Modelling the Component-based Architecture and Safety Contracts of ArmAssist in Papyrus for Robotics

Jabier Martinez, Alejandra Ruiz
Software Lifecycle Innovation group
ICT Division
Tecnalia, Basque Research and Technology Alliance (BRTA)
Derio, Spain
name.surname@tecnalia.com

Ainara Garzo, Thierry Keller
Neuroengineering area,
Health Division
Tecnalia, Basque Research and Technology Alliance (BRTA)
San Sebastian, Spain
name.surname@tecnalia.com

Ansgar Radermacher
CEA-List
Massy, France
ansgar.radermacher@cea.fr

Stefano Tonetta
Fondazione Bruno Kessler
Trento, Italy
tonettas@fbk.eu

***Abstract*—Healthcare robots are increasingly being used and the way they are engineered they still have several challenges regarding reference models and validation. In this experience report we focus on the ArmAssist robotic system and how it can be modelled including safety considerations for validation in early design phases. ArmAssist is an upper-limb robotic system for stroke rehabilitation based on serious games. The open-source tool Papyrus for Robotics was used for modelling the robotic system in close collaboration with neurorehabilitation domain experts. Papyrus for Robotics includes new functionalities that we contributed for contract-based design at component and system level, allowing to make explicit and validate the safety considerations using formal languages. In our case, the assertions are expressed in OCL and Othello. We present the resulting model and a discussion from domain experts.**

I. Introduction

Service robotics [1], those which are not part of industrial automation processes, are increasingly present in our society. A recent industrial survey on the state of the art and practice in service robotics engineering [2] indicates that several challenges regarding reference models and validation remain in this domain. Aligned with this challenge, in this experience report we focus on ArmAssist [3], [4], a robotic system for neurorehabilitation, and how the system can be modelled including safety considerations.

ArmAssist[1] is a robotic system based on serious games for upper-limb rehabilitation of stroke survivors. It is a portable device that can be used in clinical rehabilitation, but also at home with remote supervision. It allows prolonging, complementing, or continuing the regular therapies even in restricted times, e.g., due to COVID-19 pandemic [5]. The ArmAssist is an affordable, mobile and sensorized robot integrated in an assessment platform for automated quantitative evaluation of the patient's movements and interaction forces. The TeleReha software platform based on serious games includes a calibration process to identify the patient's range of motion [4]. The system set-up is shown in Figure 1 and the main system components are represented in Figure 2. The ArmAssist system has been improved thanks to the work and trials carried out with therapists and patients [6][7][8].

The robot includes a motorized base, which can be activated to help the patients to perform the movements. Thus, safety requirements are of paramount relevance, especially since the patients' arm movement and range limitations due to their impairments can cause considerable physical damage. We modeled and added safety assertions to the latest ArmAssist version in its most common variant thanks to several iterations and the collaboration with ArmAssist's domain experts, and thanks to functionalities contributed to the open-source tool Papyrus for Robotics [9], [10] regarding adding formal assertions and contract-based design to the models as in [11].

This paper is structured as follows: Section II presents the methodology and the resulting system model. Then, Section III introduces the concept of safety contracts, the addressed safety considerations for ArmAssist and how they were formalized and integrated. Section IV presents the discussion from domain experts and Section V concludes and outlines future work.

