A toolchain to ease the evaluation of abstract automotive E/E architectures

LAURENT Jean-Baptiste[1], RICHEFEU Julien[2], and MONSUEZ Bruno[3]

[1] Forvia, jeanbaptiste.laurent.pro+jdse@protonmail.com
[2] Forvia, julien.richefeu@forvia.com
[3] ENSTA Paris U2IS, bruno.monsuez@ensta-paris.fr

Abstract. Can automotive E/E architectures be produced by an unified toolchain ? maybe in the future. Yet, we propose a proof of concept implementation demonstrated on an abstract and fictive example to take a step closer to this goal. Even though it is only composed of a couple of components, it managed to hide part of the harsh design complexity, while simultaneously allowing the canvas representing the model to 1), be passed into a simulation framework, and 2) fed back with the results ready for display. All in all allowing the designer to focus on the design exploration without the need to worry about implementation details.

Keywords: Automotive E/E architectures, toolchain, model based design, generative design

1 Introduction

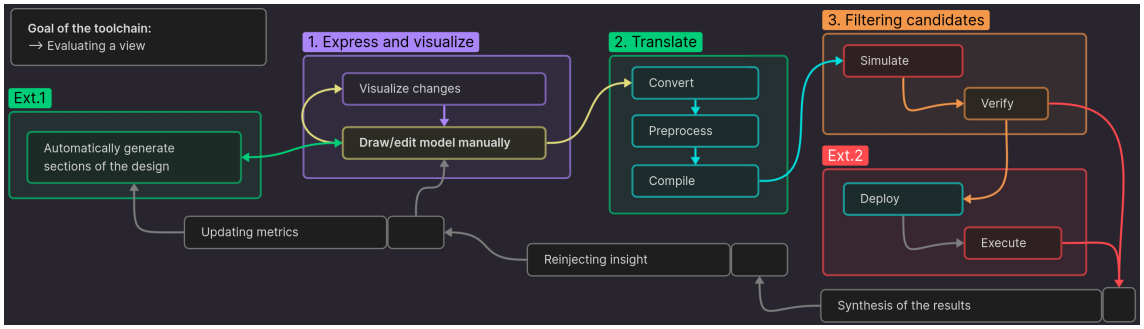


Fig. 1. A toolchain proof of concept to evaluate automotive E/E architectures

Designing the electrical/electronic (E/E) architecture part of a car is, nowadays, rather challenging. It involves the assembling of a complex mix of components made by different manufacturers, each with various levels of constraints such as safety, cybersecurity, and cost to name a few. This is furthermore reinforced by the influx of modern technologies cars have to be compatible with to remain competitive, such as autonomous driving, connected devices or cloud services.

So far, carmakers were content with the distribution of those components on a predefined, skeleton architecture. They were emitting feature requests to fill the blanks, yielding components ready to be plugged out of the box (to some extents, that is). Unfortunately, we arrive at a point where it is no longer possible to develop external features in silo. There is both a growing need for them to communicate with one another, along with the eternal search for cost reduction. For example, with the mutualization of resources.

Digging deeper and creating refined abstraction levels to add to the skeleton design won't be easy, though. In an effort to tame this complexity, we looked into the Metropolis simulation frameworks to run metamodels. That way, we would be able to explore the solution space and converge to a solution while being backed by first-hand measures. Our initial experience with it brought encouraging results, but also warranted the development of the toolchain presented in this paper, fig 1.

A practical use-case gets covered in Section 1, illustrating the partial toolchain we worked on. Section 2 covers results and discusses their implication, while Section 3 opens up with futures works before concluding.

2 Demonstration

2.1 Toolchain composition

The toolchain is composed of 3 parts, along with 2 potential extensions to enhance their potential. A modeling component (1) to express the architecture, a translator (2) to convert the model onto other formats, such as code that will be included by the simulation framework (3). As part of the experiment, a proof of concept implementation was readied for using respectively: The canvas feature of Obsidian [1], a Python script [2], and the Metropolis framework [3] who came out as a good candidate by being selected for simulation on a previous automotive study. [4]

If an implementation is available for the simulated components, then a post-extension (Ext.1) builds and deploys onto a ready physical set of hardware. At last, a pre-extension (Ext.2), upgrades the toolchain onto a generative design process [5] by generating and/or mutating part of the architectural model.

2.2 Use-case description, constraints, and architectural candidates

Our goal for the demonstration is to evaluate architectures able to produce, and display two live video streams on a screen. For the exercise, two constraints were added. Firstly, we restrained the screen to only come with single video input port left available for our own usage, as if it was already in use by other components of the architecture we have no insights on. Secondly, the streams cannot be displayed as-is, and require a converter for operations such as correcting a fish eye effect, and/or demodulation.

As a result, camera(s), converter(s), a merging unit (tasked to regroup the video streams in response to the above constraint) and a screen will be part of the architecture.

It is to be noted that architectural models will be evaluated based on two metrics: monetary cost, and performance in term of latency between the rendered image on screen; and the input streams. The former measured from the model description, the latter by the simulation.

