Modelling of Features and Feature Interactions in Nokia Mobile Phones Using Coloured Petri Nets

Louise Lorentsen[1], Antti-Pekka Tuovinen[2], and Jianli Xu[2]

[1] Department of Computer Science, University of Aarhus
IT-parken, Aabogade 34, DK-8200 Aarhus N, Denmark
`louisel@daimi.au.dk`
[2] Software Technology Laboratory, Nokia Research Centre
P.O. Box 407, FIN-00045 Nokia Group, Finland
`{antti-pekka.tuovinen,jianli.xu}@nokia.com`

Abstract. This paper reports on the main results from an industrial co-operation project[1]. The project is a joint project between Nokia Research Centre and the CPN group at the University of Aarhus. The purpose of the project was to investigate features and feature interactions in development of Nokia mobile phones through construction of a Coloured Petri Nets (CPN) model. The model is extended with domain-specific graphics and Message Sequence Charts to enable mobile phone user interface designers and software developers who are not familiar with Petri Nets to work with the model. The paper presents the CPN model constructed in the project, describes how domain-specific graphics and Message Sequence Charts are used in simulations of the CPN model, and discusses how the project and in particular the construction of the CPN model has influenced the development process of features in Nokia mobile phones.

1   Introduction

Modern mobile phones provide an increasingly complex and diverse set of features. Besides basic communication facilities there is a growing demand for facilities for personal information management, data transmission, entertainment, etc. To support flexibility and smooth operation the user interface (UI) of the mobile phone is designed to support the most frequent and important user tasks, hence enabling many features to interplay and be active at the same time. The dependencies or interplay of features are called *feature interactions* and range from simple usage dependencies to more complex combinations of several independent behaviours.

When the project started, feature interactions were not systematically documented at Nokia and often the most complex feature interactions were not fully understood before the features were implemented. In the design and development of features, focus was often on the behaviour and appearance of individual

---

[1] The project is funded by Nokia Mobile Phones.

features. The description of feature interactions was integrated in the description of the individual features involved and did not fully cover or treat feature interactions in a systematic way. This could, in the worst case, lead to costly delays in the integration phase of a set of independently developed features. Therefore, a need for more focus on the feature interactions in the development of features was identified.

The above mentioned problems have motivated the MAFIA[2] project. The main goals of the project were:

1. To increase the level of understanding of the role that feature interactions play in the UI software and its development by identification and documentation of typical patterns of feature interactions.
2. To develop a systematic methodology for describing feature interactions.
3. To provide an environment for interactive exploration and simulation of the feature interactions for demonstrational or analytical purposes.

One of the main activities in the MAFIA project was to construct a model using Coloured Petri Nets (CP-nets or CPNs) [14,17]. The use of CP-nets allowed us to have both a static graphical description of features and furthermore allowed simulation of the models and hence provided an environment for interactive exploration and simulation of feature interactions. CP-nets have been used in previous projects within Nokia, e.g., [23], and were therefore known to the project members in Nokia Research Centre.

The paper is structured as follows. Section 2 presents the MAFIA project and its organisation. Section 3 describes the CPN model constructed in the project. Section 4 discusses how the CPN model has been extended with domain-specific graphics and Message Sequence Charts. Section 5 contains a discussion of related and future work. Finally, Sect. 6 contains the conclusions and a discussion of how the construction of the CPN model has influenced the development of features in Nokia mobile phones. The reader is assumed to be familiar with the basic ideas of High-level Petri Nets [11].

2   The MAFIA Project

The MAFIA project was a joint project between Nokia Research Centre and the CPN group at the University of Aarhus. The project ran from November 2000 to November 2001 with a total amount of 15 man months of work resources. The project group consisted of two people from Nokia Research Centre and three people from the CPN group. Hence, the project group consisted of both experts from the application domain, i.e., development of mobile phones, and experts in the tools and techniques to be applied, i.e., CP-nets and its supporting tool Design/CPN [10].

---

[2] MAFIA is an acronym for Modelling and Analysis of Feature Interaction patterns in mobile phone software Architectures.

Before the joint project started, initial work was done at Nokia Research Centre to experiment with the use of CP-nets for the modelling of feature interactions. The researchers at Nokia Research Centre had practical experience with CP-nets and the Design/CPN tool from other Nokia research projects, e.g., [23]. Hence, the modelling work started immediately in the beginning of the project. When the joint project started, one researcher from the CPN group worked full time at Nokia Research Centre for six months. Other project members from the CPN group provided guidance and technical support on the modelling work.

In the first phase of the project overview information and internal documentation about the features and the mobile phone UI architecture were provided by Nokia and the project group experimented with different levels of abstraction in the CPN model. During the construction of the CPN model, the model was presented and discussed with both UI designers and software developers who were not familiar with Petri Nets. Therefore, already in the first phase of the project the project group decided to experiment with ideas for extending the CPN model with a layer of domain-specific graphics and Message Sequence Charts (MSCs) [13]. This was envisioned to enable the UI designers and software developers to work with the CPN model without interacting with the underlying CP-nets. Interactive simulations were used to validate the models. After the first phase of the project, the CPN modelling proceeded in three iterative steps each containing the following activities.

Refinement and Extension. The CPN model was refined and more features were added.
Validation. The CPN model was validated by means of both interactive and more automatic simulations.
Improvement of Visualisation Facilities. Both the domain-specific graphics and the use of Message Sequence Charts were improved and extended.
Presentation. During meetings with UI designers and software developers the CPN model was presented and its intended use was discussed.
Correction. Based on the evaluation in the meetings with the UI designers and software developers, the CPN model was corrected.

Hence, the CPN modelling proceeded with developing more and more elaborated models, each presented to the UI designers and software developers, i.e., the intended users of the CPN model. At the end of the project the CPN model and documentation were handed over to the users as a part of the results from the project. In Sect. 6 we will discuss how the CPN model relates to the other results from the MAFIA project and how the construction of the CPN model has influenced the development of features in Nokia.

