UNIVERSITY OF NEWCASTLE

# Failures: Their definition, modelling & analysis

(Submitted to DSN)

Brian Randell and Maciej Koutny

WG 10.4, Guadeloupe, Jan 2007

UNIVERSITY OF NEWCASTLE

# Summary of the Paper

- We introduce the concept of a Structured Occurrence Net (SON), based on that of an ‘occurrence net’ (ON) - a well-established formalism for an abstract record that represents causality and concurrency information concerning a single execution of a system.
- SONs consist of multiple related ONs, and are intended for recording either actual system behaviour, or evidence concerning alleged past behaviour.
- We show how SONs can enable better understanding of complex fault-error-failure chains (i) among co-existing interacting systems, (ii) between systems and their sub-systems, and (iii) involving systems that are controlling, supporting, creating or modifying other systems.
- We discuss how, perhaps using extended versions of existing tools, SONs could form a basis for improved techniques of system failure prevention and analysis.

UNIVERSITY OF NEWCASTLE

# The Failure/Fault/Error "Chain"

- A failure occurs when an error "passes through" the system-user interface and affects the service delivered by the system – a system of course being composed of components which are themselves systems. This failure may be significant, and thus constitute a fault, to the enclosing system. Thus the manifestation of failures, faults and errors follows a "fundamental chain":

. . . → failure → fault → error → failure → fault →. . .

i.e.

. . . → event → cause → state → event → cause → . . .

- This chain can flow from one system to:
  - **another system that it is interacting with.**
  - **the system which it is part of.**
  - **a system which it creates or sustains**.
- Typically, a failure will be judged to be due to multiple co-incident faults, e.g. the activity of a hacker exploiting a bug left by a programmer.

UNIVERSITY OF NEWCASTLE

# System Failures

- Identifying failures (and hence errors and faults), even understanding the concepts, is difficult when:
  - there can be uncertainties about system boundaries.
  - the very complexity of the systems (and of any specifications) is often a major difficulty.
  - the determination of possible causes or consequences of failure can be a very subtle, and iterative, process.
  - any provisions for preventing faults from causing failures may themselves be fallible.
- Attempting to enumerate a system's possible failures beforehand is normally impracticable.
- Instead, one can appeal to the notion of a "judgemental system".

UNIVERSITY OF NEWCASTLE

# Systems Come in Threes!

- The “environment” of a system is the wider system that it affects (by its correct functioning, and by its failures), and is affected by.
- What constitutes correct (failure-free) functioning *might* be implied by a system specification – assuming that this exists, and is complete, accurate and agreed. (Often the specification is part of the problem!)
- However, in principle a third system, a **judgemental system**, is involved in determining whether any particular activity (or inactivity) of a system in a given environment constitutes or would constitute – *from its viewpoint* – a **failure.**
- The judgemental system and the environmental system might be one and the same, and the judgement might be instant or delayed.
- The judgemental system might itself fail – as judged by some yet higher system – and different judges, or the same judge at different times, might come to different judgements.

UNIVERSITY OF NEWCASTLE

# Judgemental Systems

- This term is deliberately broad – it covers from on-line failure detector circuits, via someone equipped with a system specification, to the retrospective activities of a court of enquiry (just as the term “system” is meant to range from simple hardware devices to complex computer-based systems, composed of h/w, s/w & people).
- Thus the judging activity may be clear-cut and automatic, or essentially subjective – though even in the latter case a degree of predictability is essential, otherwise the system designers’ task would be impossible.
- The judgement is an action by a system, and so can in principle fail – either positively or negatively.
- This possibility is allowed for in the legal system, hence the concept of a hierarchy of crown courts, appeal courts, supreme courts, etc.
- As appropriate, judgemental systems should use evidence concerning the alleged failure, any prior contractual agreements and system specifications, certification records, government guidelines, advice from regulators, prior practice, common sense, etc., etc.

UNIVERSITY OF NEWCASTLE

# Occurrence Nets

- Directed acyclic graphs that portray the (alleged) past and present state of affairs, in terms of places (i.e. conditions, represented by circles), transitions (i.e. events, represented by rectangles) and arrows (each from a place to a transition, or from a transition to a place, representing (alleged) causality).
- For simple nets, an actual graphical representation suffices. (In the case of complex nets, these are better represented in some linguistic or tabular form.)
- We take advantage of our belated realization that the concepts of ‘system’ and ‘state’ are not separate, but just a question of abstraction, so that (different related) occurrence nets can represent both systems and their states using the same symbol - a ‘place’.
- In fact in this paper we introduce and define, and discuss the utility of, several types of relationship, and term a set of related occurrence nets a *Structured Occurrence Net* (SON).

