2023 IEEE/ACM 5th International Workshop on Robotics Software Engineering (RoSE) | 979-8-3503-0184-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/RoSE59155.2023.00007

Survey on Robotic Systems Integration

Nadia Hammoudeh Garcia
Robot and Assistive Systems Department
Fraunhofer Institute for Manufacturing Engineering and Automation (IPA)
Nobelstr. 12, 70569 Stuttgart, Germany
nadia.hammoudeh.garcia@ipa.fraunhofer.de

Andreas Wortmann
Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW)
University of Stuttgart
Seidenstr. 36, 70174 Stuttgart, Germany
wortmann@isw.uni-stuttgart.de

***Abstract*—Software integration is central to successfully developing, deploying, and operating robotics applications. Yet, the particular integration process and its challenges are poorly understood. The continuous evolution of the robotics sector, incorporating constantly a growing number of technologies, makes the unification of processes increasingly complicated. Nevertheless, current research on robotics software integration largely focuses on specific integration activities instead of considering the overall activity as a process. To provide some insight into the state of robotics software integration, we drove a survey among researchers and practitioners in the field. In this survey, we inquired how robotics software integration is currently performed in order to identify similarities with traditional software development methodologies. Through this study, we discovered commonalities in the phases of the process and potential directions of a future research to address the current challenges in the area.**

***Index Terms*—robotics, integration, methodology, SDLC**

I. Introduction

It is said that "software is eating the world" [1], which is possible as creating software thrives on reuse - in form of classes, modules, components, frameworks, libraries, or containers. Hence, integrating existing software artifacts is central in the engineering of software-intensive systems. In robotics – as in automotive, industry 4.0, and other domains – software is the main driver of functionality and added value. Thus, software integration is central to successfully developing, deploying, and operating robotics applications.

In many established domains, various software development and integration methodologies have emerged to structure and guide software integration (such as the waterfall model, tailored V-Models, or specific implementations of agile methods). This also aims to reduce the cost of integration. International studies [15] report that for professional service robots operating in manufacturing and intra-logistics, the costs for system and software integration steps make up for about 45% of the final system costs, leaving only the remaining 55% for the development of new components. Moreover, these figures are even scaled up if we consider robotic platforms for commercial distribution, where, in addition, the follow-up activities of support and maintenance have to be considered.

Instead of the full-complete process, research in robotics largely focuses on the integration of novel hardware [4], [8], [24] or software [3], [5], [19], [29] solutions. In parallel, the continuous evolution of robotics across different sectors, which incorporates an ever-increasing number of technologies, makes the integration of robotics software components increasingly challenging. The main mission of an integrator is no longer just to adapt existing solutions by making modifications to their configurations. Rather, she faces the integration of new modules coming from different vendors, following different paradigms, and various implementation technologies. These modules must be integrated not only at the level of technical architecture but also at the level of functionality.

To shed light on the state of robotics software integration, we conducted a survey among researchers and practitioners in the field. With this survey, we aim to

1) Determine the similarities in the methodologies applied by diverse profiles of robotic software developers during the integration process;
2) Analyze whether it is possible to use already formalized software development methods for the integration of robots; and
3) Identify the challenges and the type of tools that can facilitate the application of a formal development process.

Therefore, we reached out to 118 researchers and practitioners in robotics software engineering. Through this study, we identified common patterns in how integrators perform the process and derived ideas on how to advance systems engineering in robotics. The insights gained from this survey will help researchers and practitioners in better addressing the challenges of robotics software integration and provide directions for future research to improve it further. All the raw results of our study are publicly available[1]. In the remainder, Sec. II presents the state of the art in software development processes and the existing issues on the subject of software integration. Afterward, Sec. III describes the methodology of our survey, and Sec. IV presents its results. Finally, Sec. V discusses observations, and Sec. VI concludes.

II. Related Work

To systematically develop, integrate, operate, and maintain robotics software, it is necessary to consider and support its entire Software Development Life Cycle (SDLC) [25].

[1]https://ipa-nhg.github.io/RoboticSystemIntegration/

979-8-3503-0184-7/23/$31.00 ©2023 IEEE
DOI 10.1109/RoSE59155.2023.00007

Therefore, the following sections outline software integration challenges and popular SDLC methods.

A. Software Integration Challenges

Integration is the process of connecting different subsystems (or components) into a single system in a way that this assembly provides an overarching functionality [11]. Research has identified different kinds of integration [28], including data integration, functional integration, presentation integration, portal integration, and process integration. In robotics, where the systems are designed to perform complex tasks with many different alternatives for the configuration and combination, functional integration is probably the most relevant one,

The main challenges of functional integration are [2]: (1) finding the appropriate integration level; (2) reconciling different specifications; (3) managing different implementation technologies; (4) deploying subsystems; and (5) interoperability issues. This is in line with other sources that deem major challenges inconsistent, misunderstood, or missing specifications [16], the requirement to involve rigorous testing, verification, and validation [9], and as well as the ongoing increasing complexity of robotics applications.

