Automation Processes for Efficient Verification of Complex Systems: an Empirical Case Study

Corresponding Author
Rune André Haugen
University of South-Eastern Norway
rune.a.haugen@usn.no

Nils-Olav Skeie
University of South-Eastern Norway
nils-olav.skeie@usn.no

Gerrit Muller
University of South-Eastern Norway
gerrit.muller@usn.no

Abstract. This paper investigated the effect of automation processes in an industrial company engineering complex cyber-physical systems. The authors used industry-as-laboratory as research method, exploring an ongoing development project. The automation efforts focused on four areas, being 1) test setup, 2) test execution, 3) test result analysis, and 4) documentation. All four areas showed promising results on increased effectiveness and/or efficiency. Especially automation of test result analysis will help the industrial company, KONGSBERG, reduce their main bottleneck in the test process, as well as reduce the risk of costly project delays. An automated system integration test process, facilitating iterative regression testing, will leverage the efficiency of the verification test process.

Keywords — Automation, Complex Systems, Integration, Testing, Verification

Introduction

This paper contains data and evaluation of the automation processes conducted in a development project in KONGSBERG over a five-year period (2018-2023) for a complex cyber-physical system. The authors have extracted test data from the company test database, giving us high confidence in this data.

Background

The company has to execute projects faster and in parallel to cope with its future market situation. Availability of human resources is a challenge. The company needs to increase test coverage for all projects compared to today's situation. Further, improve effectiveness and efficiency of test result analyzing to cope with the amount of tests in due time. Test result documentation must be on time for all relevant testing, on different formats (pickle, markdown, and JSON) for further automatic processing, and on a pdf format for easy sharing both internally and externally. The company needs to document all test results, not only the mandatory part for customer delivery.

KONGSBERG is a large company cluster spanning multiple industry fields with its headquarter in Kongsberg and many other smaller sites located around the world. The company was founded in 1814 and has approximately 13 000 employees as of 2024. The KONGSBERG group consists of four companies:[1]

- Kongsberg Defence and Aerospace
- Kongsberg Digital
- Kongsberg Discovery
- Kongsberg Maritime

The large development project under investigation in this study started about thirty years ago and has undergone several upgrades. From the first commercial survey, KONGSBERG has been at the forefront of autonomy, with the system being capable of uninterrupted operations for its entire mission.[1]

Research questions and design

We pose the following research question (RQ) and sub-research questions (SRQ) for this case study:

RQ: How can automation of test processes improve the verification of complex systems?

SRQ1: How can automation of test setup improve the verification of complex systems?

SRQ2: How can automation of test execution improve the verification of complex systems?

SRQ3: How can automation of test result analysis improve the verification of complex systems?

SRQ4: How can automation of test document generation improve the verification of complex systems?

SRQ5: Can the use of automated test processes for complex systems make a positive impact on usage of subject matter expert hours?

We have illustrated our research design in Figure 1. Haugen and Mansouri[2] cover the first part, problem exploration, and Haugen et al.[3] cover the second part, literature review. Our aim in this paper was to do a gap analysis between different project milestones before and after implementation of automation measures within four areas of interest, being 1) test setup, 2) test execution, 3) test result analysis, and 4) documentation.

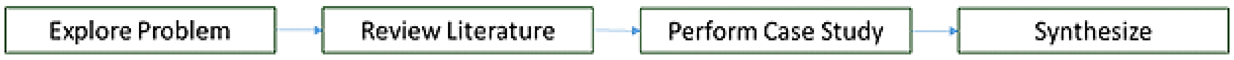

Figure 1: Research design

Contributions of the paper

The authors explored automated testing procedures in an industrial case to improve verification of complex systems. Many organizations recognize manual testing as a bottleneck. The authors advocate combining strengths of human expertise and machine capabilities to optimize the effectiveness (what) and efficiency (how) of the test process.

The proposed methodology should be applicable in general for organizations developing complex systems and having a potential for improving the human-machine task balance. The potential value of utilizing this approach is to increase the probability of on-time project delivery and reduce project delays.

Literature review

We searched relevant literature for potential benefits and concerns related to automation processes. This section is based on the literature review we conducted on detection of emergent behavior.[3] We include some highlights in the following paragraphs.

