 2019 Denny Hood

ABSTRACT

Autonomous Systems with Reuse: A Survey on the State-of-the-practice

Denny Hood

This problem report presents the results of an anonymous online survey that was used to collect information about software systems that use model-based software engineering, contain autonomy and where software reuse plays an important role. Recent advancements in computation ability and the emergence of decision-making algorithms have increased interest in and use of autonomy in application areas such as aeronautics, automotive, military, and space industries. In these application domains, autonomy is used to reduce costs, reduce reaction times, and improve performance.

Due to the emerging nature of autonomy, very little research has been done regarding the level of autonomy of autonomous systems. In addition, developing software systems involving autonomy is particularly challenging if a mission's success or human life depend on it. Also, there is limited information about autonomous systems due to the proprietary nature of the code implementing autonomy.

The main goals of our survey were to: (1) assess the current state-of-the-practice using autonomous systems, (2) identify and quantify the benefits and challenges of autonomy and reuse, (3) explore the processes and standards used to develop autonomous systems, and (4) investigate verification and validation of the models, autonomy, and reuse.

The results of the survey are based on the answers provided by 110 respondents who used model-based software engineering and/or autonomy. The main findings, as they pertain to our goals, are as follows:

1. **State-of-the-practice.** Significant percentage of autonomous components used full autonomy. Most common types of reuse were code reuse, model reuse, and data reuse.
2. **Benefits and challenges.** The prevalent challenges associated with autonomy were due to high level of environment uncertainty, system complexity, and achieving desired level of autonomy. In projects with reuse, 63% of respondents reported increased productivity, 40% reported increased quality, and 37% observed decreased cost due to reuse.
3. **Processes and standards.** 48% of respondents did not use modeling standards for their project. Only 24% of the projects went through a certification process.
4. **Verification and validation.** Only 11% of the respondents verified and validated security attributes. During the development phase the autonomous components were verified and validated using simulation based testing (20%), unit and integration testing (19%), and manual model/code reviews or inspections (18%). Only 19% of respondents verified and validated reused artifacts. 28% of respondents found bugs due the autonomous functionality, 25% found bugs due to reused artifacts, and 9% due to the use model-based approach.

Acknowledgments

The author would like to thank Dr. Goseva-Popstojanova, Dr. Johann Schumann, and Mr. Noble Nwokcha who participated in the creation of the survey questionnaire and provided their input and feedback on the work presented in this Problem Report. In addition, the author would like to thank the colleagues who participated in the pilot survey and provided their feedback, as well as all respondents to the survey for sharing their information and opinions. This work was funded by the NASA Software Assurance Research Program (SARP) in the fiscal years 2019 and 2020.

Contents

Abstract	ii
Acknowledgments	iii
List of Figures	vi
List of Tables	viii
1 Introduction	1
2 Related Work	5
3 Survey Design and Execution	8
4 Detailed Analysis	11
4.1 “Where, What, Who, How, and Why?”	11
4.1.1 RQ1a: Which areas of industry are using MBSwE and/or autonomous systems?	11
4.1.2 RQ1b: What is the level of safety criticality of the applications where MBSwE and/or autonomous systems are being used?	12
4.1.3 RQ1c: What are the programming languages used during the development and deployment of the project?	13
4.1.4 RQ1d: How was the code implementing autonomous functionality during de- velopment and deployment developed?	14
4.1.5 RQ1e: Were special hardware and/or cloud used for code implementing au- tonomous functionality?	15
4.1.6 RQ1f: What are the respondents’ roles in the project?	16
4.2 Details on Autonomy	18

4.2.1	RQ2a: Was the AUC developed using MBSwE and which MBSwE tools were used?	18
4.2.2	RQ2b: What was the level of autonomy of the AUC?	18
4.2.3	RQ2c: What algorithms and modeling paradigms were used to develop the AUC?	20
4.2.4	RQ2d: How were the requirements for the AUC specified?	20
4.2.5	RQ2e: What were the challenges associated with the AUC?	21
4.3	Reuse of Software Artifacts	24
4.3.1	RQ3a: What artifacts were reused and to what extent?	24
4.3.2	RQ3b: Are there any negative aspects of reuse?	25
4.3.3	RQ3c: What were the difficulties due to reuse?	26
4.3.4	RQ3d: What were the benefits of reuse?	26
4.4	Processes and Standards	28
4.4.1	RQ4a: Which life-cycle model was used?	28
4.4.2	RQ4b: Which modeling standards and coding standards were used by the projects?	28
4.4.3	RQ4c: Did the system go through a certification process and were the AUCs part of the certified system?	29
4.5	Verification and Validation of Autonomous Systems	30
4.5.1	RQ5a: Which quality attributes were verified and validated?	30
4.5.2	RQ5b: How were the models verified & validated?	30
4.5.3	RQ5c: How was the AUC verified & validated during development?	31
4.5.4	RQ5d: How was run-time behavior of the AUC monitored/assured?	32
4.5.5	RQ5e: Were the reused software artifacts verified & validated?	32
4.6	Bugs	35
4.6.1	RQ6a: Where there any bugs specific to model-based approach, autonomous function, and/or reuse?	35
4.7	Multiple Project Discussion Provided by The Respondents	36
5	Conclusions and Future Work	37
	Appendix A	42
	Appendix B	44

List of Figures

3.1	Flowchart of Survey	10
4.1	Areas of industry that use MBSwE	12
4.2	Safety criticality of applications	13
4.3	Programming languages used during development	14
4.4	Programming languages used during deployment	14
4.5	Origins of code	15
4.6	Hardware development using cloud-based computing	16
4.7	Respondents' role in the project	17
4.8	Number of autonomous components of entire system	19
4.9	MBSwE tools/development environments used for development of AUC	20
4.10	Level of autonomy for specified tasks	21
4.11	Algorithms and modeling paradigms for autonomous components	22
4.12	Autonomous components requirements	22
4.13	Challenges encountered during development, deployment and use of AUC	23
4.14	The extent of reuse of different software artifacts for AUC	24
4.15	The extent of reuse of different software artifacts for non-AUC	25
4.16	Negative aspects of reuse	26
4.17	Difficulties due to reuse	27
4.18	Benefits of reuse	27
4.19	Life-cycle model used by projects	28
4.20	Modeling and coding standards	29
4.21	Quality attributes that were verified and validated	30
4.22	Method of validation and verification of the models	31
4.23	Tools used for the verification and validation of the models	32

4.24	Methods used for verification and validation of autonomous components during development	33
4.25	How correct run-time of autonomous components were ensured	34
4.26	Methods used for handling the violations and errors of AUC	34

List of Tables

4.1	Use of special hardware	16
4.2	Additional information on certification	29
4.3	V&V of autonomy	31
4.4	V&V of reuse	33
4.5	Bugs Specific to Reuse, Autonomy, and MBSwE	35
4.6	Further comments provided by respondents	36
5.1	Summary of the main findings	38

Acronyms

AGC	auto-generated code
ARC	Ames Research Center
AUC	autonomous components
CUDA	Compute Unified Device Architecture
DNN	deep neural network
DO-178C	Software Considerations in Airborne Systems and Equipment Certification
GSFC	Goddard Space Flight Center
GPU	graphics processing unit
JPL	Jet Propulsion Laboratory
MBSwE	Model-based Software Engineering
NASA	National Aeronautics and Space Administration
NASA IV&V	NASA Independent Verification and Validation Facility
SWHM	software health management
SwA	software assurance
UAS	unmanned aerial system
UAV	unmanned aerial vehicle
V&V	verification and validation

Chapter 1

Introduction

In the past, decision making was a responsibility of humans for projects employing computer systems. For example, decision making during missions was traditionally a responsibility of human ground & flight controllers at NASA. However, due to computational advancements, emergence of decision making algorithms, and emphasis on efficiency, autonomous systems are becoming a larger part of the mission [1]. This means most modern missions require some form of autonomous control, which imposes novel and hard challenges for Software Assurance (SwA). NASA defines autonomy as the ability of a system to achieve goals while operating independently of external control[2].

Autonomous systems range from model-based diagnostics and prognostics to artificial intelligence based systems, and may be responsible for acquiring information, analyzing the information, decision making, and acting upon those decisions. Autonomous systems may be both safety and mission critical, where failures can place human life and the mission in jeopardy. For this reason, the level of autonomy plays an important role in the different tasks an autonomous system performs. The main tasks of autonomy are defined by Proud et. al.[3] as information acquisition, information analysis, decision and action selection, and action implementation. In addition to the tasks of an autonomous system, the levels of autonomy are defined by LaVallee et. al.[4] as: the human is primary and the computer is secondary where autonomy is used only as a tool for assistance, the computer operates with human interaction, the computer operates independently of the human where the human has limited capabilities, and computer has full autonomy with the human out of the loop. It is important to note that each task can have a different level of autonomy and not all systems implement each task.

Autonomous systems may be composed of autonomous components (AUC) which are compo-

nents or subsystems that provide autonomous capabilities or support autonomous operations. Some examples of AUCs are vision-based navigation, system health management and prognostics, flight management systems that can operate without human intervention, and subsystems of deep space missions which perform complex operations without direct or remote operator control.

Problem statement: Although the use of autonomous systems is showing an increasing trend, not much is known on the current state-of-the-practice in this area. Even less is known about the use of MBSwE and reuse in the context of autonomous systems. In order to gain a deeper understanding of the state-of-the-practice of autonomous systems, we surveyed professionals who work in this area and analyzed their responses.

The main goals of our survey are as follows:

- G1** Assess the current state-of-the-practice of using autonomous systems (i.e., answer the “Where”, “What”, “How”, “Why”, and “Who” questions),
- G2** Identify and – wherever relevant – quantify the benefits and challenges of reuse of AUC,
- G3** Explore the processes and standards used developing AUC, and
- G4** Investigate verification and validation (V&V) of the models, autonomy, and reuse.

The specific research questions that address these goals are described next. Goal G1 is addressed by research questions RQ1 and RQ2. Research questions RQ3 and RQ4 are focused on goals G2 and G3, respectively. Finally, goal G4 is addressed by research questions RQ5 and RQ6.

RQ1 “Where, What, Who, How and Why?”

RQ1a Which areas of industry are using MBSwE and/or autonomous systems?

RQ1b What is the level of safety criticality of the applications where MBSwE and/or autonomous systems are being used?

RQ1c What are the programming languages used during the development and deployment of the project?

RQ1d How was the code implementing autonomous functionality during development and deployment developed?

RQ1e Were special hardware and/or cloud used for code implementing autonomous functionality?

RQ1f What are the respondents’ roles in the project?

RQ2 Details on Autonomy

RQ2a Was the AUC developed using MBSwE and which MBSwE tools were used?

RQ2b What was the level of autonomy of the AUC?

RQ2c What algorithms and modeling paradigms were used to develop the AUC?

RQ2d How were the requirements for the AUC specified?

RQ2e What were the challenges associated with the AUC?

RQ3 Reuse of Software Artifacts

RQ3a What artifacts were reused and to what extent?

RQ3b Were there any negative aspects of reuse?

RQ3c What were the difficulties due to reuse?

RQ3d What were the benefits of reuse?

RQ4 Processes and Standards

RQ4a Which life-cycle model was used?

RQ4b Which modeling standards and coding standards were used by the projects?

RQ4c Did the system go through a certification process and were the AUCs part of the certified system?

RQ5 Verification and Validation of Autonomous Systems

RQ5a Which quality attributes were verified and validated?

RQ5b How were the models verified & validated?

RQ5c How was the AUC verified & validated during development?

RQ5d How was run-time behavior of the AUC monitored/assured?

RQ5e Were the reused software artifacts verified & validated?

RQ6 Bugs

RQ6a Where there any bugs specific to model-based approach, autonomous function, and/or reuse?

This problem report presents the results of our survey, which was focused on the use of MBSwE and autonomous systems with reuse in different industry domains, worldwide. Out of 129 respondents to the survey, 110 used MBSwE and/or autonomous systems in their projects and answered questions beyond the first question. To assure that respondents' answers are not biased, the survey was administered anonymously. Among those that voluntarily provided information about their affiliation, many respondents are affiliated with NASA, including Ames Research Center (ARC), Goddard Space Flight Center (GSFC), Jet Propulsion Laboratory (JPL), and NASA Independent Verification and Validation Facility (IV&V). Other affiliations include Shanghai Fuxin Intelligent Transportation Solutions Co. (FITSCO), Intecs, Mondragón Goi Eskola Politeknikoa (MGEP), and Technical University of Munich.

