Code Generation from Pragmatics Annotated Coloured Petri Nets
All electronic communication relies on communication protocols. It is therefore very important that protocols are correct and that protocol implementations are reliable. Coloured Petri Nets (CPNs) have been widely used to model, analyse and verify communication protocols. However, relatively limited work has been done on transforming CPN model to protocol implementations. The goal of the thesis is to be able to automatically generate high-quality implementations of communication protocols based on CPN models.

In this thesis, we develop a methodology for generating implementations of protocols using a sub-class of CPNs, called Pragmatics Annotated CPNs (PACPNs). PA-CPNs give structure to the protocol models and allows the models to be annotated with code generation pragmatics. These pragmatics are used by our code generation approach to identify and execute the appropriate code generation templates. The templates hold the information needed to transform the model to a fully working protocol implementation for a target platform. The code generation approach coupled with PA-CPNs provide a flexible way to perform code generation for communication protocols. The code generation approach has been implemented in a prototype tool called PetriCode.

We defined several criteria for our code generation approach, the approach should be scalable so that it can be used to generate code for industrial sized protocols. The models should be verifiable and it should be possible to perform efficient verification on the models. The approach and the models that are employed for code generation should be platform independent in the sense that it should be possible to generate code for a wide range of platforms based on the same model. The generated code should be integrable meaning that it should be able to use different third party libraries and the code should be easily usable by third party code. Finally, the code should be readable by developers with expertise on the considered platforms.

In this thesis, we show that our code generation approach is able to generate code for a wide range of platforms without altering the PA-CPN model that describe the protocol design. The generated code is also shown to be readable and we demonstrate that a generated implementation can be easily integrated with third party software. We also show that our approach scales to industrial sized protocols by applying our approach to generate code for the WebSocket protocol. The WebSocket protocol creates a message-based two-way channel that can be used by web applications. This allows web applications to communicate with the server much more efficiently than using the traditional request-response pattern for certain application types such as games and rich web applications. Finally, we conclude the evaluation of the criteria of our approach by using the WebSocket PA-CPN model to show that we are able to verify fairly large protocols.
Automated code generation is an important element of model driven development methodologies. We have previously proposed an approach for code generation based on Coloured Petri Net models annotated with textual pragmatics for the network protocol domain. In this paper, we present and evaluate three important properties of our approach: platform independence, code integratability, and code readability. The evaluation shows that our approach can generate code for a wide range of platforms which is integratable and readable.

Implementing the WebSocket Protocol Based on Formal Modelling and Automated Code Generation

Model-based software engineering offers several attractive benefits for the implementation of protocols, including automated code generation for different platforms from design-level models. In earlier work, we have proposed a template-based approach using Coloured Petri Net formal models with pragmatic annotations for automated code generation of protocol software. The contribution of this paper is an application of the approach as implemented in the PetriCode tool to obtain protocol software implementing the IETF WebSocket protocol. This demonstrates the scalability of our approach to real protocols. Furthermore, we perform formal verification of the CPN model prior to code generation, and test the implementation for interoperability against the Autobahn WebSocket test-suite resulting in 97% and 99% success rate for the client and server implementation, respectively. The tests show that the cause of test failures were mostly due to local and trivial errors in newly written code-generation templates, and not related to the overall logical operation of the protocol as specified by the CPN model.
PetriCode: A Tool for Template-Based Code Generation from CPN Models

Code generation is an important part of model driven methodologies. In this paper, we present PetriCode, a software tool for generating protocol software from a subclass of Coloured Petri Nets (CPNs). The CPN subclass is comprised of hierarchical CPN models describing a protocol system at different levels of abstraction. The elements of the models are annotated with code generation pragmatics enabling PetriCode to use a template-based approach to generate code while keeping the models uncluttered from implementation artefacts. PetriCode is the realization of our code generation approach which has been described in previous works.

General information

State: Published
Organisations: Department of Applied Mathematics and Computer Science, Bergen University College
Authors: Simonsen, K. I. (Intern)
Pages: 151-163
Publication date: 2014

Host publication information

Title of host publication: Software Engineering and Formal Methods: SEFM 2013 Collocated Workshops: BEAT2, WS-FMDS, FM-RAIL-Bok, MoKMaSD, and OpenCert, Revised Selected Papers
Publisher: Springer
Editors: Counsell, S., Núñez, M.
ISBN (Print): 978-3-319-05031-7
ISBN (Electronic): 978-3-319-05032-4
Pragmatics Annotated Coloured Petri Nets for Protocol Software Generation and Verification

This paper presents the formal definition of Pragmatics Annotated Coloured Petri Nets (PA-CPNs). PA-CPNs represent a class of Coloured Petri Nets (CPNs) that are designed to support automated code generation of protocol software. PA-CPNs restrict the structure of CPN models and allow Petri net elements to be annotated with so-called pragmatics, which are exploited for code generation. The approach and tool for generating code is called PetriCode and has been discussed and evaluated in earlier work already. The contribution of this paper is to give a formal definition for PA-CPNs; in addition, we show how the structural restrictions of PA-CPNs can be exploited for making the verification of the modelled protocols more efficient. This is done by automatically deriving progress measures for the sweep-line method, and by introducing so-called service testers, that can be used to control the part of the state space that is to be explored for verification purposes.