UNIVERSITY OF NEWCASTLE

# (Graphical) Representation of ONs

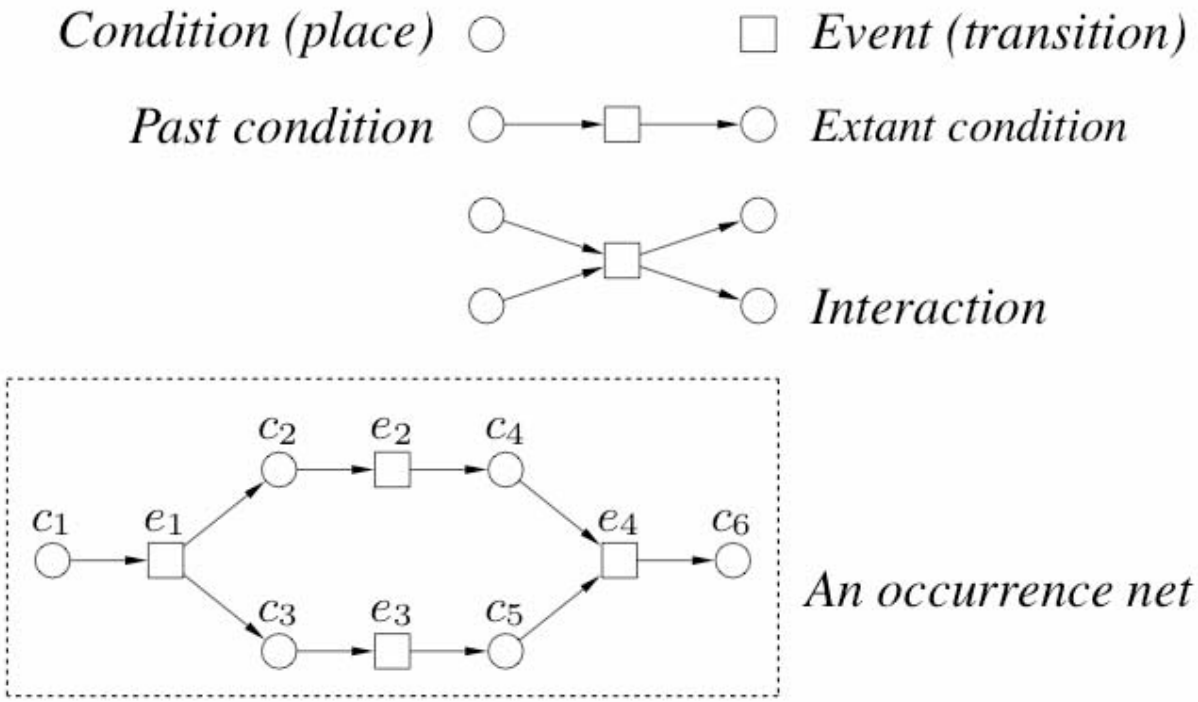


**Figure 1. Basic notation.**

UNIVERSITY OF NEWCASTLE

# System Interaction

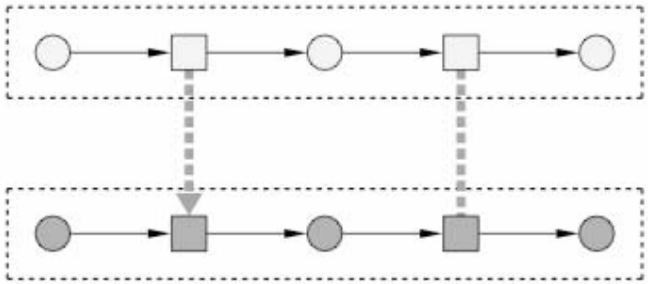

**Figure 2. System interaction.**

Thick dashed arcs indicate that one event is a causal predecessor of another event (information flow was unidirectional), and edges indicate that two events have been executed synchronously (information flow was bidirectional).

