From Analyzing System Failures, to Investigating Crimes

Brian Randell

School of Computing Science

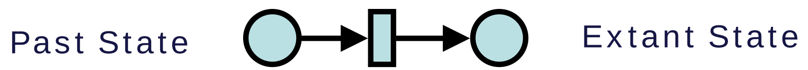
Newcastle University, UK

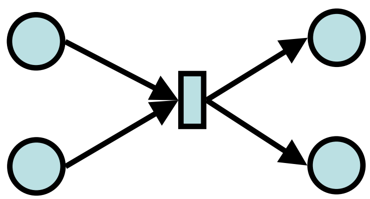
Our Current Theoretical Starting Point

- **Occurrence Nets (ONs)** are a forty-five year old mathematical formalism for representing **causality** and **concurrency** information concerning a ***single execution of a (in general asynchronous) system***.
- ONs are directed acyclic graphs that portray the (alleged) past and present, or the predicted, activity of a system, in terms of conditions (i.e. **states**), transitions (i.e. **events**) and arrows (representing known or alleged **causality**).
- (An ON can be viewed as a generalisation of a "sequential program trace".)
- They can be represented as diagrams – good for tutorials, and for illustrating the basic ideas – but ONs are often represented algebraically, hidden from their users *inside* powerful fully-automated computer tools.
- They are much used in the computer industry, e.g. inside model-checking tools for validating system designs, and even for automatically creating ("synthesizing") VLSI chip designs.

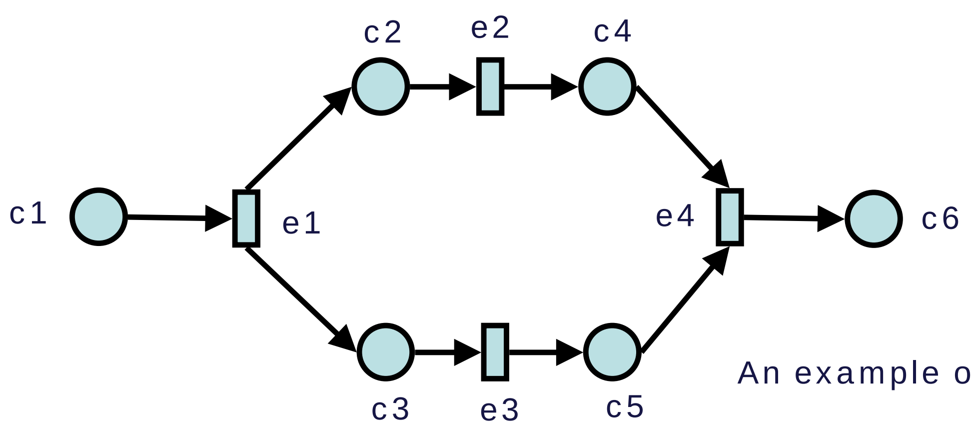
Occurrence Nets

(a <u>formal</u> graphical notation)





Interaction



An example of an Occurrence Net

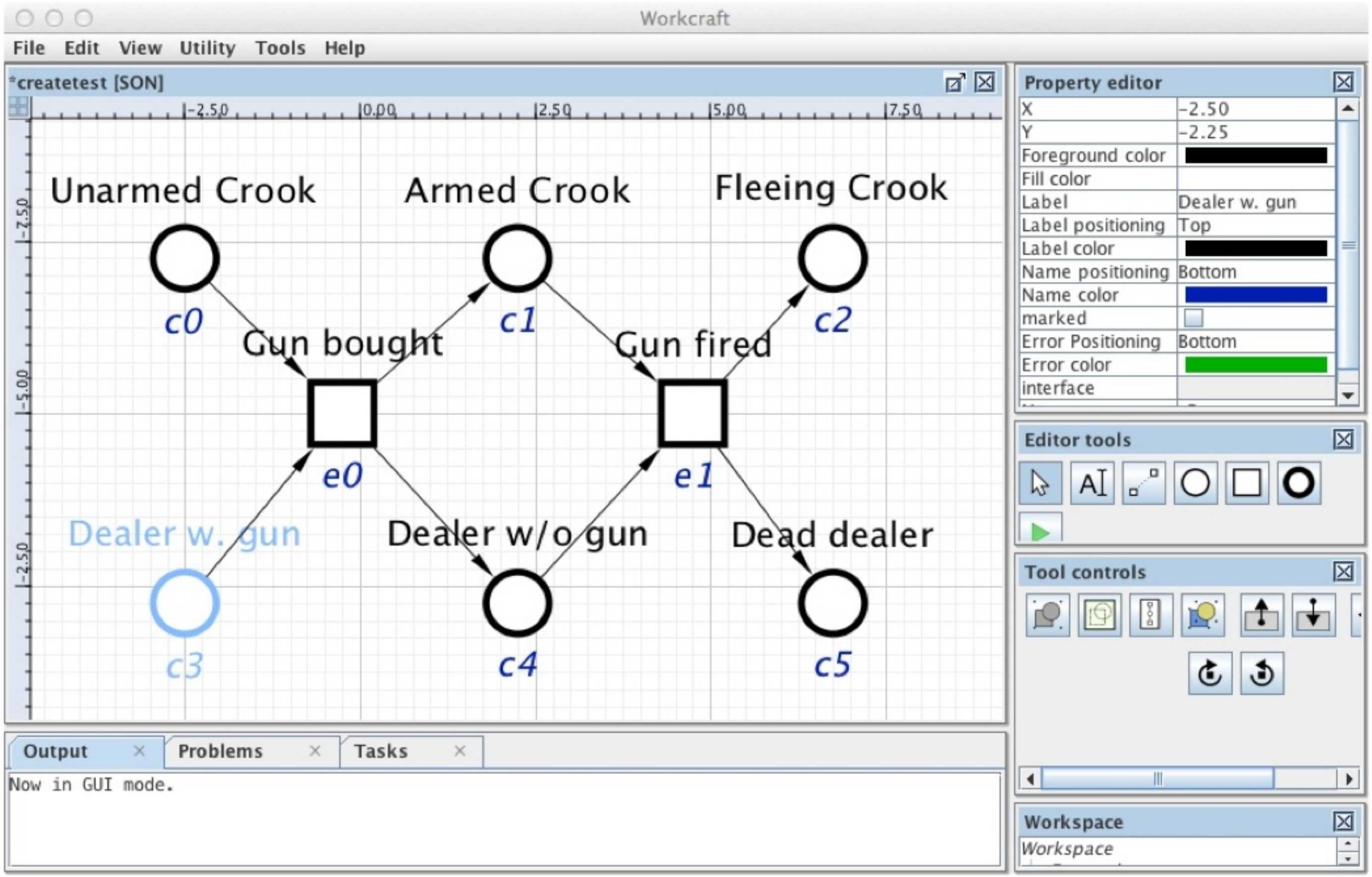
Occurrence Nets at Newcastle

Much work in ASL (Newcastle's joint EE/CS Asynchronous Systems Laboratory), on system design, validation and synthesis, uses ONs.