In the specific field of robotics, while performing the actual integration, unsurprisingly, also the main issues identified are related to the lack of common specifications at all the different levels. An empirical study [26] highlighted the following items: (1) missing standardisation of interfaces, (2) standardised flange information not available, (3) misleading compatibility specification, (4) missing property specification, (5) misleading incompatibility specification, and that (6) kits may decrease device reuse and sustainability.

A proposed methodology [27] predicts the effort and time required to complete a robot system integration by splitting the whole process into two phases, (1) high-level integration, how a software component is integrated with the development environment (i.e., how the actual software is installed and built), and (2) low-level integration, how a component is integrated to interact with the rest of the components. This methodology defines factors that influence the complexity the following: the compatibility of the APIs of the components, the language in which they are written, their performance in terms of time and resource consumption, the quality of the software (e.g., documentation, testing, ...), and the support and maintenance of the software stack.

B. Methodologies for Software Development Processes

1) Software Development Lifecycle - SDLC:

a) Waterfall: The traditional waterfall model [18] of software development assumes a linear-sequential life cycle model that consists of *Requirements Analysis*, *System Design*, *Implementation*, *Testing*, *Deployment*, and *Maintenance*. The model assumes comprehensive front-loading.

b) V-Model: The V-Model [17] splits the waterfall model into two legs of specification and development activities, which are followed by corresponding testing activities that begin on a very detailed level and complete with validation [6]. The activities of the first (development) leg are *Requirement Analysis*, *System Architecture*, *System Design*, *Software Architecture*, and *Implementation*. The activities of the second (testing) leg are *Unit Testing*, *Integration Testing*, *System Testing*, and *Acceptance Testing*. The traditional V-Model does not consider returning to previous activities.

c) Agile Methods: Agile methodologies [7] structure the development of software into different iterations in which activities are performed in a cyclic way.

As an extension of agile methods, the DevOps [14] methodology aims to accelerate the software development lifecycle and ensure continuous and high-quality delivery. Essentially, it includes the following 8 activities: *Plan*, *Code*, *Build*, *Test*, *Release*, *Deploy*, *Operate* and *Monitor*. Operations and monitoring then lead to insights that trigger changes to the software and enter the planning activity again.

2) Risk-driven process: Risk-driven process models are software development methodologies that focus on managing risks throughout the software development process. One example of a risk-driven process model is the Spiral model, it consists of the following phases: (1) Planning, where project goals, objectives, and constraints are defined, (2) Risk Analysis, the risks are prioritized based on their impact and likelihood, (3) Engineering, the software is designed, developed, and tested, and (4) Evaluation, the software is evaluated by the stakeholders.

3) Rapid application development: Rapid Application Development (RAD) is a software development methodology that prioritizes rapid prototyping and iterative development. Therefore it consists of *Requirements Planning*, *Prototype*, *Iterative Development*, *Testing* and *Deployment*.

None of these takes the specific challenges of robotics into account.

III. Survey Methodology

A. Objective and Scope

We aim to collect information from robotics practitioners on how robotic systems are currently integrated, focusing on the process followed and how it could be unified. In detail, our study[2] aims to answer the question of how can established SDLC methodologies can be (partially) applied to facilitate robotics integration.

To investigate this main topic, we identify the following research questions:

- **RQ1**. How can established software engineering methodologies be (partially) applied to robotics integration?
- **RQ2**. Which are the major challenges in robotics software integration?
- **RQ3**. Which new tools would be most necessary to support integration in applying established software integration processes?

In addition to these three aspects, and thanks to the answers obtained, we were able to establish patterns in the current state

[2]Closed questionnaire: https://forms.gle/nhtf5WSN5jScVn7NA.

of the activities followed for the integration of robotic systems, our conclusions in this matter have already been released [13].

The introduction of the survey defined the target audience of this experiment as "people with experience in any phase of the integration of various components in robotics, from the description of the solution to the system implementation and its maintenance". This definition includes roboticists from technical and organizational backgrounds, as well as practitioners and researchers. Especially, the survey is not constrained to a specific domain within robotics or to a specific type of technology and it was open from 2022-01-21 to 2022-02-15.

B. Questionnaire Design

The design of our questionnaire was guided by the recommendations [23] for the empirical validation of software engineering.

It consists of the following five sections:

1) **Responder profile:** this section is used to cluster the profile of the participants and help us in the analysis of the data.
2) **Application of established SDLC processes:** in this section, we present common activities of different established processes (cf. Sec. II) for professional software development and ask about their usability for the integration of robotic systems.
3) **Challenges and improvements**: to be able to address the problems, we need to, first, identify the actual challenges and possible improvements to mitigate them. That is what we intend to find out in this section.
4) **Existing tools to support integration:** this section investigates existing tools that support the integrator and aims to determine to what extent they are useful.
5) **Desired tools to support integration:** this section inquires the respondents' opinions on what kind of new tools they would like to have to facilitate their work on robotics software integration, as well as at which activities of the process, they are most essential.

