Appendix G: Minutes from group meetings

Minutes of meetings 11 Dec 2001

Meeting with wjk at 1pm

Present: William Knottenbelt, Raphael Goldberg, Peter Howarth, Daniel Justice, Michelle Osmond, Zhu Tan, Aleksandar Trifunovic

- Everyone given a copy of "Stochastic Petri Nets" (Bause & Kritzinger) to read over the Christmas break. Particularly:
  - Chapters 4, 7, 8.
  - Chapters 5 (Coloured Petri Nets) and 9 (Queueing Petri Nets) for further extensions to project
- Look up Petri net programs, try them out and see what are good/bad features, eg Renew (good interface).
- njd200 did a Petri net individual project last year.
- Look into PNML, Petri Net Markup Language.
- General points on the project:
  - petri nets: basic types and time dependent types
  - analysis:
    - correctness: eg invariant analysis (structural analysis), state graph (deadlock, livelock, boundedness, home states)
    - performance: eg simulator (collects statistics), Markov chain analyser (use existing or write our own), response time analysis (advanced), queueing network analyser (ambitious)
  - animation to show how petri net works.
  - linking analysis back, eg showing how deadlock happens
  - modularity, pluggable - ability to add modules while the program is running, so they appear as new menu options - need Java reflection.
  - hierarchical subnets - including a subnet interface
- Next term: decide on OMT diagram and what functionality to include.

Meeting in SCR at 1:30pm

Present: Raphael Goldberg, Peter Howarth, Daniel Justice, Michelle Osmond, Zhu Tan, Aleksandar Trifunovic

- Aleksandar Trifunovic chosen as group leader.
- Michelle Osmond chosen as group secretary.
- Next meeting: provisionally first Monday of next term.

Minutes of meeting 14 Jan 2002

Meeting opened 1.10pm

Present: Raphael Goldberg, Peter Howarth, Daniel Justice, Michelle Osmond, Zhu Tan, Aleksandar Trifunovic

Work done over holiday:

- Everyone has read through the recommended chapters of the Petri Nets book.
- Most have seen Renew; Daniel has looked at it in most detail.
- Alexander has read mw400's report (subnets), Michelle has read njd200's (PNML/SAX, reflection).
- Michelle has looked at PNML, and how to use XML with JAVA - JDOM looks promising.

Elements which we want to include or investigate further:

- An **invariant analysis module** (write this) - look up references, algorithms.
- **Markov chain analyser** - either write, or use existing one by wjk (see DaNAMiCS and Medusa (njd200), which both use **DNAmaca**).
- **Coverability trees** - see how Renew does it, look at maths/algorithms further.
- **Subnets** - see how they were implemented in mw400's project, & other ways to do it?
- **PNML, JDOM** - a candidate for the main data structure in our program - ie, not necessarily need to build up our own petri net object model.
- **Reflection** - see njd200 and mw400
- **Feedback from analysis to display**, eg deadlock tracking. Perhaps expand the module interface from that in previous projects (and write our own analysis components as pluggable modules).

To investigate by next meeting:

- Raphael: Markov chain analyser
- Aleksandar: Invariance analysis
- Michelle: Subnets, PNML (+graphics)
- Peter: Reflection (+XML)
- Daniel: JDOM, XML, PNML
- Zhu: Subnets

Next meeting: This Friday (18th), 1pm in the common room. Aleksandar will contact wjk about a meeting.

Useful links

Meeting closed 1.55pm

Minutes of meeting 18 Jan 2002

Meeting opened 1.05pm

Present: Raphael Goldberg, Peter Howarth, Daniel Justice, Michelle Osmond, Zhu Tan, Aleksandar Trifunovic, Nicholas Dingle

Discussion with Nick Dingle about Medusa

- Room 332 opposite 308 & 311 ("Antonio Garcez" on the door)
- **Runtime.exec()** to run external programs (like Dnamaca) from the command line: worked without problems.