This report is structured as follows. Chapter 2 discusses the related works. Chapter 3 gives an overview of the survey, its design and motivation and describes how the actual survey was carried out. Results of the survey are discussed in detail in Chapter 4. Finally, Chapter 5 provides a summary of the observations.

Chapter 2

Related Work

Due to the emerging nature of autonomy, evaluation of autonomous systems is limited. However, there have been a number of publications recently addressing this topic. Here we restrict the description to works that primarily used survey as a method for data collection and studied the experiences of industrial use of MBSwE and/or autonomous components.

The only survey focused on autonomy for spacecraft was conducted by LaVallee et al. [4]. The survey focused on defining an autonomous system, categorizing the level of autonomy, and the complexity of the implementation from a single component to an entire flight and ground implementation within the space domain. The authors analyzed the intelligent reasoning stages and the complexity and found that most responses used complexity at the level of autonomous components with most applications using a more human-centric reasoning system. The authors concluded that successful applications are implemented by commercial-off-the-shelf components that perform a single function.

Goseva-Popstojanova et al. [5] conducted an empirical survey on the state-of-the-practice of MBSwE and auto-generated code (AGC) within NASA and in various industry domains worldwide. The report explored the state-of-the-practice, the benefits and challenges, and software assurance of models and auto-generated code. Analysis of the results from the 114 respondents to the survey yielded the following results: (1) MBSwE and AGC are mainly used in space, automotive, and aeronautics industries for control systems applications, (2) learning to effectively use MBSwE and AGC was difficult but improved productivity, maintainability, and quality, and (3) V&V standards and certification were not used on a majority of projects.

Proud et al. [3] discussed NASA's SMART (Spacecraft Mission Assessment and Re-planning Tool) as a prototype to functionally decompose a flight management system with a suitable level

of autonomy for the desired functionality. To achieve the desired level of autonomy, the authors introduced the Level of Autonomy Assessment Tool to produce a summary of the appropriate Level of Autonomy for a desired function. The tool is designed to find the optimum level of autonomy that minimizes cost, maximizes safety and efficiency, completed on time, and with complete trust from its operators.

Parasuraman et al. [6] introduced a framework for types and levels of autonomy that act as an objective basis to determine the extent of autonomy for different tasks. The authors introduced four distinct information processing categories of autonomy: information acquisition, information analysis, decision and action selection, and action implementation. The authors also define the level of autonomy on a scale from 1 to 10 where 1 is full human decision making and 10 is full computer decision making and values in between 1 and 10 represent some combination of human and computer interaction. The level of autonomy can be applied to each information processing category using evaluative criteria to determine the appropriate extent of autonomy.

Chen et al. [7] presented concepts related to autonomous systems such as control and control level metrics to measure the level of autonomy. The architecture of autonomous Unmanned Aerial Vehicle (UAV) systems were divided into three levels: execution, coordination, and organization. In particular, the constraint conditions and the insights of Coordination Level and Organization Level were studied comprehensively. The results indicated that the important hardware and software technologies for multiple tasks are modularized depending on the requirements of mission. The software technologies are distributed to stages of flying particularly.

Schumann et al. [8] proposed a Bayesian method to diagnose and avert software faults in real-time using Software Health Management (SWHM). Using a two stage V&V process, both the model and code levels of a safety critical component are analyzed. The authors conclude that SWHM can provide an additional layer of safety during run-time, but is not suitable for replacement of V&V and certification of the system.

Schumann et al. [9] presented an advanced technique for analysis and V&V of health management models of the System Health Management (SHM). The authors used a combination of n-factor combinatorial exploration and Monte Carlo techniques for the analysis of Health Management Models. Using the aforementioned technique, potential weaknesses and unwanted parameter sensitivity in the health model were detected quickly.

Schumann et al. [10] presented a literature survey of the use of neural networks in high assurance systems of various fields of study. The authors focused on the identification and management of hazards in safety critical Artificial Neural Network systems. The authors concluded that traditional

V&V for safety-critical code is insufficient for neural network applications.

In a recent paper by Nikora et al. [11], the authors discussed V&V techniques of systems with autonomous capabilities. The authors investigated development of V&V techniques in three areas: diagnostic model, diagnostic engine, and combination of model and engine. The authors noted that test procedures and expected results about a component’s nominal functionality are more easily obtained than off-nominal behavior.

In this paper, we present the results of our survey of autonomous systems with reuse. Our survey investigated questions related to the benefits and challenges of MBSwE for autonomous systems with a focus on the extent of reuse. More importantly, our survey and the results presented in this report aimed to fill gaps in the current knowledge about the state-of-the-practice of MBSwE for autonomous systems. Our project identified the current best practices and gaps, and shows a path forward toward the assurance of autonomous software systems with reuse. Specifically, we explored respondent experiences related to software assurance practices of MBSwE, autonomous systems, and reuse. The software assurance practice included topics of processes and standards, verification and validation, and bugs.

Chapter 3

Survey Design and Execution

The survey was administered following the steps outlined in [12]: (1) define the goals, (2) transform the goals into research questions, (3) design the questionnaire, (4) evaluate of questionnaire using pilot executions, (5) execute the survey, and (6) analyze and package results. The goals of the survey were to explore the development and software assurance of autonomous systems with reuse of software artifacts. Our Survey consisted of 122 multiple choice and free response questions divided into the following sections:

1. *Introduction* section was related to use of autonomous systems and/or MBSwE.
2. *Where & What* section addressed questions about the target domain, and level of safety criticality.
3. *How* section focused on code used during development and deployment, and use of special hardware.
4. *Autonomy Details* contained questions about the level of autonomy for different tasks, modeling paradigms, requirement specification, and challenges developing autonomous systems.
5. *Reuse Details* section was concerned with the extent of reuse for AUC and non-AUC from previous projects and the benefits and challenges related to reuse.
6. *Processes and Standards* section focused on life-cycle, modeling, and coding standards, and certification.
7. *Verification and Validation* section was concentrated on questions about the V&V of the models, autonomy, and reuse.

8. *Bugs* section dealt with bugs related to the models, autonomous functionality, and reuse of software artifacts.
9. *Multiple Project Discussion* section allowed respondents to explain the differences in their projects if they worked on more than one project involving autonomy.

A flow chart of the survey is shown in Figure 3.1. The survey was divided into separate pages based on the aforementioned sections. As one can observe from the flow chart diagram, the first question was used to determine if the respondent worked with autonomous systems and the second question was used to determine if the respondent used MBSwE. If the respondent did not work on autonomous systems or use MBSwE, they were not allowed to enter the survey. For respondents who entered the survey, the first set of questions were about the nature of their work. Before proceeding to the AUC details, the respondents were asked about the number of AUCs they worked on. Based on the response to the number of AUCs the questions on the AUC details section was repeated for the number of times the respondent entered. After the respondent completed the AUC details section, they were asked if there was reuse in the project. If the respondent chose yes, they were directed to the questions on the reuse details section then to the questions about processes and standards, verification and validation, and bugs. If the respondent chose no, they were directed to final part of the survey about processes and standards, verification and validation, and bugs.

The survey was distributed via email invitations to contacts at NASA and people who work in related fields. The intended respondents were software engineering practitioners and researchers with experience in using MBSwE and/or autonomous systems in industry. For the execution of the survey we tried to reach as many potential respondents as possible using non-probabilistic convenience sampling and snowballing. Techniques included: sending invitation messages to academic and industrial contacts of the research team and to relevant mailing lists, and placing advertisements at related conferences and online forums. Additionally, over 300 authors of research papers related to autonomy were contacted via email invitations.

The data were collected during May and June 2019 by an on-line questionnaire created using Survey Monkey[®].

Figure 3.1: Flowchart of Survey

Chapter 4

Detailed Analysis

This section presents the survey results on the characteristics of software projects where autonomous systems were used. The analysis presented in this section is based on the responses of the 110 respondents who have used MBSwE and/or AUC in their projects and answered more than just the first survey question. The survey did not require the respondents to answer all questions. Therefore, the percentages given in the following figures are relative to the total number of respondents answering that specific question.

4.1 “Where, What, Who, How, and Why?”

For the first research question, we examined the necessity of autonomous systems and how autonomous systems are used in industry. Research Question 1 aims to capture the essence of autonomous systems.

4.1.1 RQ1a: Which areas of industry are using MBSwE and/or autonomous systems?

Autonomous systems are used in applications such as self-driving cars, smart robots, unmanned aerial and marine vehicles, and many other application areas. Figure 4.1 contains the results about the areas of industry where MBSwE and/or autonomous systems are used. Space industry dominated with 40% of respondents, followed by aviation at 16%, military at 13% and automotive with 9%. Domains included in “Other” were financial, government, transportation, Earth Ocean Sciences, and Department of Homeland Security.

Figure 4.1: Areas of industry that use MBSwE and/or autonomy (99 respondents, multiple answers possible)

In comparison to our earlier study in 2016 [5] which was focused only on MBSwE, the number of respondents from the automotive industry is smaller (i.e. 9% compared to 33%). This might be due to proprietary nature of autonomous systems in the automotive industry where respondents are not legally permitted to divulge information.

4.1.2 RQ1b: What is the level of safety criticality of the applications where MBSwE and/or autonomous systems are being used?

The level of safety criticality defines the impact of failures if the software or system fails. We used the levels of safety criticality as defined in the DO-178C [13] standard: catastrophic, hazardous, major, minor, and no effect. The distributions of safety criticality of the entire system is shown in Figure 4.2. 45% of respondents indicated catastrophic or hazardous level of safety criticality, followed by 39% with major safety criticality, and 8% with minor criticality. On the other hand, only 8% of respondents indicated an unknown level of safety criticality. The percentage of catastrophic and hazardous in a previous survey [5] which used different levels of criticality, is about the same with high criticality at 46%. Since a large portion of the respondents were from the space and aviation industry which typically deals with safety-critical applications, it is not surprising that many respondents indicated a high level of safety criticality.

Figure 4.2: Safety criticality of applications (97 respondents)

4.1.3 RQ1c: What are the programming languages used during the development and deployment of the project?

Next, we examined which programming languages were used. Since the development of AUC can require substantial effort during development time, for example, training of a deep neural network, we distinguished between programming languages and tools that were used in project during development and used in deployed code during operation. Programming languages used during development produce intermediate artifacts (e.g., Python[®]/TensorFlow in the cloud to train a DNN). Programming languages used in deployed code, are executed during system operation (e.g., CUDA[®] on a GPU for image classification). Figure 4.3 presents a Venn diagram of the programming languages used during development. It is important to note that the respondents could choose multiple languages. In fact, 8 respondents indicated they used all of the programming languages shown in the figure and most respondents indicated use of multiple languages. C and C++ dominate the responses, with 42 and 32 responses respectively, followed by MATLAB[®] and Python[®], with 25 and 24 responses respectively.

Programming languages used during deployment are given in Figure 4.4. As mentioned previously, the respondents could choose multiple languages. Similarly as in Figure 4.3, C and C++ dominate with 35 and 29 responses respectively followed by Python[®] with 16 responses. The number of MATLAB[®] responses decreased significantly from 25 responses in Figure 4.3 to 6 responses in Figure 4.4. There were 6 responses which indicated the use of every language during deployment. The category of “Other” programming languages in Figures 4.4 and 4.3 includes C#, assembly, Lisp, JavaScript and G2. It is interesting to note that the number of respondents who used MATLAB[®]

Figure 4.3: Programming languages used during development (97 respondents, multiple selections possible)

during deployment decreased significantly from 25 during development to 6 responses.

Figure 4.4: Programming languages used during deployment (97 respondents, multiple selections possible)

4.1.4 RQ1d: How was the code implementing autonomous functionality during development and deployment developed?

Respondents were asked about how the autonomous code was developed and could only choose one answer from the provided answers in the survey. Over one third of the respondents (36%)

indicated that the code during development was written from scratch, i.e., there was no reuse. 23% of respondents indicated use of customized existing code libraries and 20% of respondents indicated use of unmodified existing code libraries. There was very little reuse of existing libraries, in fact, only 5% of respondents selected reuse of existing software.