Automating test execution and test result analysis can remove these two bottlenecks in the test process, leveraging the efficiency.[2,4]

Utilizing metrics like Measures of Effectiveness (MoE), Measures of Performance (MoP), and Technical Performance Measures (TPM) can give us information about the system that could become valuable feedback to system top level design.[5] Raman suggests using Machine Learning techniques when monitoring MoE and MoP to look for changes that give or could give raise to undesired system behavior.[6-11]

Several research works claim to reduce the test time used for manual testing by more than 90% by automating the test procedures that are suitable for automation.[12,13] Tools for automation with built in simulators become essential for verifying and validating behavior logic in a reasonable amount of time.[14] However, a key challenge is the need for abstractions of the micro and macro levels, which is difficult to achieve in an automated manner. Hence, most approaches rely on a post-mortem observation of the simulation by a system expert.[15]

Case study

This is a generalized case for a complex cyber-physical system, based on a real KONGSBERG product, as the real case we investigated contains sensitive data and that cannot be shared outside the case company.

Test setup: We looked at automating today's manual test setup for system integration testing based on orthogonal arrays (OAs).[16] OAs are a mechanism to increase test coverage without proportionally increasing the number of tests. Since we want to test more of the parameter space, we need automation. A previous case study in the company showed promising results using OAs.[17] The researchers used the Minitab tool[18] to design an experiment based on the Taguchi method.[16] Then, the company transferred the experiment data in a machine-readable format to a simulator.

Test execution: We investigated automating today's manual tester actions, including triggers for scenario actions. The company made manual actions machine readable, see Table I, and implemented support for this in one of the company's test arenas.

Table I: Example of automatic testing procedure

Action Number	Action Phase	Action Name	Action Trigger
1	Preparation	Power on	Power available
2	Preparation	Select position	Initialization finished
3	Preparation	Select profile	Position selected
4	Preparation	Start	Start command
5	Operation	Stop	Stop command

Test result analysis: We researched automating today's manual analysis. The company scripted checks that Subject Matter Experts (SMEs) perform manually when investigating log files for different data to compare against defined acceptance criteria, making these checks machine executable. In this case study, we focused on a sub-set of these scripts. This sub-set consisted of thirty out of the total one hundred system requirements. Then, the company transformed the analysis results using a Python[19] script making a format suitable for regression analysis in Minitab.

Documentation: We examined automating today's manual documentation process. The company both tested and used the company developed Highly Automated Document System (HADES). Analyzing results and generating reports are the functions of HADES. Figure 2 shows the complete test process in a data flow diagram where the artifacts are stored in a repository facilitating further testing, processing, analysis, and reporting.

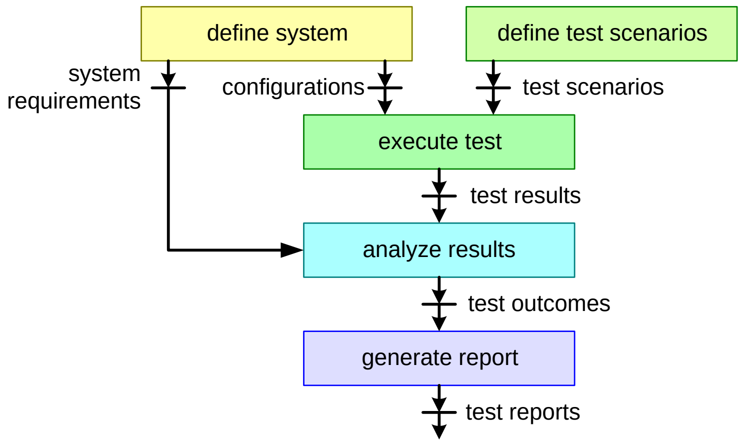

Figure 2: Test process

A test scenario and system configuration provides input to HADES for automatic creation of test descriptions. The test description is input to the simulator to execute tests. The test results from the simulator are stored in a test results database, which provides input to

HADES for automatic analysis of test results. HADES creates the test outcome based on test results and system requirements with acceptance criteria. HADES then uses the test outcome from the analysis process in automatic creation of test reports. Test outcomes are machine readable, while test reports are human readable.

Scope of research

The remainder of this paper is structured as follows: The section Methods presents methodology used for this research paper. The section Results provides a quantified comparison between two test campaigns; the first based on a manual test process and the second based on an automated test process. The section Cost estimate gives a short financial assessment in the form of SME hours. The section Discussion provides a synthesis on the results, addressing the research questions. Finally, the section Conclusion summarizes the paper, provides gained knowledge, defines limitations of the study, and proposes future research.