ASL's most recent interactive tool is WORKCRAFT (an infrastructure for interpreted graph models). See http://workcraft.org/

Editor window

Utility window



Property editor

Editor tools

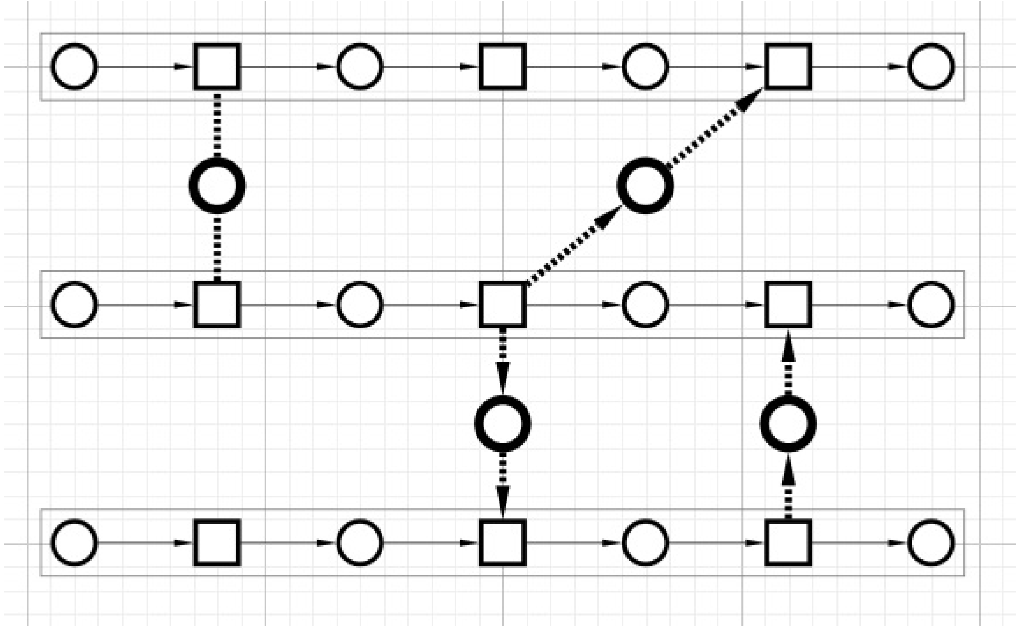
Tool controls

Workspace

Structured Occurrence Nets

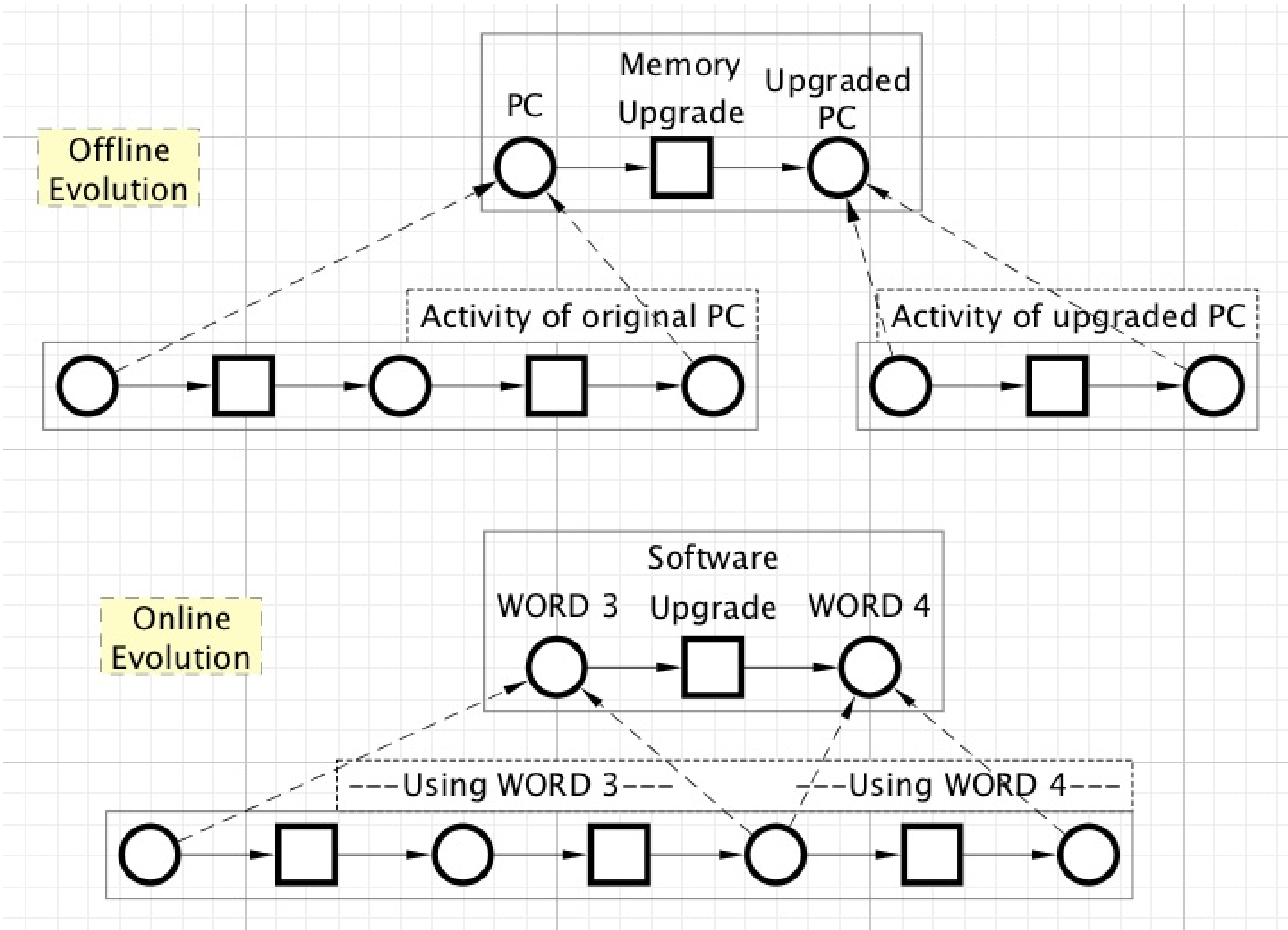
- A **Structured Occurrence Net** (SON) is a set of formally-related Occurrence Nets (using several forms of relation and hence abstraction).
- These include **behavioural abstraction** and **temporal abstraction**.
- SONs, like ONs, are acyclic – and so respect standard causality rules.
- SONs, unlike ONs:
  - enable the activities of different component systems to be readily distinguished
  - provide (through behavioural abstraction) a direct means of modelling the activity of an evolving system, so that
  - their structuring makes them suitable for much more complex systems and situations than ordinary (“flat”) ONs.
- They are of potential relevance to the tasks of verifying and synthesising (asynchronous, i.e. real) systems (of systems)
- But here I concentrate on the task of **analyzing system failures**, i.e. of determining what fault(s) caused what failure(s) – **using crimes as some of the failure examples**.

Multiple Systems



- Basic ONs are good for <u>single</u> (non-evolving) systems. (If you use a large ON to show the activity of a set of interacting different systems, it will not be clear which system is responsible for which activity.)
- But by delineating the ONs that relate to different systems, and using explicit ***communication relations*** to represent their (synchronous or asynchronous) interactions, we can construct a **Communication Structured Occurrence Net (C-SON)** to simplify the modelling and analysis of large systems of systems

(Simple) Examples of Behavioural Abstraction



Note – Such behavioural abstractions *cannot* be represented in ordinary occurrence nets

Failure Analysis

- Failure analysis can involve following links in ONs ***backwards*** from a failure in order to identify causes, and then ***forwards*** to identify further errors and hence further potential failures.
- Behaviour relations between ONs in a SON can similarly be followed in each direction, to trace fault/error/failure chains between an evolving system and the activities it performs.
- Other types of relations between ONs can also be involved in such analysis.
- The identification of failures, errors and faults as such requires additional information, e.g. obtained from system specifications, or users' complaints.
- Such identifications are in principle, and often in practice, made by other (judgemental) systems, whose activities can also be modelled by ONs.

