Common Component Market: even your mother will want one

Boom! I close the door and get home. While leaving my bag in the closet, I hear a voice calling me from the kitchen.

- Son, are you there? – asks my mom.
- Yes, it's me, I answer, while going there.
- Sit down here, and stay with me, while I prepare a lunch for you, says my mom.

My mother, nostalgic as she is, asks me if I remember what today's date is. As I usually forget, I try to think quickly if it is Mother's Day or her birthday; I quickly realize that it is none. Without success, I tell her that I can't remember anything special and apologize.

Then, she tells me that I should be more attentive, because today is a special day: it's been twenty-two years since she had made a nine hour trip and the reward was watching me to be awarded my Master's degree in Computer Science.

Surprised, I begin adding up the numbers and realize that it is true. At the same time, I realize that, despite all these years all that MSc process is still alive in my mind.

Kidding about it, my mother say kidding that she is able to remember about dissertation, and I am not. Next, she suddenly asks me what the work was about and why I spent so much time to conclude it. By the way, it was exactly one year and eleven months.

I try thinking of a short explanation for an Approach for Distributed Component-Based Software Development, in some minutes. At this moment, flash through my mind frameworks, design patterns, middleware, the MVCASE tool…at last, I give up.

- I say, Mom, it was more or less like this, in short: from since client requests, I built software components, that, were next in application development. My mom's face wrinkles in disapproval.

When I try thinking of a better explanation, she continues:

- Yes, so are you trying to say that you created something once, these so called "components", and then you created again another things, that you call applications? Why do you have to create twice to do only a thing?

So, I think to take back a particularized explanation, when, unexpectedly, another question is raised:

- And, what do you mean by "components"?

So, the problem becomes worse, I think. Why does my mother get curious like this? Ok, mother is mother. However, if forty-five experts took eighteen months to define the software component concept [1], how can I explain this to my mother now? No doubt, she is becoming worse than my Master's defense committee.

I start thinking in components definitions, interfaces, plug-ins, when I say:

- Components, mother, are like those toys that I had when I was a child that connected to one another to form something that I wanted, do you remember?

My mother says yes of course that she remembers, after all, who bought all those toys?

Software Components in the Common Component Market (CCM)

In the Common Component Market (CCM), the definition of software component used is according with [1]: "Software component is a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard".

In addition to a software component description, the CCM provides sufficient descriptive documentation to enable a consumer (company) to assemble the component into a target application. Its description, depending on the consumer's type, goes from requirement specifications until source code and test cases.

Next, I say that the components are used for this. They get together and connect to one another, until they form complete software. I look at my mother and I see that she is relieved, when, suddenly, she says to me:

- Are you saying that it is so easy to make software, and yet you spent hours at the laboratory and at home to do that?

I notice a little tone of fun; so, my mother says:

- Thus, even I can make software, and I won't spend as much time as you. Where can I find these components?

I tell her that there were none before the Common Component Market (CCM).

- CCM, what is it? - she asks.

I am surprised with my mother's interest but I am happy having to explain to her what the CCM is, instead of what an Approach for the Distributed Component-Based Software Development is, as I had to do twenty-two years ago.

I look at her and I say:

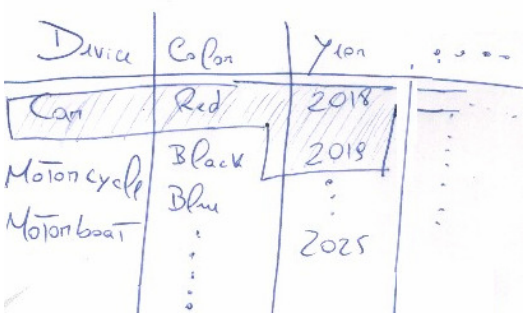
- The CCM is a large grocery store, where the members can search for components and their corresponding documentation, according to their needs and then devise own software.