Figure 4.5

Figure 4.5b represents the origins of the autonomous code during deployment. As stated previously, the respondents could only choose one of the options. The results are similar to the way software was developed during development with 38% developing code from scratch, 23% using customised existing code library, and 17% using unmodified existing code library. The results in Figure 4.5b are similar to the way software was developed during development in Figure 4.5a. We see very little reuse of existing software with only 3% of respondents indicating as such.

4.1.5 RQ1e: Were special hardware and/or cloud used for code implementing autonomous functionality?

As shown in Figure 4.6, the overwhelming majority (i.e., 70%) of respondents did not execute any software “in the cloud” or on a remote server. This indicates that an overwhelming majority of respondents are using “in-house” hardware.

We also asked the respondents about the use of special hardware during the development and operational stages of the system. As shown in table 4.1, a significant number of respondents did not use any special hardware (i.e., 70% during development and 66% during operation). Some respondents did indicate the use of special hardware, which included touch screens, NVIDIA Jetson, and custom designed space robot prototypes.

Figure 4.6: Hardware development using cloud-based computing (71 respondents)

Table 4.1: Use of special hardware (69 respondents)

	During development	During operation
Special hardware	16%	19%
No special hardware	70%	66%
NA	14%	15%

4.1.6 RQ1f: What are the respondents' roles in the project?

Figure 4.7 presents the respondents' roles in the projects. The roles of design and research constitute the largest amount of the responses with 21 and 17 respondents respectively. The roles of model development, programming, software and system integration, testing/QA/V&V, and project management are roughly evenly distributed with 12 to 13 respondents per category. The category of "Other" roles includes IV&V Analyst.

Figure 4.7: Respondents' role in the project (34 respondents, multiple selections possible)

4.2 Details on Autonomy

For this section of the survey, we asked the respondents to enter information for one or more AUC. In the survey, we defined autonomous components as a component or subsystem that provides autonomous capabilities or supports autonomous operations. Hence, we distinguish responses, each AUC detail, from respondents. Examples of autonomous components include: vision based navigation for self-driving cars, system health management and prognostics for autonomous UAS flight, UAS flight management systems that can operate the UAS during its mission without human intervention, and components of a deep-space mission which perform complex operations without direct or remote operator control. The respondents could choose between 0 and 10 AUC. Figure 4.8 shows the distribution of number of AUC entered by the respondents who could enter this part of the survey. The respondents who selected 0 AUC were directed to the reuse section of the survey, refer to Figure 3.1. The respondents who chose one or more AUC were asked to answer a set of questions about autonomy details for each AUC. From Figure 4.8, we see the number of completed responses per AUC per respondent. There were a few respondents who chose more than one AUC but did not answer the questions about each AUC. Rather than excluding respondents who did not complete the questions in this section, we chose to look at each response to each AUC separately. Therefore we distinguish the AUC responses from the number of respondents who completed the questions about each AUC. There were a total of 58 AUC responses with 28 respondents providing details about one AUC, 6 respondents entering details about two AUCs, 2 respondents providing details about 3 AUCs, and 2 respondents who provided details about four and eight AUCs respectively. Some forms of AUC mentioned include diagnostic reasoners, health management, classifiers, and navigation.

4.2.1 RQ2a: Was the AUC developed using MBSwE and which MBSwE tools were used?

Only 38% of respondents used MBSwE for development of AUC. Figure 4.9 presents a bar chart for MBSwE/development environments used. It can be seen that 17% of responses used MATLAB[®]/Simulink[®], 13% used Rational Rhapsody[®], and 4% used Rational Rose[®]. The “Other” category includes the following: Java[®], Prolog, Papyrus[14], and Generic Modeling Environment (GME).

4.2.2 RQ2b: What was the level of autonomy of the AUC?

The level of autonomy was first introduced by Sheridan et al. [15] and further refined by LaVallee et al. [4] who proposed the following six levels of autonomy: “manual”, “automatic notification”,

Figure 4.8: Distribution of the number of AUC entered per respondent (58 components entered)

“intelligent reasoning on ground with human control”, “intelligent reasoning on ground with autonomous control”, “intelligent reasoning onboard”, and “autonomous thinking spacecraft”. In our survey, for simplicity, we limited the levels of autonomy to the following:

- The human is primary and the computer is secondary,
- The computer operates with human interaction,
- The computer operates independently of human,
- Computer has full autonomy.

We adopted the different tasks an AUC may perform from Parasuraman [6]: information acquisition (monitor), information analysis (analyze), decision and action selection (decide), and action implementation (act). The tasks were derived from military strategist Colonel John Boyd’s Observe Orient Decide Act loop (OODA) cognitive model [16]. Figure 4.10 presents the results on the level of autonomy for these four tasks. From the figure, we see that a significant percentage of AUC (i.e., from 50% to 67%) operate with full autonomy. Only 5% to 12% of AUC had the human primary and computer secondary.

Figure 4.9: MBSwE tools/development environments used for development of AUC (58 AUC, multiple selections possible)

4.2.3 RQ2c: What algorithms and modeling paradigms were used to develop the AUC?

Autonomous operation can be based on different algorithms and modeling paradigms. The percent of AUC developed using different algorithms and modeling paradigms are shown in Figure 4.11. From Figure 4.11, it can be seen that 35% of AUC involved Rule-Based where decisions of AUC were implemented as program logic or using explicit rules, followed by Planning Systems/languages at 22% and Statistical and filtering methods at 18%. Both on-line and off-line Machine-learning were surprisingly low at 9% and 3% respectively. The “Other” category included nonlinear programming techniques, model checking, and nonlinear optimization.

4.2.4 RQ2d: How were the requirements for the AUC specified?

The requirement specification is shown in Figure 4.12a. The respondents could select more than one answer, but most respondents only selected one option. In fact, only 10 AUC respondents indicated more than one approach for requirement specification. 31 AUC requirements were specified to the same level as the non-AUC, 26 AUC requirements were specified using natural language and 11

Figure 4.10: Level of autonomy for specified tasks

AUC requirements were specified using specific formalisms. The “Other” specification approaches includes Langley Research Center requirements (LaRC) and interfacing with timing requirements.

Additionally, most responses indicate the level of requirements were the same as or to a higher level than the non-AUC. The survey also included a question on the level of details of requirement specification for autonomous parts of the system compared to non-autonomous parts. For this question, the answer choices were mutually exclusive. As shown in Figure 4.12b the majority of AUC (i.e., 63%) were developed using requirements with the same level of detail as the non-autonomous parts. 25% of respondents indicated the level of details of requirement specifications were higher for autonomous components compared to non-autonomous components.

4.2.5 RQ2e: What were the challenges associated with the AUC?

Using an ordinal scale, we asked the respondents about challenges associated with AUC. In Figure 4.13, the respondent chose the amount of difficulty during development and/or deployment of the AUC due to six different circumstances. The number of responses to each circumstance varied, but the number of responses were converted to percents to show the degree of difficulty for each circumstance compared to the others. It is interesting to note that system complexity led to the most challenges, with 0 AUC stating no difficulty and 67% of the AUC indicating moderate or major difficulties.

Figure 4.11: Algorithms and modeling paradigms for autonomous components (58 AUC, multiple selections possible)

Figure 4.12

Figure 4.13: Challenges encountered during development, deployment and use of AUC

4.3 Reuse of Software Artifacts

In this section we describe the results related to reuse of software artifacts. From the survey flowchart in Figure 3.1, we see that respondents could skip this section if they did not reuse software artifacts. For the reuse section, we were interested in reuse for both autonomous and non autonomous components. It is important to note that not everyone who responded to the previous section continued responding. Also, not everyone who responded to the questions in this section also answered anything on the AUC part of the survey.

4.3.1 RQ3a: What artifacts were reused and to what extent?

Figures 4.14 and 4.15 show the amount of reuse for different types of artifacts for AUC and non-AUC respectively. As can be seen in Figure 4.14, the largest amounts of at least 30% or more reuse were 63% of the responses indicating code reuse, 50% of the responses indicating software model reuse, and 39% indicating telemetry data and log file reuse. Of the at least 30% code reuse category, the offline training data had the lowest amount of reuse, with only 5% of respondents.

Figure 4.14: The extent of reuse of different software artifacts for AUC (30 respondents)

In addition to the reuse of AUC, Figure 4.15 contains the amount of reuse for the non-AUC

parts. As expected, when comparing Figures 4.14 and 4.15 the results are similar. However, there are a few notable changes. We can observe that data reuse for offline training drops from 27% to 9% for the 30% or more extent of reuse, and data reuse: trained machine learning model decreased from 23% to 9% for the 30% or more extent of reuse.

Figure 4.15: The extent of reuse of different software artifacts for non-AUC (30 respondents)

4.3.2 RQ3b: Are there any negative aspects of reuse?

In Figure 4.16, each respondent could only choose one response for each aspect of reuse. However, not everyone responded to each aspect, thus there were slightly different number of responses for each aspect. In order to facilitate reading the figures, the values were converted to percentages. The color scale of the figures contains green, yellow, orange, and red, where green represents “good” and red represents “bad”.

The negative aspects of reuse are shown in Figure 4.16. Here, we see that between 31% and 40% of respondents did not observe any negative aspects due to reuse. The largest negative aspect of reuse was due to hindering new ideas with 14% of respondents selecting “major” and 11% selecting “moderate” negative aspect. It is not surprising that 0% of respondents encountered major difficulties verifying and validating reused software since the software would have underwent a V&V

procedure before being reused. Some respondents added comments to the negative aspects stating lack of documentation and difficulty adding patches into a formal design process.

Figure 4.16: Negative aspects of reuse (30 respondents)

4.3.3 RQ3c: What were the difficulties due to reuse?

The amount of difficulty in different aspects of reuse can be observed in Figure 4.17. We see that the least amount of difficulty was in the categories of operational conditions, V&V of reuse software and reuse process. On the other hand, we can see the most amount of difficulty occurs with uncertain operational conditions/environments with 11% of respondents indicating major difficulties. One respondent commented, “The hardest part was moving from TRL2 to 4 because of the junk code that was written in TRL 2. Moving from 4 to 8 was trivial after that, up to and including those “fast patch” events that happen day of/night before big integration tests.”

4.3.4 RQ3d: What were the benefits of reuse?

Figures 4.18a & 4.18b show the benefits of reuse, in terms of productivity, quality, and cost. As can be seen in Figure 4.18a, 63% of respondents reported increase productivity and 40% reported increased quality due to reuse, and 37% observed decreased cost.

Figure 4.17: Difficulties due to reuse (29 respondents)

Figure 4.18

4.4 Processes and Standards

In the related works focused on autonomy[4] [3], neither work examined the process and standards developing the autonomous system. In this section, we present novel results regarding the processes and standards used to develop autonomous systems.

4.4.1 RQ4a: Which life-cycle model was used?

Figure 4.19 shows the different types of life-cycle models used on projects. Interestingly, twice as many of the respondents used the Agile life-cycle model than waterfall model (i.e., 38% vs 19%). Case-based development was next at 14%, followed by Rapid application development at 11%, spiral at 8% and rational unified model with 5%. The “Other” category included one respondent who stated the project started out as Agile but has gone through an enormous amount of change during the last ten years.

Figure 4.19: Life-cycle model used by projects (39 respondents)

4.4.2 RQ4b: Which modeling standards and coding standards were used by the projects?

As shown in Figure 4.20a, almost half of the respondents (i.e., 68%) did not use any modeling standard while 26% used UMLTM standards. Under the “Other” category respondents listed SysML [17], MathWorks[®] MAAB [18], and JANI [19].

Figure 4.20b shows the type of coding standards used for the projects. As can be seen in the

Figure 4.20

figure, about a quarter of the respondents did not follow any coding standards. However, higher percent of respondents used coding standards such as NASA (19%) and JPL (18%). The “Other” category included responses such as Orion, Thales, and internal coding standard.