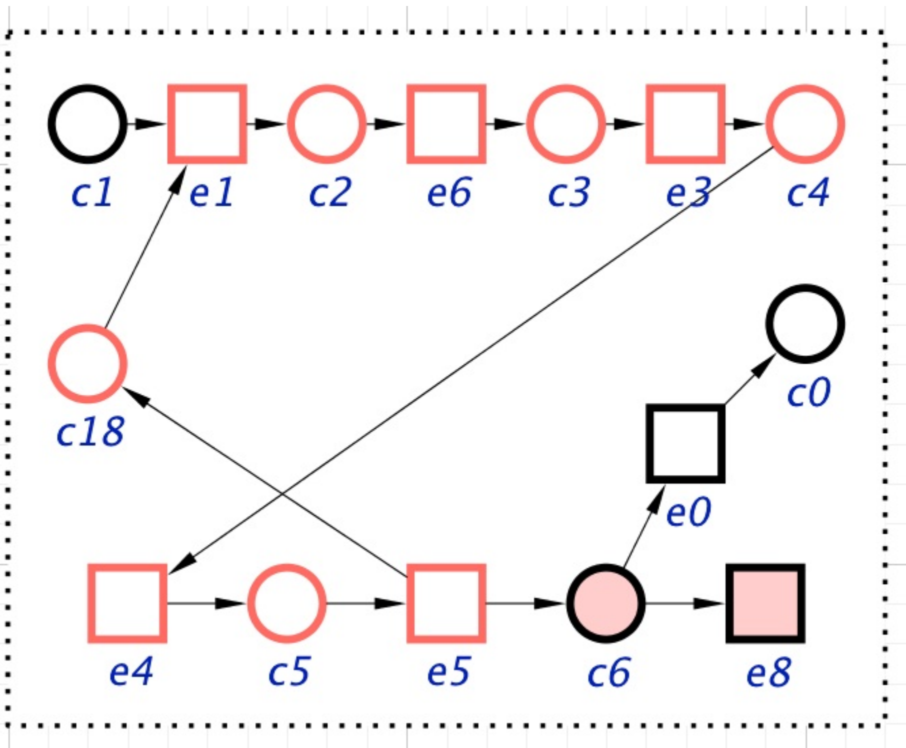
The UNCOVER Project

- A three-year project, now into its second year, funded by the Engineering and Physical Sciences Research Council (EPSRC).
- The project involves research on extending the theoretical foundations of SONs, on tool support for SONs, and on potential application areas.
- The tool support is based on the WORKCRAFT infrastructure for interpreted graph models.
- The prototype tool that has been developed in UNCOVER is called **SONcraft**.
- One of UNCOVER’s chosen illustrative application areas is crime/accident investigation support – the others are system verification (of VLSI chip designs), and on-line deadlock detection in networks.

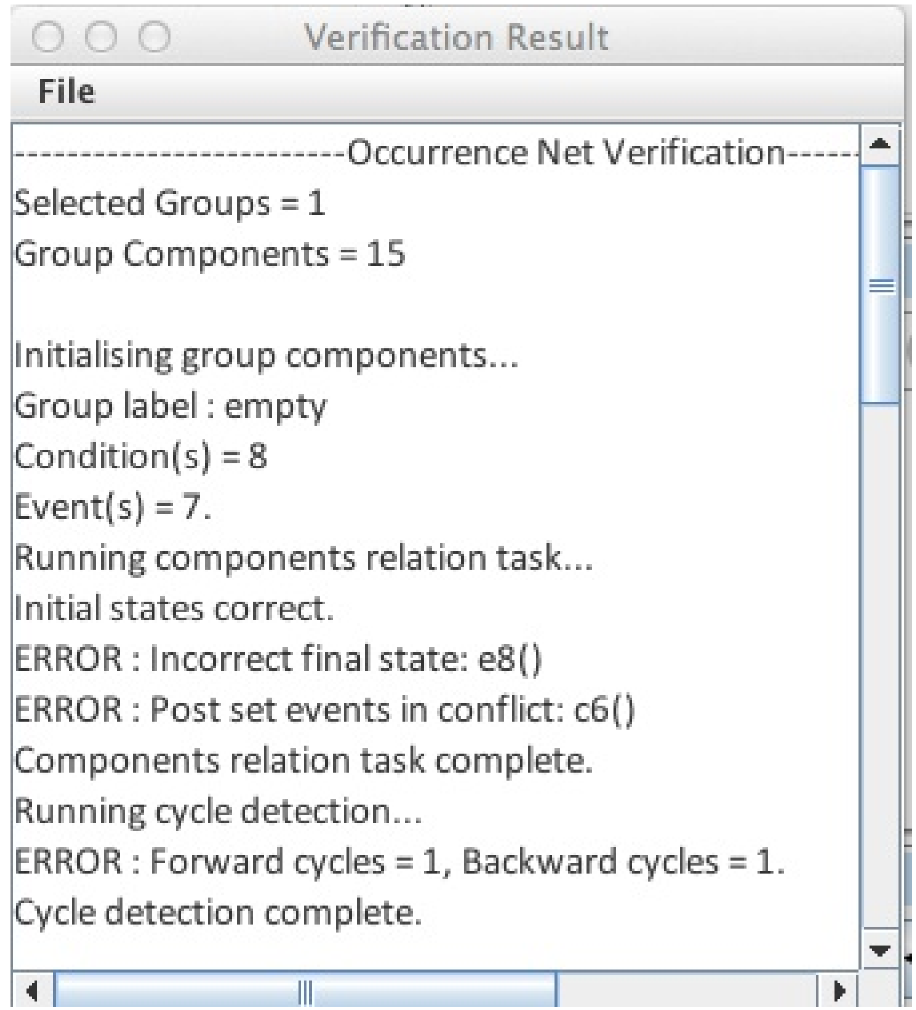
UNCOVER's SONCRAFT Tool

- Implemented (by Bowen Li) as a plug-in to WORKCRAFT
- So far it provides graphic support for editing and portraying :
  - Occurrence Nets (ONs)
  - Structured Occurrence Nets (SONs) incorporating behaviour relations
  - Multiple communicating ONs
  - (Basic) behavioural abstraction and temporal abstraction
  - Basic support for multi-page diagrams
- It enables such ONs and SONs to be verified, with respect to the formal rules of SON and ON validity, e.g.
  - freedom from cycles, caused by arcs within ONs, and by communications or behaviour relations between ONs
  - the requirement for ON fragments to start and terminate in places, not events
  - the fact that places must not have multiple incoming or outgoing arcs.
- And it allows ONs and SONs to be "simulated", i.e. for a user to explore a step at a time "what causes what", in particular what errors have been propagated from one or more identified faults.
- Some example SONcraft screenshots follow.