- **Borland JBuilder** is good (can get on magazines)
- **Simulation module** would be useful for comparison with analytical results from Dnamaca
- **Graph theory module**: was difficult to construct the coverability graph
- **PNML**: an existing tool that uses it is the PNK (Petri Net Kernel), which can deal with different types of nets, and has modules that communicate back to the main program.
- **SAX** was easy to use, but other ways may be better (didn't investigate early on). Similarly with the method he used for transferring data to modules.
- **Swing** is thread-safe - ie don't want multiple threads accessing Swing components. Had problems with it's circles!

Further discussion - what we've done since the last meeting

- **Daniel**: XML: what it is. DTD's, SAX, DOM, JDOM. (Word document)
- **Peter**: Reflection, GUI - probably not too hard, but it's worth putting effort into making it intuitive and elegant.
- **Alex**: Dnamaca, Runtime.exe. Threads - seem to be relevant for animation. Invariant analysis, looked at the matrices needed and algorithms.
- **Raphael**: read the two MSc reports from last year, looked at invariance analysis, Markov and graph theory.
- **Tan**: looked at subnets in Renew (graphical handling), and Java.
- **Michelle**: the new module (subnet) implementation in PNML and method used in last year's project. Summary of XML and PNML.

Discussion with William Knottenbelt

Key functionality and *possible extensions* for the projects are described below.

- Display and graphical editor for petri nets. *Could write a class for one 'pane', then could open multiple instances (eg edit multiple nets, or subnets).*
- Animator: click on a transition to see it fire. *Automatic animated simulator*
- Subnets
- Modules:
    - Invariant Analysis
    - Graph Theory. *Also for nets with immediate transitions*

    - Simulator. *Also with inhibitor arcs, which enable a transition when the input place is empty (not expect to support them for the other modules, it's hard!)*
    - *Interface to Markov chain analyser (Dnamaca) - takes text files as inputs and outputs*
    - *General interface - communication between modules and main program, eg for animation. Maybe return special object, or call functions?*
- *Coloured petri nets*
- *Threads - for editing different nets?*

To decide in the next week or 2:

- write requirements
- high level OMT diagram
- GUI: what should it look like
- module interface: parameters and return types

Weekly meetings with wjk: 12:30 on Fridays

Further discussion

Very likely to use JDOM for main internal data representation. Could be wrapped in a custom class with useful access functions that we determine could be useful to modules (eg a function to return a particular matrix of the net). If we pass this class to modules, they will have access to both the 'convenience' functions and the entire JDOM, allowing them to extract anything they like, including future elements that may be defined in PNML later. They can also modify the net if necessary (eg putting it into a particular marking) - may or may not be a good thing.

Need to determine interface for modules. People will look at modules and see what they require, how to do them (broadly) and what they'll return (what sort of results, and how they could be presented to the user).

We'd like a clearer view of how JDOM actually works. Also if it will work on the college machines ok.

We haven't investigated the simulation aspect properly yet.

Need to think about possible features/layouts of the GUI. Inspiration from the concurrency program? (tabs, one pane has the code and another has the graph) Would be nice to be able to view the PNML from within the program - most users will not want to deal with it (the editor is meant to make it unnecessary), but it would be nice to provide the option. Subnets: opening a new window to edit a subnet seems necessary. The subnet may be within the same pnml file as the main net, or it may be in an external file or even on the internet (the pnml allows for this: theoretically, this means there could be a library of publically available subnets on websites which could be accessed and used).

To investigate:

- Aleksandar: Invariant analysis module
- Raphael: Graph Theory module
- Peter: Simulation module
- Daniel: Details of the JDOM
- Michelle: Swing, JDOM in practice
- Tan: GUI (also other tools' GUIs) and overall design

Next meeting: Monday (21st), 1pm in the common room.