4.4.3 RQ4c: Did the system go through a certification process and were the AUCs part of the certified system?

As can be seen in table 4.2, only 24% of systems went through certification and in only 26% of cases AUC were part of the certified system. The respondents who indicated their projects went through a certification process listed the following standards: NASA, Security Technical Implementation Guide (STIG), or an internal standard.

Table 4.2: Additional information on certification (41 respondents)

	Yes	No
System went through certification process	24%	76%
Autonomous components were part of certified system	26%	74%

4.5 Verification and Validation of Autonomous Systems

While previous works have proposed V&V of autonomous systems[20], no prior work has surveyed how V&V is actually performed on autonomous systems. In this section, we investigated the V&V of autonomous systems specifically, the tools and methods used to V&V autonomous systems.

4.5.1 RQ5a: Which quality attributes were verified and validated?

The Venn diagram shown in Figure 4.21, depicts the quality attributes that were V&V. It is interesting to note that only 4 respondents performed V&V with respect to security; they have V&V all the other quality attributes as well. Most respondents performed V&V on correctness, performance, robustness, and safety.

Figure 4.21: Quality attributes that were verified and validated (36 respondents, multiple selections possible)

4.5.2 RQ5b: How were the models verified & validated?

Figure 4.22 shows the results of how V&V of the models was performed. The dominant method of verification and validation of the models was testing with 30% of the responses followed by simulation with 24%, manual model inspection/reviews with 20%, automated model analysis at 10%. Only 6% of respondents did not perform V&V.

Challenges encountered during development, deployment and use of AUC are presented in the Venn diagram shown in Figure 4.23. The number of responses is very small with only 8 respondents

Figure 4.22: Method of validation and verification of the models (37 respondents, multiple selections possible)

in total. The tools used for V&V. The most used tool was Design Verifier with 3 responses. Other tools used for V&V included UPPAAL [21], Jenkins [22], PRISM, and custom tools. It is worthwhile to note that none of the respondents chose Reactis or T-VEC tools.

4.5.3 RQ5c: How was the AUC verified & validated during development?

Table 4.3: V&V of autonomy (32 respondents)

Did you use tools for the verification and validation of the models?	Response
Yes	56%
No	22%
NA	22%

In Figure 4.24, respondents answered how AUC were verified and validated during the development phase. The majority of respondents indicated they used simulation based testing (20%), unit and integration testing (19%), and manual model/code reviews or inspections (18%). The least used V&V method was automated model analysis at only 3%. Other methods include inspection of results by stakeholders and comparing performance with human.

Figure 4.23: Tools used for the verification and validation of the models (8 respondents, multiple selections possible)

4.5.4 RQ5d: How was run-time behavior of the AUC monitored/assured?

The bar chart in Figure 4.25 indicated how respondents monitored and assured correct run-time behavior of the AUC using different monitoring and assurance methods. The most used methods were monitoring of variable values and ranges at 29%, followed by monitoring of requirements at 21% and cross-checking that commanded actions meet intended post-conditions at 16%. No respondent answered “other” for this question.

Figure 4.26 contains a bar chart regarding the V&V of the AUC where respondents answered how violations and errors were treated. Notification to human operator was the most selected at 30%, followed by control handed over to human operator at 20%, automatic autonomous action at 16%, autonomous response and adaptation of mission at 14%, and like a system failure at 14%. We see that most violations and errors resulted in increasing control to human operators with only 3% of respondents indicated control was taken away from the human operator. In addition to the methods shown in the figure, respondents also indicated failover to layered recovery mechanisms, and probability of correctness was included in predictions.

4.5.5 RQ5e: Were the reused software artifacts verified & validated?

During the survey, respondents were asked if they specifically verified and validated reused artifacts. The responses are shown in Table 4.4. From the table, we see that only 19% of respondents verified and validated reused artifacts while over half (56%) of respondents did not. It is not surprising that

Figure 4.24: Methods used for verification and validation of autonomous components during development (34 respondents, multiple selections possible)

over half of the respondents did not perform V&V on reused artifacts since most artifacts were most likely verified and validated before being reused.

Table 4.4: V&V of reuse (33 respondents)

Did you specifically Verify & Validate reused artifacts?	Response
Yes	19%
No	56%
NA	25%

Figure 4.25: Methods used for monitoring/assurance of AUC run-time behavior (34 respondents, multiple selections possible)

Figure 4.26: Methods used for handling the violations and errors of AUC (34 respondents, multiple selections possible)

4.6 Bugs

4.6.1 RQ6a: Where there any bugs specific to model-based approach, autonomous function, and/or reuse?

Our survey also explored the existence of bugs specific to reuse of software artifacts, autonomous functionality, and model-based approach. The presence of bugs with respect to different aspects of the system are given in Table 4.5. The amount of bugs related to model-based approach was fairly low at 9%. Some respondents commented about bugs specific to model-based approach stating some models were missing identifications and concerns about the amount of testing due to vagueness of the software test plan. In 25% of the projects, bugs were discovered due to reuse of software artifacts. This is not surprising since only 19% of respondents specifically verified and validated reused artifacts as shown in table 4.4. Respondents commented on the bugs in reuse of software artifacts indicating problems such as version conflicts, incompatibilities, adaptation of existing software produced unintended consequences, and bugs that were not corrected in previous release. The most prevalent aspect containing bugs found was related to autonomous functionality which constituted 28%. Respondents commented about the bugs due to autonomous functionality stating overfitting/underfitting, failures of commercial off the shelf components, and folded into subsequent releases. Overall, the prevalence of bugs was fairly low with less than a third of respondents reporting bugs due to to reuse of software artifacts, autonomous functionality, and model-based approach.

Table 4.5: Bugs specific to reuse, autonomy, and MBSwE (32 respondents)

	Reuse of software artifacts	Autonomous functionality	Model-based approach
Yes	25%	28%	9%
No	53%	47%	44%
NA	22%	25%	47%

4.7 Multiple Project Discussion Provided by The Respondents

At the end of the survey, we asked the respondents if they were involved in more than one project which included autonomous functionality to describe the differences and comment on their experiences. These comments are given in Table 4.6.

Table 4.6: Further comments provided by respondents

Response
Development processes applied with different levels of fidelity.
Widespread erroneous use of the term “Autonomous” for systems that are actually only “Automated.” Lack of knowledge of Model-based engineering best practices, life-cycle development processes, and requirement engineering. No innovation or improved quality due to over-reuse of artifacts and software.
Multiple different types of real-time autonomy running on several computer platforms. Terrain type classification (convolutional neural networks), system fault/anomaly detection (system model based), autonomous navigation and localization of mobile robots (3D terrain reconstruction and drive cost analysis)
Most of the projects I have been involved in were to develop demonstration concepts (rather than try to field a fully functioning agent). However the one in the survey was to develop a system that would work in the world for long duration missions. The level of maturity of the V&V efforts in this one was substantially greater (though probably not sufficient) than those demonstration projects.
Across a portfolio of similar military projects, we’ve employed the same basic development cycle and requirements verification regime. We lean heavily on automated testing to ensure correctness of all builds, and lean heavily on field testing to ensure correctness of human factors and human interaction. No significant differences across this portfolio of projects. Much of this was well-mirrored by the Mars 2020 path planning team, as well.
Different flight project, different language, different V&V approaches - really needs this survey done again for the different project.
I have also been involved with projects that use code generated from Simulink® that can run as a core flight system application.

Chapter 5

Conclusions and Future Work

The use of autonomous systems is growing due to advancements in computation, decision making algorithms, and emphasis on efficiency. However, due to the recent emergence of autonomy, very little research has been done regarding the current state-of-the-practice of autonomous systems. In an effort to gain insight into the development and use of autonomous systems, we conducted an online survey which was focused on the use of MBSwE and autonomous systems with reuse in different industry domains, worldwide. The main goals of our survey were to: (1) assess the current state-of-the-practice using autonomous systems, (2) identify and quantify the benefits and challenges of autonomy and reuse, (3) explore the processes and standards used to develop autonomous systems, and (4) investigate verification and validation of the models, autonomy, and reuse. Table 5.1 summarizes the main findings based on the responses of 110 respondents who used MBSwE and/or autonomous systems in their projects.

This work is a part of a larger project, which in addition to survey (i.e., opinion-based research) includes using a NASA mission as a case-study (i.e., evidence-based research).

Table 5.1: Summary of the main findings

RQ1: “Where, What, Who, How and Why?”
Space industry dominates the responses at 40%
45% of respondents indicate a catastrophic and hazardous level of safety criticality
C and C++ programming languages were used the most often at 36% and 30% respectively
RQ2: Details on Autonomy
Only 37% of respondents used MBSwE for development of AUC
Between 50% and 67% of respondents indicated the computer has full autonomy for different tasks
Most difficulties developing AUC were due to system complexity and environmental uncertainty
RQ3: Reuse of Software Artifacts
The largest amount of reuse is code reuse followed by model reuse and data reuse
The productivity increased in 63% of the projects with reuse
The quality increased in 40% of the projects with reuse
The cost decreased in 37% of the projects with reuse
RQ4: Processes and Standards
Almost half (48%) of respondents did not use modeling standards for their project
26% of respondents did not use coding standards
Only 24% of the projects went through a certification process
RQ5: Verification and Validation of Autonomous Systems
Out of 36 respondents, only 4 verified and validated for security
Only 19% of respondents verified and validated reused artifacts
RQ6: Bugs
25% of respondents found bugs due to reuse of software artifacts
28% of respondents found bugs due to autonomous functionality
9% of respondents found bugs due to model-based approach

Bibliography

- [1] Cancro, G. J.: APL spacecraft autonomy: then, now, and tomorrow. *Johns Hopkins APL Technical Digest*, vol. 29, no. 3, 2010, pp. 226–233.
- [2] Hall, L.: 2015 NASA Technology Roadmaps. 2015.
- [3] Proud, R. W.; Hart, J. J.; and Mrozinski, R. B.: Methods for determining the level of autonomy to design into a human spaceflight vehicle: a function specific approach. *Proc. Performance Metrics for Intelligent Systems (PerMIS '03), NIST Special Publication 1014, September 2003*, 2003.
- [4] LaVallee, D.; Jacobsohn, J.; Olsen, C.; and Reilly, J.: Intelligent control for spacecraft autonomy-an industry survey. *Space 2006*, 2006, p. 7384.
- [5] Goseva-Popstojanova, K.; Kahsai, T.; Knudson, M.; Kyanko, T.; Nkwocha, N.; and Schumann, J.: Survey on Model-Based Software Engineering and Auto-Generated Code. , 2016.
- [6] Parasuraman, R.; Sheridan, T. B.; and Wickens, C. D.: A model for types and levels of human interaction with automation. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, vol. 30, no. 3, 2000, pp. 286–297.
- [7] Chen, H.; Wang, X.; and Li, Y.: A Survey of Autonomous Control for UAV. *2009 International Conference on Artificial Intelligence and Computational Intelligence*, vol. 2, Nov 2009, pp. 267–271.
- [8] Schumann, J.; Mbaya, T.; Mengshoel, O.; Pipatsrisawat, K.; Srivastava, A.; Choi, A.; and Darwiche, A.: Software health management with Bayesian networks. *Innovations in Systems and Software Engineering*, vol. 9, no. 4, 2013, pp. 271–292.
- [9] Schumann, J.; Reed, E.; and Mengshoel, O.: Verification and Validation of System Health Management Models using Parametric Testing. 03 2011.