3   The CPN Model

This section presents parts of the CPN model. The CPN model does not capture the full mobile phone UI software system but concentrates on a number

of selected features that are interesting seen from a feature interaction perspective. The purpose of this section is twofold. Firstly, to give an overview of the CPN model and secondly, to give an idea of the complexity and the abstraction level chosen. Section 3.1 presents an overview of the CPN model of the mobile phone UI software system. Section 3.2 describes how the individual features are modelled and how the similarities in communication patterns can be exploited through reuse of subpages of the CPN model.

3.1   Overview of the CPN Model

The CPN model has been hierarchically structured into 25 modules (subnets). The subnets are in CPN terminology also referred to as pages, and we will use this terminology throughout the paper.

The page Nokia, depicted in Fig. 1, is the top-most page of the CPN model and hence provides the most abstract view of the mobile phone UI software system. UIController, Servers, CommunicationKernel and Applications are all substitution transitions which means that the detailed behaviours are shown on the subpages with the corresponding names.

The four substitution transitions correspond to four concepts of the mobile phone UI software system: *applications*, *servers*, *UI controller*, and *communication kernel*.

Applications. The applications implement the features of the mobile phone, e.g., calls, games and calendar. The terms feature and application will be used interchangeably in the rest of the paper.

Servers. Servers manage the resources of the mobile phone, e.g., the battery, and implement the basic capabilities of the phone.

UI Controller. Applications make the feature available to the user via a user interface. The user interfaces are handled by the UI controller. Servers do
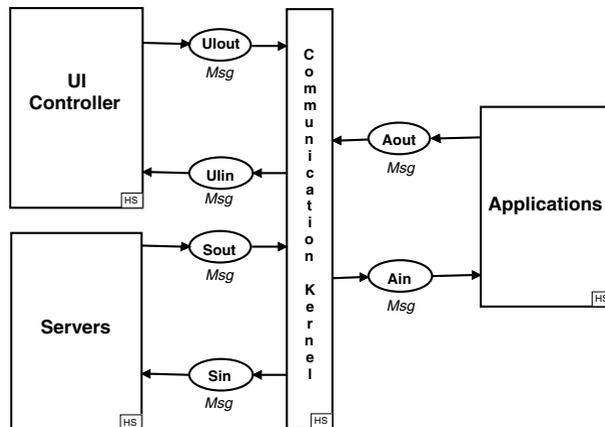

Fig. 1. Page Nokia

not have user interfaces. We will sketch the role of the UI controller by means of an example later in this section.

Communication Kernel. Servers and applications communicate by means of asynchronous message passing. The messages are sent through the communication kernel which implements the control mechanism and protocol used in the communication between the applications, servers and UI controller.

The focus in the MAFIA project is on features and feature interactions. Hence, we will not go into detail on the modelling of the UI controller, servers and communication kernel. However, we will informally sketch the role of the UI controller as it is necessary for the presentation of the CPN modelling of the applications (features).

The main role of the UI controller is to handle the user interfaces of the applications, i.e., to put the user interfaces of the applications on the display of the mobile phone and to forward key presses from the user to the right applications. Often several applications are active at the same time, for example, when the user is playing a game and a call comes in. We therefore need a priority scheme to determine which application will get access to the display and keys of the mobile phone. This is also the role of the UI controller. In Fig. 2 we give a small example to illustrate the operation of the UI controller. The example shows a scenario where the user of the mobile phone starts playing a game. While playing the game a call comes in. The scenario is shown using a Message Sequence Chart (MSC), which is automatically generated during simulation of the CPN model. The MSC contains a vertical line for the user, the UI controller and each of the applications and servers involved in the scenario (here the Game application, the Call application and the Call server). The marks on the vertical line corresponding to the UI controller indicate that the display is updated. A snapshot of the resulting display is shown below the MSC. The number next to the marks and displays indicates the correspondence. Arcs between the vertical lines correspond to messages sent in the mobile phone UI software system. The line numbers in the description below refer to the line numbers in the right-hand side of the MSC.

- The display of the mobile phone when the scenario starts is shown (line 1).
- The user presses the *left soft* key (the key just below the left-hand side of the display) to start a new game (line 2), and the Game application is notified (line 3).
- The Game application requests the display (line 4), the display is updated (line 5), and the Game application is acknowledged (line 6).
- An incoming call arrives (line 7), and the Call application requests the display (line 8).
- The Call application has higher priority than the Game application, hence the game is interrupted (line 9), the Game application acknowledges the interruption (line 10), and the display is granted to the Call application (line 11).
- Finally, the Call application is acknowledged (line 12).

The basic idea behind the model of the UI controller is that the UI controller maintains a structure of the displays (the *display structure*) that the applications
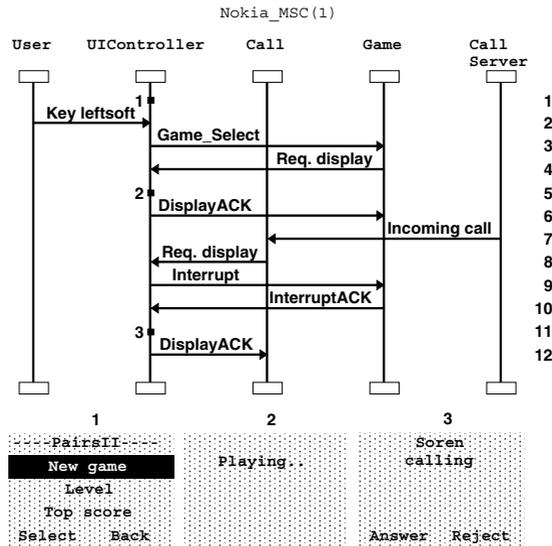
Fig. 2. Operation of the UI controller