Fig. 1: ArmAssist system set-up

[1]Video of ArmAssist: https://www.youtube.com/watch?v=0L7QSPU6QBk

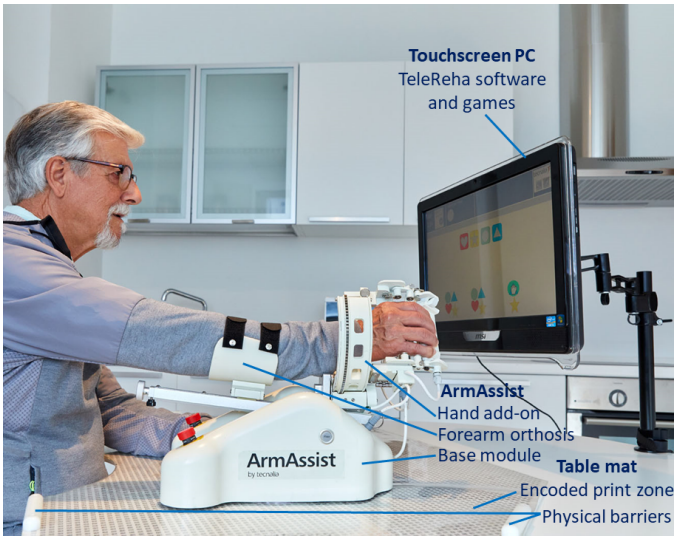

Fig. 2: ArmAssist parts

II. Modeling the component-based architecture

As a note on the methodological procedure for the modelling of the component-based architecture, we performed semi-structured interviews with an ArmAssist expert and we analysed the Bill Of Materials (BOM) of the robotic systems that was provided to us. Five iterations with the expert were needed until the model reflected properly, in an abstract level, the implemented system. The level of detail and the granularity of the components were agreed so that the resulting model should be seen as a real model but providing a high-level specification. Thus, the model is certainly hiding complexities from the implementation that could have been captured with more detailed decomposition of the components.

ArmAssist model consists of the following components:

- `PositioningSystem`: The objective of the positioning system is to provide the `MainControlUnit` with an accurate position and orientation of the robot with respect to the table, which are represented on the screen by the TeleReha software. The robot is used on top of a mat which includes a grid of QR codes (Quick Response codes) with the position coordinates, as it can be slightly observed in Figure 2 (Encoded print zone). QR codes information are continuously read by a camera installed inside the robot. The information of the robot's position is then consolidated taking into account two sources: (i) the information of the QR read by the `Camera` and processed by the `QRManager`, and (2) the relative position calculated by `Motors` using the number of spins of the wheels. Details on the motors as part of the positioning system can be found in [12]. The spin of the wheels and the position and orientation calculated by the QR recording helps also to estimate, confirm, or adjust the consolidated position of the robot.
- `Camera`: The camera is inside the base module, perpendicular to the QR mat in a way that the camera can read the QR codes. Images are sent to the `QRManager` for their processing.
- `QRManager`: This software component processes the images received by the `Camera` and gets the information from the QR codes where each one represents its position in the table.
- `Motor`: The robot uses 3 motors, one for each wheel. The system includes an algorithm for the motors and thus the robot smooth movement and turn according to the spin instructions of the `MainControlUnit`. We decided to group in this component the functionalities of the motor driver and encoder. A more detailed idea of the motor architecture can be found in [12] corresponding to a previous version of ArmAssist.
- `MainControlUnit`: The main control unit is agnostic to the activity (i.e., game) that is currently happening. It receives target positions and orientation angle of the robot, target finger forces, wrist angle, and shoulder force translated into arm weight, and according to the position of the `PositioningSystem` will ask the Motor actuators to move accordingly.
- `ActivityManager`: The activity manager takes the defined training games for the user [4], handle the screen visualisation of the game, and provide the target positions, finger forces, wrist angle, and arm weight to the `MainControlUnit` according to the sequence planned in the game. Once the target is reached the same game continues with next target until the planned time is reached. A new game can be started if the previous one has been finished, according to the therapy plan.
- `CalibrationActivityManager`: The calibration is a special activity where the current position, angles and forces are retrieved from the `MainControlUnit` to establish the limits of the patient. It corresponds to the special activities in the assessment games, where the patient is asked to perform his/her maximum movement in each case [4].
- `UserManager`: The user manager is responsible of storing the latest calibration values of the patient and to provide the list of activities to the `ActivityManager` as prescribed previously by the `TherapistConfigurationManager`.
- `TherapistConfigurationManager`: It allows the therapist to define the activities for each patient (therapy plan and calibration games), as well as monitoring the rehabilitation progress.
- `ForceSensitivityResistor`: 2 force sensing resistors (FSR), one for the thumb and other for the rest of the fingers, allow measuring the grasp movement. They are included on the hand add-on (Figure 2). The sensors translate the force the patient applies to the device and this information is sent to the `MainControlUnit`.
- `Potentiometer`: This sensor provides to the `MainControlUnit` information about the angle in which the patient turns the wrist (prono-supination exercise). It is also included in the hand add-on in the arc covering the wrist (Figure 2).