Meeting closed 3.30pm

Minutes of meeting 21 Jan 2002

Meeting opened 1.10pm

Present: Raphael Goldberg, Peter Howarth, Daniel Justice, Michelle Osmond, Zhu Tan, Aleksandar Trifunovic

What we've done since the last meeting

Only had the weekend since the last meeting, so:

- **Alex**: Looked further at invariant analysis stuff. Got algorithms, and references (see Links section).
- **Raphael**: Looked further into graph theory algorithms.
- **Michelle**: Implemented a simple Aquarium program to test JDOM in actual use, and Java2D as a possible way of drawing the graphics. Program reads in an XML file containing information on the aquarium, outputs to the console and draws a representation to a Swing window using Java2D drawing. (See Links section to try out the code/program.) It works fine in both Windows and Linux on the college machines (you need to download JDOM first).

Further discussion

Module Interface

Previous projects passed the information to modules as an XML file. This is simple, but has the disadvantage that there is no active link from the module to the main program.

The JDOM Document object contains all the information from the XML file in an easily navigable structure, as well as being the main data centre of the program, so it makes sense to pass the module a reference to this Document.

The module can then parse the information into another form if convenient. It therefore provides the equivalent functionality to previous projects' methods.
In addition, using the JDOM Document means the module can also update the main Document if it needs to modify the representation, with any changes being reflected in the program.
It can add any elements to the Document that it likes, for example it could store a representation of its results, so that the PNML file would accumulate data about itself (not necessarily in the PNML language).
It can also access any elements, regardless of whether we, as the programmers of the main program, are expecting them to exist. Eg a module which can analyse nets with Inhibitor arcs, even though the main program may not recognise them.

GUI design

A prototype design was drawn up in the meeting (Daniel).
A multiple document interface (MDI) would be ideal - that is, one main program window, and internal windows within that (like Office 97 implementations of Word, Excel). Then we can open up subnets in separate windows while still being able to get at the main net, or even have different nets open. These nets/subnets may be sourced from the same or different .xml files.
Editor - keep to the standard Toolbar for each item, with extra buttons like Rotate, Subnet. Also a standard File, Open, Print etc Toolbar. Menus will provide the same functionality as the toolbars and more. Have a Modules menu. Maybe a Subnets menu, or maybe just a Subnet item(s) under an Insert menu.
View Source - it would be nice to allow access/viewing/editing of the PNML from the editor. Not prominently, as the idea is (usually) to hide the source, but nice to have the option.

Program design

Some confusion arose when considering the program structure with the data being held in a JDOM Document object, rather than our own custom class hierarchy. Where do we put the functions, eg for displaying graphics and performing simulations/animation? Two main possibilities:

- **Data in JDOM Document, Object hierarchy with functionality in those parts of the program which need it.** The Document would be the main data structure which all other parts of the program (modules, graphics, simulation) may access/modify. One centre of information - consistency. Also modularity and flexibility because components may be easily updated without affecting other areas. Takes advantage of existing official code for dealing with XML structures. Been considering this model so far. Possible ways of implementing graphics section include:
    - Brute force and ignorance - Draw net with Java2D, detect (x,y) of mouse click, search through all the objects in the net to find any which contain (x,y). Not a good approach.
    - Use an object hierarchy (with objects like Place, Transition etc) which is intended to represent the display, but gets its information from the Document object. So the data isn't stored in these Place, Transition etc objects, rather they contain a reference to the Element in the Document

which they represent (and using that reference can get the information such as (x,y) or number of tokens). These objects would be subclasses of some useful existing Java class such as JLabel which allows MouseListeners to be assigned, and allows absolute positioning. Hence can write Listeners to deal with interactivity (eg clicking on a Place). How to deal with Simulation?

- **Use JDOM or SAX to read in data to our own object hierarchy, which contains both data and functionality. Pass Document objects or the XML file to the modules.** Follows the standard design we've been taught for designing a program from scratch - more control, can keep data and functions together. But modules don't need/want to access this hierarchy. Updating data between modules/main program, and extending capabilities of the program may be harder.