Methods

We used the industry-as-laboratory approach within the case company.[20,21] In the case company, one of the researchers had access to the system integration test group, relevant SMEs (testers and analysts), and historic test data. "The advantage with such an approach is the realism introduced into the research. Practical challenges in the theoretical frameworks can be hard to detect, unless one sees how it plays out in reality. One challenge emerging from such an approach is the potential of noise from company-specific problems that we cannot directly link to the research conducted."[4]

The digitalization process to automate earlier manual tasks included several steps of scripting. Python is used to develop script files for test setup transfer, test execution, test result analysis, and test documentation. The test input and test output are stored in separate distributed relational databases with the test input data under configuration management in a company developed system called Chaman. The test output is accessible from the company developed test web application. The reports created by HADES are stored in a product document management (PDM) system, named Enovia, from Dassault Systems.[22]

Results

The number of tests included approximately one hundred system level requirements with test as verification method. These tests, the sub-system level also used for sub-system verification. The authors have included one sub-system in this research to show data for one typical sub-system, not for all sub-systems. We present the results from two project test campaigns (the initial project and one later project update) in this section. System integration testing is the first step in system level testing, where the company through a trial and error approach ensures maturity before more formal testing. System testing is the second step in system level testing, where the company through a test plan ensures compliance to the system design. System interface testing is a part of system testing, focusing on testing input/output. System verification testing is the third step in system level testing, where the company, through a system verification plan ensures compliance to the system requirements.

Benchmark results

The benchmark results are based on manual tasks performed during a major project milestone.

Test setup: The company prioritized testing for verifying system requirements, based on the system verification plan. The test coverage of the system design was limited (10-30%) considering the parameter space of interest for the system. The time to set up testing was low (2-4 hours), only selecting and prioritizing among the described system verification tests for system integration testing.

Test execution: See Table II for an overview of the number of tests executed- and the duration of the system test campaign. The system test campaign we used is the functional testing part of the Final Design Review (FDR). The company had set the *system tests finished* milestone to February and the *verification tests finished* milestone to April. However, the company ended up doing these serial test phases more or less in parallel to finish on time, working double shifts to do so.

Table II: Number of tests in a test campaign

Test period	System level testing			Sub-system level testing		
	Integration testing	System testing	Verification testing	Integration testing	System testing	Verification testing
January	208	5	0	15	0	0
February	406	139	1	0	0	0
March	178	100	85	14	15	53
April	37	0	20	116	0	30
May	6	0	0	3	1	18
June	9	0	0	0	0	0
Total:	*844*	*244*	*106*	*148*	*16*	*101*

The project allowed some sub-system verification testing to finish in May, as they still had time before the FDR milestone meeting in June. The project continued testing in June to further increase their confidence in the delivered test reports, in case of any discussions during the FDR milestone meeting.

A search into the company test result database revealed a maximum of 546 tests executed in one month in a hectic period (February 2018), ref. Table II. The testers used two to four test arenas of two types; System Integration Lab closed loop (SILc) and System Integration Lab open loop (SILo), dependent on their availability. SILc includes close to all HW, facilitating full operational testing. SILo includes less HW than SILc, only facilitating preparation for operation type of testing. Another search in the company test result database in a less hectic period (October 2019 to March 2020), revealed an average of fifty tests executed per month.[4]

Table III: Errors revealed in different testing

Error type	**Integration testing**	**System testing**	**Verification testing**
Configuration	0	10	0
Correlation	8	5	0
Functional	0	37	6
Interface	11	68	0
Prioritization	9	0	0
Synchronization	12	0	0
Test arena	0	36	0
Test scenario	0	60	14
Undesired deviations	15	0	0
Undesired events	11	0	0
Undesired oscillations	9	3	0
Unexpected idle states	0	23	0
Unexpected results	0	58	4
Unexpected status	0	60	1

Error type	**Integration testing**	**System testing**	**Verification testing**
Total:	*75*	*360*	*25*

The testing was based on manual operations, being resource demanding. One full test in SILc involves two operators sitting in the system lab for thirty minutes including pre- and post-work, without the opportunity to do other types of work. One full test in SILo involves the same as for SILc, but only about half the time is necessary.