The survey was made available by using Google Forms and could be anonymously answered. The results were not shared with the participants to avoid biases.

Overall, it requires only 10-15 minutes to be completed and consists of 20 questions, out of which 15 are closed captions, for which the participants could select (sometimes multiple answers) from predefined answers or using a 6-point Likert scale. For 10 of them, a free-text "Other" option enables participants to expand on their answers or give additional insights.

Out of the 20 questions, only 5 are mandatory as we approach the topic very broadly and we do not expect participants to have experience in all integration activities. To obtain accurate answers, the only part where the participant is being guided is the one where we intend to evaluate the applicability of existing software development methodologies to the systems integration phase, e.g., part 2 headed as "Application of established SDLC processes". This part, composed of 6 questions, starts with an explanatory text about a particular methodology and asks the respondent for opinions about it, then leaves an open text part to either argue why it is not valid or to propose another more suitable methodology.

C. Distribution

To disseminate the questionnaire we used the following distribution channels: (1) Personalized emails to our network of professionals contacts, not only people we have collaborated with in the past but also acquaintances at events; (2) Social media (twitter, LinkedIn) using profiles related to the robotics domain; (3) Disseminating the survey in the ROS community[3];(4) Promotion through robotics mailing lists, such as euRobotics and robotics-worldwide: (5) Utilizing GitHub data mining techniques to obtain contact information for top developers within open-source robotics communities, including ROS, OROCOS, and Yarp. Following this approach, we sent a total of 180 individual emails with participation invitations to identified experts.

IV. Findings

We have received a total of 118 responses, a responsiveness rate sufficient for surveys in software engineering [23]. For these responses, we performed the data analysis quantitatively for answers where participants could select from multiple options and report the findings in form of diagrams in the following. For free-text answers, the analysis is qualitative, analyzing each answer separately and classifying them by content. To consider the potential deviation of the data by participant profile, we also analyze the responses in two different dimensions: industrial and academic practitioners.

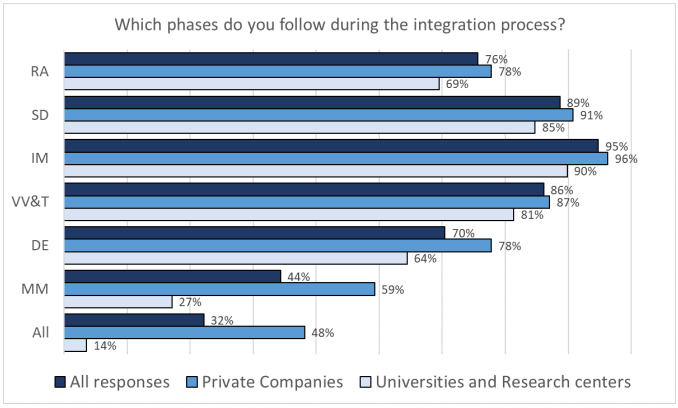
A. Company and responder profile

The first section of our survey attempted to classify the respondents by type of profile. Of the 118 surveyed 45.3% of them work in the private sector, 28.2% at the universities, and 22.2% in research centers. The outstanding 4.3% are people working for the government, retired, or freelancers. We found that 38.5% of the participants work for a company with more than 1000 employees, while 29.1% are part of small companies (under 50 employees). The rest is divided into medium size companies or institutions, 19.7% for companies between 50 and 250 people and 12.8% between 250-1000 members.

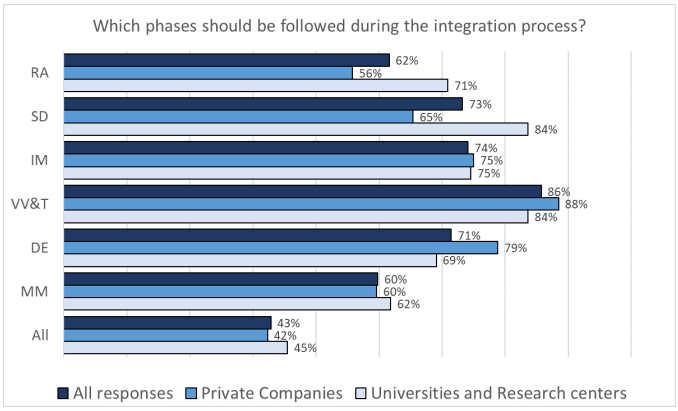
By inquiring about the main activity of the company, we found out that 56.4% of the participants work for a research company (and 49.6% exclusively in robotics), 31.6% of software provider companies, 23.9% of companies whose main activity is the robot application integration, and 16.2% of robot manufacturers.

We disseminated our survey to reach out to anyone who is involved in the robotic systems integration process but the channels we used for its distribution are mainly for professionals with a technical profile rather than a project management profile. It is therefore not surprising that the

[3]https://discourse.ros.org/t/survey-on-robot-software-integration-process/23935



(a) Distribution of the answers to the question about the activities **followed** during the integration process.



