X World Conference on Computers in Education
July 2-5, 2013; Toruń, Poland

Teaching Embedded Systems in Higher Education by Visualizing Component Relations

Steffen Büchner, *steffen.buechner@uni-siegen.de*
University of Siegen, Didactics of Informatics and E-Learning, Germany

Abstract
Scientists propose new methodologies to cope with the increasing complexity of embedded systems, while didactic research in this area is still at its beginning. This paper discusses this problem from a didactic and computer scientific viewpoint, suggesting a new approach to mediate the control of complexity for embedded system students.

Keywords
embedded systems, complexity, computational thinking, simulation, understanding

INTRODUCTION

Although embedded systems are mostly used for industrial purposes, they are nowadays progressing into our personal life (e.g. smartphone) too. Technically, embedded systems can't be assigned to one engineering discipline but are instead highly multidisciplinary, requiring skills of computer scientists, electrical engineers, physicists and mathematicians. Among other things, that is one reason why a consistent definition of the term "embedded system" is difficult to find and may change over time.

> "An embedded system is a microprocessor-based system that is built to control a function or range of functions and is not designed to be programmed by the end user in the same way that a PC is".
> (Heath, 1997, p. 2)

> "These are electronic components integrating software and hardware jointly and specifically designed to provide given functionalities, which are often critical. They are hidden in devices, appliances and equipment of any kind: mobile phones, cameras, home appliances, cars, aircraft, trains, medical devices etc." (Sifakis, 2011, p. 1)

As one can see from the citations stated, years ago the understanding of embedded systems and their technical realization (via microprocessors) was rather tangible. The second citation is in that respect vaguer, highlighting the realization independent characteristics like the close interaction between software and hardware.

The examples of possible applications stated give an indication as to why the use of embedded systems is difficult, heterogeneity and thereby complexity. Those problems need to be tackled not only in professional research but in teaching, too. The need for qualified experts in embedded systems development is furthermore crucial for industries' progression with new technologies. In addition, challenges which are common for different types of complex systems, such as embedded ones, may deepen the understanding of computer science overall (see Henzinger & Sifakis, 2006). Future developers must be aware of their responsibilities as the influence of embedded systems on society grows with their area of application.

The paper is structured as follows. The next section contains an overview about efforts already taken to ease the teaching of embedded systems in regards of internship organization, competence research and their limitations for advanced students. The third section emphasizes learning barriers as seen by the author, whereas the fourth section introduces a new metaphor to visualize the relations of system components in an embedded system. The article ends with a conclusion of the themes portrayed and the outlook for future work.

EXAMPLE FOR LEARNING AID

There are several reasons why embedded systems' engineering is difficult to convey. As Marwedel (2011) stated, one of them is the lack of appropriate teaching material and the comprehensive equipment needed for teaching embedded system design.

The University of Siegen has adapted a practical course which takes those considerations into account. It is designed for embedded systems beginners with different subsidiary subjects. The participants realize a home automation with different sensors and actuators by the use of either a microcontroller or a Field-Programmable Gate Array (FPGA).

Due to the amount of teaching material needed, it is absolutely necessary that students prepare themselves before the actual experiment takes place. This preparation is, in most cases, divided into reading task or answering questions on learning management systems like Moodle. However, hands-on experience can only be gained in the lab because solution approaches done at home can't be checked for correctness. The possibility to simulate embedded system behaviour is necessary to overcome this problem. Applications capable of this are characterized by a steep learning curve, which can only be mitigated by the teachers work.

The chair of didactics of informatics and E-Learning has developed a tool to meet this problem (Büchner & Jaschke, 2013). In brief, it is a fully functional virtual operating system, a set of necessary applications, and libraries assembled on a USB flash drive. The remarkable thing is that this tool can be leveraged for preparation purposes as well as for the laboratory task. The tool emphasizes the use of simulations in preparation tasks; for instance in the case of complex FPGA configurations. Students can experiment with their solution approaches and check instantly if the results are correct or at least as expected, before entering the laboratory.