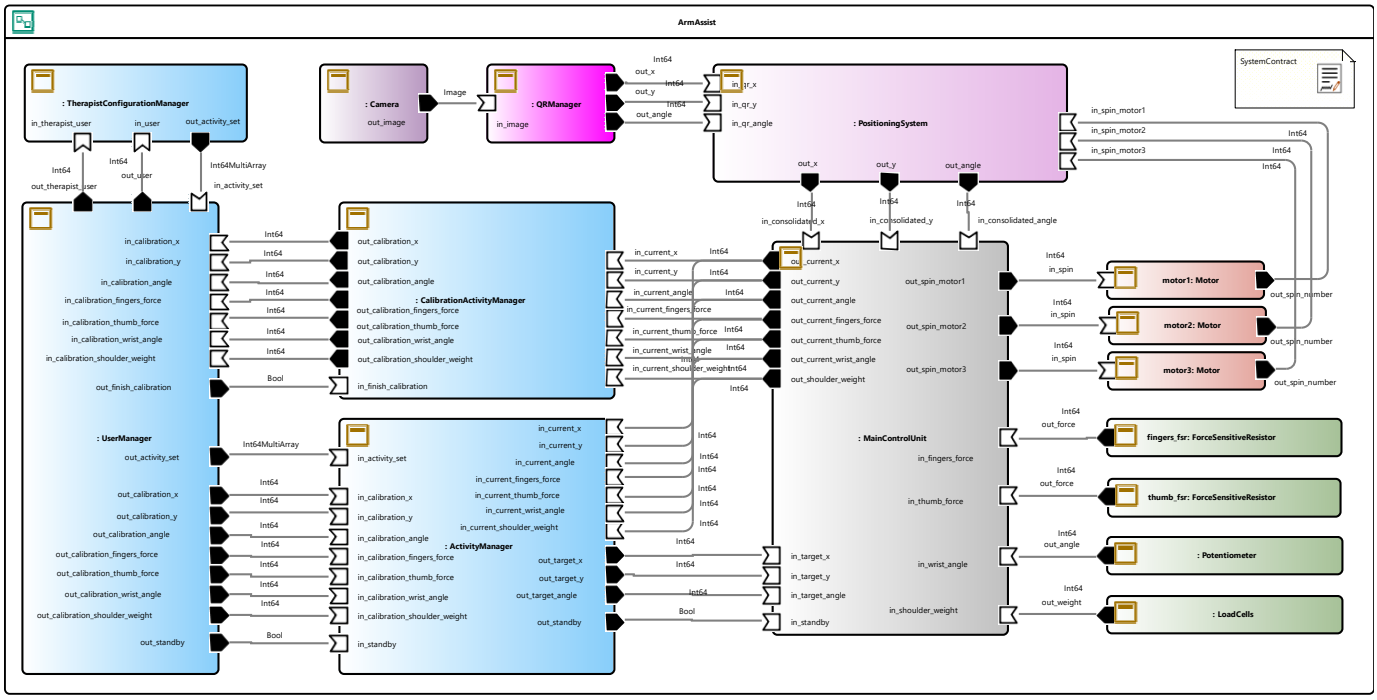

Fig. 3: ArmAssist system model

- `LoadCells`: The load cell is a sensor to measure the weight that the patient applies on the bar where the hand add-on and forearm orthosis are attached on the robot. This sensor allows shoulder elevation force measurements. In Figure 2 it is positioned in the forearm. This information is sent to the `MainControlUnit`.

Figure 3 presents the whole system model in Papyrus for Robotics with the component instances and their connections[2].

III. Safety Considerations and Contracts

Most of the ArmAssist components are commercially available. Component providers could define their own contracts that, depending on the context of operation where the component is instantiated, might have safety consequences or not. Papyrus for Robotics is extensible in the use of formal languages for defining assertions [10]. It supports OCL [13] as well as OCRA contracts [14] with Othello assertions [15] (Othello language is mapped to temporal formulas in linear temporal logic).

