

The BRITNeY Suite:  
A Platform for Experiments

Michael Westergaard  
Department of Computer Science  
University of Aarhus  
Denmark

# Motivation

- ◆ Research within the field of Coloured Petri nets requires us to perform experiments
  - ◆ State-space algorithms require prototypes to obtain experimental results
  - ◆ Language extensions should be tested
- ◆ It is often interesting to use CP-nets with other formalisms or tools/prototypes

# Motivation

- ◆ We do not want to implement a complete tool for CP-nets ourselves
- ◆ CPN Tools is a great editor for CP-nets, and it would be nice build on that
- ◆ Alas, CPN Tools
  - ◆ is closed source
  - ◆ is written in the Beta programming language
  - ◆ has a closed architecture (i.e. no plug-in or scripting facilities)

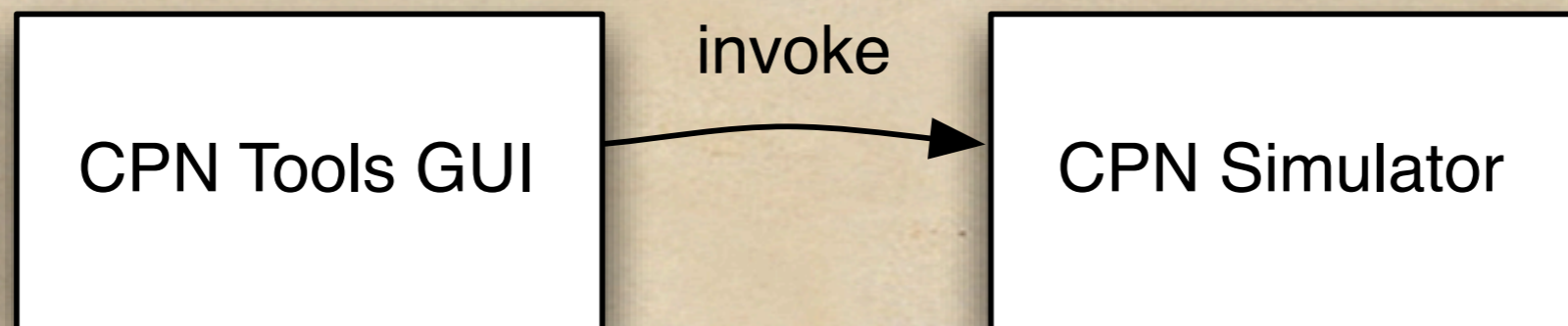
# BRITNeY Suite

- ◆ BRITNeY Suite started as a tool for visualising execution of CPN-models
- ◆ BRITNeY Suite solves the aforementioned issues, as it
  - ◆ is open source (GPL)
  - ◆ is written in a well-known programming language (Java)
  - ◆ has an open architecture (powerful scripting language and plug-in based architecture)

