Acta Cybernetica **15** (2002) 621–632.

Experiences in Modelling Feature Interactions with Coloured Petri Nets

Louise Lorentsen*, Antti-Pekka Tuovinen† and Jianli Xu†

Abstract

A modern mobile phone supports many features: voice and data calls, text messaging, personal information management like phonebooks and calendars, WAP browsing, games, alarm clock, etc. All these features are packaged into a handset with a small display and a special purpose keypad. The limited user interface and the seamless intertwining of logically separate features cause many interactions between the software components in the UI of mobile phones. In this paper, we present an overview of our approach to modelling feature interactions in Nokia's mobile phones with explicit behavioral models of features. We use Coloured Petri Nets as the modeling language and the tool Design/CPN that provides a graphical, interactive user interface for constructing and simulating Coloured Petri Nets. We describe at a general level how we have created a graphical user interface for controlling and observing the simulations of models through an on-screen mock-up of a mobile phone. Then, we discuss the conrete results we have achieved by using our approach.

1 Introduction

The context of this work is the development of the user interface (UI) software for Nokia's mobile phones. In this domain, the term *feature* means functionality of the phone that is accessible or visible to the user via the UI of the phone. The features are implemented as UI applications in a proprietary mobile phone software architecture. *Feature interaction* means a functional or behavioral (logical) dependency between features. That is features depend on other features to fulfill a service request, or the state of some feature may affect the default behavior or availability of another feature. Interactions between features are necessary and unavoidable, but they can be difficult to control intellectually in the design phase of the UI (conceptual design) and in the implementation and testing of the UI software.

*Department of Computer Science, University of Aarhus, IT-parken, Aabogade 34, DK-8200 Aarhus N, DENMARK, email: louisel@daimi.au.dk

†Software Technology Laboratory, Nokia Research Center, P.O. Box 407, FIN-00045 NOKIA GROUP, FINLAND, email:{Antti-Pekka.Tuovinen,Jianli.Xu}@nokia.com

The development of the user interface software for mobile phones is a concurrent and a distributed engineering process. There is a large family of products that share many common features but have also a high degree of variation in terms of supported features and variation in the UI style (size of display, the number and purpose of buttons and keys). There is also strong pressure for (re)using the same SW components in as many products as possible. To avoid costly delays in the integration phase of a set of independently developed features, it is important to identify and clearly specify the feature interactions at an early stage of the development. Precise descriptions of the interactions are also needed when planning the testing of the UI software.

The problem that originally motivated this research work was that feature interactions were not specified explicitly enough in the UI design specifications and the SW design documentation. This meant that SW designers and UI testers had to go through a lot of documentation to find out which interactions should be implemented and tested. Also, not having a clear understanding of the interactions of a new feature caused problems for managers because feature interactions were considered as one of the primary indicators of the cost of developing the new feature. So, having a good understanding of the interactions of the new feature would help managers to make reliable estimates of the cost (in terms of both time and money) of developing the new feature. Therefore, the goals of this work were:

- identifying categories of interactions that are specific to the domain of mobile phones,
- creating behavioral models that capture the typical feature interaction patterns in each category,
- providing an environment for interactive exploration and simulation of the interaction models, and
- providing input to the development of the UI design specification template used by UI designers to document the UIs of features.

In the core of our approach is an executable behavioral model of the underlying UI architecture and the individual features. As the modeling language we use Coloured Petri Nets. Coloured Petri Nets (CP-nets or CPN)[5] is a graphical modelling language with well-defined semantics that allows simulation of CPN models as well as formal analysis [6]. In contrast to many other modelling languages, CP-nets are both state and action oriented. CP-nets are suitable for modelling concurrent systems and a number of projects have demonstrated their usefulness in modelling and analysis of complex systems [10, 8, 3, 4]. The tool Design/CPN [7] that we use makes it possible to add domain specific graphics for visualisation and interaction purposes.

Note that we did not include automatic detection of feature interactions by analytical means (e.g. by state space analysis) as an explicit goal of this research. We first wanted to concentrate on using visualisations and interactive simulation as the means for human users to observe the behavior of the mobile phone when

features interact. Automatical feature interaction detection methods could then be investigated as follow-up work.

The structure of the paper is as follows. In Section 2 we discuss the types of interactions that are typical in mobile phones. Then, in Section 3 we describe at a general level the modelling approach and the use of domain specific graphics in the interactive simulation of the model. Then, in Section 4, we describe the concrete results we have achieved by using our approach and discuss how the results have been used by Nokia Mobile Phones. The technical details of the model and the generation of the visualisations are given in [9].