UNIVERSITY OF NEWCASTLE

# A Two-Level View of a System

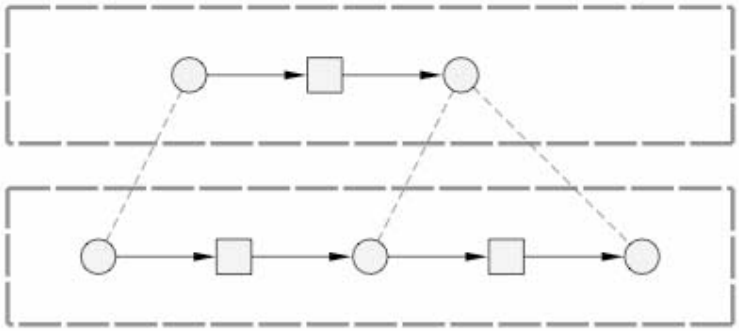

**Figure 3. Simple abstraction.**

The upper level provides a high-level view of system which went through two successive versions - the event in the middle represents a version update. The lower occurrence net captures the behaviour of the system during this period. The ‘abstracts’ relation connecting conditions in the lower part with those in the upper part which abstract them.

UNIVERSITY OF NEWCASTLE

# System Evolution and Behaviour

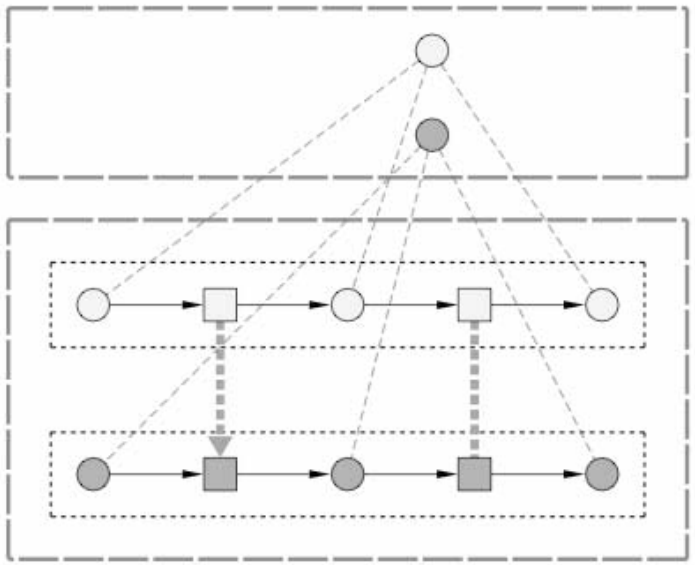


**Figure 4. Behavioural abstraction.**

This shows the existence of two systems, and some details of their (interacting) activities

