Rapid prototyping of medical device software using CamiTK

Nicolas Saubat - Pierre-Alain Barraud - Emmanuel Promayon - Céline Fouard

Univ. Grenoble Alpes, TIMC-IMAG, F-38000 Grenoble, France
CNRS, TIMC-IMAG, F-38000 Grenoble, France

1 Background

Creating a prototype of a medical device is required to validate clinician and researcher approaches to solve a specific medical problem. The team creating the prototype is usually composed of diverse actors, such as engineers, researchers of different fields and clinicians. Each actor brings its own expertise in the development of the device and may not be an expert of her/his colleague's field.

Prototyping also means trying different ideas to find the one that best answers the medical issues and include all the safety constraints. From a software development point of view, the consequence is that different implementations have to be tested for each feature needed in the medical device. The cost of this step can be very high and leads too often to an early abortion of the project. Nevertheless, it can be significantly reduced by choosing a well suited software architecture using existing open-source libraries and reuse software building blocks. Code reuse also reduces software development time.

Prototyping medical software applications usually requires the software to interface several devices such as camera, robots, and other sensors. This can lead to a situation where all the software parts have highly complex interdependencies and where a change in one specific part leads to a software architecture nightmare.

Several modular open-source medical software application frameworks exist such as 3D Slicer [1], MITK [2] and NifTK [3]. Those frameworks provide both end-user applications and a modular architecture allowing plugins extension. Intended to be used by a research audience, they are well suited to develop an experimental application, often based on the execution of a pipeline, but might lack an easy way to distribute the software to non-researchers or to a non-technical partner, such as a clinician, so that the prototype can be extensively tested.

This paper shows how using the CamiTK framework [4] can effectively organize a team of various competences, develop reusable building blocks based on complex technologies while maintaining a clean software architecture.

CamiTK (Computer Assisted Medical Intervention ToolKit) is a free, modular, cross-platform and open-source framework written in the C++ programming language, based on VTK[1] and Qt[2], created in order to: *a)* accelerate the software development of medical device prototypes (initially in, but not limited to, the CAMI (Computer Assited Medical Intervention) field, *b)* allow each expert to focus on the development of one part of the software and *c)* gather knowledge and expertise, i.e., assemble the expert contribution of each discipline in reusable software building blocks. CamiTK architecture (see Fig. 1, A) is composed of a software core and extensions (i.e., plugins). The CamiTK core goal is to allow any extension to communicate and work with one another. There are four types of pre-defined extensions: *a)* components handle data input/output. Generic (volume) images and mesh component are provided, but any other type of data can also be designed as a component (imaging equipment, biophysical signals, robots, and other sensors), *b)* actions process components and provide data processing. For instance they provide image filtering on volume image, displacement of robots to a given position, *c)* viewers provide Graphic User (GU) interaction and visualization features and *d)* applications provide an easy way to build a program by assembling viewer extensions.

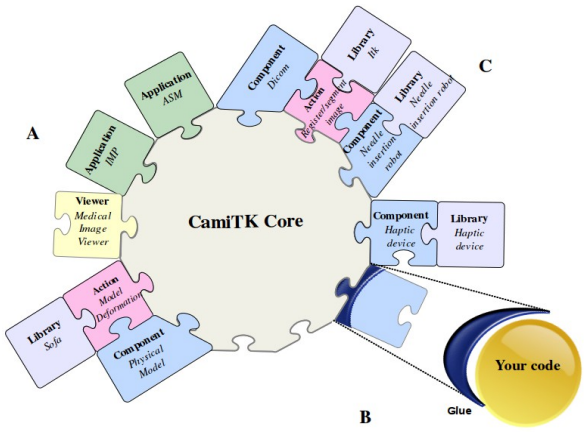

Fig.e 1. A – CamiTK architecture: the core and components (blue), actions (pink), viewers (yellow), applications (green) and libraries (grey).. B – CamiTK "*glue*" allows to create extensions. C – Use-case extensions (see section 2).

When using CamiTK for developing the software associated to a medical device, the CamiTK core (i.e., the CamiTK Software Development Kit) and some already built extensions are available (DICOM image support, image processing filters, biomechanical simulations). Moreover, CamiTK provides a set of tools to easily and quickly develop new extensions (see Fig. 1 B). This "*glue*" allows one to wrap an existing C++ code, e.g., an open-source library, into an extension, to interface it with the core. The CamiTK framework allows a clear separation between expertise with no interdependencies of extensions. CamiTK provides the *camitk-wizard* a specific developer step-by-step application that generates code skeletons and development environment to start writing this *glue*. CamiTK also provides an all-in-one application, called *camitk-imp* for trying, adjusting and testing all extensions. *camitk-imp* is the CamiTK developer workbench (Fig.2).

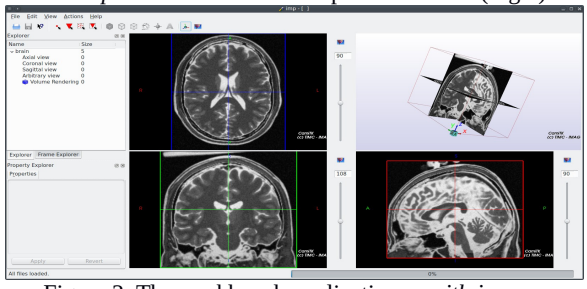

Figure 2. The workbench application *camitk-imp*.