To investigate:

- Daniel: Finish off the JDOM summary
- Michelle: Try implementing the start of a Petri net editor using the movable-label idea.
- Tan: look at the other Object based idea.

Next meeting: Tuesday (22nd), 1pm in the common room.
Meeting closed 2:55pm

Minutes of meeting 22 Jan 2002

Meeting opened 1.00pm
Present: Raphael Goldberg, Peter Howarth, Daniel Justice, Michelle Osmond, Zhu Tan, Aleksandar Trifunovic

What we've done since the last meeting

- **Michelle**: Implemented a Petri nets prototype editor using Swing and JDOM, with a MDI. It reads in a pnml file and can, so far, draw and move places, updating the DOM in memory. Implements places on screen as a subclass of JLabel, with the real data stored in a JDOM Document object. (See Links for download).
- **Daniel**: Done the DOM and JDOM summary.
- **Tan**: Worked out OMT/UML diagrams for the conventional object approach.

Discussion

Continued discussing reasons for and against using the JDOM-based model over the usual object model. No unanimous conclusion, though most in favour of JDOM due to the potential flexibility (a new feature compared to most Petri Net projects). Decide on Friday (discussion with Will). Points raised include:

- **JDOM-based:**
  - extensibility. all data automatically extracted from the input xml file, regardless of whether it's part of the pnml definition we're working with *now* (or even might not be PNML).

    - editor could be built to be dynamic to an extent, so it could extend its own capabilities without any extra coding/compiling. (pnml may/will be extended to support different types of nets.)
    - JDOM well-supported and respected; robust code written already to deal with parsing - saves us effort, and we can trust it.
    - the more in-depth details of XML could be taken advantage of (namespaces, validation).
    - modules could also modify the JDOM if they're passed a reference to the Document.
    - BUT: need to have a separate holder (class hierarchy) to store the functionality (like drawing objects), the data is not protected, and it doesn't follow the 'conventional' methods/models we're familiar with already. How to do animation?
- **Object model:**
    - tried and tested (as in previous projects), the way of doing things that we've been taught - can store data and functionality together.
    - full control over the data access.
    - easier to see how to program things like the animation.
    - BUT: extending the program to support new types of net, and elements of XML, is much harder (needs coding/compiling, probably modification of large parts of the program). also would probably be passing the JDOM Document to modules anyway (no need for data protection, but modification of the active data by the modules wouldn't be possible).

To investigate:

- Daniel: Code up a (non-active) GUI to show what we discussed in the previous meeting, and use as a basis later.
- Michelle: Continue with the Petri net editor prototype.
- Tan: Try implementing the Object based idea.

Next meeting: Friday (25th), 12:30pm with wjk and then continue meeting afterwards.
Meeting closed 1:55pm

Minutes of meeting 25 Jan 2002

Meeting with wjk at 12:30pm

Meeting opened 12.30pm
Present: William Knottenbelt, Raphael Goldberg, Peter Howarth, Daniel Justice, Michelle Osmond, Zhu Tan, Aleksandar Trifunovic

What we've done since the last meeting

- **Michelle**: Extended Petri nets prototype editor to draw and move places, transitions and arcs, updating the DOM in memory. Dynamically-generated, editable properties box to show possibilities of the JDOM system. (See Links for download).
- **Tan**: Finished OMT/UML diagrams for the conventional object approach.
- **Daniel**: Done GUI framework (menus and buttons).
- **Raphael**: First draft of Report I

Discussion

- For reasons discussed before, will use the JDOM-based system. The big reason is that it allows us to build a program which is designed to be extended to support new types of net, relatively easily compared to the usual fixed-object approach. This is a feature which most other petri net editors do not support.
- We will however maintain an object hierarchy for the graphics/animation/simulation part of the system, very similar to the conventional approach, but with pointers to the elements within the JDOM (rather than storing data in the objects).
- In the graphics objects, include access (set,get) functions to provide data protection, for the animation/simulation part to use, but the active parts of the editor (eg properties box) will need unrestricted access to the DOM. This shouldn't be a problem as that part of the system is unlikely to be extended by module authors, and may not even need to be changed when adapting the program to support new net types.
- Worth getting the editor looking good (circles), as that's what gives the first impression.