UNIVERSITY OF NEWCASTLE

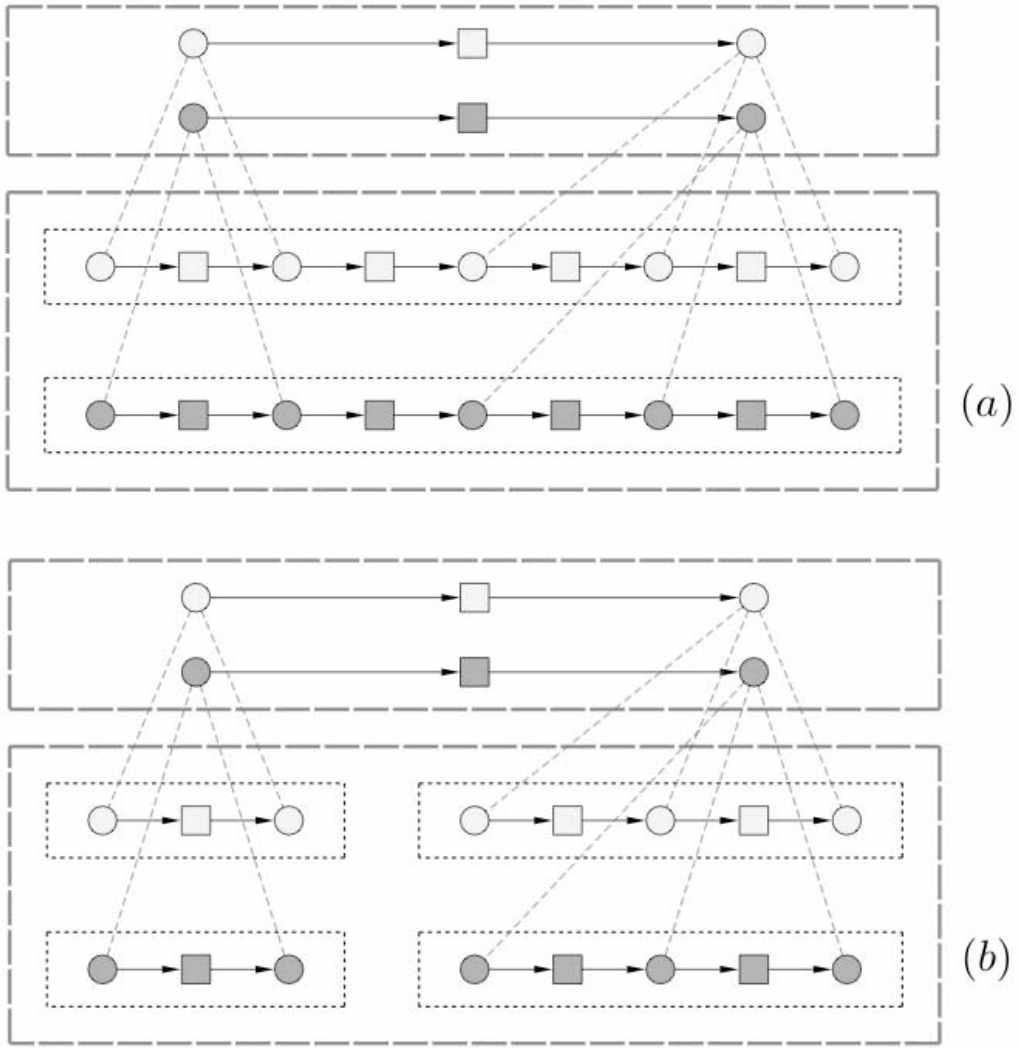


**Figure 5. System modifications.**

# On-line and off-line system modification

UNIVERSITY OF NEWCASTLE

# And system A begat system B . . .

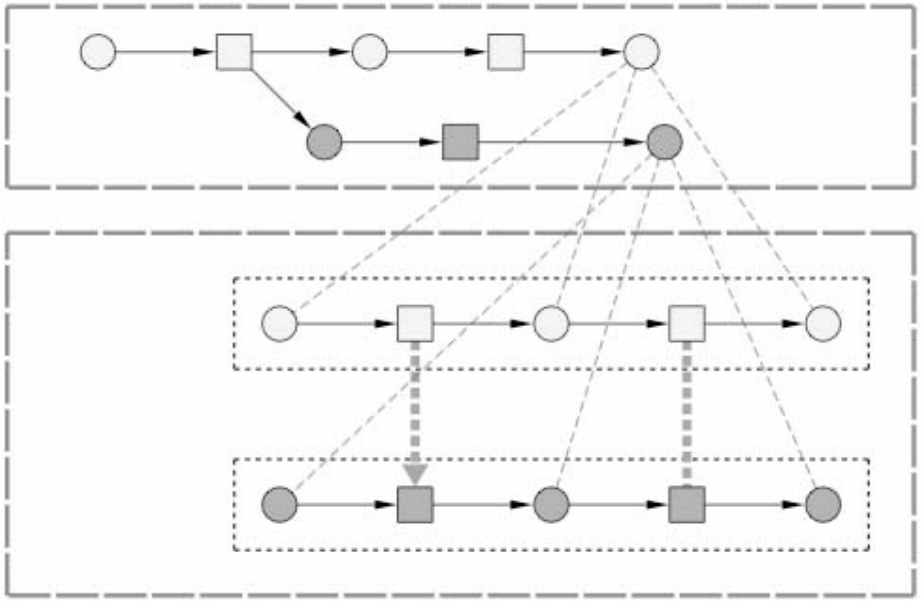

**Figure 6. System creation.**

This shows that one system has spawned another system, and after that both systems went through some independent further evolutions - and indicates how the latest versions of these systems have interacted.