- [10] Schumann, J.; Gupta, P.; and Liu, Y.: Application of Neural Networks in High Assurance Systems: A Survey. *Applications of Neural Networks in High Assurance Systems*, 2010.
- [11] Nikora, A.; Srivastava, P.; Fesq, L.; Chung, S.; and Kolcio, K.: Assurance of model-based fault diagnosis. *2018 IEEE Aerospace Conference*, March 2018, pp. 1–14.
- [12] Kitchenham, B. A.; and Pfleeger, S. L.: Personal Opinion Surveys. *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. Sjöberg, eds., Springer London, 2008, pp. 63–92.
- [13] RTCA: DO-178C/ED-12C: Software Considerations in Airborne Systems and Equipment Certification. 2012. URL <http://www.rtca.org>.
- [14] Eclipse.org: Papyrus modeling environment. URL <https://eclipse.org/papyrus/>.
- [15] Sheridan, T. B.: *Telerobotics, Automation, and Human Supervisory Control*. MIT Press, Cambridge, MA, USA, 1992.
- [16] Boyd, J. R.: The essence of winning and losing. *Unpublished lecture notes*, vol. 12, no. 23, 1996, pp. 123–125.
- [17] SysML.org: SysML Open Source Specification Project. URL <http://sysml.org>.
- [18] Mathworks: The MathWorks Automotive Advisory Board. URL <http://www.mathworks.com/solutions/automotive/standards/maab.html>.
- [19] Budde, C. E.; Dehnert, C.; Hahn, E. M.; Hartmanns, A.; Junges, S.; and Turrini, A.: JANI: Quantitative Model and Tool Interaction. *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part II*, 2017, pp. 151–168. URL https://doi.org/10.1007/978-3-662-54580-5_9.
- [20] Brat, G.; and Jonsson, A.: Challenges in verification and validation of autonomous systems for space exploration. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, IEEE, vol. 5, 2005, pp. 2909–2914.
- [21] UPPAAL. URL <http://www.uppaal.org>.
- [22] jenkins.io: Jenkins. URL <https://jenkins.io/>.

- [23] CoCoSim. URL <https://coco-team.github.io/cocosim/>.
- [24] Simulink Design Verifier. URL <https://www.mathworks.com/products/sldesignverifier.html>.
- [25] No Magic, Inc: MagicDraw. URL <http://www.nomagic.com/products/magicdraw.html>.
- [26] The Mathworks Company: Matlab, Simulink, Stateflow. URL mathworks.com.
- [27] National Instruments: NI MATRIXx. URL <http://sine.ni.com/nips/cds/view/p/lang/en/nid/12153>.
- [28] Model Advisor. URL <https://www.mathworks.com/help/simulink/ug/select-and-run-model-advisor-checks.html>.
- [29] IBM: Rational Rose Modeler. URL <http://www-03.ibm.com/software/products/en/rosemod>.
- [30] IBM: Rational Rhapsody family. URL www.ibm.com/software/awdtools/rhapsody/.
- [31] Wikipedia: Rational Rhapsody. URL https://en.wikipedia.org/wiki/Rational_Rhapsody.
- [32] Reactive Systems, I.: Reactis. URL <https://reactive-systems.com/>.
- [33] T-VEC Technologies, I.: T-Vec. URL https://www.mathworks.com/products/connections/product_detail/t-vec-tester.html.
- [34] uml.org: Unified Modeling Language. URL <https://www.uml.org/>.
- [35] Visual Paradigm: Effective IT System Design, with UML & ERD. URL <https://www.visual-paradigm.com/features/>.

Appendix A: MBSwE and Verification and Validation Tools

These tools have been reported in our survey

CoCoSim is an automated analysis tool for verifying and validating MATLAB[®] Simulink[®] and Stateflow models.[\[23\]](#)

Design Verifier is a MATLAB[®] toolbox that uses formal methods to identify hidden design errors in models and can formally verify that the design meets functional requirements [\[24\]](#).

MagicDraw[™] is a visual modeling tool for UML, SysML, BPMN and UPDM. It features customizations for domain-specific languages, model decomposition, model transformation, refactoring, and decomposition, as well as template-based document generation [\[25\]](#).

MathWorks[®] Automotive Advisory Board is an independent board that develops guidelines for using MATLAB[®], Simulink[®], Stateflow and Embedded Coder[\[18\]](#).

MATLAB[®]/Simulink[®] Simulink[®] [\[26\]](#) is a model-based design system, which works as a part of MathWorks[®]' MATLAB[®] system. Very widely distributed in engineering areas, the hierarchical graphical model features numerous blocks and libraries to design continuous and discrete simulations. Stateflow, an additional package, integrates modeling of hierarchical state machines. Production-level code can be generated using Real-Time Workshop for generic and embedded platform. MathWorks[®] sells additional tools for model analysis ("model analyzer"), testing and testcase generation, as well as static analysis of C/C++ code.

MatrixX[™] is tool chain for control design, consisting of the SystemBuild modeler and AutoCode code generator [\[27\]](#).

Model Advisor is a tool in core Simulink[®] that automatically checks your model for some common mistakes[28].

Papyrus is an industrial-grade open source Model-Based Engineering tool. Eclipse Papyrus has notably been used successfully in industrial projects and is the base platform for several industrial modeling tools[14].

Rational Rose[®] is a UML software design tool for visual software modeling and component construction [29].

Rational Rhapsody[®] Rhapsody, a modeling environment based on UML, is a visual development environment for systems engineers and software developers creating real-time or embedded systems and software [30,31].

Reactis offers model-based testing, debugging, and validation for Simulink[®]/Stateflow models[32].

System Modeling Language (SysML) is a general-purpose modeling language for systems engineering applications[17].

T-Vec Tester for Simulink[®] provides an integrated solution for model analysis, automatic test generation, test execution, and results analysis[33].

Unified Modeling Language (UML) is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems[34].

UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types [21].

Visual Paradigm Eclipse is a UML CASE modeling Tool which provides report generation and code engineering capabilities including code generation. It can reverse engineer diagrams from code, and provide round-trip engineering for various programming languages[35].

Appendix B: Survey Questionnaire

Autonomous systems have gained tremendous interest in important application areas like Aeronautics, Space, and the Automotive Industry. The development of software systems, which contain Autonomous Components (AUC) is challenging, in particular, if mission success or even human life depends on it.

In this survey, we want to find out about the state of the art in the development of software systems that contain Autonomous Components and where reuse plays an important role.

This online survey has been funded by the NASA Software Assurance Research Program (SARP) and results will be published and will provide guidance to future NASA missions that require a substantial amount of autonomy.

Our estimate is that it will take about 30-45 minutes to complete the survey. All responses are anonymous, unless you specifically choose to provide personal information and your email address. A copy of the report with the results will be mailed to respondents who provided their email addresses during the survey or requested a copy of the report by sending e-mail to SurveyAutonomousSystems@gmail.com.

Please forward this call for participation to all of your colleagues with experience in this topic.

Throughout this survey, we use the following acronyms:

Acronym	Meaning
AGC	Automatically Generated Code
AUC	Autonomous Component
CNN	Convolutional Neural Network
DNN	Deep Neural Network
MBSwE	Model Based Software Engineering
SW	Software
V&V	Verification and Validation

If you have worked on more than one project, please focus on your most representative project.

* 1. Have you worked in any form (the analysis, design, development of requirements, code and verification & validation, etc.,) of software for autonomous systems in your projects?

- ☐ No
- ☐ Yes

1

4. What were the target domains? (select all that apply)

- ☐ Agriculture
- ☐ Automotive
- ☐ Aviation
- ☐ Chemical
- ☐ Medical Systems
- ☐ Military
- ☐ Nuclear
- ☐ Power Systems
- ☐ Space
- ☐ Urban Mobility
- ☐ Other (please specify)

Level of Criticality

The level of (mission and/or safety) criticality defines the impact of failures if the software or system fails. Failure conditions are categorized as catastrophic, hazardous, major, minor, or no effect. Strict definitions are found in standards like DO-178B or C (<https://en.wikipedia.org/wiki/DO-178B>), which labels the categories A through E, which we use here as a loose indicator of the level of criticality. We would like to find out how critical your overall application is, and how critical the safe operation of the Autonomous Components are.

5. What was the level of safety-criticality for the overall application?

- ☐ Catastrophic
- ☐ Hazardous
- ☐ Major
- ☐ Minor
- ☐ No Effect

3

The development of Autonomous Components (AUC) can require substantial efforts during development time, for example, training of a deep neural network. We therefore distinguish between programming languages and tools that were:

- used in project during development before deployment to produce intermediate artifacts (e.g., Python/TensorFlow in the cloud to train a DNN).
- used in deployed code, during operation to implement code that is executed during system operation (e.g., CUDA on a GPU to for image classification).

6. Which programming languages / tools were used during development and operation? (select all that apply)

	Used during development of AUC(s)	Used during operation of AUC(s)
C	☐	☐
C++	☐	☐
CUDA®	☐	☐
Java™	☐	☐
MATLAB®	☐	☐
Prolog	☐	☐
Python™	☐	☐
R	☐	☐
TensorFlow	☐	☐

Other Programming Language (please specify)

7. Was the code for the algorithms implementing autonomous functionality during the development phase of the AUC (e.g., training of DNN) developed:

- ☐ NA
- ☐ From scratch
- ☐ Using existing code library
- ☐ Using customized existing code library
- ☐ Using existing software (commercial or open source)

If you have worked on more than one project, please focus on your most representative project.

* 2. Have you used Model-Based Software Engineering (MBSwE) in your projects?

- ☐ No
- ☐ Yes

If you have worked on more than one project, please focus on your most representative project.

If you have worked on more than one project, please focus on your most representative project.

* 3. Have you used Model-Based Software Engineering (MBSwE) in your projects?

- ☐ No
- ☐ Yes

If you have worked on more than one project, please focus on your most representative project.

Application Domain and Areas

2

4

8. Was the code for the autonomous functionality used in deployed code developed

- ☐ NA
- ☐ From scratch
- ☐ Using existing code library
- ☐ Using customized existing code library
- ☐ Using existing software

9. Was special hardware used for any autonomous code during development?

- ☐ NA
- ☐ No
- ☐ Yes (please specify)

10. Was special hardware used for any autonomous code during operation?

- ☐ NA
- ☐ No
- ☐ Yes (please specify)

11. Were any computations executed "in the cloud" (or on remote server)?

- ☐ NA
- ☐ No
- ☐ Only during design (e.g., DNN training)
- ☐ Only during operation
- ☐ Both during design and operation

Survey on Model-Based Software Engineering / Autonomous Systems

NUMBER OF AUTONOMOUS COMPONENTS

5

We define "Autonomous Component" (AUC) as a component or subsystem that provides autonomous capabilities or supports autonomous operations.

Typical example may include:

- vision-based navigation for self-driving cars,
- system health management and prognostics for autonomous UAS flight,
- UAS flight management systems that can operate the UAS during its mission without human intervention,
- components of a deep-space mission, which perform complex operations without direct or remote operator control.

* 12. Please specify the number of Autonomous Components in your system. For each Autonomous Component, one page with several questions will follow.

- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7
- ☐ 8
- ☐ 9
- ☐ 10

Survey on Model-Based Software Engineering / Autonomous Systems

AUTONOMY DETAILS

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

13. Please enter the name of the AUC.

6

14. Was the AUC developed using Model-based Software Engineering?

- ☐ No
- ☐ Yes

15. Which Model-Based Software tools / development environments did you use for this AUC? (select all that apply)

- ☐ NA
- ☐ MagicDraw®
- ☐ Matlab® / Simulink®
- ☐ MatrixX®
- ☐ Rational Rose®
- ☐ Rational Rhapsody®
- ☐ Other (please specify)

Level of Autonomy

The level of autonomy of an AUC is defined based on the degree to which the Human Operator/User is in charge. Please select the adequate level of autonomy for each of the following individual tasks of AUC:

- information acquisition (monitor)
- information analysis (analyze)
- decision and action selection (decide)
- action implementation (act)

16. Please select the level of autonomy for each task

	The human is primary and the computer is secondary, used only as a tool for assistance	The computer operates with human interaction	The computer operates independently of the human; the human has limited capabilities	Computer has full autonomy, human out of the loop
Information acquisition (monitor)	☐	☐	☐	☐
Information analysis (analyze)	☐	☐	☐	☐
Decision and action selection (decide)	☐	☐	☐	☐
Action implementation (act)	☐	☐	☐	☐

Additional comments (if any):

17. Successful AUC can require complex algorithms and modeling paradigms that originate from different research areas. (select all that apply)

- ☐ Rule-based: decisions of autonomous component(s) were implemented as program logic or using explicit rules
- ☐ Machine-learning (off-line): the autonomous component is trained during development only (e.g., DNN training); learning does not take place during operation
- ☐ Machine-learning (on-line): training and/or adaptation also takes place during system operation
- ☐ Planning systems/languages: for specification and execution of action sequences
- ☐ Symbolic or discrete search algorithms: search for a solution in a discrete space (e.g., in chess programs)
- ☐ Statistical and filtering methods: for example Monte Carlo based methods, Kalman filters, particle filters
- ☐ Other (please specify)

18. How were the requirements specified? (select all that apply)

- ☐ Same format as for the non-autonomous parts
- ☐ Natural language
- ☐ Specific formalisms and notations
- ☐ Other (please specify)

7

8

19. To what level of detail were requirements for autonomous parts specified compared to non autonomous parts?

- ☐ NA
- ☐ Lower level
- ☐ Same level of detail as the non-autonomous parts
- ☐ Higher level

Please explain

Challenges

20. Were there any difficulties due to the following:

	NA	None	Slight difficulties	Moderate difficulties	Major difficulties
System complexity	☐	☐	☐	☐	☐
Non-deterministic algorithms	☐	☐	☐	☐	☐
High level of environment uncertainty	☐	☐	☐	☐	☐
Achieving desired level of autonomy	☐	☐	☐	☐	☐
Human-computer interaction	☐	☐	☐	☐	☐
Lack of human's trust in the autonomous system	☐	☐	☐	☐	☐
Other (please specify)	<div></div>				

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

Survey on Model-Based Software Engineering / Autonomous Systems

AUTONOMY DETAILS

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

9

21. Please enter the name of the AUC.

22. Was the AUC developed using Model-based Software Engineering?

- ☐ No
- ☐ Yes

23. Which Model-Based Software tools / development environments did you use for this AUC? (select all that apply)

- ☐ NA
- ☐ MagicDraw®
- ☐ Matlab® / Simulink®
- ☐ MatrixX®
- ☐ Rational Rose®
- ☐ Rational Rhapsody®
- ☐ Other (please specify)

Level of Autonomy

The level of autonomy of an AUC is defined based on the degree to which the Human Operator/User is in charge. Please select the adequate level of autonomy for each of the following individual tasks of AUC:

- information acquisition (monitor)
- information analysis (analyze)
- decision and action selection (decide)
- action implementation (act)

24. Please select the level of autonomy for each task

	The human is primary and the computer is secondary, used only as a tool for assistance	The computer operates with human interaction	The computer operates independently of the human, the human has limited capabilities	Computer has full autonomy, human out of the loop
Information acquisition (monitor)	☐	☐	☐	☐
Information analysis (analyze)	☐	☐	☐	☐
Decision and action selection (decide)	☐	☐	☐	☐
Action implementation (act)	☐	☐	☐	☐
Additional comments (if any):	<div></div>			

25. Successful AUC can require complex algorithms and modeling paradigms that originate from different research areas. (select all that apply)

- ☐ **Rule-based:** decisions of autonomous component(s) were implemented as program logic or using explicit rules
- ☐ **Machine-learning (off-line):** the autonomous component is trained during development only (e.g., DNN training); learning does not take place during operation
- ☐ **Machine-learning (on-line):** training and/or adaptation also takes place during system operation
- ☐ **Planning systems/languages:** for specification and execution of action sequences
- ☐ **Symbolic or discrete search algorithms:** search for a solution in a discrete space (e.g., in chess programs)
- ☐ **Statistical and filtering methods:** for example Monte Carlo based methods, Kalman filters, particle filters
- ☐ Other (please specify)

26. How were the requirements specified? (select all that apply)

- ☐ Same format as for the non-autonomous parts
- ☐ Natural language
- ☐ Specific formalisms and notations
- ☐ Other (please specify)

11

21. Please enter the name of the AUC.

22. Was the AUC developed using Model-based Software Engineering?

- ☐ No
- ☐ Yes

23. Which Model-Based Software tools / development environments did you use for this AUC? (select all that apply)

- ☐ NA
- ☐ MagicDraw®
- ☐ Matlab® / Simulink®
- ☐ MatrixX®
- ☐ Rational Rose®
- ☐ Rational Rhapsody®
- ☐ Other (please specify)

27. To what level of detail were requirements for autonomous parts specified compared to non autonomous parts?

- ☐ NA
- ☐ Lower level
- ☐ Same level of detail as the non-autonomous parts
- ☐ Higher level

Please explain

Challenges

28. Were there any difficulties due to the following:

	NA	None	Slight difficulties	Moderate difficulties	Major difficulties
System complexity	☐	☐	☐	☐	☐
Non-deterministic algorithms	☐	☐	☐	☐	☐
High level of environment uncertainty	☐	☐	☐	☐	☐
Achieving desired level of autonomy	☐	☐	☐	☐	☐
Human-computer interaction	☐	☐	☐	☐	☐
Lack of human's trust in the autonomous system	☐	☐	☐	☐	☐
Other (please specify)	<div></div>				

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

Survey on Model-Based Software Engineering / Autonomous Systems

AUTONOMY DETAILS

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

10

12

29. Please enter the name of the AUC.

30. Was the AUC developed using Model-based Software Engineering?

- ☐ No
☐ Yes

31. Which Model-Based Software tools / development environments did you use for this AUC? (select all that apply)

- ☐ NA
☐ MagicDraw®
☐ Matlab® / Simulink®
☐ MatrixX®
☐ Rational Rose®
☐ Rational Rhapsody®
☐ Other (please specify)

Level of Autonomy

The level of autonomy of an AUC is defined based on the degree to which the Human Operator/User is in charge. Please select the adequate level of autonomy for each of the following individual tasks of AUC:

- information acquisition (monitor)
- information analysis (analyze)
- decision and action selection (decide)
- action implementation (act)

13

32. Please select the level of autonomy for each task

	The human is primary and the computer is secondary, used only as a tool for assistance	The computer operates with human interaction	The computer operates independently of the human; the human has limited capabilities	Computer has full autonomy, human out of the loop
Information acquisition (monitor)	☐	☐	☐	☐
Information analysis (analyze)	☐	☐	☐	☐
Decision and action selection (decide)	☐	☐	☐	☐
Action implementation (act)	☐	☐	☐	☐

Additional comments (if any):

33. Successful AUC can require complex algorithms and modeling paradigms that originate from different research areas. (select all that apply)

- ☐ **Rule-based:** decisions of autonomous component(s) were implemented as program logic or using explicit rules
☐ **Machine-learning (off-line):** the autonomous component is trained during development only (e.g., DNN training); learning does not take place during operation
☐ **Machine-learning (on-line):** training and/or adaptation also takes place during system operation
☐ **Planning systems/languages:** for specification and execution of action sequences
☐ **Symbolic or discrete search algorithms:** search for a solution in a discrete space (e.g., in chess programs)
☐ **Statistical and filtering methods:** for example Monte Carlo based methods, Kalman filters, particle filters
☐ Other (please specify)

34. How were the requirements specified? (select all that apply)

- ☐ Same format as for the non-autonomous parts
☐ Natural language
☐ Specific formalisms and notations
☐ Other (please specify)

14

35. To what level of detail were requirements for autonomous parts specified compared to non autonomous parts?

- ☐ NA
☐ Lower level
☐ Same level of detail as the non-autonomous parts
☐ Higher level

Please explain

Challenges

36. Were there any difficulties due to the following:

	NA	None	Slight difficulties	Moderate difficulties	Major difficulties
System complexity	☐	☐	☐	☐	☐
Non-deterministic algorithms	☐	☐	☐	☐	☐
High level of environment uncertainty	☐	☐	☐	☐	☐
Achieving desired level of autonomy	☐	☐	☐	☐	☐
Human-computer interaction	☐	☐	☐	☐	☐
Lack of human's trust in the autonomous system	☐	☐	☐	☐	☐

Other (please specify)

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

Survey on Model-Based Software Engineering / Autonomous Systems

AUTONOMY DETAILS

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

15

37. Please enter the name of the AUC.

38. Was the AUC developed using Model-based Software Engineering?

- ☐ No
☐ Yes

39. Which Model-Based Software tools / development environments did you use for this AUC? (select all that apply)

- ☐ NA
☐ MagicDraw®
☐ Matlab® / Simulink®
☐ MatrixX®
☐ Rational Rose®
☐ Rational Rhapsody®
☐ Other (please specify)

Level of Autonomy

The level of autonomy of an AUC is defined based on the degree to which the Human Operator/User is in charge. Please select the adequate level of autonomy for each of the following individual tasks of AUC:

- information acquisition (monitor)
- information analysis (analyze)
- decision and action selection (decide)
- action implementation (act)

16

40. Please select the level of autonomy for each task

	The human is primary and the computer is secondary, used only as a tool for assistance	The computer operates with human interaction	The computer operates independently of the human, the human has limited capabilities	Computer has full autonomy, human out of the loop
Information acquisition (monitor)	☐	☐	☐	☐
Information analysis (analyze)	☐	☐	☐	☐
Decision and action selection (decide)	☐	☐	☐	☐
Action implementation (act)	☐	☐	☐	☐

Additional comments (if any):

41. Successful AUC can require complex algorithms and modeling paradigms that originate from different research areas. (select all that apply)

- ☐ Rule-based: decisions of autonomous component(s) were implemented as program logic or using explicit rules
- ☐ Machine-learning (off-line): the autonomous component is trained during development only (e.g., DNN training); learning does not take place during operation
- ☐ Machine-learning (on-line): training and/or adaptation also takes place during system operation
- ☐ Planning systems/languages: for specification and execution of action sequences
- ☐ Symbolic or discrete search algorithms: search for a solution in a discrete space (e.g., in chess programs)
- ☐ Statistical and filtering methods: for example Monte Carlo based methods, Kalman filters, particle filters
- ☐ Other (please specify)

42. How were the requirements specified? (select all that apply)

- ☐ Same format as for the non-autonomous parts
- ☐ Natural language
- ☐ Specific formalisms and notations
- ☐ Other (please specify)

17

45. Please enter the name of the AUC.

46. Was the AUC developed using Model-based Software Engineering?

- ☐ No
- ☐ Yes

47. Which Model-Based Software tools / development environments did you use for this AUC? (select all that apply)

- ☐ NA
- ☐ MagicDraw®
- ☐ Matlab® / Simulink®
- ☐ MatrixX®
- ☐ Rational Rose®
- ☐ Rational Rhapsody®
- ☐ Other (please specify)

Level of Autonomy

The level of autonomy of an AUC is defined based on the degree to which the Human Operator/User is in charge. Please select the adequate level of autonomy for each of the following individual tasks of AUC:

- information acquisition (monitor)
- information analysis (analyze)
- decision and action selection (decide)
- action implementation (act)

19

43. To what level of detail were requirements for autonomous parts specified compared to non autonomous parts?

- ☐ NA
- ☐ Lower level
- ☐ Same level of detail as the non-autonomous parts
- ☐ Higher level

Please explain

Challenges

44. Were there any difficulties due to the following:

	NA	None	Slight difficulties	Moderate difficulties	Major difficulties
System complexity	☐	☐	☐	☐	☐
Non-deterministic algorithms	☐	☐	☐	☐	☐
High level of environment uncertainty	☐	☐	☐	☐	☐
Achieving desired level of autonomy	☐	☐	☐	☐	☐
Human-computer interaction	☐	☐	☐	☐	☐
Lack of human's trust in the autonomous system	☐	☐	☐	☐	☐

Other (please specify)

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

Survey on Model-Based Software Engineering / Autonomous Systems

AUTONOMY DETAILS

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

18

48. Please select the level of autonomy for each task

	The human is primary and the computer is secondary, used only as a tool for assistance	The computer operates with human interaction	The computer operates independently of the human, the human has limited capabilities	Computer has full autonomy, human out of the loop
Information acquisition (monitor)	☐	☐	☐	☐
Information analysis (analyze)	☐	☐	☐	☐
Decision and action selection (decide)	☐	☐	☐	☐
Action implementation (act)	☐	☐	☐	☐

Additional comments (if any):

49. Successful AUC can require complex algorithms and modeling paradigms that originate from different research areas. (select all that apply)