If we take as example the Camera, the frames per second (fps) can have a safety implications if using low values for a robot that needs image processing while moving at high speeds. This is not the case of ArmAssist but we added an illustrative contract that a Camera component provider can include. Cameras usually include fps as a parameter with a trade-off between speed and image resolution. We added an Integer parameter `fps` in the Camera component and an assertion regarding the predefined values which are accepted for this specific camera. In this case we can use OCL [13]. The assertion, using the P4R OCL language (P4R OCL is an extension of OCL to use directly P4R component's parameter names, port names, properties names, etc.), is defined as follows:

```
Set {15, 30, 45, 60, 90, 120} → includes (fps)
```

This assertion can be statically evaluated from the default value of the parameter (i.e., 15) or with the value in Camera component instances if this was modified.

Then, we added another assertion regarding the output port of the camera that transmits the Image type. We want to express that a new image is transmitted every x milliseconds, where x is derived from the `fps` parameter. Using the P4R Othello language, the assertion is as follows using milliseconds as time unit:

```
always(out_image implies then
time_until(out_image)=1000/fps)
```

The `out_image` is the port and `always`, `implies`, `then` and `time_until` are reserved words of the Othello language. The assertion says that an `out_image` is always followed by another `out_image` after `1000/fps` milliseconds. This assertion, as the previous one, helps making this fact explicit, and it can be potentially used for consistency checks with the assertions of other components, e.g., a real time image processing component that requires a minimum frequency of images to work properly. In our case study, the fps are not

[2]The ArmAssist system model and component definitions including the safety properties and contracts are publicly available. https://github.com/TecnaliaResearchAndInnovation/papyrus4robotics-models

very relevant for the safety so first we needed to identify the ArmAssist-specific safety considerations.

A. Safety considerations

The main set of safety considerations were discussed with the domain expert during the interviews which are summarized in the following points.

- S1 **Respect the calibration values**: During the calibration activity the patient is asked to perform the movements until her/his range of motion (RoM) limits to identify the thresholds in every exercise. It is already known that patients usually push themselves too much during calibration, so the threshold values are reduced by 10% during the activities to provide a enough challenging, but safe and not frustrating margin. Calibration exercises can be repeated at any moment, with the aim of updating the threshold due to the patient's improvement.
- S2 **Move only when new target values are available**: The `MainControlUnit` should not send control instructions to the `Motor`s until values for a new game are ready. The standby mode is used to refresh the wireless connections with the robot, and motors will not start working until new targets are calculated according to the calibration threshold (e.g., when there is a change from one patient to another).
- S3 **Avoid lifting up the robot**: The arm is fastened to the robot and it is desired that the patient cannot lift up the robot from the table for his or her own safety. The wheels must be in contact with the mat to provide accurate movements to the patient. This also provides proper information about the robot position in every movement to the `PositioningSystem`. To achieve this, extra weights have been added inside the robot to keep balance between maintaining the portable condition of the system and preventing that the patient could lift it from the table during the therapy. Additionally, physical barriers have been added to the mat in order to avoid that the robot could fall down from the table.
- S4 **Maximum speed of the motors**: The maximum speed of the motor is hard-coded into the system to not exceed movements above 3 centimetres per second which could result in harm of the patient by eliciting reflexes or spasticity. It always moves at the same speed, except from the accelerating and decelerating movement until reaching the defined maximum speed or zero respectively. This important safety property is handled through both the `MainControlUnit` and the `Motor`s. Notably, the commercial motors used in ArmAssist have also a way to establish a maximum speed.

Another safety consideration is the option of forcing a stop of the system in case of unexpected behavior of the robot. Two big safety buttons have been mounted on the robot. However, we do not add them to our design as it is HW power-off of the system that works completely mechanically.

