On Modelling Cloud Native Applications

Kent Inge Fagerland Simonsen

TietoEVRY
Email: kent.simonsen@tietoevry.com

Abstract. Cloud Native Applications (CNAs) have become a popular class of applications intended to run on cloud platforms. In this poster we will explore using a sub-class of Coloured Petri Nets to model CNAs.

Introduction Cloud Native Applications (CNAs) are quickly becomming one of the major application patterns for internet based applications. Although there is no generally accepted definition of CNAs, they do share some common properties [3]. Furthermore, most CNAs are comprised of a limited, albeit evolving, set of component types. Among the components types mentioned in [1] are: Microservices, Severless Applications, Event Hubs, Data Stores, and various Hosted Services. Our goal to use these elements as building blocks in order to model a large portion of CNAs.

In previous works, we have presented a sub-class of Coloured Petri Nets (CPNs) [2] called Pragmatics Annotated CPNs (PA-CPNs) [4].PA-CPNs have been used to model and generate code for network protocols. This is achieved by annotating models with so-called `pragmatics` that informs the code generator what code to generate for a given element. Furthermore, by restricting the modelling language to a set of clearly defined structures on each level of hierarchical models, the code generator is able to put the `pragmatics` in the right context.

Modelling Cloud Native Applications When modelling CNAs, we propose that the top level should include the major components of the CNA. Figure 1 shows how the top level of a simple CNA model might look. Each component is represented by a substitution transition and annotated with pragmatics that describes the service. In the figure we can see a serverless application represented by the `Serverless App` substitution transition. The serverless application is triggered by an http trigger (`http requests`) or by an event trigger (`events`). The serverless application has two outputs, one to a database `Database`, and one to an event engine `Event engine`. The events emitted from `Serverless App` are processed by the `Event engine` component which is annotated with `event_engine`. The processed events are used to trigger the `MicroService App` component, which represents a microservice. Finally, the microservice writes to a second persistence components `File Store` which represents a file system.

Figure 2 shows the page that is represented by the `Serverless App` substitution transition. On this page there are two functions, `Func` and `Durable Func`, that are represented by substitution transitions. The functions are annotated with function type: `function` and `function_orchestration`. Additionally the functions are annotated with with the type of trigger that will cause each of the functions to be executed: `httpTrigger` that triggers on http events and `eventTrigger` that is triggered by receiving an event.

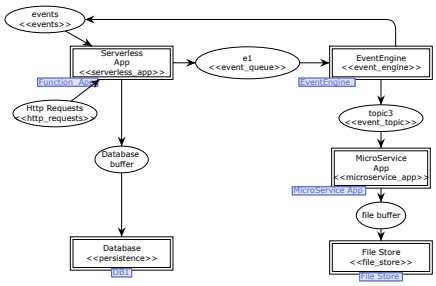

Fig. 1: The top level of an exampl CNA.

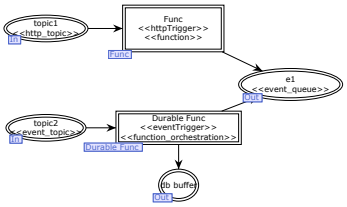

Fig. 2: The top level of an exampl CNA.

Discussion We believe that we can create a subclass of CPNs that allow us to model CNAs by using the ideas and techniques from PA-CPNs applied to CNAs. Furthermore, the models could, similarly to PA-CPNs, be used to generate infrastructure and application code for CNAs without imposing restrictions on the languages and technologies used in the implementation and operation of the CNAs. Finally, the models should also be amenable to formal verification.

In the future, we plan to precisely define the CPN subclass for CNAs, create tools for code generation for both infrastructure and application code and explore possibilities for verification of CNAs using these models. Further on, we hope to be able create tools that are useful for industry to both model and generate application and infrastructure code for CNAs.

References

1. D. Gannon, R. Barga, and N. Sundaresan. Cloud-native applications. *IEEE Cloud Computing*, 4(5):16–21, 2017.
2. K. Jensen and L.M. Kristensen. *Coloured Petri Nets - Modelling and Validation of Concurrent Systems*. Springer, 2009.
3. N. Kratzke and P.C. Quint. Understanding cloud-native applications after 10 years of cloud computing-a systematic mapping study. *Journal of Systems and Software*, 126:1–16, 2017.
4. K.I.F. Simonsen. *Code Generation from Pragmatics Annotated Coloured Petri Nets*. PhD thesis, Technical University of Denmark, 2014.