(b) Distribution of the answers to the question about the activities that **should be followed** during the integration process.

Fig. 1: Traditional process evaluation results (RA = Requirement Analysis; SD = System Design; IM =Implementation; VV& T = Validation, Verification, Testing; DE = Deployment; MM = Monitoring and Maintenance)

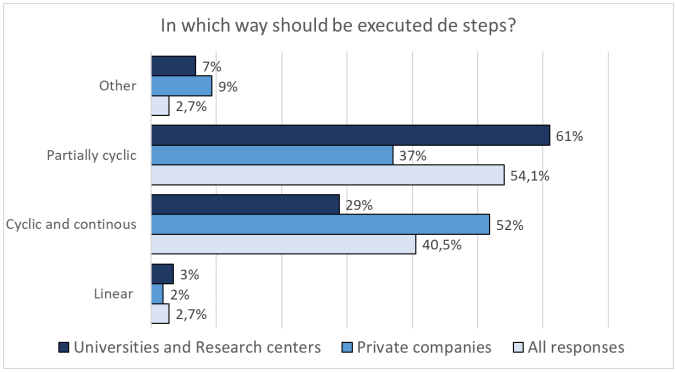


Fig. 2: Overview of the answers to the question about the way to execute the activities.

71.8% of the responders perform tasks of software developers, followed by the architecture designers (44.4%), system integrators (36.8%), project managers (31.6%), and product owners (20.5%). Among others, our interviewees have added activities like teaching at university, scientific research, or company direction. Due to the audience of our survey, mostly software developers, and the lack of a predefined integrator profile in robotics, only one person has indicated that his/her single and exclusive role in projects is system integrator.

The profile combination mentioned most often, is that of a person who is in charge of the software development *and* the design of the system architecture *and* its integration.

B. Traditional process to develop software and its application for the software integration phase

To evaluate the use of traditional software development methods, we analyzed in detail the existing SDLC methodologies II-B, looked for common activities, and describe them from a software integration perspective. The result of this study is presented to the survey participants as the header of this section:

- **Requirements Analysis**: Activity focused on the identification of the conditions that must satisfy the resulted application.
- **Design**: Phase to comprehend the solution by examining the user requirements, defining the architecture, and selecting the components to be wired. During this activity, the architecture has to be defined as well as the list of modules that will compose the final system.
- **Code Implementation**: Activity to write the code that allows the system to be launched (i.e., configure the modules and implement the code for the interaction.
- **Validation, Verification and Testing**: Validation confirms that the system is correct. While verification confirms that the system is correct to fulfill the product requirements. Testing is the process of evaluating the implementation under test through validation and verification.
- **Deployment**: Activity that comprises all the activities necessary to make the software available for use in the application. Among others, we can consider deployment tasks: setup, installation, configuration of the hardware-software pair, activation and deactivation activities.
- **Monitor and Maintenance**: Actions aimed to ensure compliance with the product requirements over time. All the activities related to the upgrade, of both, the system software and its dependencies need to be considered in this activity, as well as efforts to ensure that the end-user can operate the final system.

The first question in this section is about which of the activities presented are **currently being followed** while performing the software integration of a robot system. The ranking of the most and least completed steps(Fig. 1a) coincides for both profiles being this (1) Code Implementation, (2) Design, (3) Validation, Verification, and Testing, (4) Requirements Analysis, (5) Deployment, and (6) Monitor and Maintenance. Nonetheless, we see

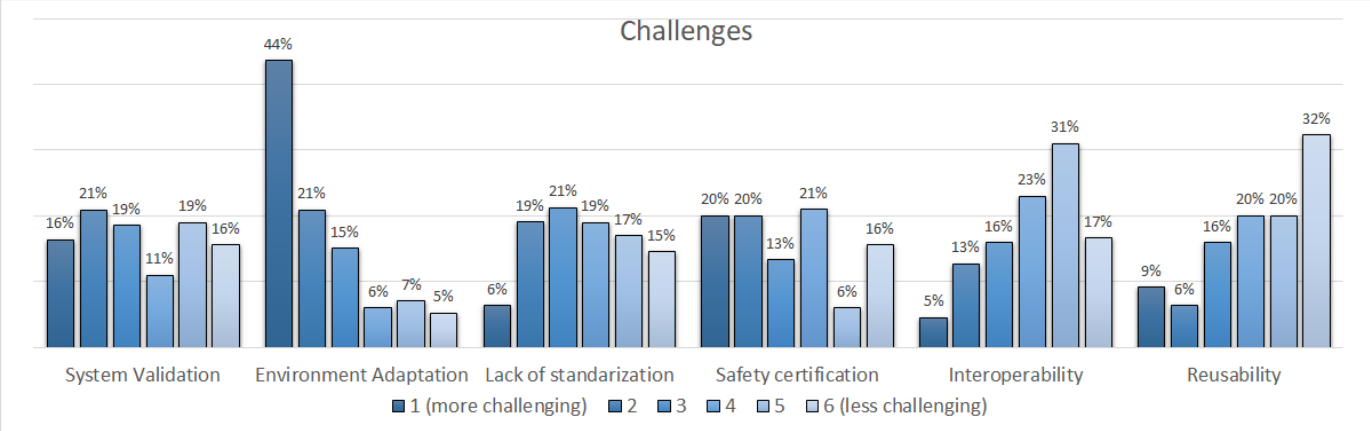


Fig. 3: Results of the question about the major challenges during the software integration process. The respondents ranked all the proposed challenges from highest(1) to lowest(5).