B. Safety contracts

The model presented in Figure 3 is the structural view of the robot's design and the safety considerations are modeled as component contracts. Contracts provide an abstraction of the behavior at the level of component interfaces. They allow to reason about the architectural decomposition without the need to specify the detailed behavior of components [11]. We present, for each of the safety considerations, how the contracts were modeled.

S1: Respect the calibration values: It is important not to send instructions to the `Motor`s through the `out_target_` ports if they are not respecting the calibration values after applying the corrective margin. The corrective margin is hardcoded in a property of the component.

In the `ActivityManager`, we have these assertions which are guarantees of the component where `calibration_safety_margin` is a component Property with value 0.9. The used language is P4R Othello where `always` is a keyword to define that the expression needs to be always satisfied.

```
  always (in_calibration_x *
calibration_safety_margin <= out_target_x)
  always (in_calibration_y *
calibration_safety_margin <= out_target_y)
  always (in_calibration_angle *
calibration_safety_margin <= out_target_angle)
```

These assertions cannot be statically evaluated. This will require a runtime validation.

S2: Do not move if not ready: We wanted to prevent that the `MainControlUnit` receives instructions to move the motors until calibration values are received. For this, we added the following assertions in the `ActivityManager` using the P4R Othello language:

```
  always (in_standby implies (in_calibration_x
releases not out_target_x)
  always (in_standby implies (in_calibration_y
releases not out_target_y)
  always (in_standby implies
(in_calibration_angle releases not
out_target_angle)
```

The Othello keywords `implies`, `releases` and `not` are used to express that, after a standby start, `out_target_` ports are available only if the corresponding calibration value is received.

S3: Avoid lifting up the robot: According to experiments if the robot is less than 7 Kilograms it is much more probable that the wheels lift up from the table. The weight of the structure is already around 3700 grams and each motor is around 100 grams. To reach the desired weight, extra weights are used, and there is a compartment in the structure of the robot for that.

The assertion at System level using the P4R OCL language is defined as:

```
  structureWeight + GlobalSum(weight) +
extraWeight >= minRecommendedWeight
```

The values of structureWeight, extraWeight, and minRecommendedWeight are added as system properties. The GlobalSum operator from P4R extended languages takes the property weight from all the component instances. In this case, the `Motors` provide weight while in the other sensors the weight is not significant. To reach the recommended weight, the compartment for the extra weights is used. In this version around 6000 grams are used to reach a total robot weight of around 10 Kilograms. This assertion can be evaluated statically.

S4: Maximum speed of the motors: The default parameter regarding the current max_speed of the Motors is high as the component do not have information about its future context of operation. This value is too high for ArmAssist context so these values are updated with the specific speed in the 3 Motor component instances. As this maximum speed value is also controlled by the `MainControlUnit`, a property `motors_speed` was added in this component.

Then, the assertions at System level using the P4R OCL language are:

```
mcu.motors_speed = motor1.max_speed
mcu.motors_speed = motor2.max_speed
mcu.motors_speed = motor3.max_speed
```

These assertions check the consistency of the `MainControlUnit` property with respect to the `Motors` parameters. It can be evaluated statically.

IV. Discussion and lessons learned

In this section we report the results of this experience from the perspective of the ArmAssist's domain expert that was directly involved during the modelling both for the presented component-based architecture and for the assertions. This domain expert is the main developer of the ArmAssist asset who has been working and researching on this system since 2016. Feedback was also provided by the head of the neurorehabilitation area of Tecnalia with extensive experience in healthcare robotics and ArmAssist, both at functional and technical level, since its initial development in 2008 and first version in 2009. We are aware of the inherent bias of using experts for evaluation so we only claim here to report qualitative feedback.

Component-based architecture modelling was not new as a concept for the ArmAssist team, however, they did not rely on this formalism for its design. ArmAssist design documents are based on different documentation including rich spreadsheet files containing very detailed information about the Bill Of Materials (BOM). As the ArmAssist is a medical device, most documentations of this system has been produced in a similar way as a technical file for medical devices. The BOM contains descriptions, manufacturer information, quantity, relevant quality attributes, etc. ranging from low-level elements such as capacitors, resistors, cables to more high-level components. Those high-level components include main control unit, camera, user interface, light, motor drivers, motor, emergency button, load cells, potentiometer, battery, charger, external connectors etc. where several of them we later interpreted and modelled in this work.