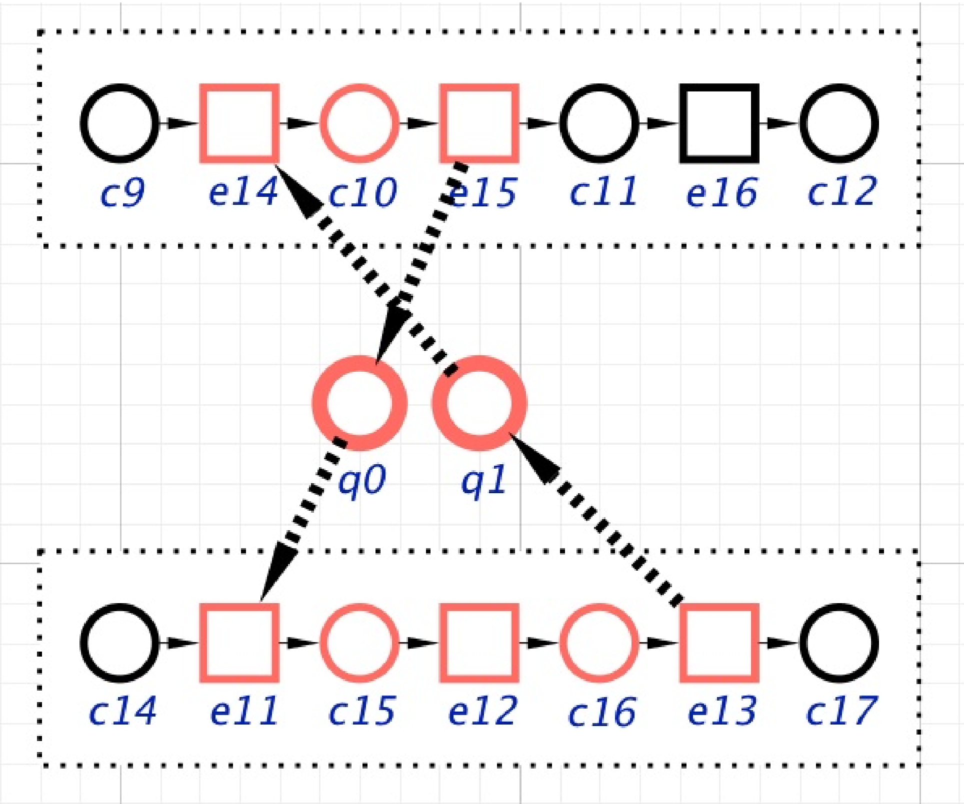
Verification of an ON

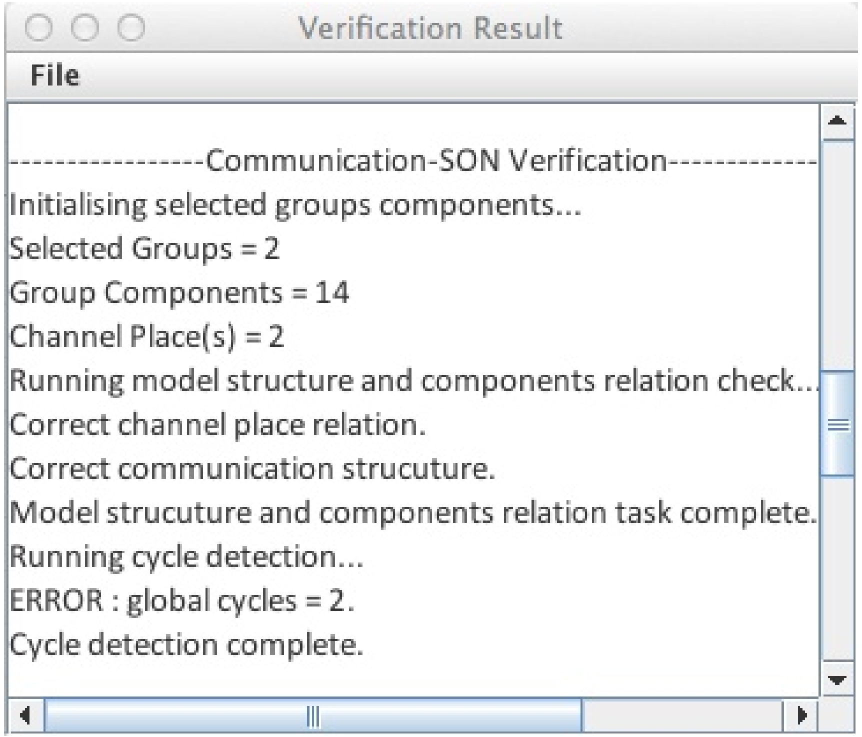


Shaded events and conditions are invalid, ones with red borders are part of a cycle



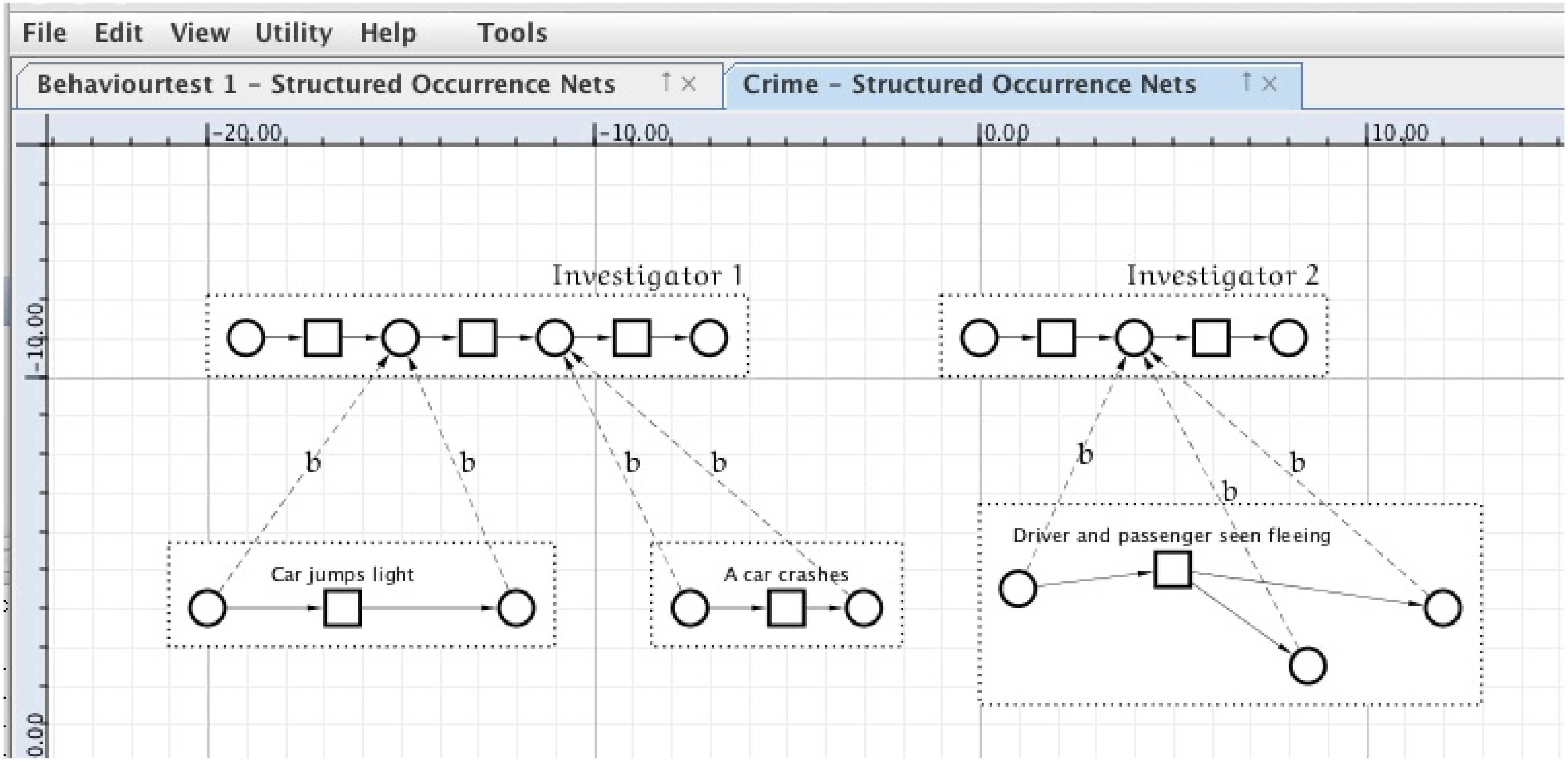
Verification of a Communication SON





Events and conditions with red borders are part of a cycle involving asynchronous communications links joining multiple ONs