Test result analysis: SMEs reported that they on average used 39% of their time for analysis work in hectic periods, but only 4% of their time in non-hectic periods.[4] This analysis effort only covers 9% of tests executed, not considering potential overlap, leaving 91% of test data not used for analysis.[4] 8% of analysis conducted did reveal an issue.[4]

Analysts reported varying times to detect an error, but they estimated an average of two hours. Further, they reported the time to find the causal factor to vary between weeks and months.[4] One analyst conducting analysis for fifteen hours per week during hectic periods is then likely to detect seven to eight errors per week. Table III shows errors and numbers that a selection of seven analysts reported during a six months integration testing period (October 2019 to March 2020).[4] In addition to these 75 reported errors, we found 385 other errors from a project team status list used by the project team to follow-up issues during the system testing period (February and March 2018) and a verification testing period (March and April 2018).

Documentation: The documentation part of this case study consisted of two test reports, one at system level, and one at sub-system level. SMEs created both documents manually. The documents held compliance status and rationale for all hundred system requirements with test as verification method. SMEs roughly estimated the time spent creating these two test reports to one week (37.5 hours) each.

Comparison results

The comparison results are based on automated tasks performed during three project update increments, which we can see as regression testing.[23]

Test setup: The company has not tried automating the requirements-based test setup but has tested automating test setup based on OAs. OAs and automation are independent choices, which both will improve the company test process and the product of them even more so. We tried using one OA to set up testing to increase the test coverage of the parameter space of interest for the system (70-90%) and did so by selecting a suitable OA from the Taguchi framework.[16] For our example case, we used a L25 OA to fit our need. We wanted to test six parameters at five different levels each, which gave 15 625 combinations (entire system parameter space). However, the L25 OA provided us with sufficient test coverage in only twenty-five tests (parameter space of interest). To set up a test execution according to this matrix for our company test arena, we exported this OA to the simulator and generated the twenty-five test cases from the L25 OA setting values in a simulator input file (simulated environment data).

Test execution: The test coverage included thirty system level test cases of the one hundred existing, plus nineteen more system level interface test cases. The company down selected manually to avoid similar type of testing, only testing distinctly different functionality.

Table IV: Number of tests in a regression test campaign

Test period	**System level testing**		**Sub-system level testing**	
	System testing	**Interface testing**	**System testing**	**Interface testing**
March	11	19	13	0
April	7	0	2	0
June	12	0	0	0
Total:	*30*	*19*	*15*	*0*

See Table IV for an overview of tests executed and the period of the regression test campaign. The company still performed the testing in the test arenas allocated for verification, but they also did some testing in an alternate test arena built for automatic testing. One test in the alternate test arena required one person to start the test, but allowed this operator to do other tasks in the office while the test arena was executing the test based on an automated procedure (machine-readable version of the earlier manual list of actions).

Test result analysis: The company automated the information flow from the test results database to the HADES system, removing this earlier manual step and avoiding new test results not being analyzed. The company used HADES to analyze the test results, which they could do for a specified set of test cases or for all test cases being part of the test report. The time to conduct one analysis was about five seconds. Analyzing thirty interface and system level test cases took three minutes. Table V shows forty-three errors that the company detected during this regression test period.

Out of sixty-four system and sub-system level tests, they discovered forty-three errors. This gives a detection rate of 0.67 errors per test. In comparison, they discovered 385 errors during 467 system- and verification tests in the benchmark test campaign, giving a detection rate of 0.82 errors per test. The regression test campaign based on automated tasks has a detection rate of 0.67 errors per test when the company could expect zero, which is a significant contribution for the automation effort. We found 25% of the errors to originate from new functionality/interfaces (11 errors in 21 tests) while the remaining 75% came from updated functionality/interfaces (32 errors in 43 tests) previously verified.

Documentation: The company used HADES to create three test reports at system level, covering compliance and rationale for forty-nine interface and system level test cases. The time HADES needed for creating these three test reports were approximately 1.5 minutes per report.