in the system have requested to be put on the display of the mobile phone. Due to the limited UI resources of the mobile phone, not all of the display requests can be fulfilled. The displays are assigned a priority to assure that more important features, e.g., an incoming call, will interrupt less important features, e.g., a game. For a given priority the display structure contains a stack of display requests, i.e., if two applications request the display with the same priority the latest request will get the display. Hence, when applications request the UI controller to use the display of the mobile phone, the UI controller will put the display request in the display structure (on top of the stack in the corresponding priority). The UI controller will grant the display to the top of the first, i.e., highest priority, non-empty stack in the display structure. Figure 3 shows an example of the display structure where three priorities are shown: foreground, desktop and background with three, four and two display requests in the corresponding stacks, respectively. The display of the mobile phone will indicate that there is an incoming call (the top element of the stack with foreground priority).

3.2   Modelling of the Features

We will now go into more detail about the modelling of the features of the mobile phone. In the following we will use the Call feature as an example.

Figure 4 shows the detailed behaviour of the page Call modelling the Call feature. The Call feature is an essential feature of the mobile phone and is also the most complex feature included in the CPN model. To enhance readability the Call feature has been divided into a number of subpages. First we will give
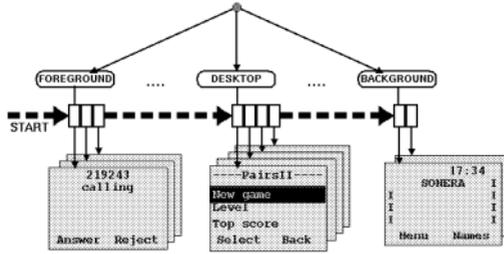
Fig. 3. Stack like structure of application displays

a brief overview of the Call feature, then we will concentrate on the part of the Call feature handling an incoming call.

The two places, Ain and Aout in the upper left corner in Fig. 4, model the message buffers to/from the Call application. It should be noted that there is an arc from Ain to each of the transitions in the page and an arc to Aout from each of the transitions in the page. However, to increase readability these arcs have been hidden in the figure.

The three places Idle, InCall (one active call) and In2Call (two active calls) model possible states of the Call application. Initially, the Call application is Idle. A new call can be established as either an Incoming Call or an Outgoing
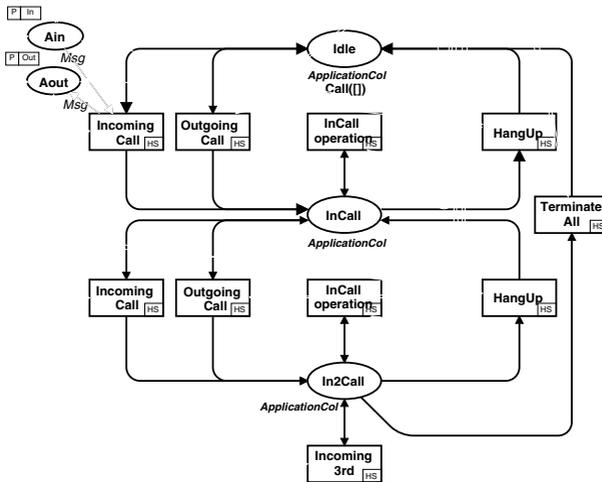

Fig. 4. Page Call

Call making the Call application change its state from Idle to InCall. When the Call feature is InCall, a second call can be established as either an Incoming Call or an Outgoing Call making the Call application change its state from InCall to In2Call. In InCall and In2Call a single call can be terminated (modelled by the substitution transitions Hangup) making the Call applications change its state to Idle or InCall, respectively. When the Call application is In2Call, both calls can be terminated (modelled by transition Terminate All) causing the Call application to change its state to Idle. When the Call feature has two active calls, i.e., is in state In2Call a third incoming call can be received (modelled by transition Incoming3rd). The third call cannot be established before one of the two active calls are terminated. Hence, the Call application will remain in In2Call.

Figure 5 shows the page IncomingCall which models the part of the Call application handling the reception of an incoming call. This is an example of the most detailed level of the CPN model of the features. As before, there is an arc from Ain to each of the transitions in the page and an arc to Aout from each of the transitions in the page. The arcs from/to the two places Ain and Aout have again been hidden to increase readability of the figure.

The places Idle, Indicating and InCall all have colour set ApplicationCol, which denotes an application and some internal data of the application (here the call and the internal data of the call, i.e., the number of the caller and the status of the call). These three places model the possible states of the Call feature related to the reception of an incoming call. We will explain the rest of the places (the three Init places) later.
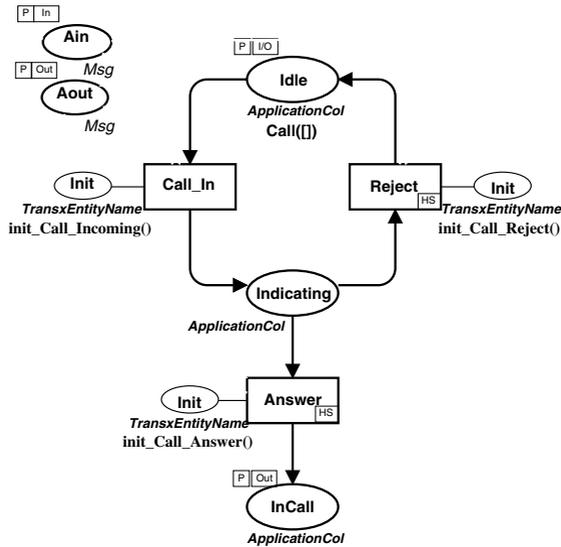

Fig. 5. Page IncomingCall

In Fig. 5 the arrival of an incoming call is modelled by the substitution transition Call_In, which causes the Call application to change its state from being Idle to Indicating. Now the call can be either answered or rejected by the user. The substitution transition Reject models the rejection of the incoming call and causes the Call application to change its state back to Idle. The substitution transition Answer models the answering of the call and causes the Call application to change its state from being Indicating to InCall. All the transitions in Fig. 5 are substitution transitions, hence the detailed behaviour is modelled on separate pages.