2 Types of Feature Interactions

The user-interface (UI) of a mobile phone can be characterized as *task-oriented*. This means that the phone UI is designed to directly support the main functions of the device. This is different from a PC that has a generic UI that supports a much wider range of applications. For example, a phone has keys assigned permanently for answering and ending a call. Furthermore, when browsing the contact information stored in the phonebook, it must be possible to call a person by a single press of a key. This design philosophy reflects the requirements and needs of mobile users, and the physical and economical constraints of the devices.

Each mobile phone product follows a certain UI style ranging from basic (simple) models to more complex business-use oriented models and exclusive fashion products. The style is an important part of the product brand and it has a relatively long lifetime. It describes the physical structure of the UI and the basic mechanisms (keys and their operation) of user interaction. The UI specification of a product defines the features of the product by showing the GUI design and by describing the detailed user interaction for each feature.

A mobile phone is a multi-tasking device: many applications (features) can be active at the same time. Even during a phone call and while talking, the user can activate the calendar or start a game through the in-call menu of the phone. The status of external actors can affect how some features behave in a mobile phone. For example, network status affects some features immediately. Also, the accessories that are connected to the phone change the default behavior of some features. From the UI software point of view, these external actors are independent sources of events; they can issue events that the features have to react to and change their behavior accordingly. Inside the phone, the energy manager (battery status supervisor) can also issue events indicating a critical condition (e.g. 'battery low' -event) that need immediate reaction from some applications (a game is stopped) or the whole phone software system (complete shutdown).

It is difficult to manage the behavioral complexity of feature interactions without explicit models of the interactions. When new types of features and more complex UI styles are introduced in the future, the problems become even more difficult. UI designers, UI testers, and UI software developers all need a solid understanding about the implications that feature interactions have on their work.

2.1 Feature Interaction Categories

Feature interactions come from different sources in a mobile phone. The first category of interactions is the use interaction between features. For instance, the task-oriented user interface design requires that when browsing the phone numbers stored in the phone, a call can be made to a number directly from the browser. This represents an interaction between the 'phonebook' and 'mobile originated call' features.

The second category of interactions comes from the need to share the limited UI resources (screen, keypad) between many features that can be activated independent of each other. Because of the prioritization of the features, important events may interrupt less important activities. For example:

- an incoming call screens phonebook browsing for the duration of the call but the browser application does not know it,
- hang-up key stops search from phonebook (the browser is killed), and
- an incoming call suspends a game, but the game is saved and it can be continued.

The third category involves interactions where one feature affects other features by making them unavailable or by modifying their default behavior. For instance, the 'any key answer' feature makes it possible to answer an incoming call by pressing any key on the keypad and the 'key guard' feature locks the keypad for accidental key presses. The combined effect of these features is that if 'any key answer' is enabled and 'key guard' is on, an incoming call can be answered only by pressing the 'send' (off-hook) key. However, once the call is active, 'key guard' is temporarily disabled during the call and then automatically enabled again after the call. This scenario can be made ever more complex by adding other simultaneous events, for instance 'calendar alarm'.

The use interactions are specified in the UI specifications implicitly but in detail. The use of word 'implicit' means that the use interactions are not called interactions directly; they are described as a part of the flow of user's actions. Use interactions are not so difficult to control during implementation. However, the interactions of the second and the third categories are much more difficult to manage and the behavioral specifications of the interactions are scattered in the specifications of the involved features. Therefore, the focus of our work is on modeling and documenting the typical feature interaction patterns that belong to the latter two categories.

3 Modelling and Visualising Feature Interactions

We do not intend to model the entire phone UI software system. Instead, we concentrate on a representative set of features that have interesting interactions. Also, the main focus of the work described here is on the visualisation of feature interactions rather than on the automatic analysis and detection of feature interactions.

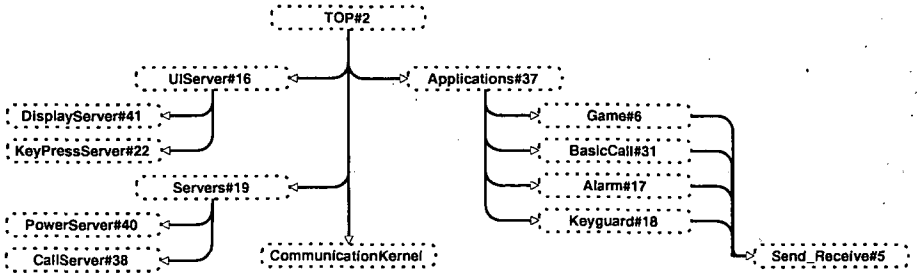

Figure 1: The hierarchy page.