Table V: Errors revealed in different regression testing

Error type	**System testing**	**Interface testing**
Correlation	1	0
Functional	7	0
Interface	1	2
Test arena	7	0
Test scenario	24	1
Total:	*40*	*3*

Cost estimate

The company will have to invest money to implement and maintain the proposed automation processes. The cost of implementing and maintaining the different automation processes is dependent on the number of SME hours (Sh) needed and the cost of one SME hour (C_h). The total number of SME hours (Sh) for test result analysis depends on the SME hours needed for initial scripting (Sh_{is}), script maintenance (Sh_{sm}), and manual analysis (Sh_{ma}), see Equation 1.

$$Cost = C_h * Sh, where\ Sh = Sh_{is} + Sh_{sm} + Sh_{ma} \qquad (1)$$

The benchmark results in our case study revealed a poor test result analysis coverage in the system integration phase. These results correspond well with survey results from one company in KONGSBERG, where SMEs stated they only use 4% of their time on test result analysis.[4] Further, the test result analysis coverage in the system- and verification test phases were high, which also corresponds with the survey, stating SMEs use 39% of their time on test result analysis.[4]

SMEs in the case company use about 12% of their time to create scripts for automatic analysis and about 6% of their time on average to maintain these scripts through different project increments. We have extracted these numbers from the case company's time management system. Furthermore, we assess SMEs only use about 4% of their time on test result analysis in the system- and verification phases when automatic test result analysis is in place. The latter, we base on an estimated 90% test analysis coverage in the system integration phase based on automatic analysis compared to only 9% coverage previously reported.[4]

Based on the above numbers, we clearly see the benefit of the company investing in the automation process for test result analysis. For establishment of the automatic test results analysis framework, we have a workload ratio of 3:1 compared to performing the analysis manually. During maintenance of this framework through different project increments, we have a workload ratio of 3:2 compared to otherwise manual efforts. Using this framework in the system testing and verification testing phases, we have a workload ratio of 1:10 compared to manual operations. Treating these three ratios equally, we get a total ratio of 7:13 (46%) in favor of the automation process. We found the ratio to be 26:35 (26%) for the case company project update. This assessment is further strengthened by the cost of detecting errors late in a project development being higher than detecting errors early. A case study from Carnegie Mellon University claims that the cost of detecting errors in the system- and verification phase is 2-3 times higher than in the previous system integration phase.[24] Also, the systems engineering handbook claims that the cost of extracting defects in late project testing is 500-1000 times higher than the cost in earlier testing.[25]

Discussion

We answer the research questions in light of the results and the cost estimate.

SRQ1: How can automation of test setup improve the verification of complex systems?

The company did a minor test related to test setup where we used a L25 Taguchi OA, which showed promising results in supplying test data input to twenty-five tests without the need to set up the twenty-five tests manually. The company can set up testing effectively (test the right things) and efficiently (test the right way) through this approach, combining statistical based experiment design and automation.

SRQ2: How can automation of test execution improve the verification of complex systems?

The company did a small-scale testing effort looking at how the company could run tests automatically in a suitable test arena, which showed good results. The test arena conducted the tests while the operator could do other types of work anywhere. The company can benefit from a potential increased test coverage using the system scenario sequencer (automated test arena), resolving the bottleneck with availability of test arenas in the system test lab.

SRQ3: How can automation of test result analysis improve the verification of complex systems?

The company did a large-scale test of automated test result analysis scripts, which has proven to be both beneficial and challenging. The company spent a significant amount of time creating these scripts, but the potential gain is ever growing as we iterate and do regression testing. However, we have seen a weakness in maintainability of these scripts as we apply changes to the system interfaces. The company must update the scripts manually for them to still work as intended. The company has a huge potential to increase the analysis coverage through use of scripts and HADES, resolving the main bottleneck of available SMEs for manual analysis. We see the importance of regression testing in the comparison test campaign, where 75% of the errors came from functionality already verified in the benchmark test campaign.

SRQ4: How can automation of test document generation improve the verification of complex systems?

The company has used HADES in its effort to automate analysis and documentation. In this case, we saw that the HADES system is helpful in significantly reducing the time for documentation related to testing. The company was able to produce both test descriptions and test reports during a one-month test campaign, which they would never have been able to do previously. Where HADES uses 1.5 minutes to generate a document automatically, a SME uses about 37.5 hours to do the same manually. The time to manually create one test description prior to testing and one test report after testing consumes two weeks, which only leaves two weeks for testing in a one-month test campaign. However, the review process is the same whether the documents are automatically or manually generated. Typically, a review process takes twelve days (two days to prepare the peer review, two days to conduct the peer review, one week to update the document, and one day to release the document through a formal review).

SRQ5: Can the use of automated test processes for complex systems make a positive impact on usage of subject matter expert hours?

