Joel Greenyer, Malte Lochau, and Thomas Vogel (Hrsg.): Explainable Software for Cyber-Physical Systems (ES4CPS), GI-Dagstuhl Seminar 19023, January 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 11

Towards Modeling Languages for Explainable Robotics

Andreas Wortmann[1]

Robotics are cyber-physical systems (CPS) that increasingly take over many aspects of our lives: automated vehicles bring us to work[2] where we cooperate with industrial or service robots[3] until we have dinner in a robotic restaurant[4]. Simultaneously, robotics is one of the most challenging fields of CPS research. The successfully deployment of non-trivial robotics systems demands for the multi-disciplinary collaboration of experts from a variety of different domains, including, *e.g.,* mechanical and electrical engineering, navigation, artificial intelligence, manipulation, human-robot interaction (HRI), and systems engineering. All of these domains traditionally employ different kinds of (mental) models to describe or prescribe parts of their views on the CPS under development. For instance, mechanical engineering might contribute CAD models or differential equations, artificial intelligence might contribute artificial neural networks, HRI might leverage user interface models, and systems engineering might use SysML [FMS14] diagrams. In the multi-disciplinary engineering of robotics systems, these models, which reify different ways of thinking about systems, must be aligned and integrated carefully to ultimately enable production of hardware and software of a robotic system. To prevent the costly and error-prone manual integration of models through, *e.g.,* best-practices or modeling guidelines, research on software language engineering (SLE) [HRW18] investigates making the languages behind the different experts' (mental) models explicit and accessible to automation.

Explaining is threefold as it requires that an *explainer* to explain *something* to a *receiver*. Moreover, it can require different question and explanation *shapes* to inquire about facts or contrasts [DTG18]. A research roadmap to explainable robotics needs to consider all three explanation roles as well as the different shapes of explanations. Properly expressing explanations in the light of multi-disciplinary collaboration therefore is one of the key challenges in explainable software for CPS. This demands for

[1] Software Engineering, RWTH Aachen University, Templergraben 55, 52062 Aachen, Germany wortmann@se-rwth.de

[2] Self-driving electric bus propels Swiss town into the future (CNN): https://edition.cnn.com/2018/06/27/sport/trapeze-self-driving-autonomous-electric-bus-switzerland-spt-intl

[3] Meet the Cobots: Humans and Robots Together on the Factory Floor (National Geographic): https://news.nationalgeographic.com/2016/05/financial-times-meet-the-cobots-humans-robots-factories/

[4] The Rise Of The Restaurant Robot (Forbes): https://www.forbes.com/sites/andrewrigie/2018/09/24/the-rise-of-the-restaurant-robot/

1. **Modeling languages to describe (partial) explanations at design time and at runtime**. Through such languages, developers and CPS can act as explainers by enriching existing models with semantic explanation parts in such a way that these are accessible to automated derivation of more complete explanations based on run-time models, accessible data, and the systems' history. Whether these languages are dedicated explanation languages, domain-specifically derived from systems modeling languages, or attached to existing languages needs to be investigated.
2. **Integration of explanations of different domains and across different levels of abstraction throughout the CPS lifecycle.** Explainable software for CPS requires comprehensive explanations of system behaviors across software modules provided by different domain experts. As these encompass concepts reified with different modeling languages, deriving explanations demands for automated integration of heterogeneous explanation models. Due to the different levels of software artifacts contributing to system behavior, this integration also must consider different levels of explanation abstraction (*e.g.,* by combining code with run-time models and documentation) and different phases of the CPS lifecycle (*e.g.,* design time, runtime, maintenance phases). In model-driven processes, this especially requires to reify explanations in the target code and retrieve these at runtime to combine them with other explanation models.
3. **Receiver-specific and time-specific tailoring of explanation granularity.** Explanations to different receivers and at different phases of a CPS lifecycle demand different explanation contents and shapes. Instead of creating individually tailored explanations, (partially) derived explanations should be adjusted to views matching the receivers interests automatically. To match the shapes and contents a receiver is interested in, this further demands for querying techniques over explanation models.

These three challenges demand for research in explanation languages for CPS. Research in SLE and on deriving modeling languages [HRW18] can facilitate making explanations explicit and accessible to automated construction and reasoning in futures' explainable CPS. However, to provide meaningful contributions to these challenges, research in SLE needs to be guided by domain experts to properly reify the relevant concepts necessary to express explanations.

References

[DTG18] Dam, Hoa Khanh; Tran, Truyen; Ghose, Aditya: Explainable software analytics. In: Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results. ACM, pp. 53–56, 2018.

[FMS14] Friedenthal, Sanford; Moore, Alan; Steiner, Rick: A practical guide to SysML: the systems modeling language. Morgan Kaufmann, 2014.

[HRW18] Hölldobler, Katrin; Rumpe, Bernhard; Wortmann, Andreas: Software language engineering in the large: towards composing and deriving languages. Computer Languages, Systems & Structures, 54:386 – 405, 2018.