- ☐ Rule-based: decisions of autonomous component(s) were implemented as program logic or using explicit rules
- ☐ Machine-learning (off-line): the autonomous component is trained during development only (e.g., DNN training); learning does not take place during operation
- ☐ Machine-learning (on-line): training and/or adaptation also takes place during system operation
- ☐ Planning systems/languages: for specification and execution of action sequences
- ☐ Symbolic or discrete search algorithms: search for a solution in a discrete space (e.g., in chess programs)
- ☐ Statistical and filtering methods: for example Monte Carlo based methods, Kalman filters, particle filters
- ☐ Other (please specify)

50. How were the requirements specified? (select all that apply)

- ☐ Same format as for the non-autonomous parts
- ☐ Natural language
- ☐ Specific formalisms and notations
- ☐ Other (please specify)

20

51. To what level of detail were requirements for autonomous parts specified compared to non autonomous parts?

- ☐ NA
- ☐ Lower level
- ☐ Same level of detail as the non-autonomous parts
- ☐ Higher level

Please explain

Challenges

52. Were there any difficulties due to the following:

	NA	None	Slight difficulties	Moderate difficulties	Major difficulties
System complexity	☐	☐	☐	☐	☐
Non-deterministic algorithms	☐	☐	☐	☐	☐
High level of environment uncertainty	☐	☐	☐	☐	☐
Achieving desired level of autonomy	☐	☐	☐	☐	☐
Human-computer interaction	☐	☐	☐	☐	☐
Lack of human's trust in the autonomous system	☐	☐	☐	☐	☐

Other (please specify)

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

Survey on Model-Based Software Engineering / Autonomous Systems

AUTONOMY DETAILS

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

21

53. Please enter the name of the AUC.

54. Was the AUC developed using Model-based Software Engineering?

- ☐ No
- ☐ Yes

55. Which Model-Based Software tools / development environments did you use for this AUC? (select all that apply)

- ☐ NA
- ☐ MagicDraw®
- ☐ Matlab® / Simulink®
- ☐ MatrixX®
- ☐ Rational Rose®
- ☐ Rational Rhapsody®
- ☐ Other (please specify)

Level of Autonomy

The level of autonomy of an AUC is defined based on the degree to which the Human Operator/User is in charge. Please select the adequate level of autonomy for each of the following individual tasks of AUC:

- information acquisition (monitor)
- information analysis (analyze)
- decision and action selection (decide)
- action implementation (act)

22

56. Please select the level of autonomy for each task

	The human is primary and the computer is secondary, used only as a tool for assistance	The computer operates with human interaction	The computer operates independently of the human, the human has limited capabilities	Computer has full autonomy, human out of the loop
Information acquisition (monitor)	☐	☐	☐	☐
Information analysis (analyze)	☐	☐	☐	☐
Decision and action selection (decide)	☐	☐	☐	☐
Action implementation (act)	☐	☐	☐	☐

Additional comments (if any):

57. Successful AUC can require complex algorithms and modeling paradigms that originate from different research areas. (select all that apply)

- ☐ **Rule-based:** decisions of autonomous component(s) were implemented as program logic or using explicit rules
- ☐ **Machine-learning (off-line):** the autonomous component is trained during development only (e.g., DNN training); learning does not take place during operation
- ☐ **Machine-learning (on-line):** training and/or adaptation also takes place during system operation
- ☐ **Planning systems/languages:** for specification and execution of action sequences
- ☐ **Symbolic or discrete search algorithms:** search for a solution in a discrete space (e.g., in chess programs)
- ☐ **Statistical and filtering methods:** for example Monte Carlo based methods, Kalman filters, particle filters
- ☐ Other (please specify)

58. How were the requirements specified? (select all that apply)

- ☐ Same format as for the non-autonomous parts
- ☐ Natural language
- ☐ Specific formalisms and notations
- ☐ Other (please specify)

23

59. To what level of detail were requirements for autonomous parts specified compared to non autonomous parts?

- ☐ NA
- ☐ Lower level
- ☐ Same level of detail as the non-autonomous parts
- ☐ Higher level

Please explain

Challenges

60. Were there any difficulties due to the following:

	NA	None	Slight difficulties	Moderate difficulties	Major difficulties
System complexity	☐	☐	☐	☐	☐
Non-deterministic algorithms	☐	☐	☐	☐	☐
High level of environment uncertainty	☐	☐	☐	☐	☐
Achieving desired level of autonomy	☐	☐	☐	☐	☐
Human-computer interaction	☐	☐	☐	☐	☐
Lack of human's trust in the autonomous system	☐	☐	☐	☐	☐

Other (please specify)

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

Survey on Model-Based Software Engineering / Autonomous Systems

AUTONOMY DETAILS

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

24

61. Please enter the name of the AUC.

62. Was the AUC developed using Model-based Software Engineering?

- ☐ No
☐ Yes

63. Which Model-Based Software tools / development environments did you use for this AUC? (select all that apply)

- ☐ NA
☐ MagicDraw®
☐ Matlab® / Simulink®
☐ MatrixX®
☐ Rational Rose®
☐ Rational Rhapsody®
☐ Other (please specify)

Level of Autonomy

The level of autonomy of an AUC is defined based on the degree to which the Human Operator/User is in charge. Please select the adequate level of autonomy for each of the following individual tasks of AUC:

- information acquisition (monitor)
- information analysis (analyze)
- decision and action selection (decide)
- action implementation (act)

25

67. To what level of detail were requirements for autonomous parts specified compared to non autonomous parts?

- ☐ NA
☐ Lower level
☐ Same level of detail as the non-autonomous parts
☐ Higher level

Please explain

Challenges

68. Were there any difficulties due to the following:

	NA	None	Slight difficulties	Moderate difficulties	Major difficulties
System complexity	☐	☐	☐	☐	☐
Non-deterministic algorithms	☐	☐	☐	☐	☐
High level of environment uncertainty	☐	☐	☐	☐	☐
Achieving desired level of autonomy	☐	☐	☐	☐	☐
Human-computer interaction	☐	☐	☐	☐	☐
Lack of human's trust in the autonomous system	☐	☐	☐	☐	☐

Other (please specify)

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

Survey on Model-Based Software Engineering / Autonomous Systems

AUTONOMY DETAILS

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

27

64. Please select the level of autonomy for each task

	The human is primary and the computer is secondary, used only as a tool for assistance	The computer operates with human interaction	The computer operates independently of the human; the human has limited capabilities	Computer has full autonomy, human out of the loop
Information acquisition (monitor)	☐	☐	☐	☐
Information analysis (analyze)	☐	☐	☐	☐
Decision and action selection (decide)	☐	☐	☐	☐
Action implementation (act)	☐	☐	☐	☐

Additional comments (if any):

65. Successful AUC can require complex algorithms and modeling paradigms that originate from different research areas. (select all that apply)

- ☐ Rule-based: decisions of autonomous component(s) were implemented as program logic or using explicit rules
☐ Machine-learning (off-line): the autonomous component is trained during development only (e.g., DNN training); learning does not take place during operation
☐ Machine-learning (on-line): training and/or adaptation also takes place during system operation
☐ Planning systems/languages: for specification and execution of action sequences
☐ Symbolic or discrete search algorithms: search for a solution in a discrete space (e.g., in chess programs)
☐ Statistical and filtering methods: for example Monte Carlo based methods, Kalman filters, particle filters
☐ Other (please specify)

66. How were the requirements specified? (select all that apply)

- ☐ Same format as for the non-autonomous parts
☐ Natural language
☐ Specific formalisms and notations
☐ Other (please specify)

26

69. Please enter the name of the AUC.

70. Was the AUC developed using Model-based Software Engineering?

- ☐ No
☐ Yes

71. Which Model-Based Software tools / development environments did you use for this AUC? (select all that apply)

- ☐ NA
☐ MagicDraw®
☐ Matlab® / Simulink®
☐ MatrixX®
☐ Rational Rose®
☐ Rational Rhapsody®
☐ Other (please specify)

Level of Autonomy

The level of autonomy of an AUC is defined based on the degree to which the Human Operator/User is in charge. Please select the adequate level of autonomy for each of the following individual tasks of AUC:

- information acquisition (monitor)
- information analysis (analyze)
- decision and action selection (decide)
- action implementation (act)

28

72. Please select the level of autonomy for each task

	The human is primary and the computer is secondary, used only as a tool for assistance	The computer operates with human interaction	The computer operates independently of the human, the human has limited capabilities	Computer has full autonomy, human out of the loop
Information acquisition (monitor)	☐	☐	☐	☐
Information analysis (analyze)	☐	☐	☐	☐
Decision and action selection (decide)	☐	☐	☐	☐
Action implementation (act)	☐	☐	☐	☐

Additional comments (if any):

73. Successful AUC can require complex algorithms and modeling paradigms that originate from different research areas. (select all that apply)

- ☐ Rule-based: decisions of autonomous component(s) were implemented as program logic or using explicit rules
- ☐ Machine-learning (off-line): the autonomous component is trained during development only (e.g., DNN training); learning does not take place during operation
- ☐ Machine-learning (on-line): training and/or adaptation also takes place during system operation
- ☐ Planning systems/languages: for specification and execution of action sequences
- ☐ Symbolic or discrete search algorithms: search for a solution in a discrete space (e.g., in chess programs)
- ☐ Statistical and filtering methods: for example Monte Carlo based methods, Kalman filters, particle filters
- ☐ Other (please specify)

74. How were the requirements specified? (select all that apply)

- ☐ Same format as for the non-autonomous parts
- ☐ Natural language
- ☐ Specific formalisms and notations
- ☐ Other (please specify)

29

77. Please enter the name of the AUC.

78. Was the AUC developed using Model-based Software Engineering?

- ☐ No
- ☐ Yes

79. Which Model-Based Software tools / development environments did you use for this AUC? (select all that apply)

- ☐ NA
- ☐ MagicDraw®
- ☐ Matlab® / Simulink®
- ☐ MatrixX®
- ☐ Rational Rose®
- ☐ Rational Rhapsody®
- ☐ Other (please specify)

Level of Autonomy

The level of autonomy of an AUC is defined based on the degree to which the Human Operator/User is in charge. Please select the adequate level of autonomy for each of the following individual tasks of AUC:

- information acquisition (monitor)
- information analysis (analyze)
- decision and action selection (decide)
- action implementation (act)

31

75. To what level of detail were requirements for autonomous parts specified compared to non autonomous parts?

- ☐ NA
- ☐ Lower level
- ☐ Same level of detail as the non-autonomous parts
- ☐ Higher level

Please explain

Challenges

76. Were there any difficulties due to the following:

	NA	None	Slight difficulties	Moderate difficulties	Major difficulties
System complexity	☐	☐	☐	☐	☐
Non-deterministic algorithms	☐	☐	☐	☐	☐
High level of environment uncertainty	☐	☐	☐	☐	☐
Achieving desired level of autonomy	☐	☐	☐	☐	☐
Human-computer interaction	☐	☐	☐	☐	☐
Lack of human's trust in the autonomous system	☐	☐	☐	☐	☐

Other (please specify)

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

Survey on Model-Based Software Engineering / Autonomous Systems

AUTONOMY DETAILS

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

30

80. Please select the level of autonomy for each task

	The human is primary and the computer is secondary, used only as a tool for assistance	The computer operates with human interaction	The computer operates independently of the human, the human has limited capabilities	Computer has full autonomy, human out of the loop
Information acquisition (monitor)	☐	☐	☐	☐
Information analysis (analyze)	☐	☐	☐	☐
Decision and action selection (decide)	☐	☐	☐	☐
Action implementation (act)	☐	☐	☐	☐

Additional comments (if any):

81. Successful AUC can require complex algorithms and modeling paradigms that originate from different research areas. (select all that apply)

- ☐ Rule-based: decisions of autonomous component(s) were implemented as program logic or using explicit rules
- ☐ Machine-learning (off-line): the autonomous component is trained during development only (e.g., DNN training); learning does not take place during operation
- ☐ Machine-learning (on-line): training and/or adaptation also takes place during system operation
- ☐ Planning systems/languages: for specification and execution of action sequences
- ☐ Symbolic or discrete search algorithms: search for a solution in a discrete space (e.g., in chess programs)
- ☐ Statistical and filtering methods: for example Monte Carlo based methods, Kalman filters, particle filters
- ☐ Other (please specify)