In the following we will give a detailed explanation of the substitution transition Call_In modelling an incoming call to the mobile phone. Figure 6 shows in detail what happens when an incoming call arrives in the mobile phone UI software system. The figure is a MSC generated from a simulation of the CPN model. Below we give an textual description. The line numbers refer to the line numbers in the right-hand side of the MSC.

The MSC contains a vertical line for the user, the UI controller and each of the applications and servers involved in the scenario (here the Call application, the Phonebook application, the Any key answer application, the Keyguard application and the Call server). When the simulation starts the phone is idle (line 1).

– The Call application is triggered by an incoming call (via the Call server) (line 2).
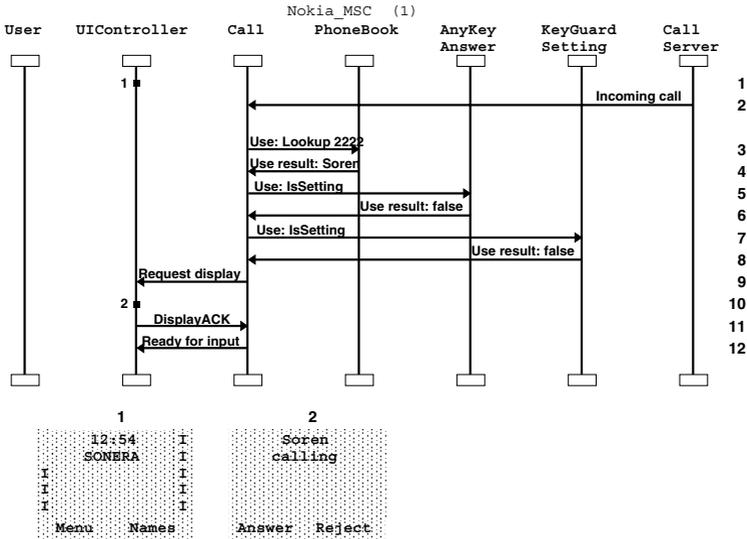

Fig. 6. An incoming call arrives in the mobile phone

- The Call application requests the UI controller, servers, and other applications in the mobile phone UI software system and uses the results to carry out its own task, i.e., to receive and notify the user about the incoming call.
  - The Phonebook application is requested to check whether the calling number is listed in the phonebook (in that case the name of the caller will be written in the display instead of the calling number) (lines 3-4).
  - The Any key answer application is requested to check whether the Any key answer setting is on (in that case the call application will allow the call to be answered using any key of the mobile phone, otherwise the call can only be answered using the *offhook* key, i.e., the key with the red phone) (lines 5-6).
  - The Keyguard application is requested to check whether it is active (in that case the call can only be answered using the *offhook* key independent of the state of the Any key answer application) (lines 7-8).
  - The UI controller is requested to put the user interface of the Call application on the display of the phone and to send information about user key presses to the Call application (line 9). The UI controller updates the display (line 10). Finally, the Call application is acknowledged (line 11). Note that the number was listed in the phonebook (the display indicates that Soren is calling), any key answer was not set, i.e., only information about the *offhook* key should be sent to the Call application (this cannot be seen from the figure).
- The call application changes its state and notifies the UI controller (line 12).

In the above description of the Call application's handling of the incoming call, the events and messages sent in the mobile phone UI software system have been divided into three parts: *trigger*, *request/result loop*, and *proceed*. In fact all state changes of the Call application (as well as the rest of the applications in the mobile phone UI software system) can be characterised by the same three steps. Hence, all of the state changes of the applications are modelled using instances of the same subpage modelling the general communication pattern sketched above. The page modelling the three steps is shown in Fig. 7.

The page GenerelSendReceive contains five *port places*: Start, End, Ain, Aout and Init. The two places Start and End (in the left and right of Fig. 7) are bound to the input place and the output place of the substitution transition (when the page is used for the Call_In substitution transition in Fig. 5, Start and End are bound to Idle and Indicating). Places Ain and Aout are bound to the message buffers to/from the application. Place Init is bound to the place connected to the substitution transition with a thin line (a line corresponds to an arc with arrowheads in both directions).

- The first step (trigger) is modelled by the Trigger transition modelling that the state change of the application is triggered by some event sent to the application via the Ain place causing the application to change its state from Start to Loop.
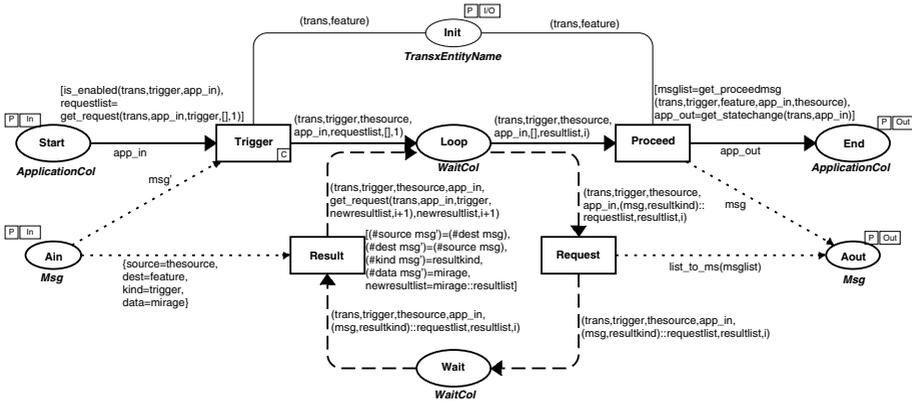
Fig. 7. Page GenerelSendReceive

- The second step (request/result loop) is modelled by the two transitions
  Request and Result (the loop in Fig. 7 with the dashed arcs). Request models
  the sending of a request to another entity in the system (an application,
  server or the UI controller) via the Aout buffer and causes the application
  to change its state from Loop to Wait. When a result appears on place Ain
  transition Result can occur, causing the application to change its state back
  to Loop. The loop can be executed several times until the application has no
  more requests.
- The third step (proceed) is modelled by the transition Proceed modelling
  that a message is sent to the Aout buffer. The application changes its state
  to End, thus modelling a complete state change of the application.