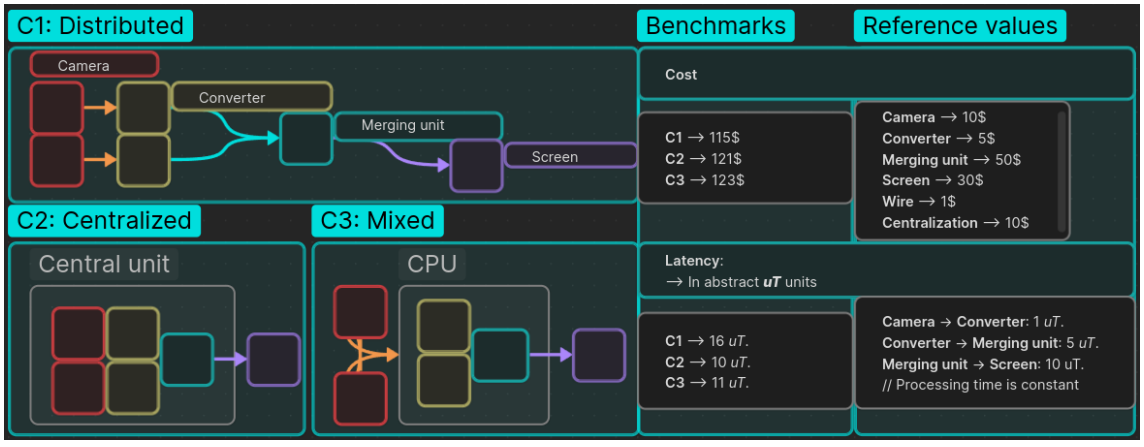


Fig. 2. Evaluation of 3 architectural meta-models with various centralization levels, using arbitrary metrics

We arranged three valid candidates models C1, C2 and C3 (see fig 2). C1 was the first solution we though about and simply consist of a distributed setup: each component has its own hardware. As a result, both cameras stream to their respective converter, which in turn forward the clean stream to a merging computing unit before sending it on screen. Unlike C1, C2 is a fully centralized solution. Even though their functional flow are identical, only a single network link is now needed: between the centralized unit and the screen. Finally, C3 is a middleground solution limiting itself to the centralized of the converters and the merging unit.

3 Discussion

Running the toolchain on the candidates brings results regarding their cost, and performance. Yet, the important portion doesn't come from this evaluation alone, but from the ability to test any variations solely by drawing a new model. This concerns topological changes, the addition or removal of components (if pre-implemented in the simulation framework) or adjusting any cost or performance values to reflect a change in the hardware.

Moreover, third non-expert party (or the client himself) can tweak the canvas as it see fits and apply a range of architectural changes such as: the topology of the system, how strong or weak components are in regards to implemented properties, the kind of network link in use, and/or add new equipment to scale the solution. If the designer manages to draw and run the toolchain on C1, then not much remains to be done for C2 and C3 to work as well.

Beside those early results, a few things are left to consider. Firstly, the presented toolchain remains a proof of concept. More meta components need to be developed to maintain the design exploration freedom, and layers of communication, such as middlewares are yet to be part of the model. Secondly, expressing E/E architecture with all their constraints is a design theory subject on his own, and successfully doing it in a single canvas would need to be part of a dedicated study. Thirdly (and lastly), our toolchain is intended to be a design enhancing work rather than a fully fledged framework. This role, and the decisions associated with it, are still in the hand of the designer in charge.

Lastly, the work presented here is part of a more global methodology. It encompasses the use of the toolchain, and takes care of the how to converge to an automotive E/E architecture by iteratively refining the models until it is fit enough to answer a feature request.

4 Conclusion

Instrumenting the design process makes it easier to converge to a clear solution. With its visual model representation, it reduces the odds of miscommunication between the involved parties while simultaneously allowing untrained people to conduct, to some extent, the solution exploration. This is also a mean to centralized and log the design process to get ride of information redundancy that could otherwise be lost between multiple representations.

It is technically possible to adapt the overall methodology to be used in areas outside the automotive. As long as the designer is able to model the new constraints, electrical and electronic architectures should be in reach. As for now, even though the toolchain prototype capabilities are promising, it is limited by both its simulation, and modeling framework.

We noted that synthesizing the description of automotive E/E architecture into a unified, visual representation is a complex topic on its own deserving a research of its own. In the meantime, work will be conducted into expanding the existing toolchain in order to allow more detailed architectures to be simulated.

References

[1] Shida Li and Erica Xu, *A personal knowledge base and note-taking software application that operates on Markdown files*, en, Mar. 2020. [Online]. Available: `https://obsidian.md/`.

[2] *Welcome to Python.org*, en, Aug. 2023. [Online]. Available: `https://www.python.org/`.

[3] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. Sangiovanni-Vincentelli, "Metropolis: An integrated electronic system design environment," en, *Computer*, vol. 36, no. 4, pp. 45–52, Apr. 2003, ISSN: 0018-9162. DOI: `10.1109/MC.2003.1193228`. [Online]. Available: `http://ieeexplore.ieee.org/document/1193228/` (visited on 11/23/2022).

[4] K. Sri and Z. Haibo, *Exploring Trade-off's Between Centralized versus Decentralized Automotive Architectures Using a Virtual Integration Environment*, 2006. [Online]. Available: `https://ieeexplore.ieee.org/document/1656943` (visited on 07/19/2022).

[5] *Generative design*, en, Page Version ID: 1166246748, Jul. 2023. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=Generative_design&oldid=1166246748`.