The implementation of automated test processes and further maintenance of these will require a significant amount of SME hours initially and throughout the project development. However, the amount of SME hours used in the final stages of the project seems to be reduced more than the investment cost. Based on a previous survey in a company in KONGSBERG[4] and data from the case company's time management system, we clearly see the benefit of automation processes. The most critical automation process being the test result analysis, where we saw a 26% reduction of a typical SME position's usage of time on test result analysis tasks in a development project.

RQ: How can automation of test processes improve the verification of complex systems?

We see an increased system robustness through the different SRQ discussions above. All the four areas we have looked at regarding automation processes have proved to be beneficial for the overall product robustness. The company will be able to detect more of the inherent errors and undesired behaviors in a complex system, and do so earlier, by increasing the test and analysis coverage and reducing the time spent to perform these actions. The proposed semi-automated test process with a better human-machine task balance can reduce project delays for the company significantly. Based on prior knowledge from a previous case study,[17] the company is able to test a system more effectively through use of OAs matching the provided test input data to generate necessary test scenarios, contributing to earlier detection of errors.

Conclusion

This empirical case study serves as a "proof of concept" for automation processes when it comes to increased test coverage. We have been able to see promising results within all the four categories we have looked at. First, we can set up testing to ensure the desired test coverage in an effective and efficient way using OAs. E.g., the case company only needs to provide one input file to trigger twenty-five different tests as part of one OA, being the tests necessary for the company to extract the desired level of information. Second, we can execute testing efficiently through removal of human-in-the-loop. E.g., the case company can save fifteen to thirty minutes per test. Third, we can analyze significantly more test results during a given period through use of scripting. E.g., the case company can analyze one test in a few seconds compared to typically two hours. However, an interface update strategy would be beneficial to reduce the maintenance cost. Fourth, we can create test documentation in a fraction of the time by the use of an automated test documentation process. E.g., the case company can produce a test report in 1.5 minutes compared to one week. Fifth, the overall project cost could be significantly reduced even with investment and maintenance cost for automated test processes. E.g., a typical SME can reduce 26% of time spent on test result analysis by investing in automation processes and avoiding lengthy manual test result analysis tasks.

We assess these findings to be useful for the industry developing complex cyber-physical systems in general and for the case company in specific. However, this study is limited in the way that we have done one case study in one project in one company. The described techniques may have a bias toward this particular case. Further research is necessary to establish a best practice for human-machine task balance, substantiating the effectiveness and efficiency in different systems in various contexts.

Acknowledgements. We would like to acknowledge the Research Council of Norway for their funding, grant number 321830. Also, the KONGSBERG company for sharing relevant industry information through their SMEs and databases. Especially, Bjoern Victor Larsen and Oluf Tonning from KONGSBERG have contributed with their systems engineering- and industrial expertise. Finally, Elisabet Syverud at the University of South-Eastern Norway (USN) has provided guidance to make this paper part of a larger PhD project. Authors have no conflict of interest relevant to this article.

References

1. KONGSBERG. Accessed 14th October, 2022. www.kongsberg.com
2. Haugen RA, Mansouri M. Applying Systems Thinking to Frame and Explore a Test System for Product Verification; a Case Study in Large Defence Projects. Wiley; 2020:78-93.