While the tool offers organizational support, it does not directly mediate development competences. One research topic, which is more oriented in that direction, is realized within the DFG funded project "Competence development with embedded micro- and nano-systems" (KOMINA). Within this project several competence structure models have been developed to highlight which competences are important for future embedded systems engineers. The normative competence structure model (NCSM) was the first model obtained by the analysis of module descriptions of excellent rated universities and curricula recommendations (Schäfer et al., 2011). The gained competences have been assigned to four different categories:

- Competences as preconditions,
- Development competences,
- Competences for a multi-level development,
- Non-cognitive competences.

Afterwards the project members created a survey in which experts in embedded systems engineering could rate the importance of every competence description included in the NCSM (very important, rather important, rather unimportant, very unimportant). As a result, all competences have been assigned to six categories of importance (A-F). These ratings influenced the second stage of the model, the so called empirical competence structure model (ECSM) as well as decisions on laboratory design (Schäfer et al., 2012). Those A-rated competences should be of major interest for educators in the field of embedded systems engineering and will also proof to be helpful hereinafter. Other research on this topic exists in various forms from curricula (Caspi et al., 2005) to practical course recommendations (Marwedel, 2005) and more general overviews (Jackson & Caspi, 2005).

LEARNING BARRIERS

One of the main problems in developing embedded systems is their complexity. Similar to huge software systems they contain many components sharing difficult relations among each other. External influences like temperature or humidity alter the runtime behaviour of the system, making predictions very difficult (Sifakis, 2011).

An embedded system for the automotive market will serve as an example. While the hardware components of desktop computers and notebooks usually have operating temperatures from 0 °C to 70 °C, those used in the automotive sector have to operate reliably at temperatures from −55 °C to 150 °C (International Rectifier). This requirement has far-reaching influences on other design decisions. Many conventional electronic components do not satisfy this constraint (e.g. transistors used for desktop computers or notebooks). As a result, many properties from dependability to execution time need to be reconsidered due to new hardware characteristics. To know the cornerstones of the environment in which the embedded system will work is therefore a crucial part of a good development proceeding. Didactic research has found no way to mediate the required competencies. This is not surprising, since even science hasn't found a suitable one-fits-all approach to describe complex component relations with respect to the special requirements for embedded systems till now. Several solution attempts include tools/languages like SysML or Simulink/Matlab. As to be seen later, those are not completely suitable for the sensitization of students for impacts of design decisions in embedded system development.

The urgency to mediate requirements and component relations of embedded systems is also proven lately by the ECSM results mentioned earlier. One of the A-rated competence descriptions depicts the stated demand:

> "They know the special edge conditions of the design of embedded systems."
> - Important: 57.6 % (very), 28.8 % (rather), Unimportant: 13.5 % (rather), 0.0 % (very) -

This is important because choosing a development or design methodology means to know and appraise crucial parameters of the system to be created. Inspecting typical relations between system attributes may therefore be a way to lay a common ground for the understanding needed to apply state-of-the-art development methodologies. This argumentation is in line with the demand stated by the ARTIST education group. They recommend teaching general principles instead of specific tool usage, because the latter is very likely to be obsolete tomorrow (Caspi et al., 2005).

The next section of this paper introduces a new approach to visualize and explore the mentioned relations of embedded systems, enabling the students to inquire the behaviour of an embedded system.

VISUALIZING RELATIONS IN EMBEDDED SYSTEMS

The knowledge about component relations in embedded systems is fundamental for the perceiving of new development methodologies and their differences to traditional approaches. Complexity may be highlighted by visualizing parts of a system and their relations between each other as the variety of graph-based approaches proves (e.g. Mcgrath, Krackhardt & Blythe, 2003). Other types of modelling include, for instance, interaction, composition or requirement diagrams. While those types of visualization are very useful for programming and design, they lack the ability to show the impact of design decisions on the system. Instead they describe the behaviour of system components. The proposed approach makes the former possible by enabling the students to alter system parameters to see the effect which those changes entail. In this way, students explore the systems behaviour by constructing their own understanding of it. While this approach is no new design methodology it can be used to introduce students to the special requirements of embedded systems.

The Metaphor

The approach makes extensive use of sliders to visualize component relations inside a system. Every attribute of its components may be described by a slider consisting of the components name and the unit of measurement. A range is needed to change the actual value within its bounds. This range is denoted with a minimum and a maximum value of the components' attribute. Obviously, by using the slider, the actual value can be changed. If relations between sliders exist, changing one value will automatically change the others, too.