82. How were the requirements specified? (select all that apply)

- ☐ Same format as for the non-autonomous parts
- ☐ Natural language
- ☐ Specific formalisms and notations
- ☐ Other (please specify)

32

83. To what level of detail were requirements for autonomous parts specified compared to non autonomous parts?

- ☐ NA
- ☐ Lower level
- ☐ Same level of detail as the non-autonomous parts
- ☐ Higher level

Please explain

Challenges

84. Were there any difficulties due to the following:

	NA	None	Slight difficulties	Moderate difficulties	Major difficulties
System complexity	☐	☐	☐	☐	☐
Non-deterministic algorithms	☐	☐	☐	☐	☐
High level of environment uncertainty	☐	☐	☐	☐	☐
Achieving desired level of autonomy	☐	☐	☐	☐	☐
Human-computer interaction	☐	☐	☐	☐	☐
Lack of human's trust in the autonomous system	☐	☐	☐	☐	☐
Other (please specify)	<div></div>				

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

Survey on Model-Based Software Engineering / Autonomous Systems

AUTONOMY DETAILS

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

33

85. Please enter the name of the AUC.

86. Was the AUC developed using Model-based Software Engineering?

- ☐ No
- ☐ Yes

87. Which Model-Based Software tools / development environments did you use for this AUC? (select all that apply)

- ☐ NA
- ☐ MagicDraw®
- ☐ Matlab® / Simulink®
- ☐ MatrixX®
- ☐ Rational Rose®
- ☐ Rational Rhapsody®
- ☐ Other (please specify)

Level of Autonomy

The level of autonomy of an AUC is defined based on the degree to which the Human Operator/User is in charge. Please select the adequate level of autonomy for each of the following individual tasks of AUC:

- information acquisition (monitor)
- information analysis (analyze)
- decision and action selection (decide)
- action implementation (act)

34

88. Please select the level of autonomy for each task

	The human is primary and the computer is secondary, used only as a tool for assistance	The computer operates with human interaction	The computer operates independently of the human, the human has limited capabilities	Computer has full autonomy, human out of the loop
Information acquisition (monitor)	☐	☐	☐	☐
Information analysis (analyze)	☐	☐	☐	☐
Decision and action selection (decide)	☐	☐	☐	☐
Action implementation (act)	☐	☐	☐	☐
Additional comments (if any):	<div></div>			

89. Successful AUC can require complex algorithms and modeling paradigms that originate from different research areas. (select all that apply)

- ☐ **Rule-based:** decisions of autonomous component(s) were implemented as program logic or using explicit rules
- ☐ **Machine-learning (off-line):** the autonomous component is trained during development only (e.g., DNN training); learning does not take place during operation
- ☐ **Machine-learning (on-line):** training and/or adaptation also takes place during system operation
- ☐ **Planning systems/languages:** for specification and execution of action sequences
- ☐ **Symbolic or discrete search algorithms:** search for a solution in a discrete space (e.g., in chess programs)
- ☐ **Statistical and filtering methods:** for example Monte Carlo based methods, Kalman filters, particle filters
- ☐ Other (please specify)

90. How were the requirements specified? (select all that apply)

- ☐ Same format as for the non-autonomous parts
- ☐ Natural language
- ☐ Specific formalisms and notations
- ☐ Other (please specify)

35

91. To what level of detail were requirements for autonomous parts specified compared to non autonomous parts?

- ☐ NA
- ☐ Lower level
- ☐ Same level of detail as the non-autonomous parts
- ☐ Higher level

Please explain

Challenges

92. Were there any difficulties due to the following:

	NA	None	Slight difficulties	Moderate difficulties	Major difficulties
System complexity	☐	☐	☐	☐	☐
Non-deterministic algorithms	☐	☐	☐	☐	☐
High level of environment uncertainty	☐	☐	☐	☐	☐
Achieving desired level of autonomy	☐	☐	☐	☐	☐
Human-computer interaction	☐	☐	☐	☐	☐
Lack of human's trust in the autonomous system	☐	☐	☐	☐	☐
Other (please specify)	<div></div>				

If you selected more than one Autonomous Component (AUC), this page will be repeated for each AUC.

Survey on Model-Based Software Engineering / Autonomous Systems

REUSE

36

* 93. Was software, data, or other artifacts reused in your project(s)?

- ☐ No
☐ Yes

Survey on Model-Based Software Engineering / Autonomous Systems

REUSE DETAILS

94. Were artifacts from other / previous projects reused in AUC and to what extent?

	NA	Less than 30%	30% or more
Model Reuse -- system	☐	☐	☐
Model Reuse -- software models	☐	☐	☐
Code Reuse	☐	☐	☐
Data-Reuse -- training data (for offline training)	☐	☐	☐
Data-Reuse -- trained machine learning model	☐	☐	☐
Data-Reuse -- telemetry data, log files	☐	☐	☐
Reuse -- other artifacts (please describe)	☐	☐	☐

Other (please specify)

95. Were artifacts from other / previous projects reused for on AUC and to what extent?

	NA	Less than 30%	30% or more
Model Reuse -- system	☐	☐	☐
Model Reuse -- software models	☐	☐	☐
Code Reuse	☐	☐	☐
Data-Reuse -- training data (for offline training)	☐	☐	☐
Data-Reuse -- trained machine learning model	☐	☐	☐
Data-Reuse -- telemetry data, log files	☐	☐	☐
Reuse -- other artifacts (please describe)	☐	☐	☐

Other (please specify)

37

96. Were specific preparatory activities for future reuse done during the development phase of the project?

- ☐ No
☐ Yes (please specify what activities)

97. Were there any negative aspects of reuse and to what extent:

	NA	None	Slight	Moderate	Major
Increased start-up cost	☐	☐	☐	☐	☐
Additional cost due to reuse sustainability	☐	☐	☐	☐	☐
Hindering new ideas because of reuse	☐	☐	☐	☐	☐
Added complexity because of reuse	☐	☐	☐	☐	☐

Other (please specify)

98. Were there any difficulties due to the following and to what extent:

	NA	None	Slight	Moderate	Major
Lack of information for reused software	☐	☐	☐	☐	☐
Integrating the reused parts into the development environment	☐	☐	☐	☐	☐
Uncertain operational conditions / environment	☐	☐	☐	☐	☐
Verifying and validating reused software	☐	☐	☐	☐	☐
Lack of reuse process	☐	☐	☐	☐	☐
Lack of planning for reuse in advance	☐	☐	☐	☐	☐
Lack of reuse awareness / training / education	☐	☐	☐	☐	☐

Other (please specify)

38

99. Benefits

How would you estimate the benefits of Software Reuse on your project, in terms of Cost, Quality, and Productivity?

	NA	Decreased	About the same	Increased
Cost	☐	☐	☐	☐
Quality	☐	☐	☐	☐
Productivity	☐	☐	☐	☐

Survey on Model-Based Software Engineering / Autonomous Systems

PROCESSES AND STANDARDS

100. Which life-cycle model was used on your project?

- ☐ Agile
☐ Case-based development (with a-priori knowledge)
☐ Rapid application development
☐ Rational Unified Model
☐ Spiral
☐ Waterfall
☐ Other (please specify)

101. Which Modeling Standards did you use / enforce?

- ☐ None
☐ MathWorks® Automotive Advisory Board (MAAB)
☐ Systems Modeling Language (SysML)
☐ Unified Modeling Language (UML™)
☐ Other (please specify)

39

102. Which Coding Standards did you use / enforce?

- ☐ None
☐ Motor Industry Software Reliability Association (MISRA®)
☐ NASA coding standard
☐ JPL coding standard
☐ Other (please specify)

103. Were the lifecycle model and processes specifically adapted for the development of the AUC?

- ☐ No
☐ Yes (please specify)

104. Did the system (or major portions) go through a certification process?

- ☐ No
☐ Yes (please explain, including applicable standards and approach)

105. Were the AUCs part of the certified system?

- ☐ No
☐ Yes (please explain)

106. Additional comments on certification, if any.

Survey on Model-Based Software Engineering / Autonomous Systems

40

VERIFICATION AND VALIDATION (V&V)

107. Did you Verify & Validate (select all that apply):

- ☐ Correctness
- ☐ Performance (with respect to time/iness)
- ☐ Robustness
- ☐ Safety
- ☐ Security

Verification & Validation of Models

108. How did you perform Verification & Validation of the models? (select all that apply)

- ☐ NA
- ☐ Did not perform V&V
- ☐ Manual model inspection / reviews
- ☐ Testing
- ☐ Simulation
- ☐ Automated model analysis
- ☐ Other (please specify)

109. Did you use tools for the Verification & Validation (V&V) of the models?

- ☐ NA
- ☐ No
- ☐ Yes

41

110. If you answered yes to the question above, what tools did you use for Verification & Validation of the models? (select all that apply)

- ☐ CoCoSim
- ☐ Design Verifier™
- ☐ Model Advisor
- ☐ Reactis®
- ☐ T-VEC®
- ☐ Other (please specify which tools you used)

Verification & Validation of Autonomy

111. Did you specifically Verify & Validate autonomous functionality?

- ☐ NA
- ☐ No
- ☐ Yes (please specify)

42

112. How did you perform Verification & Validation of AUCs during developmental phase? (select all that apply)

- ☐ NA
- ☐ Automated model analysis
- ☐ Fault injection
- ☐ Manual model/code reviews or inspections
- ☐ Model checking (formal method)
- ☐ Monte Carlo or statistical testing
- ☐ Simulation-based testing
- ☐ Static code analysis
- ☐ Unit and integration testing
- ☐ Other (please specify)

113. How was correct runtime behavior of AUCs monitored / assured? (select all that apply)

- ☐ NA
- ☐ Checking of pre-conditions for mitigation and recovery procedures
- ☐ Contingency architecture for detection and mitigation (e.g., ARTIS/F-3269)
- ☐ Cross-checking that commanded actions meet intended post-conditions
- ☐ Execution of consistency checks
- ☐ Monitoring of requirements
- ☐ Monitoring of variable values and ranges
- ☐ Other (please specify)

43

114. How were violations and errors in the autonomous components handled? (select all that apply)

- ☐ NA
- ☐ Automatic autonomous action (e.g., land immediately, return home)
- ☐ Autonomous response and adaptation of mission (autonomous contingency management)
- ☐ Control handed over to human operator
- ☐ Control taken away from human operator
- ☐ Like a system failure
- ☐ Notification to human operator
- ☐ Other (please specify)

Verification & Validation of Reuse

115. Did you specifically Verify & Validate reused artifacts?

- ☐ NA
- ☐ No
- ☐ Yes (please specify and compare to V&V of newly developed artifacts)

Survey on Model-Based Software Engineering / Autonomous Systems

BUGS

116. Were there any bugs specific to model-based approach?

- ☐ NA
- ☐ No
- ☐ Yes (please specify)

44

117. Were there any bugs specific to autonomous functionality?

- ☐ NA
- ☐ No
- ☐ Yes (please specify)

118. Were there any bugs specific to reuse of software artifacts?

- ☐ NA
- ☐ No
- ☐ Yes (please specify)

Survey on Model-Based Software Engineering / Autonomous Systems

ROLE

119. How would you characterize your role in this project? (select all that apply)

- ☐ Design
- ☐ Model development
- ☐ Certification
- ☐ Programming
- ☐ Project management
- ☐ Research
- ☐ SW and System Integration
- ☐ Testing / QA / Verification & Validation
- ☐ Tool provider or developer
- ☐ Other (please specify)

45

Survey on Model-Based Software Engineering / Autonomous Systems

MULTIPLE PROJECT DISCUSSION

120. If you were involved in more than one project which included autonomous functionality, please describe the differences and comment on your experiences.

Survey on Model-Based Software Engineering / Autonomous Systems

THANK YOU

Thank you for participating in our survey. Your contribution will help evaluating the experiences with Autonomous systems.

If you would like to receive the survey results, please provide your email address below. Alternatively, you may request the survey results by sending an email to SurveyAutonomousSystems@gmail.com

121. This survey is anonymous. Providing any information below is entirely optional.

Name	<input type="text"/>
Institution	<input type="text"/>
Email Address	<input type="text"/>

122. Country

46