Recording an Investigation



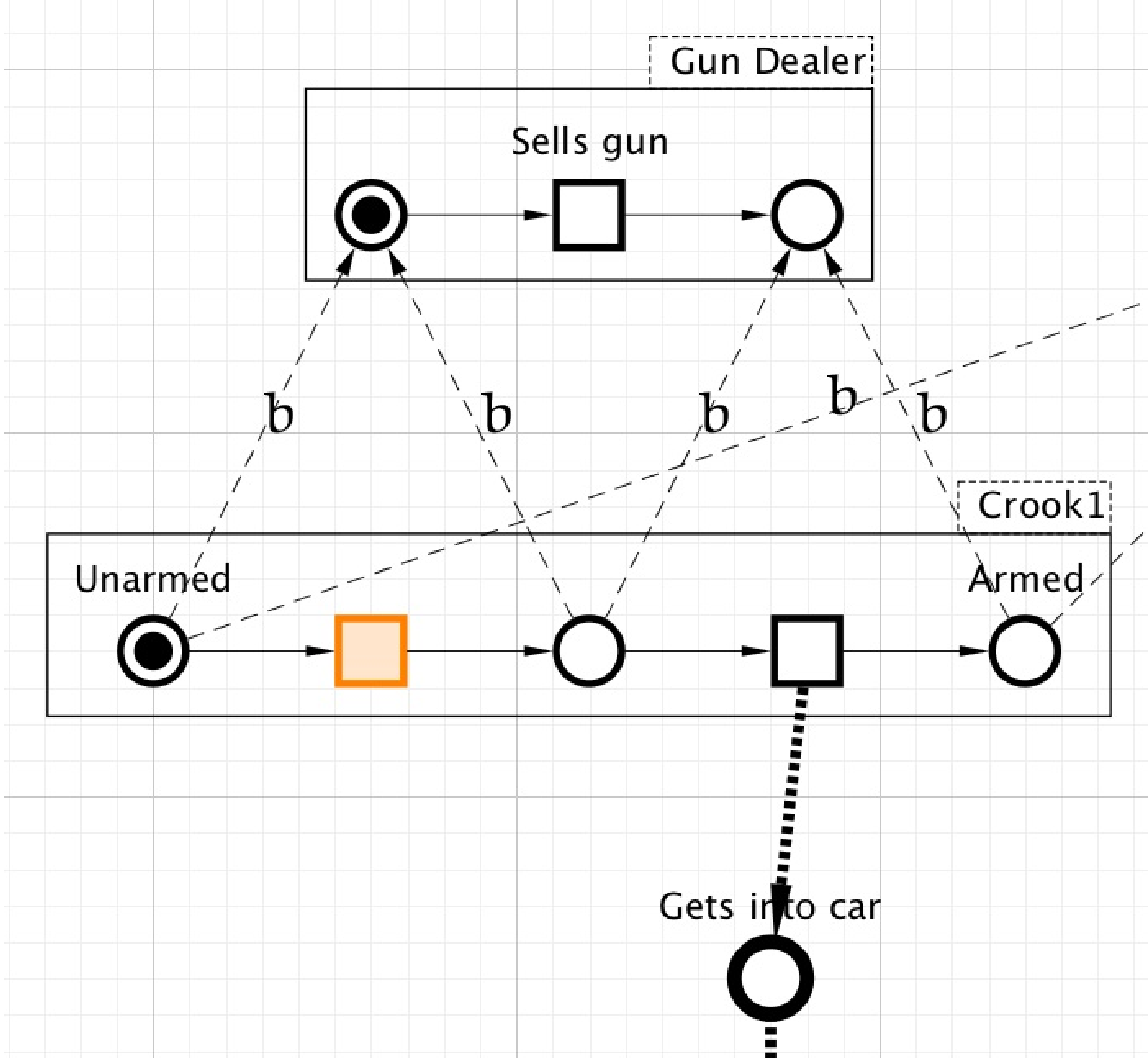
This SONCRAFT screenshot is interpreted as showing which investigator acquired and recorded what information when about a possible crime; it makes use of behavioural abstraction.

SON Simulation

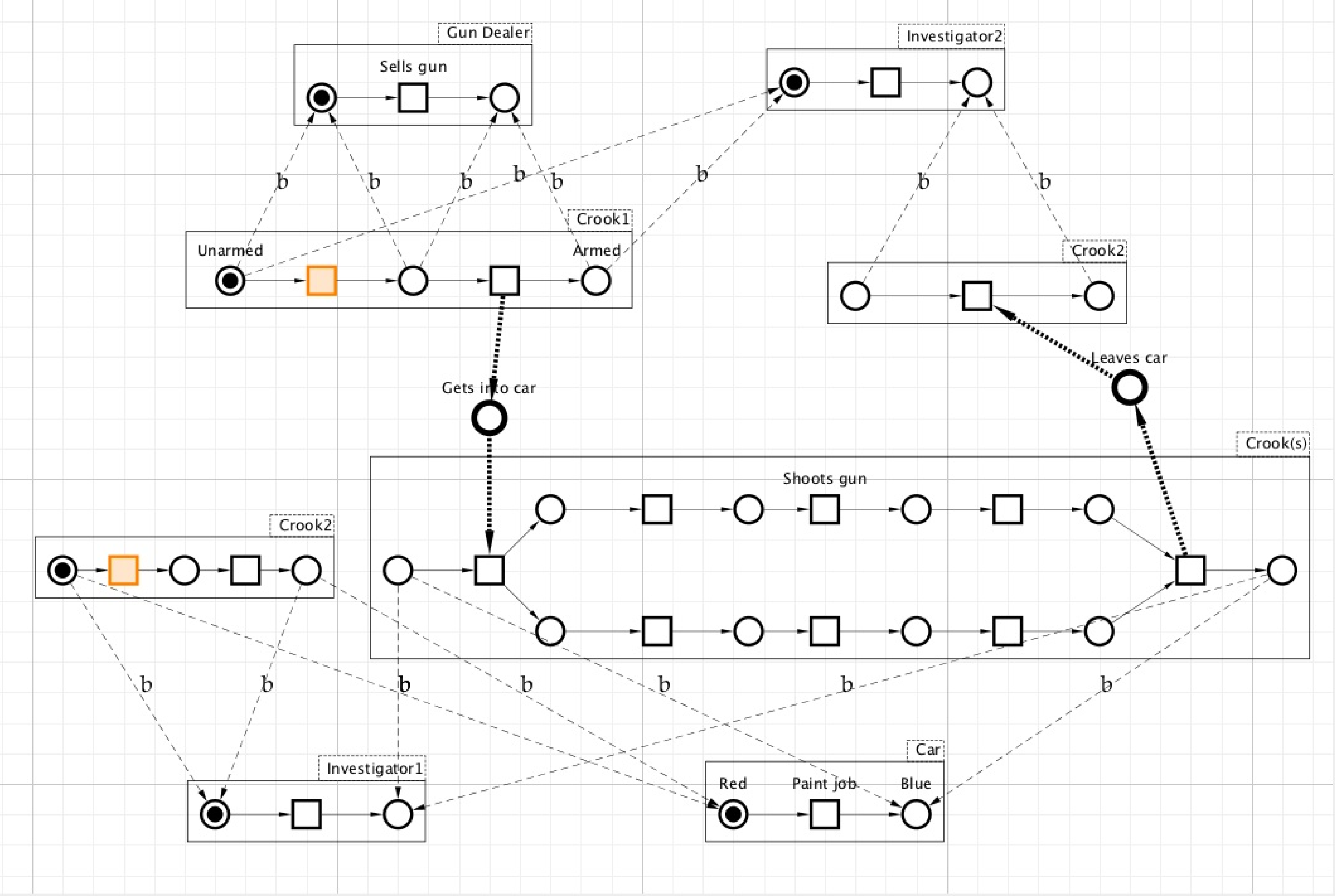
Simulation models error propagation – “reverse simulation” can model fault identification.