Towards a model-based development approach for wireless sensor-actuator network protocols

Model-Driven Software Engineering (MDSE) is a promising approach for the development of applications, and has been well adopted in the embedded applications domain in recent years. Wireless Sensor Actuator Networks consisting of resource constrained hardware and platform-specific operating system is one application area where the advantages of MDSE can be exploited. Code-generation is an integral part of MDSE, and using a multi-platform code generator as a part of the approach has several advantages. Due to the automated code-generation, it is possible to obtain time reduction and prevent errors induced due to manual translations. With the use of formal semantics in the modeling approach, we can further ensure the correctness of the source model by means of verification. Also, with the use of network simulators and formal modeling tools, we obtain a verified and validated model to be used as a basis for code-generation. The aim is to build protocols with shorter design to implementation time and efforts, along with higher confidence in the protocol designed.
Code Generation for Protocols from CPN models Annotated with Pragmatics

Model-driven engineering (MDE) provides a foundation for automatically generating software based on models. Models allow software designs to be specified focusing on the problem domain and abstracting from the details of underlying implementation platforms. When applied in the context of formal modelling languages, MDE further has the advantage that models are amenable to model checking which allows key behavioural properties of the software design to be verified. The combination of formally verified models and automated code generation contributes to a high degree of assurance that the resulting software implementation satisfies the properties verified for the model. Coloured Petri Nets (CPNs) have been widely used to model and verify protocol software, but limited work exists on using CPN models of protocol software as a basis for automated code generation. In this report, we present an approach for generating protocol software from a restricted class of CPN models. The class of CPN models considered aims at being descriptive in that the models are intended to be helpful in understanding and conveying the operation of the protocol. At the same time, a descriptive model is close to a verifiable version of the same model and sufficiently detailed to serve as a basis for automated code generation when annotated with code generation pragmatics. Pragmatics are syntactical annotations designed to make the CPN models descriptive and to address the problem that models with enough details for generating code from them tend to be verbose and cluttered. Our code generation approach consists of three main steps, starting from a CPN model that the modeller has annotated with a set of pragmatics that make the protocol structure and the control-flow explicit. The first step is to compute for the CPN model, a set of derived pragmatics that identify control-flow structures and operations, e.g., for sending and receiving packets, and for manipulating the state. In the second step, an abstract template tree (ATT) is constructed providing an association between pragmatics and code generation templates. The ATT then directs the code generation in the third step by invoking the code templates associated with each node of the ATT in order to generate code. We illustrate our approach using an example of a unidirectional data framing protocol.

Generating Protocol Software from CPN Models Annotated with Pragmatics

Model-driven software engineering (MDSE) provides a foundation for automatically generating software based on models that focus on the problem domain while abstracting from the details of underlying implementation platforms. Coloured Petri Nets (CPNs) have been widely used to formally model and verify protocol software, but limited work exists on using CPN models of protocols as a basis for automated code generation. The contribution of this paper is a method for generating protocol software from a class of CPN models annotated with code generation pragmatics. Our code generation method consists of three main steps: automatically adding so-called derived pragmatics to the CPN model, computing an abstract template tree, which associates pragmatics with code templates, and applying the templates to generate code which can then be compiled. We illustrate our method using a unidirectional data framing protocol.
PetriCode: A Tool for Template-based Code Generation from CPN Models

Code generation is an important part of model driven methodologies. In this paper, we present PetriCode, a software tool for generating protocol software from a subclass of Coloured Petri Nets (CPNs) that is the realization of previous work [17, 19, 18]. The CPN subclass is comprised of hierarchical CPN models describing a protocol system at different levels of abstraction. The elements of the models are annotated with code generation pragmatics enabling PetriCode to use a template based approach to generate code while keeping the models uncluttered.

Towards a CPN-Based Modelling Approach for Reconciling Verification and Implementation of Protocol Models

Formal modelling of protocols is often aimed at one specific purpose such as verification or automatically generating an implementation. This leads to models that are useful for one purpose, but not for others. Being able to derive models for verification and implementation from a single model is beneficial both in terms of reduced total modelling effort and confidence that the verification results are valid also for the implementation model. In this paper we introduce the concept of a descriptive specification model and an approach based on refining a descriptive model to target both verification and implementation. Our approach has been developed in the context of the Coloured Petri Nets (CPNs) modelling language. We illustrate our approach by presenting a descriptive specification model of the Websocket protocol which is currently under development by the Internet Engineering Task Force (IETF), and we show how this model can be refined to target both verification and implementation.
Period: 01/03/2011 → 21/11/2014
Number of participants: 6
Phd Student:
Simonsen, Kent Inge (Intern)
Supervisor:
Kristensen, Lars Michael (Ekstern)
Main Supervisor:
Kindler, Ekkart (Intern)
Examiner:
Baumeister, Hubert (Intern)
Daniel, Moldt (Ekstern)
Meling, Hein (Ekstern)

Financing sources
Source: Internal funding (public)
Name of research programme: Stipendie fra udlandet
Project: PhD