a considerable difference in the percentage of developers following all the steps, while in the industry this data is 48.1% for academia this figure is reduced to only 13.6%. *Monitoring and Maintenance* of the system and *Requirements Analysis* are the root of this disturbance, both are skipped by a large segment of the academic sector.

Finding 1: While in industry 48.1% say they complete all the activities we proposed, in research this percentage is significantly reduced to only 13.6%.

However, when asked about the activities that **should be followed** during the integration process, we find different answers, again we can distinguish between the preferences of the industry and the academic sector (Fig. 1b). Here trends change, we identify that the academy sees a much greater need to follow all the activities (45.5% support this), and substantially increases the need for a *Monitoring and Maintenance* phase. They see the need for improvement in how they do the integration and encourage the idea of accomplishing additional activities. Another curious fact, is if we look at the *Requirements Analysis* phase, we see that while research would like to give it more importance, for the private industry it is the opposite, they believe that it is not a necessary phase to be done by the integrator. This can probably be explained by the fact that they prefer another role within the team to be in charge of the requirements analysis, and that the integrator has a technical profile and focuses more on the design and implementation steps.

To propose any other activity in the process we leave a free text question. In the answers, we find suggestions such as analysis of existing solutions, system prototyping phase, simulation, documentation, demonstration, and customer training linked to support. Also, some responders proposed the known V-Model as a solution applicable to the robotics software integration process. To conclude this section, we inquired in which mode the activities should be executed and give three default options: (1) Purely linear, i.e., one activity after the other; (2) Cyclic and continuous, i.e., after the last activity you start again with the first activity (DevOps style), or (3) Partially cyclic, after certain activities, you go back to adjust tasks from previous phases.

Once again the difference in profile preferences is evident (Fig. 2), with 61.0% of academia supporting the partially cyclical mode, compared to 51.9% for the private sector favoring the continuous cyclical mode. Among other proposals, several participants suggest a hybrid mode between a partially cyclical process and a continuous cyclical process, i.e., only some phases should be conducted continuously. Other respondents point out that the way the process is executed is highly dependent on the product to be developed, so that it must be adapted individually to each project and product.

C. Challenges and improvements

In this section, to frame the views on the challenges, we asked two extra questions our respondents:

- The framework used for the development, since the limitations of this framework can be the cause of the challenges. Here, 80.2% of the respondents indicated ROS, 57.7% indicated ROS 2, 6.6% indicated OPC-UA, and 6.3% indicated Yarp.
- The use of tools that assist them during the integration phase, here a clear 68.7% said yes, they use tools.

Interestingly, when ranking which of the main challenges in robotics are the most relevant during the integration phase, we found no notable differences between the responses of each profile (Fig. 3). Both rank *Dynamic adaptation to the environment* as the major constraint faced by a robotic systems developer (43.6% of them put it in the first place). The second place is assigned to *Safety certification* (20.0%) and *Validation of the resulting system* (16.4%).

At the other side of the spectrum, we find the *Software Reusability* and *Interoperability between middlewares* and systems. This is not surprising, since if most of the respondents use ROS (or ROS 2), and reusability is assured by the nature of this framework, and they do not face