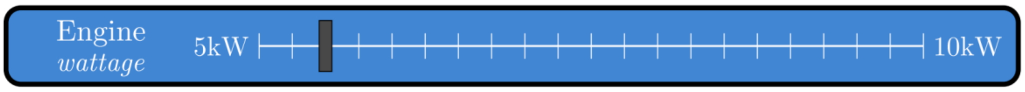


Figure 1: Slider for a component named engine with its power denoted in wattage and the actual design-value.

The slider is a well-known instrument to parameterize properties and therefore can be used to display and change values. While the traditionally used UML diagrams can be much more powerful in regards of description and distinction, they usually can't be executed nor simulated. In addition, the power of UML comes at the cost of a rather complex syntax and a sometimes confusing amount of diagram types available, making its use difficult for embedded system beginners. An often used alternative would be a graph based approach which for complex systems oftentimes results in a network. The approaches' advantage, the full account of all relationships in a system assembled in one image, is at the same time its biggest disadvantage. Single relations out of complex networks are oftentimes unrecognizable due to the large amount of overall relations. Like UML diagrams it serves viewing and not simulation purposes. This distinguishes the metaphor from UML and graph based solutions. Two or more component attributes, sharing a relation, effect each other's values if one of them is changed. By changing system properties through the sliders, the approach makes use of the Law of Common Fate, known from “Gestalt-principles”.

> Things that are moving in the same direction appear to be grouped together. (Mcgrath, Krackhardt & Blythe, 2003)

Nesbitt and Friedrich (2002) observed that this perception applies to objects as well which move in different directions at the same time. The proceeding for the approaches' application is the following:

1. **The system description** is the starting point for the approach and is in most cases given. An example can be seen later.
2. **The extraction of components** and attributes is a step known from most UML-based modelling techniques. The system description given in step one is divided in functional components. For embedded systems, two major types of components - hardware and software - can be distinguished. When a component is derived, one needs to add task important attributes additionally (e.g. power consumption, execution speed).
3. **Inserting simple relations** is mandatory for the approach to work. Those connections oftentimes only effect two of the components derived in step two.

Each of those can have multiple attributes which may be related to each other. The most common observation will, however, be the relation to other components.

4. **Exploring the relations** visualizes the path of dependencies which may implicitly affect the whole system. While the first three steps are common for most modelling techniques, the simulation highlights the far reaching consequences of dependencies.

Before a single relation can be modelled, the components involved have to be parameterized. This is not an easy task for students of embedded system engineering, because they need to assemble the hardware- and software-view into a system-view. For example, students have to pass the first obstacles by choosing an appropriate scale for the description of components (Step 2). Questions like the following may be asked by software biased students, in regards to a simple temperature measurement task:

1. How high is the resolution of a typical Analog/Digital converter (ADC) needed to read temperature?
2. What kind of unit is the temperature sensors' output (V, Ω, °C)?
3. What temperature do I expect to be measured?
4. How fast can measurements be updated?

In contrast, hardware biased students may struggle when defining logical elements like the checksum implementation for a communication protocol or similar tasks also performed in step 2. Those questions widen the view of students and introduce them to unfamiliar concepts of electrical engineering, computer science, or physics.

Two or more component attributes, sharing a relation, effect each other's values if one of them is changed. Those are - at first - only simple connections like the energy provided by a power supply and the energy needed by an engine (Step 3). If the task requires a stronger engine, there may be a need for a stronger power supply too. However, it has to be noted that dependencies may work in both or in only one direction. For the example given, the latter would be the case, because a stronger power supply may not result in the need for a stronger engine.

While those relations are quite simple, they can stack up to complicated ones, resulting in unnoticed dependencies of diverse components. The approach visualizes those relations without the need for line-based references, which oftentimes clutter the representation (Step 4). By moving one slider, others move too and thus reveal a relation to the attribute changed. Thus, a chain of simple relations form complex ones.