UNIVERSITY OF NEWCASTLE

# Compositional (Spatial) Abstraction

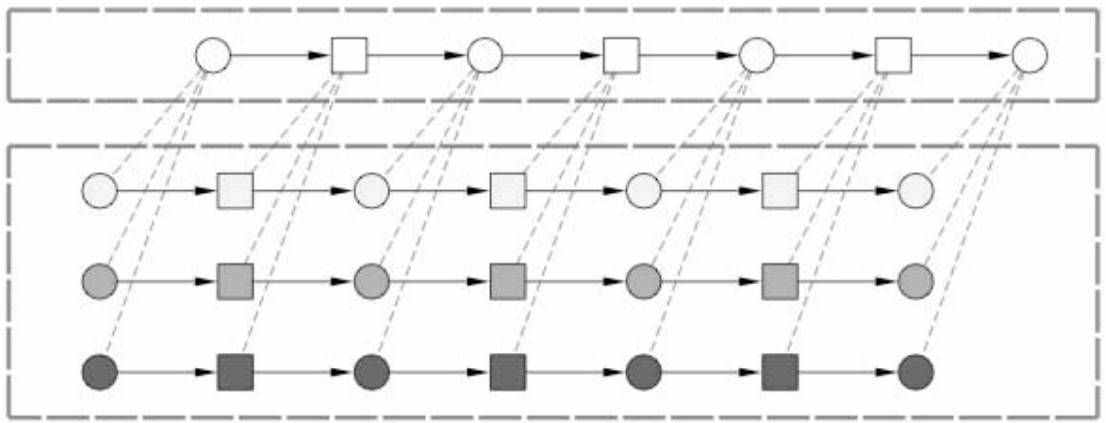

**Figure 7. System composition.**

This shows the behaviour of a system and of its three component systems, and how its behaviour is related to that of its components. (It does not represent the matter of how, or indeed whether, the component systems are enabled to interact, i.e., what design is used, or what connectors are involved.) Each component system has the other two as its *environment.*

UNIVERSITY OF NEWCASTLE

# Abbreviation (Temporal Abstraction)

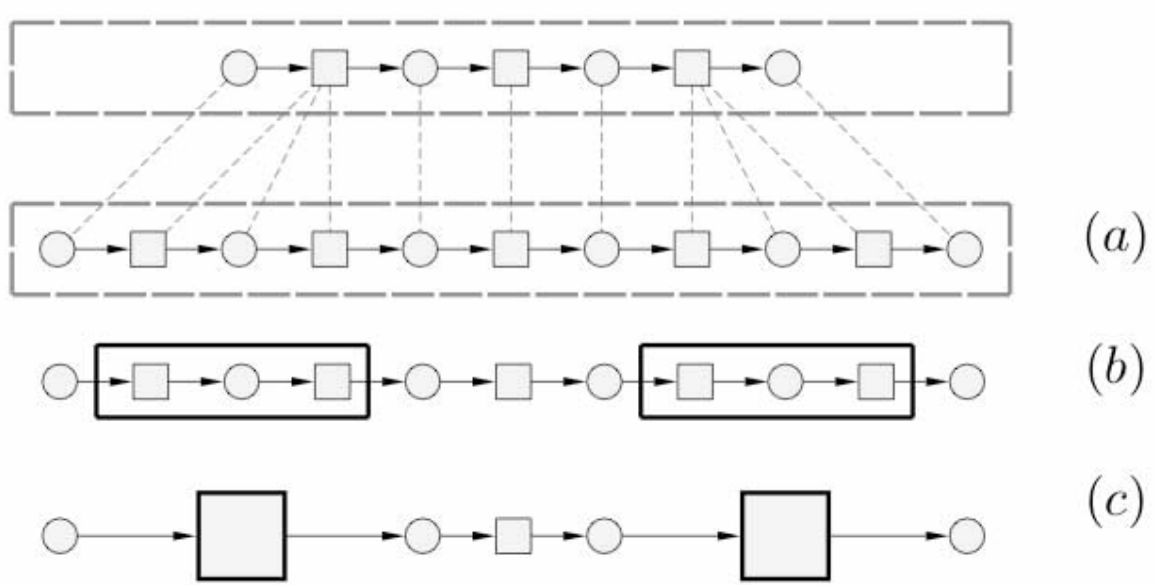


**Figure 8. System abbreviation.**

'Abbreviating' parts of an occurrence net in effect defines atomic actions, i.e., actions that appear to be instantaneous to their environment.