3. Haugen RA, Skeie N-O, Muller G, Syverud E. Detecting emergence in engineered systems: A literature review and synthesis approach. *Systems Engineering*. 2023;26(4):463-481. doi:10.1002/sys.21660
4. Kjeldaas KA, Haugen RA, Syverud E. Challenges in Detecting Emergent Behavior in System Testing. Wiley; 2021:1211-1228.
5. Skreddernes O, Haugen RA, Haskins C. Coping with Verification in Complex Engineered Product Development. Wiley; 2023:482-502.
6. Raman R, Jeppu Y. An Approach for Formal Verification of Machine Learning based Complex Systems. 2019:544-559.
7. Raman R, Jeppu Y. Formal validation of emergent behavior in a machine learning based collision avoidance system. In: *SYSCON 2020 - 14th Annual IEEE International Systems Conference, Proceedings*. Institute of Electrical and Electronics Engineers Inc.; 2020:
8. Raman R, D'Souza M. Decision learning framework for architecture design decisions of complex systems and system-of-systems. Article. *Systems Engineering*. Nov 2019;22(6):538-560. doi:10.1002/sys.21517
9. Raman R, Jeppu Y. Does the Complex SoS Have Negative Emergent Behavior? Looking for Violations Formally. In: *15th Annual IEEE International Systems Conference, SysCon 2021 - Proceedings*. Institute of Electrical and Electronics Engineers Inc.; 2021:
10. Raman R, Gupta N, Jeppu Y. Framework for Formal Verification of Machine Learning Based Complex System-of-System. Wiley; 2021:
11. Murugesan A, Raman R. Reinforcement Learning for Emergent Behavior Evolution in Complex System-of-Systems. IARIA; 2021:
12. Enoiu E, Sundmark D, Causevic A, Pettersson P. A Comparative Study of Manual and Automated Testing for Industrial Control Software. ResearchGate; 2017:412-417.
13. Øvergaard A, Muller G. System Verification by Automatic Testing. Wiley; 2013:356-367.
14. Giammarco K. Practical modeling concepts for engineering emergence in systems of systems. 2017:1-6.
15. Szabo C, Teo Y. An integrated approach for the validation of emergence in component-based simulation models. 2012:2739-2749.
16. Roy RK. *A Primer on the Taguchi Method*. 2nd ed. Society of Manufacturing Engineers; 2010:304.
17. Haugen RA, Ghaderi A. Modelling and Simulation of Detection Rates of Emergent Behaviors in System Integration Test Regimes. Linköping Electronic Conference Proceedings; 2021:
18. Minitab. Minitab. Accessed 29th March, 2023. www.minitab.com
19. Python. Accessed 29th March, 2023. www.python.org
20. Five Years of Multi-Disciplinary Academic and Industrial Research; Lessons Learned. http://www.gaudisite.nl/.
21. Industry-as-Laboratory Applied in Practice: The Boderc Project. http://www.gaudisite.nl/.
22. Dassault. Accessed 10th October, 2023. www.3ds.com/
23. SEBoK. Robert Cloutier EiC, ed. *System Integration*. San Diego, CA: International Council on Systems Engineering (INCOSE); 2023. Accessed 20th November 2023. https://sebokwiki.org/w/index.php?title=System_Integration&oldid=67584

24. Feiler PH, Hanson J, de Niz D, Wrage L. *System Architecture Virtual Integration: An Industrial Case Study*. 2009. http://www.sei.cmu.edu
25. Incose, Wiley. *INCOSE Systems Engineering Handbook*. John Wiley & Sons, Incorporated; 2015.

Biography

Rune André Haugen is an industrial-PhD candidate at the University of South-Eastern Norway (USN). He was in service with the Royal Norwegian Air Force (RNoAF) from 1997 to 2003, including graduation from the RNoAF Officer Candidate School in Stavern (1999) and the RNoAF Academy in Trondheim (2001). He holds both a BSc (2006) and a MSc (2013) in Systems Engineering from USN. He has worked as a design engineer at FMC Kongsberg Subsea from 2006 to 2008 (3D modeling), and as a system engineer at Kongsberg Defence and Aerospace since 2008 (system design and system test).

Nils-Olav Skeie got his MSc in Cybernetics from Norwegian University of Science and Technology (NTNU) in 1985. He worked with system development within the computer, aviation and maritime industry for more than 20 years before receiving a PhD within machine learning in 2008 in a cooperation between NTNU and the University of South-Eastern Norway (USN). In 2006 he went back to the academia and has been teaching BSc, MSc and PhD students in software engineering and system engineering. He continued to work as a part time system architect for the maritime industry from 2008 to 2015. He became a professor of industrial machine learning at USN in 2020.

Gerrit Muller, originally from the Netherlands, received his MSc in physics from the University of Amsterdam in 1979. He worked from 1980 until 1997 at Philips Medical Systems as a system architect, followed by two years at ASML as a manager of systems engineering, returning to Philips (Research) in 1999. Since 2003 he has worked as a senior research fellow at the Embedded Systems Institute in Eindhoven, focusing on developing system architecture methods and the education of new system architects, receiving his PhD in 2004. In January 2008, he became a full professor of systems engineering at the University of South-Eastern Norway. He continues to work as a senior research fellow at the Embedded Systems Institute in Eindhoven in a part-time position. All information (System Architecture articles, course material, curriculum vitae) can be found at: Gaudí systems architecting http://www.gaudisite.nl/