In this section, we first provide an overview of the CPN model. CP-nets will be informally introduced along with the presentation of the model. Then, we explain how we have added domain specific graphics to the model for visualisation and user interaction purposes.

3.1 CPN Models

Figure 1 gives an overview of the CPN model by showing how it has been hierarchically structured into 14 modules (also referred to as subnets or pages). Each node in Fig.1 represents a subnet of the CPN model. An arc between two nodes indicates that the source node contains a substitution transition whose behaviour is described in the subnet represented by the destination node.

The CPN model consists of four main parts that correspond to the four concepts of the phone UI software system: applications, servers, UI controller, and communication kernel. Servers implement the basic capabilities of the phone and Applications implement the behaviour of features by using the services of servers. Applications make the feature available to the user via a user interface; servers do not have user interfaces. Servers and applications communicate by means of asynchronous message passing through the communication kernel. The UI controller manages the user interaction and handles the displays of applications.

The subnet Top depicted in Fig.2 is the top-most page of the CPN model and provides the most abstract view of the CPN model. The page consists of four substitution transitions that correspond to the four parts mentioned above. The detailed behaviour of UIController, Servers, CommunicationKernel, and Applications is modelled in subnets associated with the substitution transitions.

3.1.1 Semantics of CP-nets

A CP-net is created as a graphical drawing with textual inscriptions. A state of a CP-net is represented by means of places which are drawn as ellipses with a name positioned inside. The places contain tokens, which carry data values, or *colours*. Each place has a type (a colour set) which specifies the kind of tokens that the place can hold.

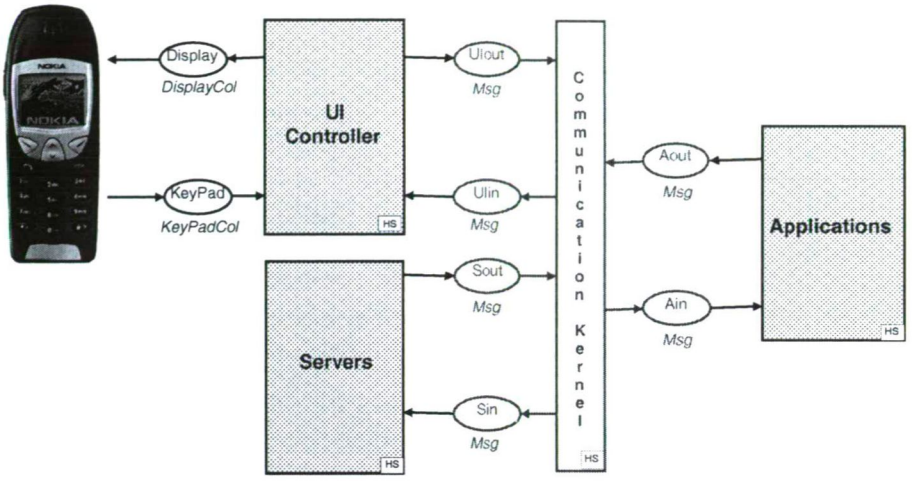

Figure 2: Page Top.

Actions of CP-nets are represented as transitions which are drawn as rectangles with a name positioned inside. The transitions and places are connected by arcs. Transitions remove tokens from places connected by incoming arcs and add tokens to the places connected by outgoing arcs. The tokens removed and added are determined by arc expressions which are textual inscriptions positioned next to the arcs. In the Design/CPN tool, the inscription language is Standard ML. The transitions can have guards that are boolean expressions written in ML.

When simulating a model, a step in a simulation means calculating a new marking for the CP-net. Calculating a new marking means calculating a new assignment of tokens to places as dictated by the firing of a multiset of enabled transitions. Before the transistions can fire, the set of binding elements is calculated first. This means finding all the possible assignments of the tokens in the places to variables (place holders for tokens) in the arc expressions that satisfy the guards of the transitions. Then, a non-conflicting set of binding elements can be chosen to enable a set transitions. Note that a transition can actually be fired more than once during a step depending on the binding elements chosen (see [5] for a detailed definition).