Most of the components are commercial off-the-shelf (COTS) components but the ArmAssist team developed firmware and software for their integration as well as the main control unit to satisfy the ArmAssist's purpose. Several patents are associated to ArmAssist and it has been licensed to companies in Europe and China. The BOM for building and replicating the system, and obviously, all the tests and test documentation needed for validation and verification purposes, are sufficient assets for its validation and commercialization. Thus, a more high-level viewpoint from the system as proposed in this work can have its benefits, but it is not mandatory for a successful delivery. More challenging are all the certification requirements for medical products to which this approach can contribute, or whether the approach can help to discover safety violations that would be very difficult to detect using the current testing techniques. In this context, we discussed with them the potential benefits and drawbacks they see in integrating these modelling practices in their future developments and for ArmAssist evolution in particular.

The first positive aspect that they mention is the diagram visualisation which offers a global overview of the system (e.g., components, ports, connections) much faster than the one you can obtain inspecting the BOM spreadsheets. They also mentioned that they considered the tool relatively easy to use so the learning curve is small compared to the benefit of obtaining the diagrams. The elements to create the diagrams, i.e., the concrete syntax of the Papyrus for Robotics metamodel, was expressive enough to capture the system overview. Other information which is not directly represented in the diagram, but that can be added in the tool such as the quality attributes of the components, descriptions etc. could have been used to include almost all the data currently represented in the spreadsheets.

The new functionality of Papyrus for Robotics to include contracts was found interesting by the ArmAssist expert. It can help to make explicit safety considerations which are currently implicit knowledge, or explicit but at very low-level (e.g., source code). Those considerations descriptions are based on formal language, not on natural language which was also positively considered. The currently supported languages OCL and Othello are correctly documented. The continuous evolution of ArmAssist in different versions and variants, could end up identifying inconsistencies during design-time although the assertions and contracts are all currently satisfied. The early detection of mechanical issues would avoid expending time and money in the implementation phase if those problems are later identified during the source code development, on configuration or on testing phases. So the proposed approach is considered sound and promising for the ArmAssist team.

It has been also considered that the modelling of the component-based architecture and the safety considerations could help in the communication among different stakeholders including software and hardware architects and developers, quality managers and experts in safety. The global experience

and the model built in this work also opens the door to new research directions for the the neuroengineering team involved in ArmAssist, notably to leverage model-driven techniques for healthcare robotics.

V. Conclusion

This experience report presented the modelling of the ArmAssist robotic system for hand and arm rehabilitation including its safety considerations. This was possible with the new functionalities of Papyrus for Robotics which is an open-source tool available for the robotics community. The discussion and lessons learned from this process aims to encourage robotic practitioners to adopt these modelling practices that can help to make more explicit their designs, support in the communication among stakeholders, and support the validation in early phases thanks to formal assertions and contracts at component and system levels.

As further work, research on how to integrate runtime verification and validation of the assertions, using data flow simulation, or hardware- or software-in-the-loop runtime monitoring, possibly automatically deriving the monitors from the assertions. Also, design space exploration evaluating design alternatives is a possible research direction. This will require to try to optimize quality attributes while satisfying the assertions. Other practical directions include an assessment of the scalability, investigating how the approach can help to avoid unpredictable situations, or trying to assess the development cost and difficulties of maintaining the contracts in hardware and software evolution.

Acknowledgment

This work has been funded by the SafeCC4Robot Integrated Technical Project which received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 732410, in the form of financial support to third parties of the RobMoSys Project. We would like to thank Angel López, Elixabete Ostolaza, Matteo Morelli, and Huascar Espinoza for their help during the tool design and development. The authors also would like to thank to Iñigo Dorronsoro, Javier Arcas Ruiz-Ruano, Gabriel Gaminde, Beñat Garcia-Mendizabal, Je Hyung Jung, Cristina Rodriguez-de-Pablo, Joel Perry, Aitor Belloso, David Valencia and Haritz Zabaleta for their contributions to the ArmAssist system development.