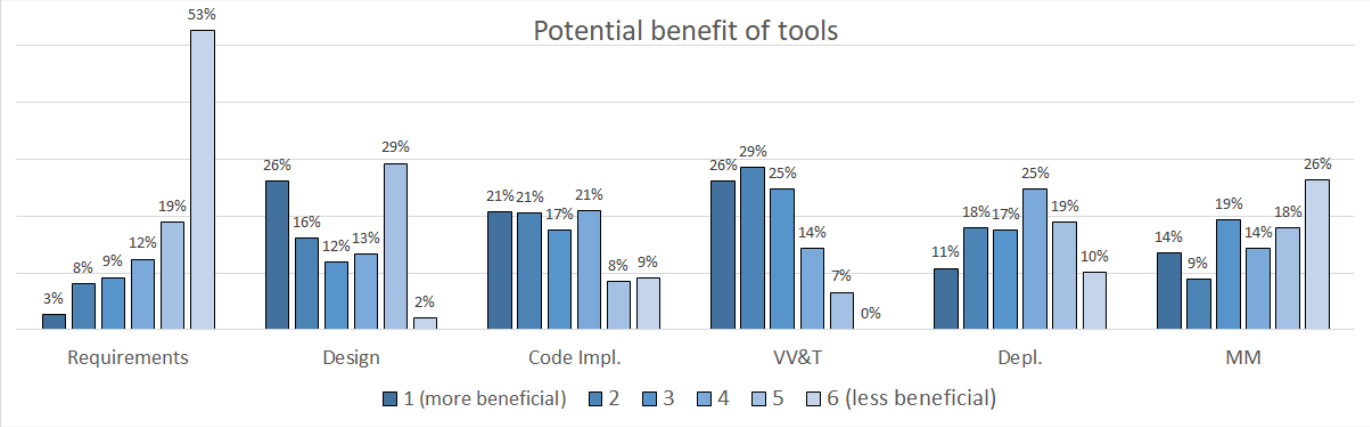


Fig. 4: Summary of the answers to the stage for which a tool would be more beneficial question.

interoperability problems since they do not target hybrid systems.

Finding 2: 43.6% of the participants ranked the *Dynamic adaptation to the environment* as the major challenge faced by robotic systems developers.

D. Tools to support integration

It is also relevant for us to know what instruments we have available to assist the integrator. Therefore we asked about the tools that are today in use. Even leaving the answer as open free text, we could easily find patterns in the type of tools that are employed. A 50% of the respondents assert that version control systems (such as GitHub or GitLab) are a great help for the implementation of the code, usually, it is combined with continuous integration systems. The second most mentioned tool types, 7 out of 53 responses, were for deployment containerization tools, such as Docker. The same percentage of people indicated IDEs as good support. The list of tools is completed by simulators (concretely Gazebo [20]) and ROS-specific tools (like rosgraph [22] or rosbag [21]). Another type of tool to highlight from the answers are model-based tools, among them SysML, SmartMDSD, or Simulink were mentioned.

To complement this part, we also wanted to investigate in which phases these tools are used. Unsurprisingly, the answers obtained are consistent with the tools mentioned above. The 76.3% of the responses pointed to the use of tools during the *Implementation* phase, 73.7% for *Validating and Testing* purposes, and 55.3% for the *Deployment* phase. Then, 34.2% reported the use of tools for *System design*, while 32.9% use them for *Monitoring and Maintenance* of the system. Last, only 7.9% said that use tools for the *Requirements Analysis* phase.

Finding 3: The 76.3% of the responses pointed the use of tools during the *Implementation* phase, 73.7% for *Validating and Testing* purposes.

E. Desired tools to support integration

In this last section of the survey, we want to find out the needs of the robotics community in terms of new tools to unify the procedure. A more than overwhelming 95.5% believe that tools can help, simplify and standardize the integration process.

The majority of respondents indicated in their free-text responses (20 out of 78 answers) that they see the use of tools to manage the use of common interfaces, and to some extent standards, as essential. Also, the unification of documentation is highlighted by the responders. Another valuable feature of a potential new tool is the assistance in the validation, verification, and testing stages of the system (17 out of 78 respondents). These tasks are very tedious to perform manually, any form of automation in this aspect would be very beneficial. Other users bring up code generator tools that not only save time and effort but also make the code less error-prone. A handful of participants go further and talk about model-based tools that allow the formal description of both, hardware and software characteristics of components, and even one says “making aspects explicit and achieve traceability from the requirements down to the source code”. In line with this last statement, to the question of what feature used in other domains would be beneficial in robotics, 74.1% point to the model-based nature, a 65.7% refer to the benefit of including graphical interfaces that make the process more intuitive and a 61.1% also see the need for tools that incorporate code generators. Among other suggestions, we have captured formal verification of systems, simulators, and data-driven tools.

When surveyed about in which phase of the development of the integration would be most beneficial to have tools, the results (Fig. 4) of the most selected option in the first position are very close. If we also consider the option selected as a second choice, we can say that the robot developers wish to have a tool that facilitates the *Validation, Verification, and Testing* activity. The second stage selected, with little difference, is the *Design* phase. If we go to the opposite extreme, the responders clearly identify the *Requirements Analysis* phase as the one where they see the

least need for help. On this topic, the responses between the scientific community and private industry have no remarkable differences, both sectors reach similar conclusions.

Finding 4: Robot developers wish foremost to have a tool that facilitates the *Validation, Verification, and Testing* activity.

V. Discussion

A. Lessons Learned

RQ1. Applicability of SDLC Methodologies: To obtain a conclusion on this aspect, we must look at Fig. 1b, when asking whether the phases we propose (which have been completely extracted from existing SDLC methodologies) should be carried out during the system integration. All the phases have obtained more than 50% support. We conclude from this that the community generally supports using SDLC methodologies but with nuances:

- The phases should be adapted to accommodate specific integration tasks, as the proposed steps are too generic for software development.
- The phases do not all have the same importance, depending on the Technology Readiness Level (TRL) some may even be omitted in the case of rapid prototyping and, even if all of them are conducted, not all must be executed continuously or cyclically, some will suffice to be performed only once.