The concrete information about which messages trigger the event, which requests
are sent, how the results are used by the application, and what change in data
are performed, etc. is determined from the (initial) marking of the Init place. The
same page (shown in Fig. 7) is used as a subpage for all of the substitution tran-
sitions modelling state changes of the applications with individual installations
of the corresponding Init places. The reuse of the page shown in Fig. 7 means
that even though the CPN model of the mobile phone UI software system only
consists of 25 different pages it actually contains 98 page instances.

   We have now presented the CPN model of the Call application to give an
idea about the complexity of the CPN model and the efforts required to include
a feature in the model. All features in the CPN model follow the same idea
as the Call feature. Hence, when adding a new feature to the CPN model, a
page modelling the overall state changes is provided together with a description
of triggers, requests/results and proceed messages. This way of modelling the
features reflects the way the features are currently documented in the written
specifications of features in Nokia mobile phones. Here the features are described
using a textual description of the possible states and state changes of the feature
together with a description of the user interfaces and the use of other features and

the reactions of the feature to user key presses. Hence, the CPN model closely follows both the UI designers' and software developers' current understanding of features as well as the current available documentation of the features.

4   Simulations and Visualisation

We model the individual features of the mobile phone using the same ideas as described for the Call feature. The feature interactions are captured in the CPN model as the communication and interaction between the individual features in the CPN model. The UI designers and software developers who are developing new features will use the CPN model to identify and analyse the interaction patterns of their new features as well as existing features. One way of using the CPN model is by means of simulations; both interactively (step-by-step) and more automatically for investigations of larger scenarios. In this section we will present techniques which allow UI designers and software developers who are not familiar with Petri Nets to control and gain information from simulations without directly interacting with the underlying CP-net and its token game.

Two extensions have been made to the CPN model allowing the current state and behaviour of the CPN model to be visualised during simulations. Firstly, the state of the phone as the user observes it on the handset is visualised via an animation of the display. Secondly, the CPN model is extended with Message Sequence Charts (MSCs) which are automatically constructed as graphical feedback from simulations. MSCs were chosen to visualise the behaviour of the CPN model because diagrams similar to MSCs are already in use in the design process at Nokia. Therefore, MSCs allow the behaviour of the CPN model to be visualised in a way that is familiar to the UI designers and software developers. The use of domain-specific graphics and MSCs allow the state of the model to be visualised. However, the users of the CPN model are also interested in controlling the simulations without interacting directly with the underlying CP-nets, i.e., without explicitly selecting the transitions and bindings to occur. The CPN model has therefore also been extended with facilities for controlling the simulations without interacting with the underlying CP-nets.

In the following we will present techniques for visualising information about the simulation and for controlling the simulation using domain-specific graphics. Figure 8 shows how the domain-specific graphics appears as a single page in the CPN model of the mobile phone UI software system. In the following we will present the elements of the page shown in Fig. 8 in detail.

4.1   Animation of the Display

The CPN model is extended with a picture of a mobile phone. The picture is used both to present information about the simulation to the user and to get information from the user to control and guide simulations of the CPN model. The state of the phone as the user observes it via the handset is visualised via an animation of the display. The left-hand side of Fig. 8 shows a snapshot of the
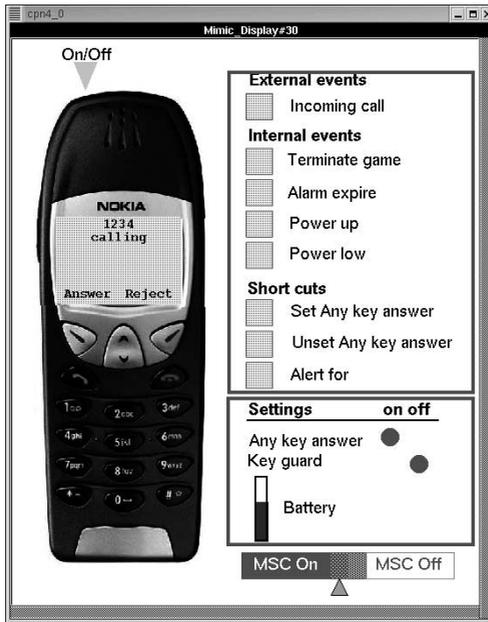
Fig. 8. The domain-specific graphics for the mobile phone UI software system

animation taken during a simulation of the CPN model. The snapshot shown corresponds to a state of the CPN model where the Call feature is Indicating. The user now has the possibility to either answer (transition Answer in Fig. 5) or reject (transition Reject in Fig. 5) the incoming call. The picture of the mobile phone is also used to control the simulation. By means of key presses (using mouse clicks) on the picture of the mobile phone, the user can control the simulation, i.e., start a game or answering an incoming call. In the following we will sketch how the use of domain-specific graphics has been integrated in the CPN model of the mobile phone UI software system.