Report

- Initial draft looks good.
- Add description of what JDOM is and why we're using it (extensibility).
- OMT diagram of the object (label) hierarchy, and high-level description of how the rest of the program should work.

Meeting closed 1:30pm

Meeting in 343

Present: Raphael Goldberg, Peter Howarth, Daniel Justice, Michelle Osmond, Zhu Tan, Aleksandar Trifunovic

Meeting opened 1:30pm

- Looking at Renew: nice interface. Automatic creation/deletion of transitions, connecting arcs. Simulator - nice, but too fast - would be nice to include a changeable timestep. Simulator runs and then returns to original marking when it finishes.

- Editor/display are closely linked (~same thing). Get the basics up and running fast, then do the clever stuff (like Renew's features) later
- Animator (click on transition to fire) also needs good knowledge of the editor/display part, so better to not make it a module.
- Modules: Invariance Analysis, State Space, Simulator (later), Markov Chain (later)

To do:

- Aleksander: Invariant Analysis module
- Tan: update the OMT diagram and send to Peter
- Michelle: write JDOM summary, email PNML people, sort out GUI
- Daniel: sort out GUI
- Peter: continue writing the report, look at subnets
- Raphael: state space, graph theory module

Meeting closed 2:30pm

Next meeting: Monday (28th), 12:30pm in 343 (or common room)

Minutes of meeting 28 Jan 2002

Meeting opened 12.45pm
Present: *Peter Howarth, Daniel Justice, Michelle Osmond, Zhu Tan, Aleksandar Trifunovic*
Apologies: *Raphael Goldberg*
What we've done since the last meeting

- **Alex**: Almost finished invariant analysis algorithm.
- **Daniel, Michelle**: Put together the prototype editor and GUI. set out files in group directory, sorted out packages, classpaths, permissions.
- **Michelle**: Done most of PNML/JDOM part of report.
- **Tan**: Looked at how to use Swing, JDOM, patterns for writing programs in Java.
- **Peter**: Done report (except PNML/JDOM). Looked at subnets.

Discussion
Module interface: the few functions required by reflection, and an object which may contain various information like sequences of transitions, to allow the module to communicate back to the program (eg display path to deadlock). Module knows where object is, object knows where the appropriate Animator is. Useful for state space module, probably not the invariance analysis one.
Assignment of tasks: Alex and Raphael continue with their modules, the rest share the editor/animator/overall program/extensibility.
To do:

- Alex: Invariant Analysis module
- Daniel, Tan: start implementing menu functions (file->save, etc) and toolbar (add new place etc).

- Michelle: finish JDOM summary, sort out existing demo classes.
- Peter: try implementing a dummy module to sort out using reflection.
- Raphael: state space module

Meeting closed 1:20pm

Next meeting: Wednesday (30th), 1:15pm in 343 - sort out Report I (due Thursday)

Minutes of meeting 30 Jan 2002

Meeting opened 1pm

Present: *Peter Howarth, Daniel Justice, Michelle Osmond, Zhu Tan, Aleksandar Trifunovic, Raphael Goldberg*

Meeting was more of a group work session.