The figure shows part of a SON representation of a crime and Its investigation.

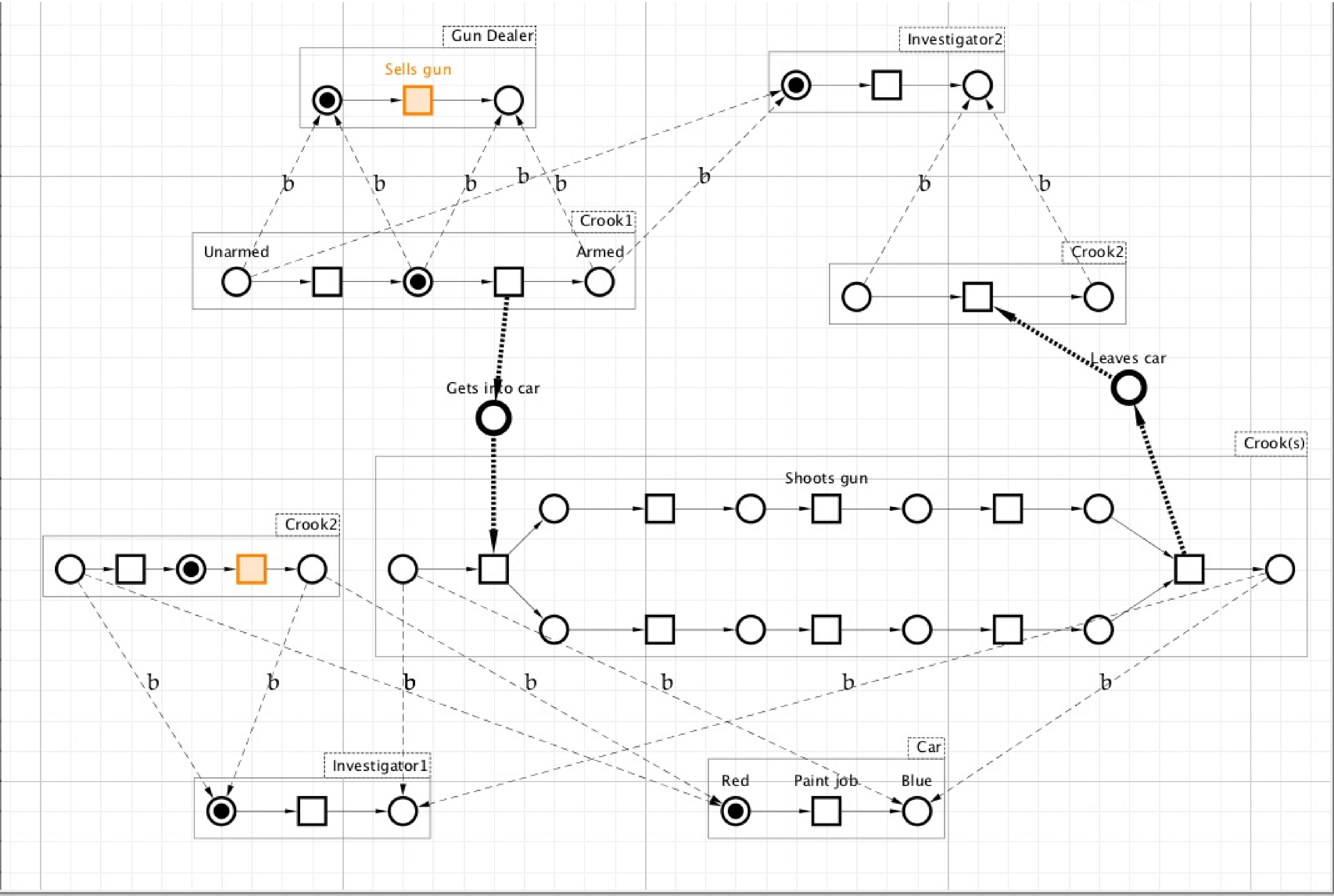
Places with black tokens are “active’ - i.e. ready to contribute to an event. One can click on a chosen event that is coloured (i.e. able to proceed), to see the results of a single execution step.



A crime and its investigation (1)

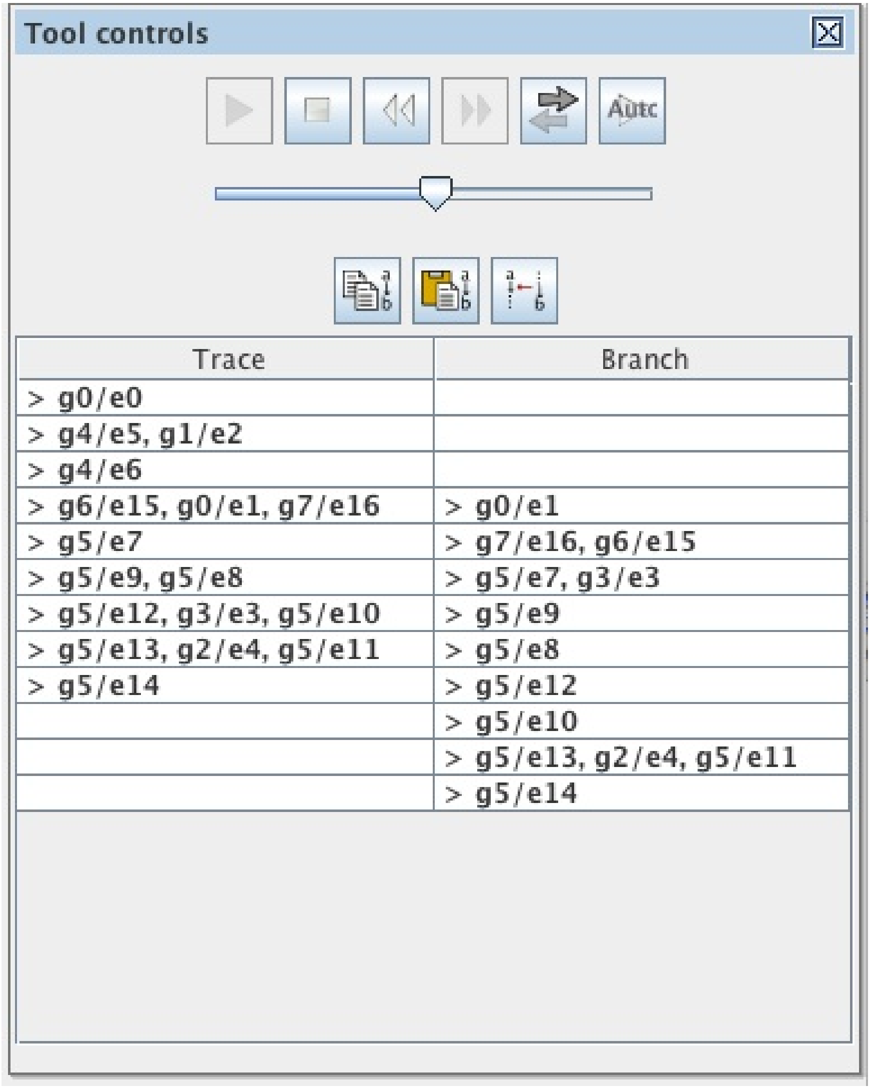


A crime and its investigation (2)