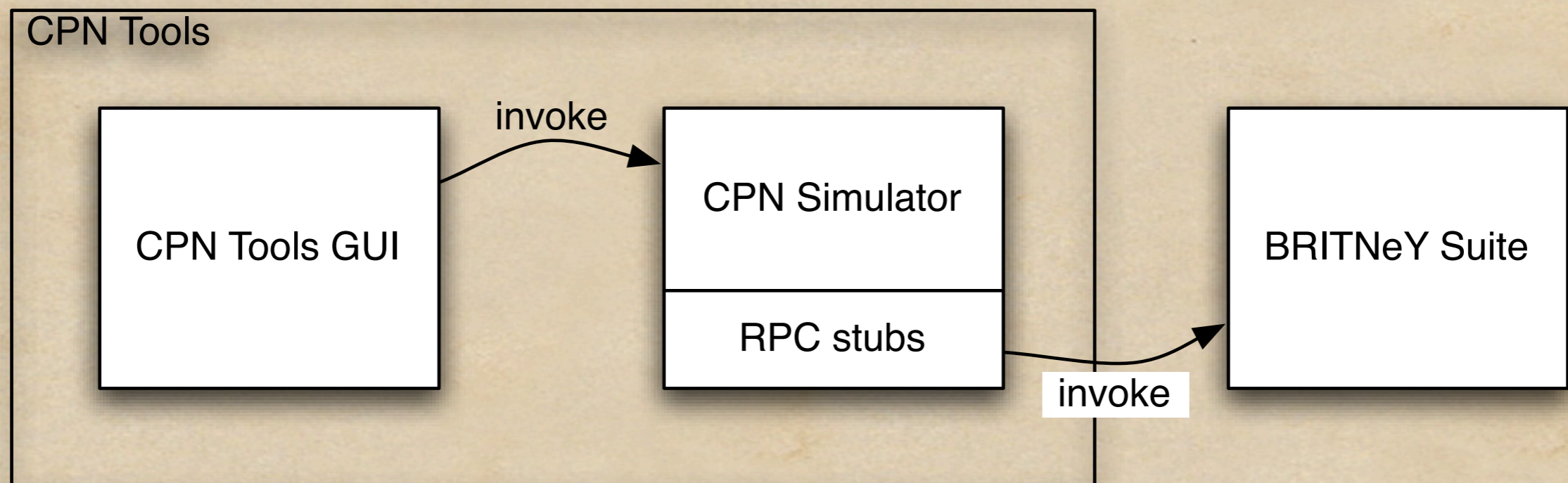
# Overview

- ◆ Architecture of the BRITNeY Suite
- ◆ The scripting features of BRITNeY
  - ◆ A language extension: prioritised transitions
- ◆ A completely new feature: sound