UNIVERSITY OF NEWCASTLE

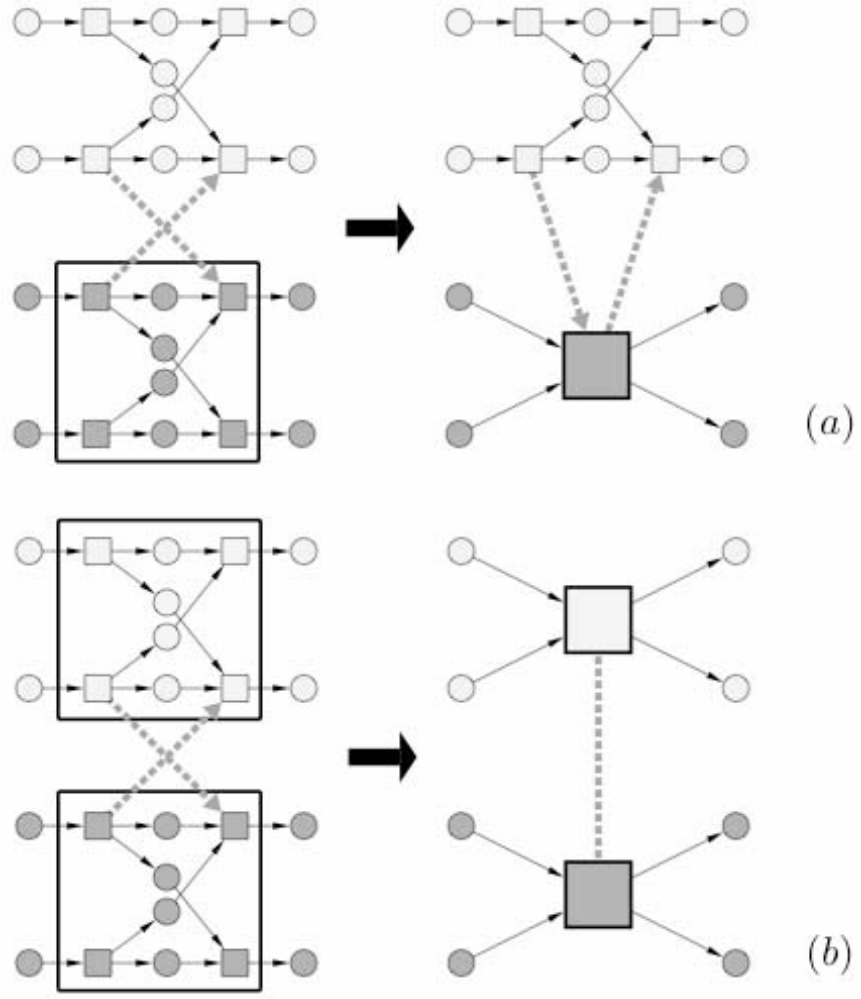


Figure 9. Two valid collapsings.

# Abbreviating ('collapsing') interacting activities

The rules that enable one to make such abbreviations are non-trivial when multiple concurrent activities are shown in the net - one has to avoid introducing cycles into the resulting graph.

UNIVERSITY OF NEWCASTLE

# Recovery Points

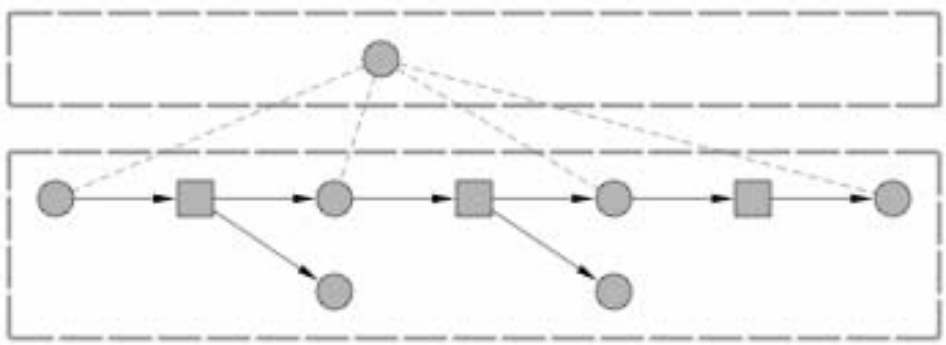

**Figure 10. State retention.**

To allow for the possibility of failure a system might, e.g., make use of ‘recovery points’. Such recovery points can be recorded in retained states that take no further (direct) part in the system’s ongoing (normal) behaviour, as shown above.

UNIVERSITY OF NEWCASTLE

# Judgemental Systems

- The notion of a ‘failure’ event involves, in principle, three systems — the given (possibly failing) system, its environment, and a judgemental system.
- The judgemental system may interact directly and immediately with the given system, in which case it is part of the system’s environment, e.g., a built-in checking circuit, or in a very different world, a football referee!
- Alternatively the judgemental system may be deployed after the fact using an occurrence net that represents how the failing event (is thought to have) occurred.
- Such an occurrence net can be recorded in a retained state, e.g., that of the judgment system.