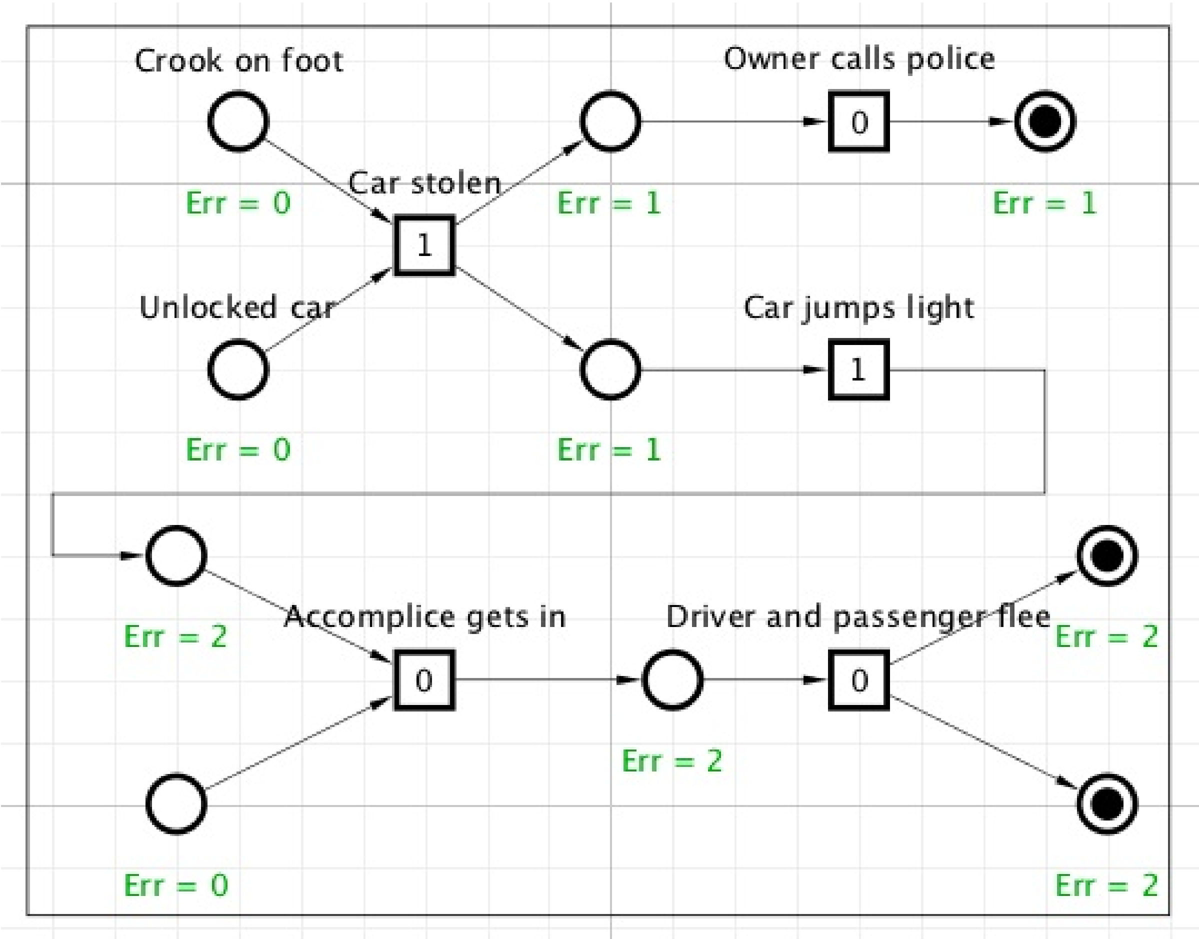
Simulation Control

- Simulation can be performed forwards or backwards, and repeatedly reversed, while exploring fault/error/failure chains.
- As a simulation proceeds a 'trace' record is made of what event(s) occurred, and used to control the simulator.
- Such traces can also loaded with firing sequences produced by tools such as the reachability analyzer.
- Traces can be used to explore (forwards or backwards) through the sequence of recorded events, and to explore alternative choices of which enabled event(s) to select.
- Or they can be run through automatically, at a chosen speed.

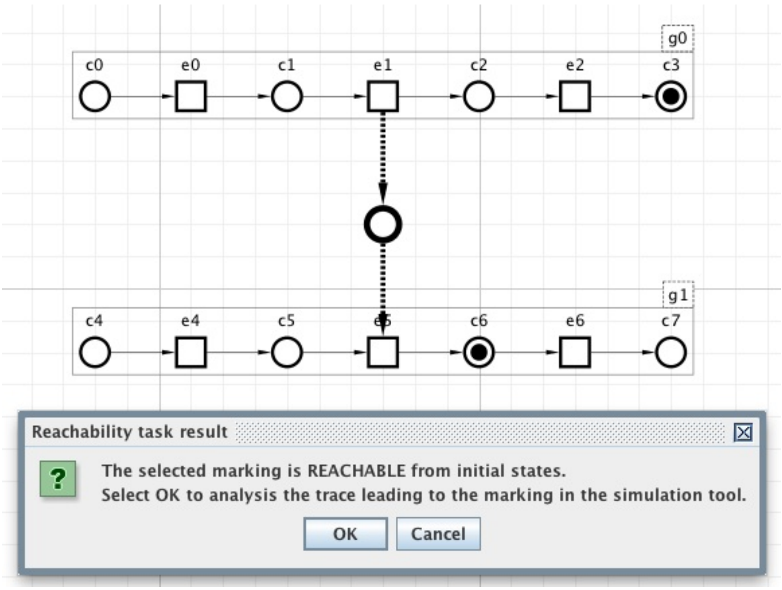


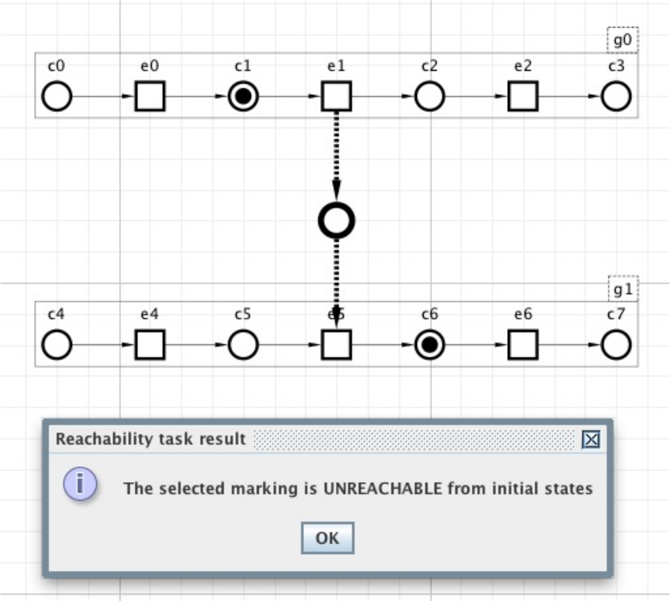
Automated Error Tracking

- Each event can have a Fault bit, to indicate whether the user wishes to regard the event as a faulty one.
- If *error tracing* is turned on then the fault status of each event is flagged with a “1” or a “0”, “1” indicating a simulated fault.
- An error count is shown below each condition, set initially to “0
- During simulation, each condition’s count shows the number of faults that have been passed before it was reached.