Example of Use

Temperature measurements are needed for quality assurance in many different areas. The following example depicts a possible application:

> A hub for food has to take special measurements to monitor the temperature of the products handled. Every packaging unit therefore contains an embedded system consisting of a temperature sensor and a microcontroller attached to it. For service purposes the temperature history has to be logged as long as the unit is in the building. Whenever a critical value is reached, the unit emits an audio signal.

According to Step 1, the system description is the starting point for further investigations. Temperature measurements can be done using different techniques, such as laser measurement, infrared measurement or with the aid of physical components like temperature sensors. For the sake of simplicity the latter is chosen for this example. In this case, the following parts are needed to perform the described task:

Table 1: Components needed for temperature measurement.

Physical	**Logical**
• Temperature sensor • Micro-controller • Analog/Digital converter • Reference resistor • Circuit board & fasteners • Power supply	• ADC interface • Storage interface • Temperature retrieval

As one can see from table 1, oftentimes components can be derived from physical and logical sets. Physical components are for example actors (e.g. engines, servos) and sensors (e.g. temperature, humidity, pressure). In some way those components represent the hardware parts of the system. This includes components such as Analog/Digital converter, multiplexer, relays, transistors, LEDs etc. The core of this group is in most cases located in a micro-controller, FPGA, or comparable device.

Logical components regulate the behaviour of the physical ones. This includes non-tangible components like software controlling systems, communication modules, protocols or interfaces.

Each of the named components has different attributes which, depending on the importance for the overall system function, have to be considered. In regards to the example given, especially the attributes for measurement resolution, measurement speed and measurement accuracy are of interest. Those are the first three sliders which have to be added (see figure 2).

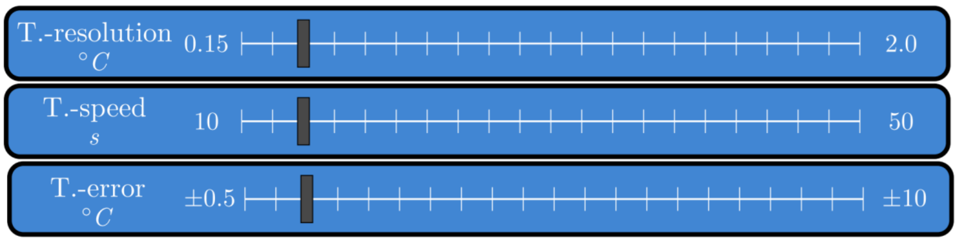


Figure 2: Slider for temperature-resolution, temperature measurement speed, and maximum temperature error

These depend heavily on attributes of other system components described in table 1. For instance, how fast a temperature measurement can be performed is determined by the conversion speed of the ADC as well as the thermal time constant of the temperature sensor itself. In addition, the way the ADC is interfaced is as much important because it specifies the number of conversions needed for one averaged value.

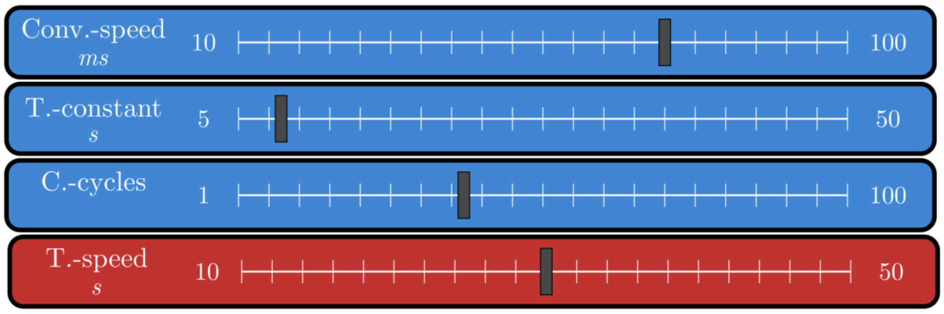


Figure 3: Effects on temperature measurement speed (red) with three sources (Conversion-speed, thermal time constant and conversion cycles)