The Design/CPN tool provides ways to control the selection of bindings and the selection of enabled transitions. Furthermore, the tool provides two modes of simulation: interactive and automatic. The interactive mode can be used to debug a CP-net by manually selecting bindings and enabled transitions. In automatic mode, stop criteria can be set to limit the number of steps performed by the simulator.

3.1.2 Extensibility

An important property of the Design/CPN tool is that transitions can have code segments in Standard ML. The code segments are excecuted when transitions fire.

The code segments can employ functions and types imported from user defined ML libraries to create any kind of side effects. For instance, the Mimic library [2] can be used to create animated domain specific graphical representations of the concepts modelled by the CP-nets. Then, the graphical representation can be updated during simulation from the code segments in the transitions to reflect the changes in the state of the model.

The Mimic library provides functions for querying input from the user. That is, the user can point and click the graphical objects on the Mimic windows (called pages). The library captures the raw window events and converts them to ML objects. However, the CP-nets comprising the model must be structured so that they query for user input at the appropriate steps of the simulation. The (blocking) calls to the Mimic query functions are contained in the code segments of transitions.

3.1.3 Modularity

The use of substitution transitions allows the user to relate a transition to a more complex CP-net. The idea is similar to the module concepts found in many programming languages. Substitution transitions can act like macros or functions in terms of how the replacement/substitution is done.

Currently, the model constitutes of 98 page instances. However, there are only 25 different kinds of pages. The applications are modelled using a two or three level hierarchy of pages containing substitution transitions. The state changes of applications are modelled using a generic page that describes a single state transitions in the behavioral model of application. This generic page is hooked up with the input and output message buffers that an application uses to send and receive messages from the UI controller.

Furthermore, CP-nets have the concept of fusion place. This means that the user can specify that a set of places in a CP-net actually represent the same place (object) even though the places of the set are drawn as individual places. So, fusion places have the effect of global variables. Using these two constructs together with Design/CPN's ability to import and export subnets, we have constructed a CPN model of the phone UI software system where features can easily be added and removed. Hence, large parts of the CPN model can be reused in the models of other products with new features.

3.2 Visualisation

An important aspect of our work is the addition of domain specific graphics to the model. This makes it possible to observe, configure and control simulations without interacting directly with the CP-nets. We have made two extensions to the CPN model. First, the state of the phone (as the user observes it) is visualised by an animation of the phone display and the keypad of the phone.

The animation is generated using the Mimic library [2]. So, during an interactive simulation run, the user can press the keys on the animated phone and the

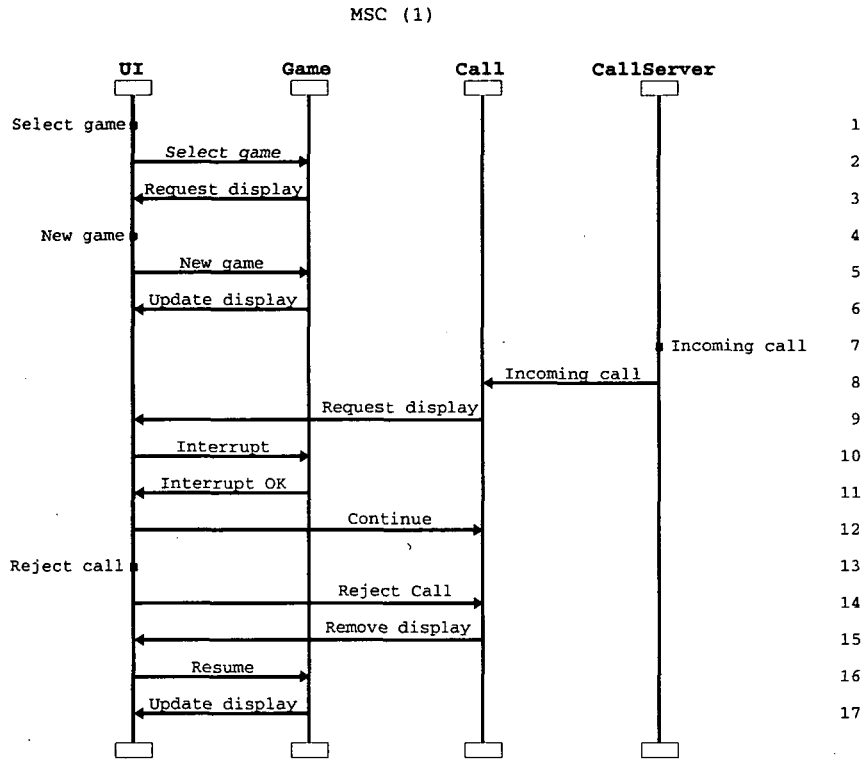

Figure 3: An MSC automatically generated during a simulation run.

underlying model responds correspondingly to the user's actions by changing the display.