The use of domain-specific graphics is implemented using the Mimic library [21] of Design/CPN, which allows graphical objects to be displayed, updated and selected during simulations. The animation in the left-hand side of Fig. 8 is constructed as number of layered objects:

- A background picture of a Nokia mobile phone is used to make the use of graphics look like a real Nokia mobile phone as it is known to the users of the CPN model.
- On top of the display of the background picture there are a number of regions (one for each area of the mobile phone display). Updating the contents of those regions allow us to animate the display of the mobile phone during simulation. We have implemented a function update_display() that scans through the display structure and finds the top element (display request) in the first non-empty stack, i.e., the one to be put on the display of the

mobile phone. The regions on top of the display of the background picture are updated with the contents of this "top most" display.
– On top of each of the keys of the background picture there is a region. We have implemented a function `get_input()` which enables the user to select one of the regions corresponding to the keys of the mobile phone and return information about which region, i.e., which key, is pressed.

The two functions `update_display()` and `get_input()` are called from code segments in the CPN model. Hence, the graphics is updated during simulations; user key presses are also read into the CPN model and can be used to control the simulations.

The animation of the display shows the state of the CPN model as the user of the handset observes it. Often the user is also interested in getting information about the internal state of the CPN model, e.g., the current personal settings of the mobile phone. Hence, the CPN model is also extended with graphics visualising the internal state of the phone. The lower part in the right-hand side of Fig. 8 shows how the internal state of the CPN model of the mobile phone UI software system is visualised using the Mimic library. The graphics visualise the state of Any key answer and Keyguard applications (whether they are *on* or *off*) and the amount of power in the battery. The graphics is updated dynamically during simulations to reflect the current state of the CPN model.

4.2   Controlling the Simulations

We have already described how the user can control the simulations by means of mouse clicks on the picture of the mobile phone. However, this often turns out to be insufficient. There are a number of events which are not generated as a result of user key presses, e.g., an incoming call to the mobile phone. The top-most part in the right-hand side of Fig. 8 shows how the CPN model has been extended with the possibility of generating such events by pressing buttons during simulations. The events are divided into three categories:

External Events i.e., events that are caused by the environment of the mobile phone (here incoming calls from the network).
Internal Events i.e., events that are generated by the mobile phone software without the influence of the user (here termination of a game, expiration of the alarm clock, battery level low warning, and battery charging).
Short Cuts allow the user of the CPN model to perform an operation without pressing a series of keys (here set or unset the Any key answer with a single key press instead of through the menu and change which phone numbers to alert for).

4.3   Message Sequence Charts

The animation of the display and the visualisation of the internal state of the mobile phone will provide information about the state of the CPN model as the

user of the mobile phone will observe it. However, the software developers are also often interested in gaining information about the UI software system at a more detailed level. This is obtained through the use of MSCs which capture the communication between applications, servers and the UI controller in the mobile phone UI software architecture. Figure 2 shows an example of a MSC automatically generated from a simulation of the CPN model. The MSC is described in detail in Sect. 3.

The MSC shown in Fig. 2 has a vertical line for both the user, the UI controller and each of the applications and servers involved in the scenario. During meetings with both UI designers and software developers we identified that the desired level of detail in the MSCs depends highly on the background of the user. Hence, we found it necessary to be able to customise the MSCs to the individual users of the CPN model. Hence, we now provide four possible levels of detail (see the lowest part in the right-hand side of Fig. 8). It should be noted that the MSCs can dynamically be turned on and off during simulations. Also the level of detail in the MSC can be changed dynamically. A small triangle below the buttons indicate which level is currently in use.

1. MSC On will generate detailed MSCs with vertical lines corresponding to the user, the UI controller as well as for all applications and servers in the system and arcs for all messages sent in the system.
2. The two buttons in the middle (between MSC On and MSC Off) generate MSCs in two intermediate levels of detail
   – The leftmost button generates MSCs like 1. but omits arcs corresponding to acknowledgement messages. The MSC in Fig. 2 is generated using this level of detail.
   – The rightmost button generates MSCs where only the communication between the user and the user interface is captured. This kind of MSC is mostly aimed at the UI designers who design the layout and the use of the display.
3. MSC Off turns off the use of MSCs.

In this section we have presented separate techniques for visualising information about the simulation and for controlling the simulation without interacting directly with the underlying CP-nets. It should be noted that during simulation of the CPN model, the page in Fig. 8 is the only page of the CPN model that needs to be visible to the user. Hence, the CPN model can be demonstrated and used without showing the underlying CPN model.

5   Related and Future Work

A number of published papers, e.g., [7,18,23], report on projects where CP-nets successfully have been used in industrial settings. Feature interactions have been studied in several application areas, e.g., telecommunications systems (see [15] for a survey), process planning [12], and computer-aided design [20]. To our knowledge there are no applications of CP-nets to feature interactions.

Especially in the area of telecommunication services much research work have been done on feature interactions and there is a biannual workshop on the topic [16,3]. Our work in the MAFIA project does not relate to what is traditionally known as *telecom feature interactions*; our problem is more general: how to coordinate concurrent interrelated services. However, we can draw some parallels. Much of the work in the area of feature interactions in telecommunications, e.g., [4,19,1], concentrate on the use of formal methods to automatically detect feature interactions. This is, however, not the aim of work the MAFIA project. In our work we focus on the development process of features, i.e, how the use of CP-nets to model the basic call model and the additional features can increase the level of understanding of the feature interactions and aid the future development of new features. Thus, we will categorise our approach to belong to the software engineering trend identified in [2] as one of the three major trends in the field of telecommunications services. However, the use of an underlying formal method, i.e., CP-nets, enables us to both obtain a formal description of the features and feature interactions and an environment for interactive exploration and simulation of the feature interactions. Furthermore, the construction of the CPN model as well as simulations were used to resolve inconsistencies in the specifications and to gain knowledge about features and feature interactions.