Work done during meeting

- Raphael: Collated information and produced Report 1
- Alex: Coded Invariance Analysis module
- Tan, Peter, Michelle: UML Diagram
- Daniel: designed/coded the menu and Listener system: File->Open, File->New (also explained to Tan)
- Michelle: reorganised/cleaned up the demo code to conform to our UML diagram (also explained to Peter)

Meeting closed 5pm-ish

Next meeting: Friday (1st Feb), 12:30pm with Will

Minutes of meeting 1 Feb 2002

Meeting opened 12:45 pm

Present: *Peter Howarth, Daniel Justice, Michelle Osmond, Aleksandar Trifunovic, Raphael Goldberg*

Apologies: *wjk*

Absent: *Zhu Tan*

To Do

- Daniel (+Tan?): implement the "Add Place", etc functionality in the Editor
- Peter: manage the subnets part. Think about design, how it should be managed by the user.
- Raphael: continue with module
- Alex: deal with the module interface. continue with module
- Michelle: provide JDOM code examples (particularly for module people)

Meeting closed 1pm

Next meeting: Tuesday (5th Feb), 1:00pm with Will

Minutes of meeting 5 Feb 2002

Meeting opened 1:00 pm

Present: *Peter Howarth, Daniel Justice, Michelle Osmond, Aleksandar Trifunovic, Raphael Goldberg, Zhu Tan, William Knottenbelt*

Discussion

- Invariant analysis - going ok apart from a small bug - sort out after meeting
- Have put together the demo editor and GUI. Open and Save XML are working.
- State space - half way through coding. Points to consider:
    - Hash table for storing/searching different markings
    - Java memory problems - can't rely on garbage collection happening fast enough - better to reuse state 'shells' rather than discard them and create new ones. Also speeds up by x2. (look into an equivalent of 'delete' in Java?)
    - Compaction of markings (rather than storing a vector of integers) - eg 1 byte per place?
    - Paper on a probabalistic state space exploration method, deals with the state space explosion problem - Stern & Dill 1995
- Petrinator?

To Do:

- Daniel: implement the "Add Place", etc functionality in the Editor
- Peter: subnets
- Raphael: continue with state space module
- Alex: continue with invariant analysis module
- Michelle: animator (highlight active transitions, click on transitions->fire)
- Tan: module interface

Meeting closed 1:25pm

Next meeting: Friday (8th Feb), 12:30pm in 345

Next meeting with Will: Friday (15th Feb), 12:30pm

Minutes of meeting 8 Feb 2002

Meeting opened 12:50 pm

Present: *Daniel Justice, Michelle Osmond, Aleksandar Trifunovic, Raphael Goldberg, Zhu Tan*

Absent: *Peter Howarth*

Discussion

People are progressing with their coding.

Michelle done web-based timesheet system - update it yourselves :)

Discussed how to implement the module loader.

Everyone to report on what stage they're at, and explain what they've done, next meeting.

Meeting closed 1:00pm

Next meeting: Friday (15th Feb), 12:30pm

Minutes of meeting 15 Feb 2002

Meeting opened 12:30 pm

Present: *Daniel Justice, Michelle Osmond, Aleksandar Trifunovic, Raphael Goldberg, Zhu Tan, Peter Howarth, William Knottenbelt*

Discussion

Progress continues.
Alex - invariant analysis working & demonstrated.
Raphael - state space working - improve memory use?
Michelle - dynamic label creation, animation basis done.
Daniel - adding places etc working. Continued with GUI.
Tan - module interface working, but needs modifying/extending.
Meeting (with wjk) closed 1:10pm

Further Discussion

Alex, Raphael and Tan will work together on module interface, and connecting in the modules. (Alex & Raphael will also continue with their modules.)

- Want to be able to load in multiple modules at one time, and also have some loaded on startup (specified by an .ini file or similar)
- Pass reference to JDOM Document (rather than xml file) to module - eg could implement a subnet-flattening module.
- Could store module results in the Document, as custom XML data, and also store an md5 checksum of the current state of the file, so that if the module is run soon after, it can see if its old results are still valid (and avoid recalculating).
- Pass a reference to an 'interface' object of some sort which allows the module to communicate back to the main program. Eg: the module could store data in the object's fields (eg a list of transitions to fire, or a node to highlight) and then call a trigger function in the object which acts upon the contents of its fields (or maybe it could be given instructions as an argument to the trigger function).