Second, the CPN model is extended with Message Sequence Charts (MSCs) [1] that are automatically generated as graphical feedback. The reason MSCs are chosen is that they allow the behaviour of the CPN model to be visualised in a way that is familiar to Nokia's UI designers and software developers.

Fig. 3 shows an example of such an MSC automatically generated during a simulation run of the CPN model. The MSC contains a vertical line for each of the applications and servers in the phone UI software system. The arrows between the vertical lines correspond to messages sent in the system. A mark on a vertical line corresponds to an external event such as an user action. The communication sequence considered corresponds to a scenario where the mobile phone receives an incoming call while the user is playing a game. The scenario demonstrates an interaction between the 'game' and the 'call' features that belongs to the category of UI resource sharing interactions. The scenario consists of the following sequence of events:

- The user selects a game from the menu (line 1)
- The game feature is notified and it requests the display (lines 2-3)
- The user selects a new game (line 4)
- The 'game' feature is notified and it changes the contents of the display accordingly (lines 5-6)
- An incoming call arrives. The 'call' server notifies the 'call' feature (lines 7-8)
- The 'call' feature requests the display (line 9)
- The display is currently in the use of the 'game' feature. The UI controller interrupts the 'game' feature and after the interruption has been acknowledged the display is granted to the 'call' feature (lines 10-12)
- The user rejects the call (line 13)
- The 'call' feature is notified and the display is removed (lines 14-15)
- The 'game' feature is resumed (lines 16-17)

In the scenario above, the UI controller (see Fig. 2) is responsible for interrupting (lines 10-12) and resuming (lines 16-17) the execution of features. So, the features do not have to know which features they potentially interrupt or which features interrupt them. This makes it possible to add and remove features from the CPN model without changing the subnets modelling the rest of the features.

4 Conclusions

4.1 Results

The model provides the basic UI infrastructure where we can plug in features. Currently, the model includes the features listed below. All these features can be simulated by interactive graphical simulation and the corresponding MSCs can be generated during simulation. The MSCs can then be used as static documentation about the events that occurred during simulation and about the messages that were passed in the system during the simulation session.

Idle State
: The Idle State feature handles the mobile phone when it is idle, i.e., is ready to establish incoming and outgoing calls, browse the menus etc.

Menu
: The Menu feature handles the menu structure and browsing of menus of the mobile phone.

Call
: The Call feature handles the incoming and outgoing calls of the mobile phone.

Any Key Answer
: The Any Key Answer allows incoming calls to be answered by the user of the mobile phone by pressing any key of the mobile phone.

Key Guard
: The Key Guard feature blocks the keys of the mobile phone to prevent the user from accidentally pressing keys when not using the phone.

Phone Book
: The Phone Book feature allows names and numbers to be stored in memory.

Profiles and Caller Groups
: The Profiles and Caller Groups feature allows the user to divide the numbers stored in the Phone Book into *caller groups* and assign different ringing tones to each caller group.

Alarm Clock
: The Alarm Clock feature allows the user to set an alarm clock. When the clock expires the mobile phone will indicate the alarm using tones and the mobile phone display.

Power
: The Power feature observes the status of the battery of the mobile phone and indicates when the battery is low or charging.

Game
: The Game feature allows the user to play games.

These features allow us to visualise all three types of interactions described in Section 2. More features were planned to be included in the model (multi-party calls and in-call menu) but these had to be dropped due to the time constraints of the project.

The most valuable result of the modeling work was the understanding of the nature and the causes of feature interactions gained by the people participating in the project. This knowledge was refined and transferred to the customer at the Nokia R&D unit that funded the research.

First, we helped the people responsible for the development of the Nokia mobile phones UI design specifications to add a section about feature interactions into the UI design specification template. The feature interaction section in the new template uses the categorisation described in Section 2. Second, using the MSCs generated from simulating the model, we wrote a document that describes the feature interaction categories with concrete interaction examples. We also identified all the situations caused by events (from internal and external sources) that need to be considered when specifying the UI of a feature.