Based on the above, an interesting question is therefore whether the modelling approach of features developed in the MAFIA project can be used to automatically analyse and detect feature interactions. Using the Design/CPN Occurrence Graph Tool (OG Tool) [6] initial attempts have been done to evaluate the applicability of the CPN model for analysis purposes. State spaces have been generated for different combinations of features (including the features in isolation, i.e., the basic CPN model with only the analysed feature included). Not all combinations of features have been analysed and no attempts have been done to generate the state space for the full CPN model of the mobile phone UI software system presented in Sect. 3; only state spaces for selected combinations were generated with the goal of evaluating if (and how) state space analysis can be used to detect feature interactions in the CPN model developed in the MAFIA project.

In the following $\mathcal{S}_{f_1,f_2}$ denotes the full state space of the basic CPN model including the features f_1 and f_2. $\mathcal{S}_f \models P$ means that the property P can be verified from $\mathcal{S}_f$. One possible way of formally expressing that two features f_1 and f_2 interact is that for a property P we have that $\mathcal{S}_{f_1} \models P$ but $\mathcal{S}_{f_1,f_2} \not\models P$. We will use two properties P_1 and P_2 as examples:

$P_1=$ There are no *deadlocks* and the initial state is a *home state*.

$P_2=$ If a Call comes in and the Any Key Answer is set, then pressing any key on the mobile phone will answer the Call.

P_1 and P_2 can be formulated as *queries* in the OG tool. The answers to the queries can then be automatically determined by the OG tool when a state space has been generated. P_1 is general property of the CPN model and can be expressed as a standard query of a CPN model. P_2 relates to specific features

of the CPN model and can be formulated using temporal logic [9]. Properties
expressed in temporal logic can be verified using the OG tool library ASK-
CTL [8] which makes it possible to make queries formulated in a state and action
oriented variant of CTL [5]. We will not go into detail with how the properties
are expressed as queries. Instead we will show how P_1 and P_2 can be used to
detect feature interactions in the CPN model of the mobile phone UI software
system. Analysing different combinations of features we find that

1. $\mathcal{S}_{\mathsf{Call}} \models \neg P_1$ but $\mathcal{S}_{\mathsf{Call,AnyKeyAnswer,PhoneBook,KeyGuard}} \not\models \neg P_1$
2. $\mathcal{S}_{\mathsf{AnyKeyAnswer}} \models P_2$ but $\mathcal{S}_{\mathsf{AnyKeyAnswer,KeyGuard}} \not\models P_2$

The first feature interaction found is an interaction between the Call, Any key
answer, Phonebook and Keyguard features. The interaction is a result of the
Call application's use of other features in the mobile phone software system.
Figure 6 shows how the Call applications requests the Phonebook application,
the Any key answer application and the Keyguard application when an incoming
call arrives at the mobile phone. Hence, a deadlock appears when the modelled
Call application exists in isolation, i.e., without the Phonebook application, the
Any key answer application and the Keyguard application.

The second feature interaction found is an interaction between the Any key
answer and Keyguard features. The interaction appears as a result of the fea-
tures' capability of modifying the behaviour of other features. Here the Keyguard
application disables the Any key answer application to prevent the user from an-
swering an incoming call by accident.

After having performed the basic analysis presented in this section we con-
clude that the CPN model developed in the MAFIA project seems applicable for
analysis and detection of feature interactions; no major changes or adjustments
are needed. However, we expect the state space of the full CPN model to be
very large. Since the features of the mobile phone are asynchronous a possible
direction for future analysis would be to use the stubborn set method [22] to
reduce the size of the state space.

6   Conclusions

This paper describes the MAFIA project and one of its main activities: the
construction of a CPN model of the mobile phone UI software system. Previous
sections have reported on the CPN model and its construction. In this section
we will conclude the paper by discussing how the CPN model was used within
the MAFIA project and how the construction of the CPN model has influenced
the development process of features in Nokia mobile phones.

The CPN model has been constructed in a period of six months and has been
constructed in several iterative steps with more and more elaborated models
each presented to the intended users of the CPN model, i.e., the UI designers
and software developers. The fact that the CPN model was presented to users
not familiar with Petri Nets meant that the CPN model was extended with

domain-specific graphics at a very early stage. The graphics was developed and extended in parallel with the underlying CP-net.

An important aspect of the CPN model developed in the MAFIA project is that the model is intended to be used in very different settings:

High Level (UI Oriented). UI designers design the features and their inter-actions at the level of the user interface of the mobile phone. The CPN model provides possibilities for simulations with feedback by means of the anima-tion of the display and MSCs with detailed information about the flow of user interaction and snapshots of the contents of the mobile phone display. The UI designers can simulate the CPN model using only the page with Mimic graphics.

Conceptual Level. The architecture and protocol for the mobile phone UI software system is designed particularly to be well suited for development of features. The CPN model provides possibilities for simulations with more de-tailed feedback about the individual features and their interactions (messages sent between the individual applications, servers and the UI controller).

Low Level. Software developers specify and implement the new features of mobile phones. The CPN model is developed to make it possible to easily add new features in the model without changing the existing model. Hence, new features can be included in the CPN model and simulated with the features already included in the CPN model.

The construction of the CPN model of the mobile phone UI software system is only one of the activities in the MAFIA project. Other activities include development of a categorisation of feature interactions. During development of such a categorisation, the CPN model was used to experiment with the categories found. The CPN model have also been used to produce static figures (like the MSC in Fig. 2) to a document explaining feature interaction to UI designers and UI testers in Nokia. We have already described how the CPN model was constructed in close cooperation with UI designers and software developers of Nokia mobile phones. The UI designers and software developers also suggested features to be included in the CPN model to ensure that the categorisation of feature interactions developed in the MAFIA project is covered by features included in the CPN model. Simulations were used to understand the feature interactions and as a starting point for discussion.