Michelle will continue with the display (priority - arcs and animation), and extensibility.
Daniel will start on copy, paste, maybe undo functions.
Peter will continue with designing/implementing the subnet system.
Meeting closed 1.25pm

Next meeting: Friday (22nd Feb), 12pm then 12:30pm with Will

Minutes of meeting 22 Feb 2002

Meeting opened 12:30 pm

Present: *Daniel Justice, Michelle Osmond, Aleksandar Trifunovic, Raphael Goldberg, Zhu Tan, Peter Howarth, William Knottenbelt*

Discussion

Progress continues.
Module loading is ok for Rafi's but not Alex's (multiple classes something to do with it?), though they're not reading data from the JDOM Document.
Flattening module done.
Require a module bridge class which provides functions to the modules so they don't need to know about the internal program structure:

- Reload from DOM (makes new labels, repaints the DisplayPane

- Highlight(given id)
- Animate(given list of transition id's)

Flattening module should be integrated into the program. The program should automatically flatten the net for the module if necessary - write a function as part of the module interface which returns a boolean to say whether the module can deal with subnets or not.
Better arcs and basics of animation are working. Animation interface needs creating.
Meeting closed 1:30pm
Next meeting: Friday (1st Mar), 12pm then 12:30pm with Will

Minutes of meeting 1 Mar 2002

Meeting opened 12:00 pm
Present: *Daniel Justice, Michelle Osmond, Aleksandar Trifunovic, Raphael Goldberg, Zhu Tan, Peter Howarth, William Knottenbelt (12:30-1:00)*
Discussion
Progress continues.

- Module loading has null pointer problems with passing the JDOM Document (probably). Still needs the automatic detection of subnet-capability, and the bridge class is not done.
- Analysis modules have memory issues to sort out for large nets.
- Subnets are progressing: animation and linking to do.
- Cut and paste are working.
- Animation is better - slider to implement.
- Extensibility features to add.
- Undo functionality?
- Dragging labels - on-screen display needs sorting.
- **Have a working demo for next Friday.**

Meeting closed 1:30pm

Next meeting: Friday (8th Mar), 12pm then 12:30pm with Will

Minutes of meeting 8 Mar 2002

Meeting opened 12:30 pm
Present: *Daniel Justice, Michelle Osmond, Raphael Goldberg, Zhu Tan, Peter Howarth, William Knottenbelt*
Apologies: *Aleksandar Trifunovic*
Discussion
Progress continues.

- Module loading is sorted, only issue is passing back information from flattened nets.
- Subnets are progressing.
- Animation interface done - still have to disable other buttons.
- Extensibility core is working, need interface (choosing net type)
- Support for timed/immediate transitions to do.

- Dragging labels - sorted
- **Have a finished version for next Friday.**

Meeting closed 1:00pm
Meeting opened 1:00 pm
Present: *Daniel Justice, Michelle Osmond, Aleksandar Trifunovic, Raphael Goldberg, Zhu Tan*
Apologies: *Peter Howarth*
To Do

- Directory organisation (Tan, Alex)
    - Find out how to jar up the files.
    - Need a bat file and a shell script to run the program, that should set the classpath appropriately too.
    - Need subdirectories that are visible (after the program has been jar'd up) for modules, templates, and the xml hierarchy file (nettypes.xml)
- Modules (Alex, Raphael, Tan)
    - Put module classes in the right directory
    - Translation of flattened net id's back to real id's - do after subnets are sorted.
    - Alex's module - sort out the dialog box + stop program shutting down when close module.
    - Raphael's module - track down a bug, maybe add in finding path to unbounded.
    - Dnamaca module interface
- Extensibility (Michelle)
    - GSPN support - new Transition type and deal with insertion
    - Properties Box for Stochastic Transitions
    - Properties Box and menu/right-click item for whole net
    - Creating new net - provide type options, templates(?)
- Animator (Michelle)
    - Disable editing functions while animating
    - GSPN Animator code
- Subnets (Peter, Dan)
    - Get the components connected and working
    - Display and selection of the subnet instance
    - Interface issues - eg insert new subnet
- Editor (Dan)
    - Key binding/button for delete

