Using Event-B to Verify the Kmelia Components and Their Assemblies

P. Andre, Gilles Ardourel, Christian Attiogbe, Arnaud Lanoix

Using Event-B to Verify the Kmelia Components and their Assemblies

Pascal André, Gilles Ardourel, Christian Attiogbé and Arnaud Lanoix

COLOSS Team
LINA CNRS UMR 6241 - University of Nantes
{firstname.lastname}@univ-nantes.fr

Component-based software engineering is a practical approach to address the issue of building large software by combining existing and new components. However, building reliable software systems from components requires to verify the consistency of components and the correctness of their assemblies. In this context we proposed an abstract and formal model, named Kmelia [1,2], with an associated language to specify components, their provided and required services and their assemblies; we also developed a framework named COSTO [3] and re-used some verification tools [1,4] to study the Kmelia specifications.

A Kmelia component is equipped with invariants and with pre/post-conditions defined on services. A Kmelia assembly defines a set of links between required and provided services of various components, with respect to their pre/post-conditions. Our main concern is to establish the correctness of Kmelia components and their assemblies. Among the formal analysis necessary to ensure complete correctness, we consider: (i) the component invariant consistency vs. pre-/post-conditions of services; (ii) the Kmelia assembly link contract correctness, that relates services which are linked in the assemblies. We use the notion of contract as in the classical works and results such as *design-by-contract* [5] or *specification matching* [6]: on the one hand the pre-condition of a required service is stronger than the pre-condition of the linked provided service; on the other hand the post-condition of the provided service is stronger than the post-condition of the linked required service. This motivates the choice for using Event-B and the Rodin framework to check the consistency of Kmelia components and the correctness of their assembly contracts, by discharging generated proof obligations.

Figure 1 gives an overview of the necessary Event-B models, generated from parts of the Kmelia specifications we want to verify. We design Event-B patterns to guide the translation and build the necessary proof obligations.

In order to verify the Kmelia invariant consistency rules, we systematically build appropriate Event-B models, by translating the necessary Kmelia elements in such a way that the Event-B proof obligations (POs) correspond to the specific rules we needed to check at the Kmelia level. Three kinds of Event-B models are to be extracted:

- a first Event-B model C_obs corresponds to the observable part of the Kmelia component ;
- another Event-B model (C) is built as a refinement of the previous one C_obs to consider the whole component, not only its observable part;
- for each required service, an Event-B model A_servR is built.

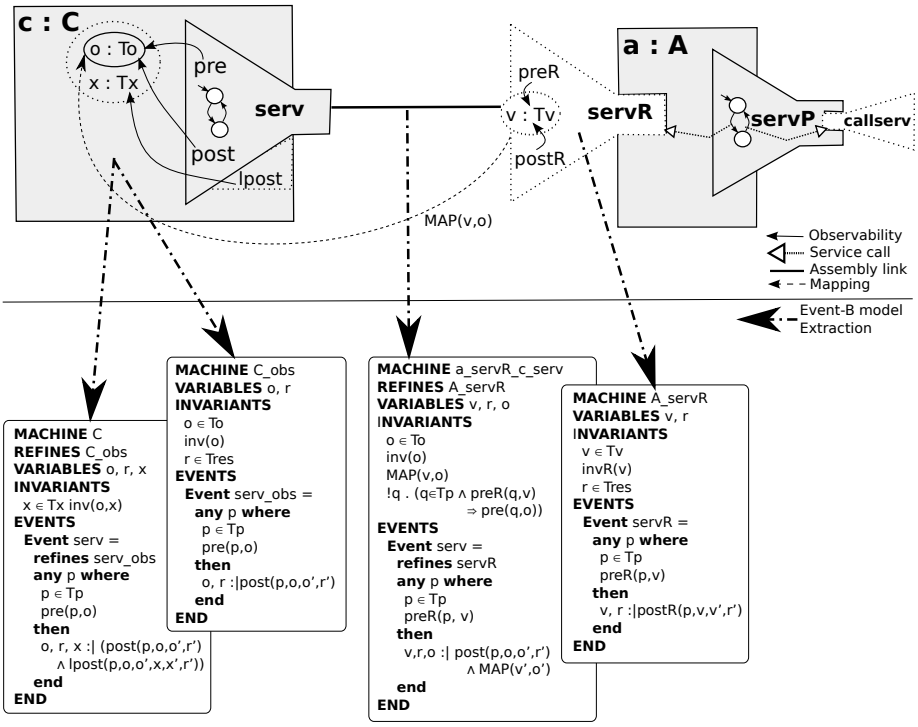

Fig. 1. Event-B Extraction patterns

We describe how the proofs of the Event-B models are linked with the attempted proofs at the Kmelia level. As an illustration, consider the generated POs about the invariant preservation [7] by the event serv_obs:

$$
\begin{array}{l}
o \in To \wedge inv(o) \ \wedge r \in Tres \ \wedge p \in Tp \\
\wedge \ pre(p,o) \quad \wedge post(p,o,o',r') \\
\Rightarrow \quad o' \in To \wedge inv(o') \quad \wedge r' \in Tres
\end{array}
$$

This corresponds exactly to the intended invariant consistency of the observable part at the Kmelia level.