Naturally, the CPN model and all related technical documentation was also delivered to the customer. The model may be valuable for the future development of the mobile phone UI software architecture. However, from the customer's point of view, the most immediate value of the research comes from the input to the UI specification work.

4.2 Coloured Petri Nets vs. UML models

When we started the work on analysing and modelling feature interactions, we considered also other formalisms than Coloured Petri Nets. The UML statecharts and activity diagrams were also strong candidates. However, we chose CPNs from the following reasons (in no particular order):

- CPNs are both state and action oriented. This gives a choice in the kind of models that are constructed (i.e. data flow or discrete state models).
- The hierarchical structure of nets, the use of separate pages, and the fusion places makes it possible to build modular models using either a top-down or a bottom-up approach.
- Even incomplete models can be immediately simulated (executed) during construction
- Tool support (Design/CPN) for simulation and visualisation of models.
- We had good earlier experiences in using CPNs in modelling and analysing the mobile phone software architecture.

4.3 Final Remarks

The most immediate benefit of our modeling work is the increased understanding of feature interactions. The model provides a sound base to build a body of knowledge about feature interactions in Nokia's mobile phones and how to handle them at the feature level.

We have found that CPNs are well suited for the purpose of modelling feature interactions in mobile phones. As a modelling language, CPNs provides the flexibility and modularity that is needed in the construction of non-trivial models. The extensibility and progammability of CPNs in the Design/CPN tool have been very important properties for our work.

There are several interesting possibilities for future tasks based on the current work. For example, we could link the interaction patterns to existing implementation patterns in the software. This would be achieved through static documentation. Other possible uses of the model include regression testing of the behavior of the phone UI when changing the logic of the features included in the model. The Design/CPN tool gives several possibilities to add support also for automated regression testing. For instance, we could check that invariants expressed as markings (contained tokens) of specific states are preserved (using automatic simulation or state space analysis). In [9], we discuss some initial ideas about automatic detection of feature interactions using the CPN model and some additional tools.

Acknowledgements

This work has been funded by Nokia Mobile Phones. We want to thank Andy Turner, Jyrki Okkonen, Piia Yliranta, and Sirpa Ruokangas for their input.

References

[1] ITU (CCITT). Recommendation Z.120: MSC. Technical report, International Telecommunication Union, 1992.

[2] S. Christensen. *Message Sequence Charts. User's Manual*, January 1997. Available from http://www.daimi.au.dk/designCPN/.

[3] S. Christensen and J.B. Jørgensen. Analysis of Bang and Olufsen's BeoLink Audio/Video System Using Coloured Petri Nets. In P. Azéma and G. Balbo, editors, *Proceedings of ICATPN'97*, volume 1248 of *Lecture Notes in Computer Science*, pages 387–406. Springer-Verlag, 1997.

[4] D.J. Floreani, J. Billington, and A. Dadej. Designing and Verifying a Communications Gateway Using Coloured Petri Nets and Design/CPN. In J. Billington and W. Reisig, editors, *Proceedings of ICATPN'96*, volume 1091 of *Lecture Notes in Computer Science*, pages 153–171. Springer-Verlag, 1996.

[5] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts.* Monographs in Theoretical Computer Science. Springer-Verlag, 1992.

[6] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 2, Analysis Methods.* Monographs in Theoretical Computer Science. Springer-Verlag, 1994.

[7] K. Jensen, S. Christensen, P. Huber, and M. Holla. *Design/CPN Reference Manual.* Department of Computer Science, University of Aarhus, Denmark, 1995.
Online: `http://www.daimi.au.dk/designCPN/`.

[8] L. Lorentsen and L.M. Kristensen. Modelling and Analysis of a Danfoss Flowmeter System. In M.Nielsen and D.Simpson, editors, *Proceedings of ICATPN'2000*, volume 1825 of *Lecture Notes in Computer Science*, pages 346–366. Springer-Verlag, 2000.

[9] L. Lorentsen, A.-P. Tuovinen, and J. Xu. Modelling Features and Feature interactions of Nokia Mobile Phones Using Coloured Petri Nets. In J. Esparza and C. Lakos, editors, *Proceedings of the 24th International Conference on Application and Theory of Petri Nets (ICATPN'2002)*, Lecture Notes in Computer Science. Springer-Verlag, 2002.

[10] J. Xu and J. Kuusela. Analyzing the Execution Architecture of Mobile Phone Software with Colored Petri Nets. *Software Tools for Technology Transfer*, 2(2):133–143, December 1998.