My mother smiles and asks:

- Simple as doing my groceries?
- It is similar, mom. You access the CCM and you give your informations. Based on it, the CCM opens the doors, where you can inform what you want or look for specific shelves.

My mother, interested, asks, while she is preparing lunch:

- What do you mean by "inform what you want"?
- It's similar to choosing your car, when you bought one, do you remember? You specified, with your own words, that you wanted either a red or black car, that had been used for two years, with six doors, etc… and it brought a list of options. Then, there were a few more choices and that was it: the car was delivered on time. The same thing happens with components. Based on information provided, the CCM brings the possible components, for example: I can specify to the market that I want components to be executed in a medical system, keeping patient, doctor, empty and occupied bed data, that will be executed in the internet, using a specific program, on a computer with the quantity "X" of memory, with a speed "Y". With such data, it builds a sort of table relating the data and brings me specific components.
- Do you understand? I ask. Look at this drawing, and you will understand.



Common Component Market (CCM)

In the movie “Back to the Future” [2], the hero, Marty Macfly, travels back to 1955 in the Delorean time machine. Macfly’s objective was to change his future. Our proposal will be “simpler”: to know how the component market has been developed in the last thirty-five years and propose an efficient solution.

In 1968, at the NATO Software Engineering Conference, McIlroy [3], in his evolutionary paper “Mass Produced Software Components”, presents his thesis which states that: ”the software industry is weakly founded and that one aspect of this weakness is the absence of a software component sub industry”. The author proposes to investigate mass-production techniques in software, according to some ideas from the industry of construction.

McIlroy idealized, with the components sub industry, to see standard catalogues of routines, classified by precision, robustness, time-space performance, size limits, and binding time of parameters; to apply routines in the catalogues to any one of a larger class of often quite different machines; to have confidence in the quality of the routines; and, at least, that the different types of routine in the catalogue that are similar in purpose to be engineered uniformly, so that two similar routines should be available with similar options and two options of the same routines should be interchangeable in situations indifferent to that option.

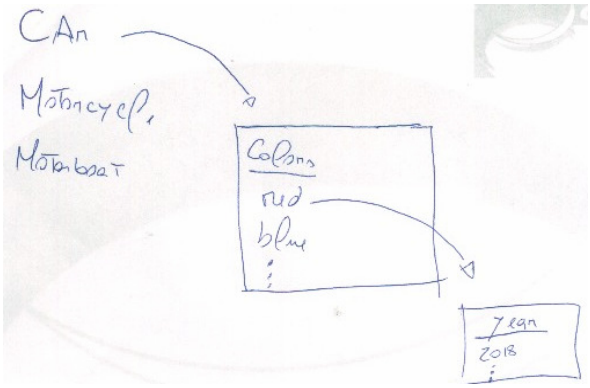
For almost four decades, extensive researches has been carried out and presented in conferences, like International Conference on Software Engineering (ICSE), International Conference on Software Reuse (ICSR) and in journals like Communications of ACM and IEEE (Software and Transaction on Software Engineering), in the area of Component-Based Development (CBD), trying to follow the directions pointed by McIlroy. However, none of these research results have achieved all the goals set forth by McIlroy.