For each assembly link between a required service servR and a provided one serv, we build an Event-B model as a refinement of the Event-B model previously generated for the required service servR. The observable variables of the provided service are added and the invariant is completed with the mapping MAP(v,o). Then Event-B refinement proof obligations are generated and discharged:

1. *Invariant preservation*

$$
\begin{array}{l}
v \in Tv \wedge inv(v) \quad \wedge res \ \in Tres \ \wedge \\
o \in To \wedge inv(o) \ \wedge MAP(v,o) \wedge \forall \, q \ . \ (\ q \in Tp \wedge preR(q,v) \Rightarrow pre(q,o) \ ) \\
p \in Tp \wedge preR(p,v) \ \wedge
\end{array}
$$

$$
\begin{array}{l}
\mathrm{post}(p,o,o',r') \quad \wedge \mathrm{MAP}(v',o') \\
\Rightarrow \\
o' \in \mathrm{To} \wedge \mathrm{inv}(o') \wedge \mathrm{MAP}(v',o') \wedge \forall\, q \;.\; (\; q \in \mathrm{Tp} \wedge \mathrm{preR}(q,v') \Rightarrow \mathrm{pre}(q,o') \;)
\end{array}
$$

With an $\wedge$-elimination, we consider $\forall q.\ (q \in \mathrm{Tp} \wedge \mathrm{preR}(q,v') \Rightarrow \mathrm{pre}(q,o'))$ in the right hand side. Then, the use of $p \in \mathrm{Tp} \wedge \mathrm{preR}(p,v)$ in the left hand side, combined with MAP(v',o') enables us to conclude that $\mathrm{pre}(q,o')$ holds.

2. *Action simulation*

$$
\begin{array}{l}
v \in \mathrm{Tv} \wedge \mathrm{inv}(v) \quad \wedge \mathrm{res} \in \mathrm{Tres} \wedge \\
o \in \mathrm{To} \wedge \mathrm{inv}(o) \wedge \mathrm{MAP}(v,o) \wedge \forall\, q \;.\; (\; q \in \mathrm{Tp} \wedge \mathrm{preR}(q,v) \Rightarrow \mathrm{pre}(q,o) \;) \\
p \in \mathrm{Tp} \wedge \mathrm{preR}(p,v) \wedge \\
\mathrm{post}(p,o,o',r') \quad \wedge \mathrm{MAP}(v',o') \\
\Rightarrow \\
\exists\, v'.\ \mathrm{postR}(p,v,v',r')
\end{array}
$$

These POs establish the Kmelia assembly link contract correctness rules.

The refinement technique of Event-B is used to manage both the structuring of the generated Event-B models and also the proofs to be discharged. Yet we have applied the technique to small and medium size case studies. Using classical B to validate components assembly contracts has been investigated in [8]. Our approach is quite similar with respect to the use of the refinement to check the assembly, but we start from complete component descriptions and target Event-B to prove properties. Compared with existing works, our work contributes at the level of correct-by-construction components and also at the level of the consistency of component assemblies. The results of the current work constitute one more step for rigorously building components and assemblies using the Kmelia framework.

References

1. Attiogbé, C., André, P., Ardourel, G.: Checking Component Composability. In: 5th Intl. Symposium on Software Composition, SC'06. Volume 4089 of LNCS., Springer (2006)
2. André, P., Ardourel, G., Attiogbé, C.: Defining Component Protocols with Service Composition: Illustration with the Kmelia Model. In: 6th International Symposium on Software Composition, SC'07. Volume 4829 of LNCS., Springer (2007)
3. André, P., Ardourel, G., Attiogbé, C.: A Formal Analysis Toolbox for the Kmelia Component Model. In: Proceedings of ProVeCS'07 (TOOLS Europe). Number 567 in Technichal Report, ETH Zurich (2007)
4. André, P., Ardourel, G., Attiogbé, C., Lanoix, A.: Using Assertions to Enhance the Correctness of Kmelia Components and their Assemblies. In: 6th International Workshop on Formal Aspects of Component Software(FACS 2009). LNCS (2009) to be published.
5. Meyer, B.: Applying "design by contract". IEEE COMPUTER **25** (1992) 40–51
6. Zaremski, A.M., Wing, J.M.: Specification Matching of Software Components. ACM Transaction on Software Engeniering Methodolology **6**(4) (1997) 333–369
7. Abrial, J.R., Hallerstede, S.: Refinement, Decomposition, and Instantiation of Discrete Models: Application to Event-B. Fundamenta Informaticae **77**(1-2) (2007) 1–28
8. Lanoix, A., Souquières, J.: A Trustworthy Assembly of Components using the B Refinement. e-Informatica Software Engineering Journal (ISEJ) **2**(1) (2008) 9–28