As a first instantiation of a methodology for software integration in robotics, the exercise of evaluating SDLC methods seems not to have gone so far astray. The basis is good, but we are also aware that such generic methods for product development cannot be applied in a literal and bare-bones way. All the information we have gathered with our study is of great value to adapting the methodologies to cover the particularities of the integration topic and to identify the filings in the process depending on the maturity level of the solution to be built.

RQ2. Challenges: Based on our findings, the most important challenges to robotics software integration are:

1) *Dynamic adaptation to the environment:* This is the challenge of most concern to respondents and is no surprise. Robotic systems have to operate in a real environment and interact with it. On a technical level, we have increasingly better sensors and perception software to help us. But on a practical level, it is a very complex matter to solve for an integrator, there are too many peculiarities of each particular scenario to be considered.
2) *Safety certification:* In most cases, the certification of a robotic application is a very arduous task, as it has to be done in a specific and individual way for each different setup. In this respect, standardization is probably the best way to address this challenge, as these standards comprise a set of requirements whose conformity can be evaluated for the different robotic systems.
3) *Validation of the resulted system:* As *Finding 4* suggests, robotics integrators would appreciate having tools to assist in this task. But there is very little development in this area. This is because to automate (fully or partially) the validation, we need first a formal definition of the customer specifications as well as of the built system, in both aspects, there is a complete lack of standardized solutions.

RQ3. Tools: Several conclusions can be drawn from the results:

1) The community supports the development of tools for integration.
2) There is a great need for tools for the unification of the process, specifications, and interfaces. Many respondents claim standardization in robotics and point out that the use of tools can be part of the solution.
3) The preferred choice is model-based tools.
4) Validation, verification, and test functionalities, as well as system design should be the main priorities to be supported by tools.

It is very challenging to design a set of tools or a complete toolchain that can address all of these points altogether. Nonetheless, in other, more consolidated industries of complex systems, e.g., automotive or aeronautics, we can find some traceability between the use of tools, standardization, utilization of models, and validity and verification of requirements [10], [12]. These industries have a strong base of standards. The tools available to software solution developers use models that conform to these standards, which greatly facilitate the integrators' work (including VV&T). These domains should be taken as inspiration when developing new tools. However, we see an analog development arduously complicated since in robotics we do not have the first foundation, a base of unified and consolidated standards.

B. Threats to validity

To discuss the threats to validity of our survey we analyze the standard classification [23].

1) Construct Validity: In this aspect, the biggest threat to our study has been vocabulary. Many of the terms used are abstract (e.g., system, tool, or process). In some of the answers, we detected that users did not interpret the terms in the context we were addressing. This has only happened in some free-text answers, nevertheless, their answers have been partly considered for the general evaluation of the study.

2) Internal Validity: For the distribution of the survey we promoted it through channels outside our circle of partners and collaborators, in this way, we have reached anonymous and unbiased parties. However, it must be considered while looking at the results that most of the respondents are software developers. This is especially relevant for non-software-related questions like the challenges part.

3) External Validity: By tackling the issue of integration and this being a common process in any type and category of robotic application, we certainly cannot consider threats in the aspect of applicability to other robotic domains, all of them have been de facto included.

VI. Conclusion

We conducted a survey on robotics software integration that was answered by 118 participants from industry and academia. Their answers are helpful to understand the state of the art in robotics software integration, as well as the demands of the community. Judging from the results there are three clear things. There is still an opportunity to consolidate a generic process for software integration in robotics, there is a clear consensus on the current issues that an integrator faces and there is room for tools that provide solutions to the community.

In the future, on one hand, we plan to consolidate the insights from this survey into a proposal of a generic robotics software integration process to systematically guide researchers and practitioners in efficiently engineering robotics applications. A draft of this proposal, along with a summary of the survey responses, will be made public using the same distribution channels. On the other hand, we will investigate tools to unify the process and address the challenges. Primarily, we will look into other traditional domains of complex software engineering, such as automotive and avionics, and more specifically those based on model-driven development (since 74% of the responses encourage the use of this type of solution). Both efforts converge in a clear mission: unify and facilitate the integration of software in robotics.

Acknowledgment

We sincerely thank all participants of our survey.

This research is sponsored by the Baden-Württemberg Ministry of Economic Affairs, Labour and Tourism Baden-Württemberg at the AI Innovation Center "Learning Systems and Cognitive Robotics".

References

[1] M. Andreessen, "Why software is eating the world," *Wall Street Journal*, vol. 20, no. 2011, p. C2, 2011.

[2] K. Balasubramanian, D. C. Schmidt, Z. Molnár, and Á. Lédeczi, "System integration using model-driven engineering," in *Designing software-intensive systems: methods and principles*. IGI Global, 2009, pp. 474–504.

[3] R. Bormann, T. Zwölfer, J. Fischer, J. Hampp, and M. Hägele, "Person recognition for service robotics applications," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2013, pp. 260–267.