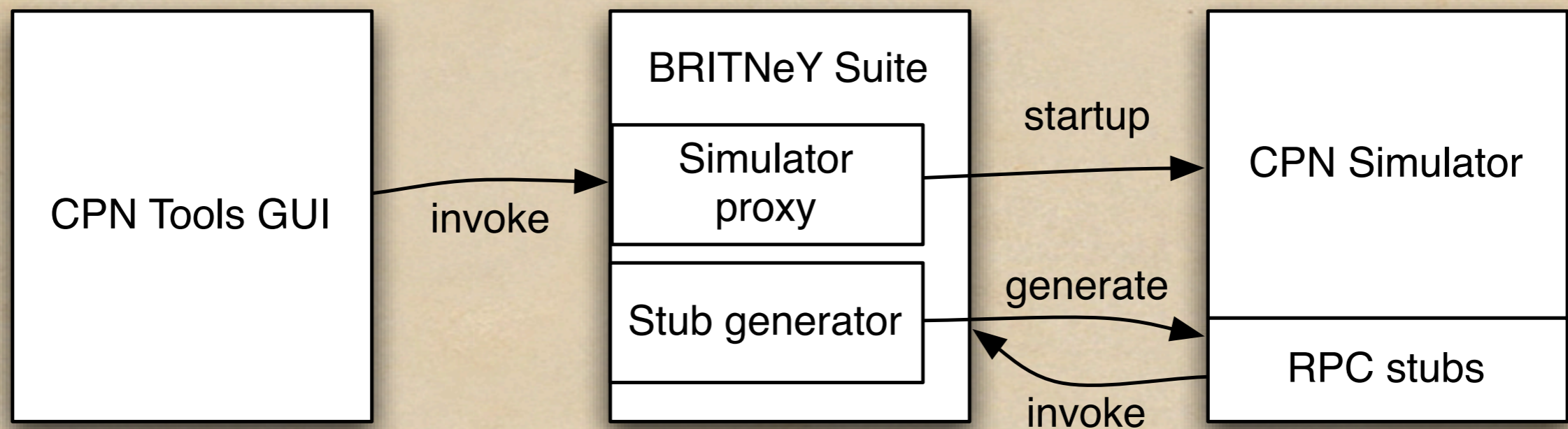
# Architecture: CPN Tools



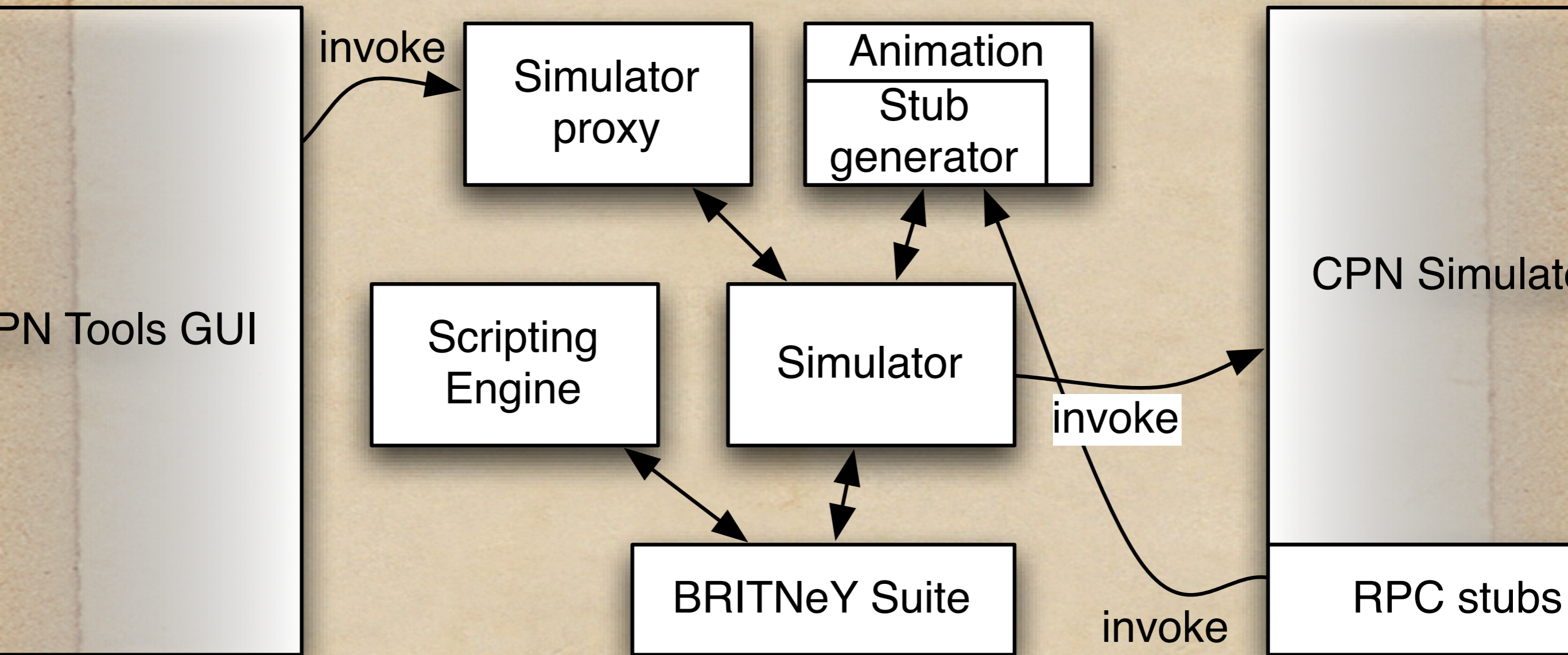
# BRITNeY + CPN Tools



# BRITNeY + CPN Tools



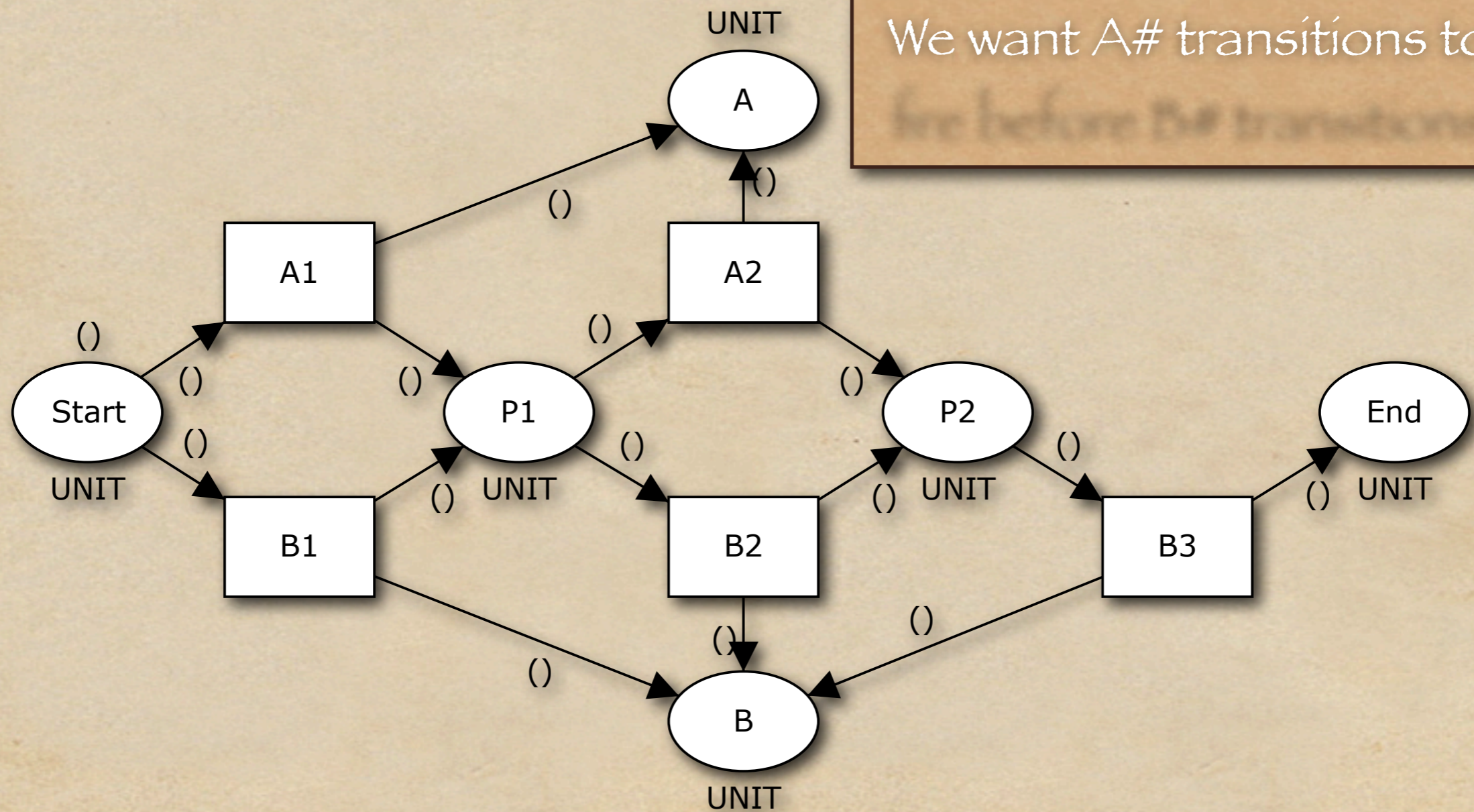
# BRITNeY Suite



# Scripting in BRITNeY

- ◆ Written in BeanShell (relaxed Java syntax)
- ◆ Uses hooks built into the tool
  - ◆ All commands (menu items)
  - ◆ Services provided by BRITNeY, e.g. showing messages to the user, starting simulators
  - ◆ Utility classes for adding items and menus
  - ◆ Everything built into BRITNeY & Java

# Prioritised Transitions



# Prioritised Transitions

- ◆ Strategy
  - ◆ Build lists of A# resp. B# transitions
  - ◆ Fire an A# transition (if one is enabled) or a B# transition (if no A# transition is enabled) as long as any transitions are enabled

# Build Lists of A# + B#

```
As = new ArrayList();  
Bs = new ArrayList();  
for (page : netmodel.getPages())  
    for (transition : page.getTransitions())  
        for (instance : transition.instances())  
            if (transition.getName().startsWith("A"))  
                As.add(instance);  
            else  
                Bs.add(instance);
```

# Execute Transitions

```
simulator.initialState();  
do {  
    if (simulator.fireAny(As))  
        executed = true;  
    else  
        if (simulator.fireAny(Bs))  
            executed = true;  
        else  
            executed = false;  
} while (executed);
```

# Execute Transitions

```
simulator.initialState();  
do {  
    executed = simulator.fireAny(As) ||  
                simulator.fireAny(Bs);  
} while (executed);
```

# Tying it all Together

- ◆ Demo
  - ◆ Execution with normal scheduler
  - ◆ Execution with our new scheduler

# Creating New Features

- ◆ The scripting engine is very powerful, and it is, in principle, possible to write every conceivable extension as scripts
- ◆ Some features are easier to write in plain Java, however

# Plug-ins

- ◆ As BRITNeY is plug-in based, we can implement new features as plug-ins

- ◆ The “core feature” of BRITNeY, animation, is “just” a plug-in



# An Extension: Sound

- ◆ In the paper the complete source code for a plug-in for adding sound to the BRITNeY Suite is shown
  - ◆ Sound can be played directly from the user-interface
  - ◆ Sound can be played from CPN models

Let's give BRITNeY the  
Final Word...