UNIVERSITY OF NEWCASTLE

# Post-hoc Judgement

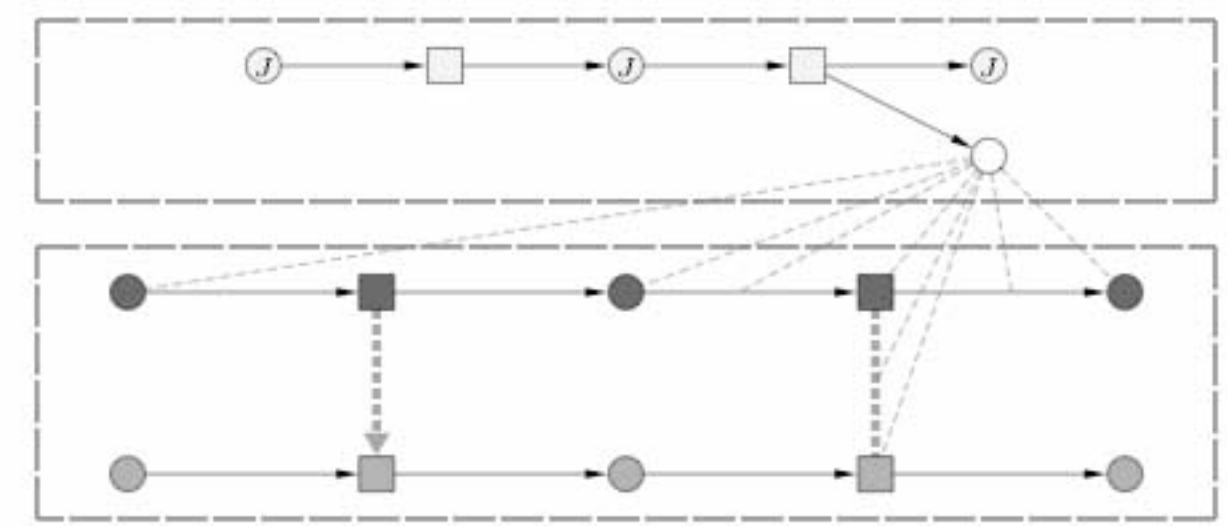


**Figure 11. Post-hoc judgement involving a judgemental system (top) and an active system (bottom).**

This deliberately portrays a situation in which a judgement system has obtained only incomplete evidence of the systems' states and events and even the causal relationships between conditions and events.

UNIVERSITY OF NEWCASTLE

# Failure Analysis

- SONs could be used to represent actual or assumed past behaviour, or possible future behaviour, and to record F-E-F chains between systems.
- They could be generated and recorded (semi?) automatically – alternatively they might need to be generated retrospectively, from whatever evidence and testimony is available.
- Analysis of a SON typically involves following (possibly in both directions) causal arrows *within* ONs, and the various different sorts of relations *between* ONs.
- Such analysis is of course limited by the accuracy and the completeness of the SON – and might be interspersed with efforts at validating and enhancing the SON.

UNIVERSITY OF NEWCASTLE

# Concluding Remarks

- Our various types of abstractions are all ones that could facilitate the task of understanding complex systems and their failures, and analyzing the cause(s) of such failures.
- They would in most cases be a natural consequence of the way the systems, have been conceived and perceived. Thus they can be viewed as providing a means of naturally structuring what would otherwise be an impossibly large and complex occurrence net.
- Alternatively, they can be viewed as a way of reducing the combinatorial complexity of the information accumulated and the analyses performed in following fault-error-failure chains after the fact.
- In either case, computer assistance is needed, something we plan to investigate, building on existing work at Newcastle and elsewhere.
- Our paper provides the formalizations of the various types of abstraction that are needed as a starting point for this investigation. (It's not just a set of pretty pictures!)

UNIVERSITY OF NEWCASTLE

# Examples of What You've Been Spared

**Definition 4.3 (evolutional SON)** *An* evolutional structured occurrence net *is a tuple* $\mathcal{ESON}=(\mathcal{EON},\mathcal{ION},\alpha)$, *where* $\mathcal{EON}$ *and* $\mathcal{ION}$ *are as in Def. 4.2 and 3.1, respectively, and* $\alpha : \mathbf{C} \to C$ *is a mapping such that:*