Reachability Analysis



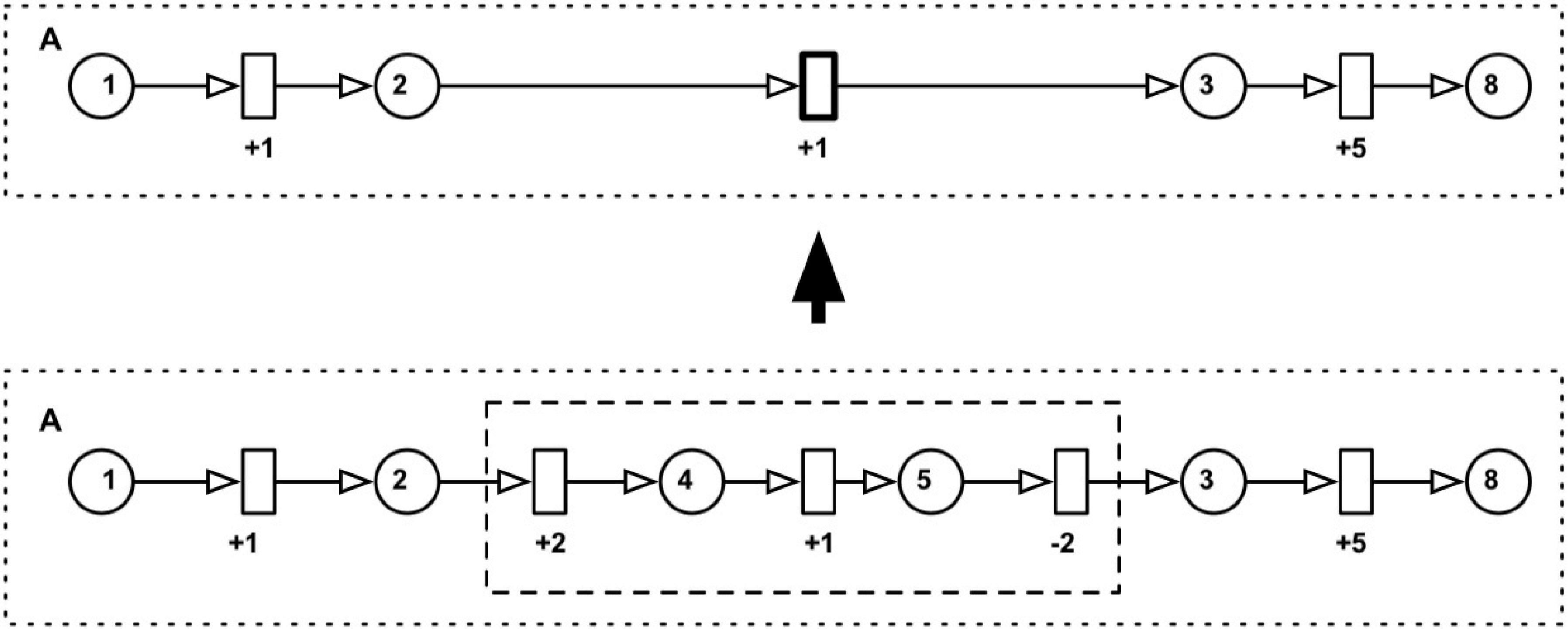


- The Reachability tool allows a user to check whether a given set of states (conditions and/or channel places) can be reachable *at the same time* from a model's initial states.
- (It checks whether any two of the given states are causally linked, i.e. must happen before the other.)
- If the given set of states is reachable (e.g. if crook A and crook B could have been in Bristol Docks simultaneously) then a trace of the events that would lead to this situation is passed to the simulation tool for playback or further analysis.

Concluding Remarks

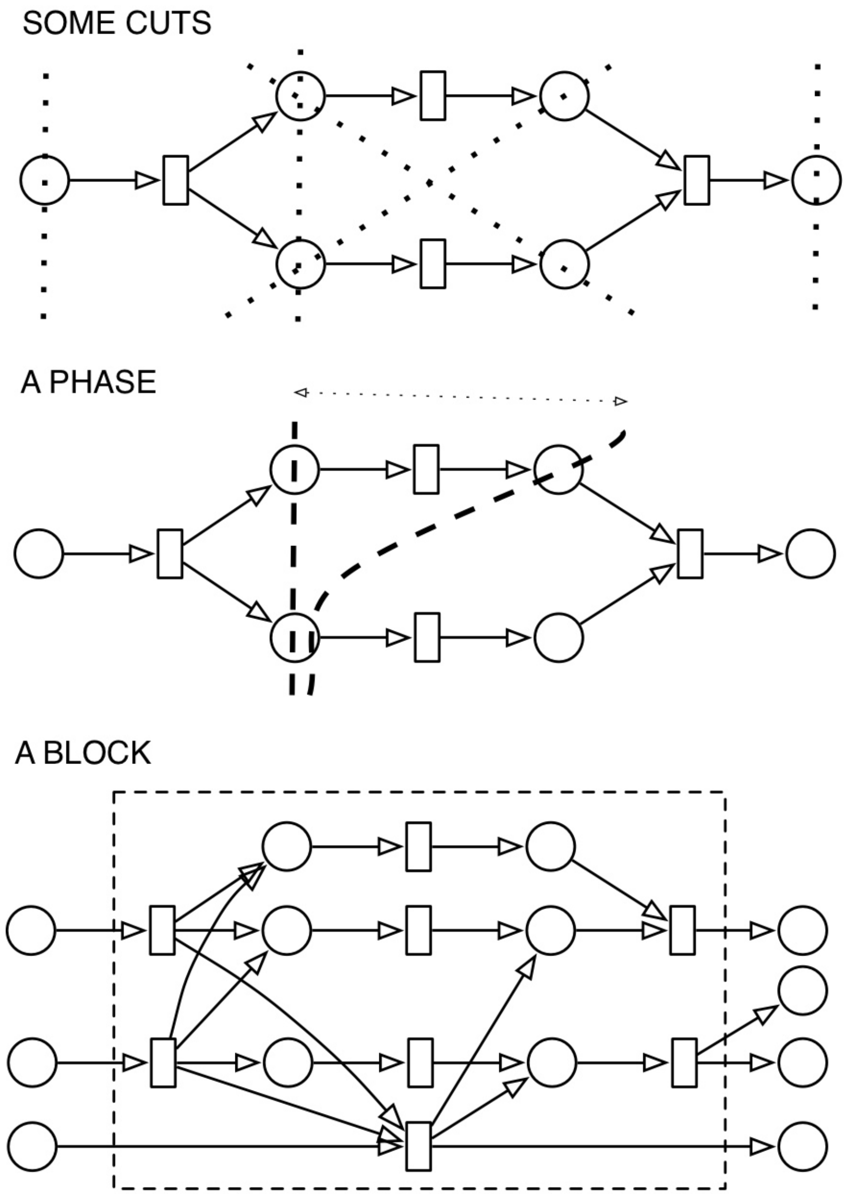
- SONs help reduce the complexity of complicated activity records, and to deal with evolving systems – and are we believe quite novel, and of wide applicability.
- Our EPSRC-funded "UNCOVER" research project involves both further theory development, and implementation of means of representing, analyzing and exploiting fully-general SONs.
- SONCRAFT is to be extended:
  - to deal with large (multi-sheet) diagrams,
  - to link with data sources (e.g. phone records),
  - to represent time (of event occurrences) and durations (of states), and
  - to support the modelling of multiple alternative (assumed) scenarios, associating probability estimates with these alternatives.
- We are discussing the potential of SONCRAFT as a front end to the CLUE crime investigation software, in order to provide investigators with new and enhanced analysis tools.
- SONs are involved in several other recently-submitted research project proposals (to EPSRC and EU), on topics including synthetic biology, cloud-based cybercrime, dynamic real-time system reconfiguration, risk reduction in marine transport, and automated traffic control.

(Simple) Temporal Abstraction – an arithmetic example



Note – The apparent simplicity of both behavioural and temporal abstraction soon vanishes when one starts considering asynchronous systems!

Cuts, Phases and Blocks



Phases are used for behavioural abstraction, **blocks** for temporal abstraction