1 http://www.vtk.org/
2 http://www.qt.io/

Once all requirements and features have reached a mature state, CamiTK provides another application called *camitk-asm* (ASM stands for Action State Machine), that assembles these same extensions in an ergonomic wizard-like GU interface, easy way to let clinicians test and validate the prototype using a step-by-step guided process (Fig. 3).

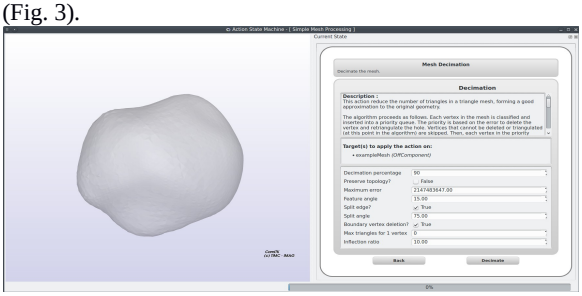

Figure 3. The prototype testing application *camitk-asm*.

The CamiTK framework is published under the LGPL v3 license which allows the developers to choose between a compatible open-source or a proprietary license for their extensions. This is especially interesting for projects with an industrial partnership or intellectual property protection.

2 Use-Case:a tele-operated, image guided robot

To illustrate the process of rapid prototyping in CamiTK, let's consider a research project which goal is developing a tele-operated image guided robot for interventional radiology [5]. The MRI compatible robot has to insert a surgical tool (e.g., a biopsy needle) inside a targeted organ. First, a pre-operative image showing both the patient and the insertion robot is taken, then the clinician specifies the exact position of the target organ on this image and remotely manipulates the robot. He/she uses, for instance a 7-DOF mouse or a haptic device to position the robot adequately and to insert the needle within the patient. Thus, the targeted medical software applications include several features: *a)* an I/O process to open and visualize medical images (DICOM format), *b)* an image analysis algorithm to segment and extract the position of the surgical tool according to the patient, *c)* an interactive visualization system for the clinician to interact with these images (e.g., to choose a target and an entry point for the surgical tool), *d)* the synchronization with a specific interactive device, and , *e)* a system to drive the insertion robot. The high technicality of each sub-system requires to build a team of people from different research and expertise areas.

3 Methods

For each part of the application, CamiTK offers a good support that facilitates and accelerate the prototyping. First, CamiTK natively supports DICOM image I/O (from either MRI, CT scan or UltraSound modality) and volume image visualization and standard interactivity (e.g., using standard mouse). Part *a)* and *c)* described above are therefore straightforward. The haptic device expert chooses a specific device with its proprietary driver API. The *camitk-wizard* allows him/her to generate the code skeleton of the *glue* between CamiTK and this driver. This *glue* is a CamiTK component that handles all the input/outputs from/to the haptic device. The robotic expert develops her/his own library to drive the insertion robot, wrapped it into a CamiTK component, which can be tested independently from the haptic device component. The image processing specialist uses an existing open-source library to segment and register medical images. This *glue* is a CamiTK action applied on medical images, taking the robot component model for registration. Once, all the specialists implemented, tuned parameters and tested their extensions using *camitk-imp*, the exact execution pipeline and its parameters can be written. in an XML document, used as an input by *camitk-asm. camitk-asm* provides a step-by-step interface to play this pipeline. *and* uses the exact same components and actions extensions as *camitk-imp*.

3 Results

Fig. 1 – C shows the different extensions built by the team. In total this use-case project leads to the development of three specific extensions, all independent and reusable as building blocks in other projects.

4 Discussion and conclusion

By creating these building blocks the team saved time, reused existing blocks and gather knowledge for the future: actors can use these blocks for a future project. The next step for the prototype would be to perform clinical studies on patients. This requires ethical comity quality assessment and approval. CamiTK can be considered as a Software of Unknown Pedigree (SOUP). CamiTK offers many requirements for passing this evaluation such as continuous integration system, unit testing, and bug reports), making it a good choice from the development phase to the evaluation phase.

Acknowledgments

This work was supported by French state funds managed by the ANR within the Investissements d'Avenir programme (Labex CAMI) under reference ANR-11-LABX-0004.

References

[1] Pieper S, Halle M, Kikinis R (2004) 3D slicer. In: Biomedical imaging: nano to macro, 2004. IEEE international symposium on, pp 632–635. IEEE, http://www.slicer.org

[2] Nolden M, Zelzer S, Seitel A, Wald D, Müller M, Franz AM, Maleike D, Fangerau M, Baumhauer M, Maier-Hein L, et al. (2013) The medical imaging interaction toolkit: challenges and advances. Int J Comput Assist Radiol Surg, 1–14, http://www.mitk.org

[3] Clarkson, M. J., Zombori, G., Thompson, S., Totz, J., Song, Y., Espak, M., ... & Ourselin, S. (2014). The NifTK software platform for image-guided interventions: platform overview and NiftyLink messaging. *International journal of computer assisted radiology and surgery*, 1-16.

[4] C. Fouard, A. Deram, Y. Keraval, E. Promayon. CamiTK: a Modular Framework Integrating Visualization, Image Processing and Biomechanical Modeling. In Soft Tissue Biomechanical Modeling for Computer Assisted Surgery, Y. Payan (ed.), pp. 323-354, 2012.

[5] Dorileo, E.A.G; Hungr, N. ; Zemiti, N. ; Fouard, C. ; Poignet, P. A modular CT/MRI-guided Teleoperation Platform for Robot Assisted Punctures Planning. In 28th CARS - Computer Assisted Radiology and Surgery, Fukuoka, Japan, 2014.