References

[1] ISO, "ISO - Robotics," 2012. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en

[2] S. García, D. Strüber, D. Brugali, T. Berger, and P. Pelliccione, "Robotics software engineering: A perspective from the service robotics domain," *ESEC/FSE*, pp. 593–604, 2020.

[3] J. C. Perry, H. Zabaleta, A. Belloso, and T. Keller, "Armassist: A low-cost device for telerehabiltation of post-stroke arm deficits," in *World Congress on Medical Physics and Biomedical Engineering, September 7 - 12, 2009, Munich, Germany*, O. Dössel and W. C. Schlegel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 64–67.

[4] C. Rodriguez-de-Pablo, J. C. Perry, S. Balasubramanian, A. Belloso, A. Savic, T. D. Tomic, and T. Keller, "Serious games for assessment and training in post-stroke robotic upper-limb telerehabilitation," in *Proceedings of the 2nd International Congress on Neurotechnology, Electronics and Informatics, NEUROTECHNIX 2014, Rome, Italy, October 25-26, 2014*. SciTePress, 2014, pp. 126–134.

[5] A. Garzo, J. A. Ruiz-Ruano, I. Dorronsoro, G. Gaminde, J. H. Jung, J. Téllez, and T. Keller, "Merlin: upper-limb rehabilitation robot system for home environment," in *Converging Clinical and Engineering Research on Neurorehabilitation IV. ICNR 2020*, 2020.

[6] J. C. Perry, C. Rodriguez-de Pablo, F. I. Cavallaro, A. Belloso, and T. Keller, "Assessment and training in home-based telerehabilitation of arm mobility impairment," vol. 3, pp. 44–75, Nov. 2013. [Online]. Available: http://www.jacces.org/index.php/jacces/article/view/12

[7] C. Rodriguez-de Pablo, A. Savić, and T. Keller, "Game-based assessment in upper-limb post-stroke telerehabilitation," in *Converging Clinical and Engineering Research on Neurorehabilitation II*, J. Ibáñez, J. González-Vargas, J. M. Azorín, M. Akay, and J. L. Pons, Eds. Cham: Springer International Publishing, 2017, pp. 413–417.

[8] T. J. D. Tomić, A. M. Savić, A. S. Vidaković, S. Z. Rodić, M. S. Isaković, C. Rodríguez-de Pablo, T. Keller, and L. M. Konstantinovic, "Armassist robotic system versus matched conventional therapy for poststroke upper limb rehabilitation: A randomized clinical trial," pp. 1–6, 2017.

[9] Eclipse, "Papyrus for Robotics, v0.8," https://www.eclipse.org/papyrus/components/robotics/, 2020.

[10] J. Martinez, A. Ruiz, A. Radermacher, and S. Tonetta, "Assumptions and guarantees for composable models in papyrus for robotics," in *ICSE workshops, 3rd International Workshop on Robotics Software Engineering (RoSE)*, 2021.

[11] A. Cimatti and S. Tonetta, "Contracts-refinement proof system for component-based embedded systems," *Sci. Comput. Program.*, vol. 97, pp. 333–348, 2015.

[12] J. H. Jung, D. B. Valencia, C. Rodriguez-de-Pablo, T. Keller, and J. C. Perry, "Development of a powered mobile module for the armassist home-based telerehabilitation platform," in *IEEE 13th International Conference on Rehabilitation Robotics, ICORR 2013, Seattle, WA, USA, June 24-26, 2013*. IEEE, 2013, pp. 1–6.

[13] OMG, "Object Constraint Language," http://www.omg.org/spec/OCL/, 2014.

[14] A. Cimatti, M. Dorigatti, and S. Tonetta, "OCRA: A tool for checking the refinement of temporal contracts," in *ASE*. IEEE, 2013, pp. 702–705.

[15] A. Cimatti, M. Roveri, A. Susi, and S. Tonetta, "Validation of requirements for hybrid systems: A formal approach," *ACM Trans. Softw. Eng. Methodol.*, vol. 21, no. 4, Feb. 2013.