[4] L. Chin, M. C. Yuen, J. Lipton, L. H. Trueba, R. Kramer-Bottiglio, and D. Rus, "A simple electric soft robotic gripper with high-deformation haptic feedback," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2765–2771.

[5] R. Dietrich and S. Dörr, "Deep learning-based mutual detection and collaborative localization for mobile robot fleets using solely 2d lidar sensors," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 6706–6713.

[6] I. Drave, S. Hillemacher, T. Greifenberg, B. Rumpe, A. Wortmann, M. Markthaler, and S. Kriebel, "Model-based testing of software-based system functions," in *2018 44th Euromicro conference on software engineering and advanced applications (SEAA)*. IEEE, 2018, pp. 146–153.

[7] M. Fowler, J. Highsmith *et al.*, "The agile manifesto," *Software development*, vol. 9, no. 8, pp. 28–35, 2001.

[8] T. Froehlich and U. Reiser, "Design and implementation of a spherical joint for mobile manipulators," in *Proceedings of ISR 2016: 47st International Symposium on Robotics*, 2016, pp. 1–8.

[9] F. Gerhardt, "Integration of programming environments for platform migration." Ph.D. dissertation, University of Tübingen, Germany, 2003.

[10] V. I. GmbH. DaVinci Developer - Designing AUTOSAR Software Components. Accessed: 2022-08-06. [Online]. Available: https://www.vector.com/int/en/products/products-a-z/software/davinci-developer/

[11] J. Grady, *System Integration*. CRC Press, 1994.

[12] M. Halle and F. Thielecke, "Avionics next-gen engineering tools (avionet): Experiences with highly automised and digital processes for avionics platform development," in *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, 2021, pp. 1–8.

[13] N. Hammoudeh Garcia and A. Wortmann, "Patterns and tools in robotic systems integration," in *2022 IEEE International Conference on Robotic Computing (IRC)*, 2022.

[14] M. Hüttermann, *DevOps for developers*. Apress, 2012.

[15] IFR, "International federation of robotics - annual report." [Online]. Available: https://ifr.org/

[16] A. M. Madni and M. Sievers, "Systems integration: Key perspectives, experiences, and challenges," *Systems Engineering*, vol. 17, no. 1, pp. 37–51, 2014.

[17] S. Mathur and S. Malik, "Advancements in the v-model," *International Journal of Computer Applications*, vol. 1, no. 12, pp. 29–34, 2010.

[18] K. Petersen, C. Wohlin, and D. Baca, "The waterfall model in large-scale development," in *International Conference on Product-Focused Software Process Improvement*. Springer, 2009, pp. 386–400.

[19] S. Realpe, F. G. Roldan, J. M. Fajardo, J. D. Hernández, and P.-F. Cardenas, "Benchmark of sampling based motion planners in bin picking manipulation task," in *2022 27th International Conference on Automation and Computing (ICAC)*. IEEE, 2022, pp. 1–6.

[20] O. Robotics. Gazebo: Robot simulation made easy. Accessed: 2022-08-06. [Online]. Available: http://gazebosim.org/

[21] ——. rosbag: Set of tools for recording from and playing back to ROS topics. Accessed: 2022-08-07. [Online]. Available: http://wiki.ros.org/rosbag

[22] ——. rosgraph: command-line tool that prints information about the ROS Computation Graph. Accessed: 2022-08-07. [Online]. Available: http://wiki.ros.org/rosgraph

[23] F. Shull, J. Singer, and D. SjÃ¸berg, *Guide to Advanced Empirical Software Engineering*. Springer, 01 2008.

[24] J. T. Stoll, K. Schanz, and A. Pott, "A compliant and precise pneumatic rotary drive using pneumatic artificial muscles in a swash plate design," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3088–3094.

[25] B. Summers, *Effective Methods for Software and Systems Integration*. CRC Press, 04 2016.

[26] D. Tola, E. Madsen, C. Gomes, L. Esterle, C. Schlette, C. Hansen, and P. G. Larsen, "Towards easy robot system integration: Challenges and future directions," in *2022 IEEE/SICE International Symposium on System Integration (SII)*, 2022, pp. 77–82.

[27] P. Triantafyllou, R. Afonso Rodrigues, S. Chaikunsaeng, D. Almeida, G. Deacon, J. Konstantinova, and G. Cotugno, "A methodology for approaching the integration of complex robotics systems: Illustration through a bimanual manipulation case study," *IEEE Robotics & Automation Magazine*, vol. 28, no. 2, pp. 88–100, 2021.

[28] D. Trowbridge and M. Corporation, *Integration Patterns*, ser. Patterns & practices. Microsoft Corporation, 2004.

[29] E. Wete, J. Greenyer, A. Wortmann, O. Flegel, and M. Klein, "Monte carlo tree search and gr (1) synthesis for robot tasks planning in automotive production lines," in *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 2021, pp. 320–330.