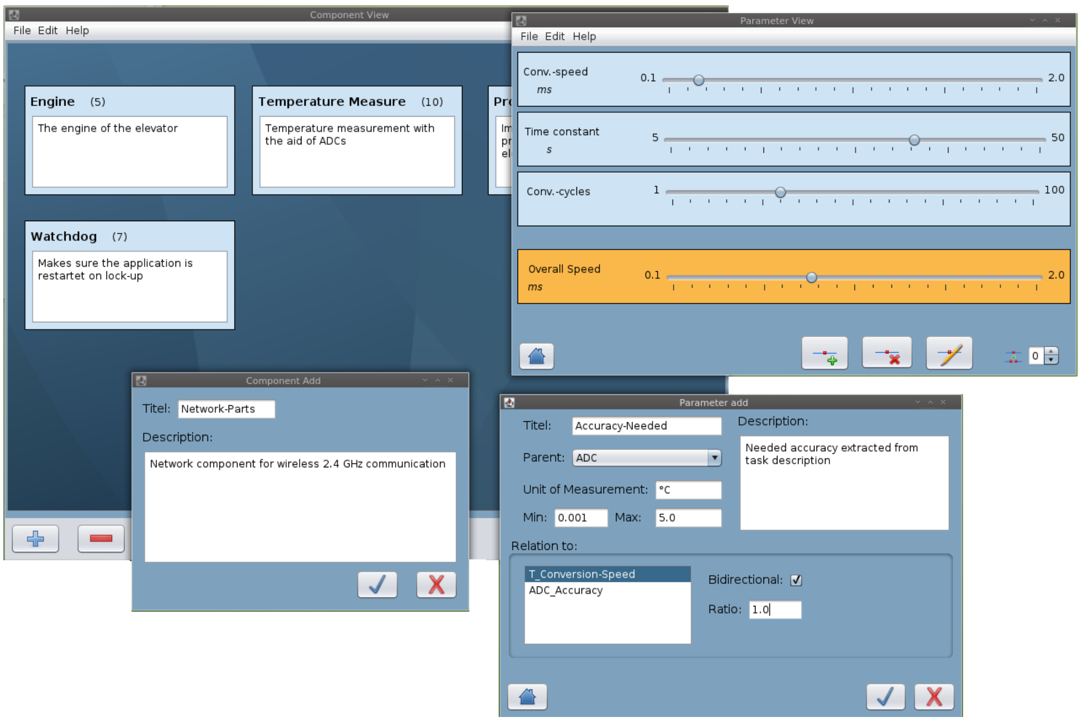
In embedded systems development, the appreciation of different solutions to a problem is important. As one can see from the example given in figure 3, the measurement speed can either be improved by changing the ADC interfacing to do less conversion cycles or change the slow temperature sensor for a faster one. However, by moving the corresponding sliders this may result in a greater error for each measurement. By inquiring both solution approaches to the problem, the students can check which trade-off is superior.

It is to mention that relations are in most cases not linear. For instance, if the food hub example would be extended to include two embedded systems streaming the temperature history over air, the whole infrastructure has to be rethought (e.g. Collision Detection, Checksums, Multi-Master mode). Every of such units above the two already realized, would not result in far-reaching changes because the mentioned extensions have then already been implemented. This means, that the students and teachers have to interpret the changes and determine critical points for the design.

A tool which implements this metaphor has already been built (see figure 4). It includes simulation capabilities between component attributes using the visualization via the Law of Common Fate. Different attribute relations like ratio and direction can be used to specify component dependencies more in detail. For the sake of a better usability, a component view was added.

CONCLUSION

After taking a closer look on current research work, the paper presented the progress of embedded systems teaching in higher education and introduces a novel approach to mediate the complexity of component dependencies usually found in such systems. Unlike industrial tools, the proposed metaphor makes use of "Gestalt-principles" to reveal component relations and their impact on design decisions. The metaphor is adoptable in regards to detail and size and can thus be used for different learning situation. While this approach is no new design methodology it may be used to introduce students to the special requirements of embedded systems. A tool which implements the approach has already been built using the Java programming language.



Engine (5)
The engine of the elevator

Temperature Measure (10)
Temperature measurement with the aid of ADCs

Watchdog (7)
Makes sure the application is restartet on lock-up

Conv.-speed *ms* 0.1 2.0
Time constant *s* 5 50
Conv.-cycles 1 100
Overall Speed *ms* 0.1 2.0

Titel: Network-Parts
Description:
Network component for wireless 2.4 GHz communication

Titel: Accuracy-Needed
Parent: ADC
Unit of Measurement: °C
Min: 0.001 Max: 5.0
Description:
Needed accuracy extracted from task description
Relation to:
T_Conversion-Speed
ADC_Accuracy
Bidirectional: ✓
Ratio: 1.0

Figure 4: Temperature measurement example implemented with the software.