The CPN model is intended to be used in future feature development in Nokia; both as a means for presenting and experimenting with the different features and feature interactions captured in the CPN model, but also as a framework for development of new features. New features can be designed and included in the CPN model and the behaviour and interactions with other fea-tures can be observed using simulations of the CPN model. However, even if the CPN model was not to be used in feature development, we find that the project group and the users involved in the development process have benefitted from the construction of the CPN model. During construction of the CPN model, a number of inconsistencies in the descriptions of the features have been identified. The modelling process has identified what kind of information about a feature's

interaction with other features need to be provided by the designer of the feature and what kind of problems need to be resolved when a feature is designed. In parallel with the MAFIA project the UI designers involved in the construction of the CPN model were designing a new feature for a Nokia product. During development of the feature, focus was pointed at interactions with other features based on experiences gained from the MAFIA project, and this resulted in a suggestion for new design templates and development practises.

The construction of the CPN model has also generated new ideas for the mobile phone UI architecture. The first version of the CPN model was constructed by the project group based on written documentation. A number of inconsistencies were found, and some of these could not be resolved before the first meeting with the users. Hence, a solution was chosen and presented to the users as a starting point for discussion. In fact, the chosen solution was not the correct solution with respect to the current implementation of the mobile phone UI software system. However, the solution modelled was found to be interesting and in fact a possible suggestion for a change in the mobile phone UI architecture to make it more oriented towards feature interactions.

In conclusion we can say that the MAFIA project has improved knowledge about features and feature interactions and has influenced an ongoing change of design practises of features in new products of Nokia mobile phones. The CPN model developed in the MAFIA project is intended to be used in future design of features of Nokia mobile phones. Furthermore, the modelling activities have raised interesting questions and ideas that can lead to design changes of the mobile phone UI architecture.

References

1. D. Amyot, L. Charfi, N. Gorse, T. Gray, L. Logrippo, J. Sincennes, B. Stepien, and T. Ware. Feature Description and Feature Interaction Analysis with Use Case Maps and LOTOS. In M. Calder and E. Magill, editors, *Feature Interactions in Telecommunications and Software Systems*, volume VI, Amsterdam, May 2000. IOS Press. 309
2. M. Calder, M. Kolberg, E. H. Magill, and S. Reiff-Marganiec. Feature Interaction: A Critical Review and Considered Forecast. Submitted for publication. On-line version: `http://www.dcs.gla.ac.uk/~muffy/papers/calder-kolberg-magill-reiff.pdf`. 309
3. M. Calder and E. Magill. Feature Interactions in Telecommunications and Software Systems VI. IOS Press, 2000. 309
4. M. Calder and A. Miller. Using SPIN for Feature Interaction Analysis - a Case Study. In *Proceedings of SPIN 2001*, volume 2057 of *Lecture Notes in Computer Science*, pages 143–162. Springer-Verlag, 2001. 309
5. A. Cheng, S. Christensen, and K. Mortensen. Model Checking Coloured Petri Nets Exploiting Strongly Connected Components. In M. Spathopoulos, R. Smedinga, and P. Kozák, editors, *Proceedings of WODES'96*. Institution of Electrical Engineers, Computing and Control Division, Edinburgh, UK, 1996. 310
6. S. Christensen, K. Jensen, and L. Kristensen. *Design/CPN Occurrence Graph Manual*. Department of Computer Science, University of Aarhus, Denmark. On-line version: `http://www.daimi.au.dk/designCPN/`. 309

7. S. Christensen and J. 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. 308
8. S. Christensen and K. H.Mortensen. *Design/CPN ASK-CTL Manual*. Department of Computer Science, University of Aarhus, Denmark, 1996. Online: `http://www.daimi.au.dk/designCPN/`. 310
9. E. Clarke, E. Emerson, and A. Sistla. Automatic Verification of Finite State Concurrent Systems using Temporal Logic. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986. 310
10. Design/CPN Online. `http://www.daimi.au.dk/designCPN/`. 295
11. H. Genrich. Predicate/Transition Nets. In K. Jensen and G. Rozenberg, editors, *High-level Petri Nets*, pages 3–43. Springer-Verlag, 1991. 295
12. J.-S. Hwang and W. A. Miller. Hybrid Blackboard Model for Feature Interactions in Process Planning. *Computers and Industrial Engineering*, 29(1–4):613–617, 1995. 308
13. ITU (CCITT). Recommendation Z.120: MSC. Technical report, International Telecommunication Union, 1992. 296
14. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts*. Monographs in Theoretical Computer Science. Springer-Verlag, 1992. 295
15. D. O. Keck and P. J. Kuehn. The Feature and Service Interaction Problem in Telecommunication Systems: A survey. *IEEE Transactions on Software Engineering*, 24(10):779–796, October 1998. 308
16. K. Kimbler and L. G. Bouma. Feature Interactions in Telecommunications and Software Systems V. IOS Press, 1998. 309
17. L. Kristensen, S. Christensen, and K. Jensen. The Practitioner's Guide to Coloured Petri Nets. *International Journal on Software Tools for Technology Transfer*, 2(2):98–132, December 1998. 295
18. L. Lorentsen and L. 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. 308
19. M. Nakamura, Y. Kakuda, and T. Kikuno. Feature Interaction Detection using Permutation Symmety. In K. Kimbler and L. G. Bouma, editors, *Feature Interactions in Telecommunications and Software Systems*, volume V, pages 187–201, Amsterdam, September 1998. IOS Press. 309
20. D.-B. Perng and C.-F. Chang. Resolving Feature Interactions in 3rd Part Editing. *Computer-Aided Design*, 29(10):687–699, 1997. 308
21. J. L. Rasmussen and M. Singh. *Mimic/CPN. A Graphical Simulation Utility for Design/CPN. User's Manual*. On-line version: `http://www.daimi.au.dk/designCPN/`. 306
22. A. Valmari. Error Detection by Reduced Reachability Graph Generation. In *Proceedings of the 9th European Workshop on Application and Theory of Petri Nets*, pages 95–112, 1988. 310
23. 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. 295, 296, 308