People write up sections they've been working on - expand from Report 2. (not all bullet points!)
Reuse Petri net intro from Report 1, and summary of program (all together) from Report 2. Then go into details.
Need at least 3 specifically user-oriented sections:
* how to use the program
* how to write a module for the program
* how to extend the program

Meeting closed 1:30 pm

Next meeting: Friday (15th Mar), 12pm then 12:30pm with Will

Minutes of meeting 15 Mar 2002

Meeting opened 12:30 pm

Present: *Daniel Justice, Michelle Osmond, Raphael Goldberg, Zhu Tan, Peter Howarth, Aleksandar Trifunovic*

To do

- Prompt to save changes on closing windows - **(Dan)**
- Subnets:
  - Opening subnets is complicated - give user instructions **(Peter)**
  - Flattening : be able to deal with modules in external files **(Tan)**
  - Be able to open up nets which have subnets in external files **(Peter)**
  - Opening nets/subnets - make reference links appear **(Peter)**
  - Linking of subnet elements **(Dan)**
  - Animation of subnets **(Michelle)**
- Make normal transitions (ptNet) black **(Michelle)**
- Help in program - use html, from Dan's user guide **(Dan)**
- Name: decided on EXtensible Petri net Tool (EXPT) - fix current references **(Michelle)**
- Animation feedback from module - make it use the right path **(Michelle)**
- Modules: package up the helper classes in subdirectory, so the user knows which class to open. **(Alex)**
- Rename rafi.class **(Rafi)**
- Delete option - link the delete key to the ctrl-v cut function **(Dan)**
- Directory organisation, jar, bat file etc **(Tan)** so that others can make code work **(Dan, Michelle, Peter)**

Final Report

Alex will collate and format the contributions, with help from Rafi.

Main report:

- **Intro, Specifications** - done (Alex put together)
- **What we set out to achieve:**
  - **GUI functionality advantages/disadvantages** - Dan
  - **PNML, JDOM** - done (Michelle)
  - **Modular Extensibility (intro)** - Tan
  - **Extensions** - support for multiple net types intro - Michelle
    Dnamaca module intro - Tan
- **How we did it**
  - **Program intro** - mostly done (Michelle), needs a few additions from Peter/Dan/Rafi/Alex
  - **GUI** - Dan
  - **Display** - Michelle
  - **Subnets** - Peter
  - **Animation** - Michelle

    - **Module interface** - Tan
    - **Graph Theory Module** - Rafi
    - **Invariant Analysis** - done (Alex)
    - **Extensions**
        - **Extensibility (net types)** - Michelle
        - **Dnamaca module interface** - Tan
- **Evaluation of tool**
    - **Testing of functionality** - Alex
    - **Comparison with other tools** - Alex
    - **Extensions, improvements** - Alex, Michelle, anyone else

Appendices:

- **Users Guide** - done (Dan)
- **How to write a module** - Tan
- **Adding support for a new net type** - Michelle
- **Minutes of meetings** - done (Michelle)
- **Logbook of work** - done (Michelle)

Next meeting: Sunday (17th Mar) - final fixing and putting together reports

Appendix H: Summary

Whilst a project of this nature does not lend itself to division into separate components, as so much of it is interrelated, the division of the work can be said roughly to have been as follows. Daniel and Michelle worked predominantly on the GUI, the extensibility of the tool and the overall structure of application. Peter worked on the subnet element of the tool in addition to working on the Javadoc and more general design issues within the application. Tan produced the interface between the editor and the dynamically loaded modules, the net-flattening tool and the module that integrates with the DANAMICA program. Alex and Raphael focused their efforts on producing the complex invariant analysis and state space analysis modules respectively. The group worked as a whole towards the three reports and the general success of our project!