FURTHER WORK

With respect to the recommendations proposed by the European Network of Excellence Artist, students should get the option to display the component relations in other views like UML to transfer their gained knowledge to those representations (Caspi et al.,2005). A long time goal to mitigate this transition would be the development of a didactic model which describes the strength and weaknesses of the most commonly used design techniques and draw up a way to link them together.

While the proposed approach provides a good way to explore the behaviour of a given embedded system, much has to be researched in the way it is used. Many scenarios for the integration in courses are conceivable. For instance, teachers could instruct groups of students to align systems' parameters to achieve a specific target (e.g. minimum of money required, best trade-off between speed and accuracy) and thus use the tool for challenges or games. Another important topic for further work is how the gained understanding can be summarized, structured and discussed. Because medical education oftentimes relies heavily on the use of simulations, researchers have taken up the concept of debriefing to answer similar questions (see Fanning & Gaba, 2007). Whether this proceeding can be applied to embedded systems education or not is topic of future research.

REFERENCES

Büchner, S. & Jaschke, S. (2013). Preparation for Embedded Systems Laboratories The Virtual Workspace Approach. *IEEE Global Engineering Education Conference 2013*, 171-175 (in print)

Caspi, P., Sangiovanni-Vincentelli, A., Almeida, L., Benveniste, A., Bouyssounouse, B., Buttazzo, G., Crnkovic, I., …, Yi, W. (2005). Guidelines for a graduate curriculum on embedded software and systems. *ACM Trans. Embed. Comput. Syst.*, *4* (3):587–611

Fanning, Ruth M. & Gaba, David M. (2007). The role of debriefing in simulation-based learning. *Simulation and Healthcare, Vol. 2* (2)

Heath, S. *Embedded Systems Design* (1997). Butterworth-Heinemann, Newton, MA, USA, 1st edition

Henzinger, T. A. & Sifakis, J. (2006). The embedded systems design challenge. In *Proceedings of the 14th International Symposium on Formal Methods* (FM), Lecture Notes in Computer Science, pages 1–15.

International Rectifier. Automotive level qualification requirements for ic products. Retrieved from http://www. irf.com/product-info/reliability/qraic.pdf.

Jackson, D. J. & Caspi P.(2005). Embedded systems education - future directions, initiatives, and cooperation. *ACM SIGBED Review, 2*(4):1–4

Marwedel, P. (2005) Towards laying common grounds for embedded system design education. *ACM SIGBED Review*, *2*(4):25–28

Marwedel, P. (2011) Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems. *Embedded Systems.* Springer Science+Business Media B.V, Dordrecht

Mcgrath, C., Krackhardt, D., & Blythe, J. (2003). Visualizing complexity in networks: seeing both the forest and the tress*. Connections 25*(1). p. 37-47

Nesbitt, K. V. & Friedrich, C. (2002). Applying gestalt principles to animated visualizations of network data. In *Information Visualisation*, pages 737–743

Schäfer, A., Brück, R., Büchner, S., Jaschke, S., Schubert, S., Fey, D., Kleinert, B., Schmidt, H. (2012): The empirically refined competence structure model for embedded micro- and nanosystems, *Innovation and technology in computer science education - ITiCSE '12*. ACM Press

Schäfer, A., Brück, R., Jaschke, S., Schubert, S., Fey, D., Kleinert, B., & Schmidt, H. (2011). A normative competence structure model for embedded micro- and nanosystems development. In *Proceedings of the 16th annual joint conference* *on Innovation and technology in computer science education - ITiCSE'11,* ACM Press

Sifakis, J. (2011). A vision for computer science - the system perspective. *Central European. Journal Computer Science*, *1*(1):108–116

Biography

Steffen Büchner is a member of the Chair of Didactics of Informatics and E-Learning at the University of Siegen, Germany. His main research interests are in the area of competence research for the proceeding in embedded system development.