- $\ell(\alpha(\mathbf{C})) = \mathcal{SYS}$;
- $\alpha(C_i) \cap \alpha(C_j) \neq \varnothing$ *implies* $i = j$, *for all* $i, j \leq k$;
- $\alpha(C_i)$ *is an interval and* $|\ell(\alpha(C_i))|=1$, *for all* $i \leq k$;
- *for every* $i \leq k$ *and every condition* $c \in C$ *with* $\alpha^{-1}(c) \subseteq C_i$, *the sets* $Min_c$ *and* $Max_c$ *of, respectively, all minimal and maximal elements of* $\alpha^{-1}(c)$ *w.r.t. the flow relation* $F_i$ *are cuts of* $\mathcal{ON}_i$;
- *for every* $i \leq N$ *and all conditions* $b, c, d \in \mathbf{C}$ *such that* $\ell(\alpha(b))=\ell(\alpha(d))=i$, *if* $(\alpha(b), \alpha(c)) \in F^+$ *and* $(\alpha(c), \alpha(d)) \in F^+$, *then we have* $\ell(\alpha(c))=i$;
- $Prec_{\mathcal{ESON}} = Prec_{\mathcal{ION}} \cup Prec$ *is an acyclic relation, where* $Prec$ *is the union of sets* $Max_c \times Min_d$, *for all* $e \in E$ *and* $(c, d) \in pre(e) \times post(e)$.

$\mathcal{ION}' = (\mathcal{ON}'_1, \ldots, \mathcal{ON}'_k, \kappa', \sigma')$ *is an interaction occurrence net with* $\mathcal{ON}'_i = (C'_i, E'_i, F'_i)$ *(for* $i \leq k$*), and* $\xi : \mathbf{C}' \cup \mathbf{E}' \to \mathbf{C} \cup \mathbf{E}$; *and, moreover, the following are satisfied, for every* $i \leq k$ *(below* $\mathbf{C}' = \bigcup_i C'_i$, $\mathbf{F}' = \bigcup_i F'_i$ *and* $\mathbf{E}' = \bigcup_i E'_i$*):*

- $\xi(C'_i \cup E'_i) = C_i \cup E_i$, $\xi^{-1}(C_i) \subseteq C'_i$ *and* $\xi(E'_i) = E_i$;
- $\xi^{-1}(e)$ *is a block of* $\mathcal{ON}'_i$, *for every* $e \in E_i$;
- $|\xi^{-1}(c)| = 1$, *for every* $c \in C_i$;
- $F_i = \{(x, y) \mid (\xi^{-1}(x) \times \xi^{-1}(y)) \cap F'_i \neq \varnothing\}$;
- $\kappa = \{(e, f) \mid (\xi^{-1}(e) \times \xi^{-1}(f)) \cap \kappa' \neq \varnothing\}$; *and*
- $\sigma = \{(e, f) \mid (\xi^{-1}(e) \times \xi^{-1}(f)) \cap \sigma' \neq \varnothing\} \cup \{(e, f) \mid (((\xi^{-1}(e) \times \xi^{-1}(f)) \cap \kappa' \neq \varnothing) \wedge ((\xi^{-1}(f) \times \xi^{-1}(e)) \cap \kappa' \neq \varnothing))\}$.

UNIVERSITY OF NEWCASTLE

# Some of our References


- Best, E. and Randell, B. (1981). A Formal Model of Atomicity in Asynchronous Systems, *Acta Informatica*, Vol. 16 (1981), pp 93-124. Springer-Verlag Germany. http://www.cs.ncl.ac.uk/research/pubs/articles/papers/397.pdf
- Chatain, T. and Jard, C. (2004). Symbolic Diagnosis of Partially Observable Concurrent Systems. Proc. of FORTE'04, LNCS 3235, 326–342.
- Grahlmann, P and Best, E: PEP - More than a Petri net tool. Proc. of TACAS'96, LNCS 1055, 1996, pp.397-401 [PEP]
- Holt, A.W., Shapiro, R.M., Saint, H., and Marshall, S., "Information System Theory Project", Appl. Data Research ADR 6606 (US Air Force, Rome Air Development Center RADC-TR-68-305), 1968.
- Khomenko, V. and Koutny, M.: Branching Processes of High-Level Petri Nets, Proc. of TACAS'03, LNCS 2619, 2003 pp.458-472, http://www.cs.ncl.ac.uk/research/pubs/articles/papers/425.pdf
- Merlin, P.M. and Randell, B. State Restoration in Distributed Systems, In *Proc FTCS-8*, Toulouse, France, 21-23 June 1978 pp. 129-134. IEEE Computer Society Press 1978 http://www.cs.ncl.ac.uk/research/pubs/articles/papers/347.pdf