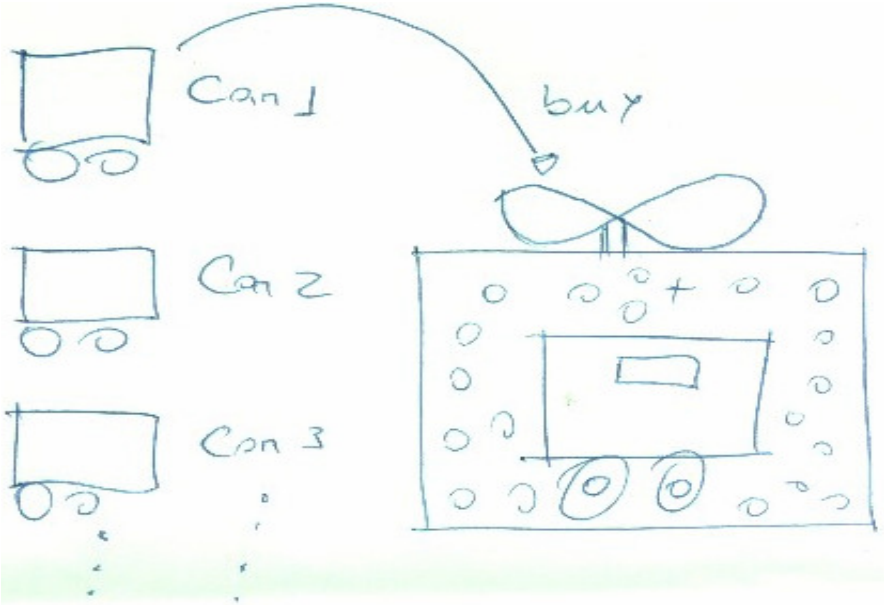
Among the main reasons for the gap between McIlroy’s ideals and the current state-of-the-art, are the software industry lagging behind the hardware industry, in terms of manufacturing principles and catalogues of standard parts; the cultural change in developers, who always use the verb “to build”, instead of the verb “to reuse” (Wrong – “What caching mechanisms do we build?” Right – “What caching mechanisms do we reuse?”); the lack of effective techniques for a couple of familiar purposes; the fact of the current repository systems rarely consider Domain Engineer or Product Line process for development of artifacts, and processes for Software Reuse Assurance before publishing the artifacts for reuse; and, finally, these research results do not consider this piece of McIlroy’s ideas: ”To develop a useful inventory, money and talent will be needed. Thus, the whole project is an improbable one for university research”.

Differently from Marty Macfly we cannot physically go back to the past and change the future. But, anyway, we can make this trip. We have the opportunity to mould the component industry future, learning the lessons since the evolution of the previous works and interaction with industry, and to give one more step to resolving the presented questions with development of CCM.

Like this the process of purchasing a car was done. But, we don’t see this table. You only have seen the images, navigated through the cars and only. Like this other scrawl.

- Ah, it is truth! Now I understand – she answers. But, what about the idea of shelves? - She asked again.





- After choosing from the information, cars are presented (components).
- I prefer this, says my mother. And she continues: I don't really trust these machines. Imagine if it forgets to bring just my car?
- Mom, don't be so extreme.
- Ok, and after that what can I do? asks my mother.
- Then, you bring the necessary components, request execution and it is ready. They automatically connection to one another and the software can be used.
- Oh, my son and why did you spend so much time in the laboratory, working on holidays and weekends, if, with components, making software is so fast?
- Mom, we had not created the CCM.
- Ah, and how did you create this thing?
- Do you remember the hours of study, laboratories, trips to conferences, holidays and weekends? So, let's have lunch. Maybe one day, on the beach, if you have patience, I will explain that to you.

Salvador, Bahia, Brazil, 2025.

References

1. Heineman, G., T., Council, W., T. *Component-Based Software Engineering: Putting the Pieces Together*, Addison-Wesley. 2001.

2. Universal Studios. *Back to the Future*. 1985. Available by Universal Studios URL: http://www.bttfmovie.com. Consulted in July 1, 2003.

3. Naur, P., Randell, B. *Software Engineering*, Report on a conference sponsored by the NATO SCIENCE COMMITTEE. 1968.

EDUARDO SANTANA DE ALMEIDA (esa2@cin.ufpe.br) is PhD Candidate in Computer Science at Federal University of Pernambuco, Brazil.

JONES OLIVEIRA DE ALBUQUERQUE (joa@ufrpe.br) is PhD in Computer Science at Federal University of Minas Gerais, Brazil, and assistant professor at Federal University of Pernambuco.

SILVIO ROMERO DE LEMOS MEIRA (srlm@cin.ufpe.br) is PhD in Computer Science at University of Kent, at Canterbury, England, and full professor at Federal University of Pernambuco, Brazil.