Newcastle University

System Complexity, Dependability and Failures

Brian Randell

Newcastle University

Menu

- On (False) Simplicity
- Dependability and Security Concepts
- (Formal) Modelling of Failures in Complex Evolving Systems
- A Topical Example of (Avoidable) System Complexity
- Concluding Remarks

Complexity and Simplicity

- *Complexity* in general usage is the opposite of *simplicity*.
- Tony Hoare on *simplicity*:
  - "The price of dependability is extreme simplicity, and this is a price that most computer and software companies find to be excessively high."

Newcastle University

Complexity and Simplicity

- *Complexity* in general usage is the opposite of *simplicity*.
- Tony Hoare on *simplicity*:
  - "The price of dependability is extreme simplicity, and this is a price that most computer and software companies find to be excessively high."
- Albert Einstein on *simplicity*:
  - "Everything should be as simple as possible, but no simpler!"

Complexity and Simplicity

- *Complexity* in general usage is the opposite of *simplicity*.
- Tony Hoare on *simplicity*:
  - "The price of dependability is extreme simplicity, and this is a price that most computer and software companies find to be excessively high."
- Albert Einstein on *simplicity*:
  - "Everything should be as simple as possible, but no simpler!"
- Me:
  - "One of the best ways of making things *excessively simple* is to ignore possible causes of system failure!"
  - "Another is to avoid bothering to be very careful to define one's terms, or to ensure that terminological variations are fully understood by all parties to a discussion."
- Hence my view of the importance of the subject of system dependability, and of teasing out and understanding its fundamental concepts.

Newcastle University

Dependability

- There are severe terminological and conceptual confusions in and around the dependability field.
- Words like flaw, bug, error, failure, malfunction, incident, exception, mistake, etc., are bandied around, vaguely.
- New terms, such as trustworthiness, high confidence systems, survivability, resilience, and autonomic systems are introduced, and defined in terms that are virtually identical to the pre-existing definition of dependability.
- The relationship between dependability and security is problematic and controversial.
- But what matters is clarity of concept, rather than uniformity of nomenclature.

On Failures

- To me, **failures** are the central issue, *the* most basic concept - in the field of dependability (however this is named).
- Particular types of failures (e.g. producing wrong results, ceasing to operate, revealing secret information, causing loss of life, etc.) relate to what can be regarded as different attributes of dependability: reliability, availability, confidentiality, safety, etc.
- Complex real systems, made up of, and by, other systems (e.g. of hardware, software and people) do actually fail from time to time (!), and reducing the frequency and severity of their failures is the major challenge.
- Hence *a dependable system is one whose failures are not unacceptably frequent or severe* (from some given viewpoint).

Three Basic Concepts

- A system **failure** occurs when the delivered service is adjudged to have deviated from fulfilling the system function.
- An **error** is that part of the system state which is *liable to lead to subsequent failure:* an error affecting the service is an indication that a failure occurs or has occurred. The *adjudged or hypothesised cause* of an error is a **fault**.
  - (Note: errors do not necessarily lead to failures – this may be avoided by chance or design; component failures do not necessarily constitute faults to the surrounding system – this depends on how the surrounding system is relying on the component).
- These three concepts (an *event*, a *state*, and a *cause*) must be distinguished, whatever names you choose to use for them.

Basic Concepts and Taxonomy of Dependable and Secure Computing, Avizienis, A., Laprie, J.-C., Randell, B. and Landwehr, C. *IEEE Transactions on Dependable and Secure Computing*, Vol. 1, No. 1, pp 11-33, 2004

Newcastle University

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".

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. (But 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 term judgemental system 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.)
- 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.

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.

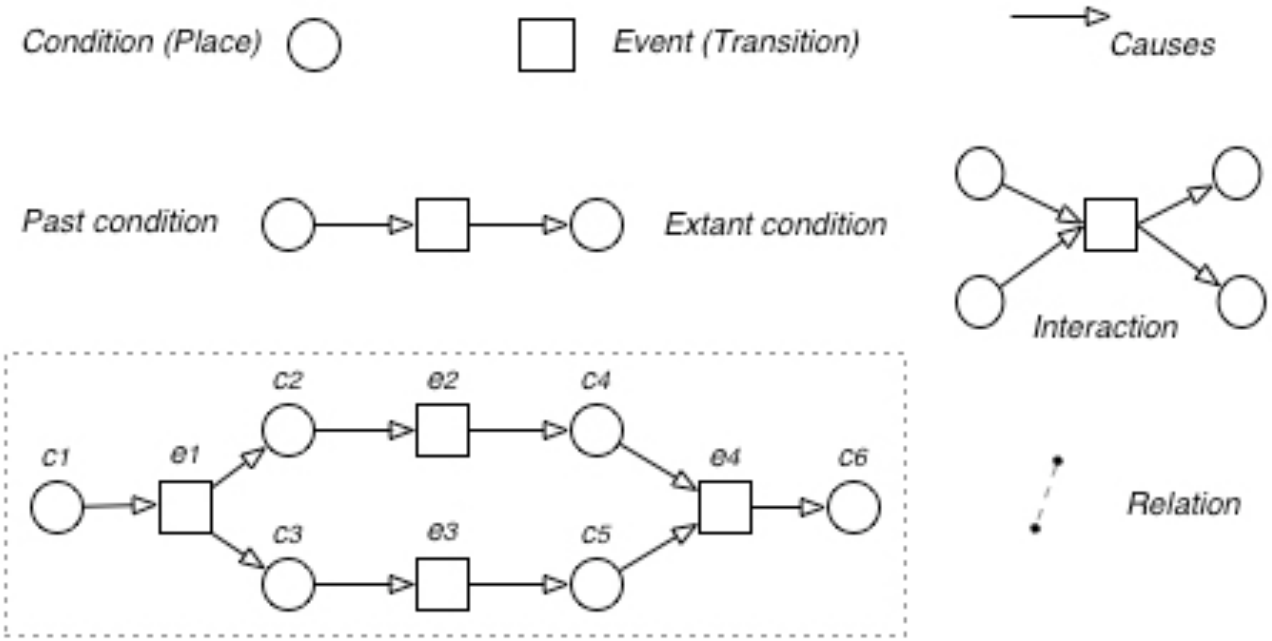
. . . → event → cause → state → event → cause → . . .

- 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.

A Role for (Formal) Models

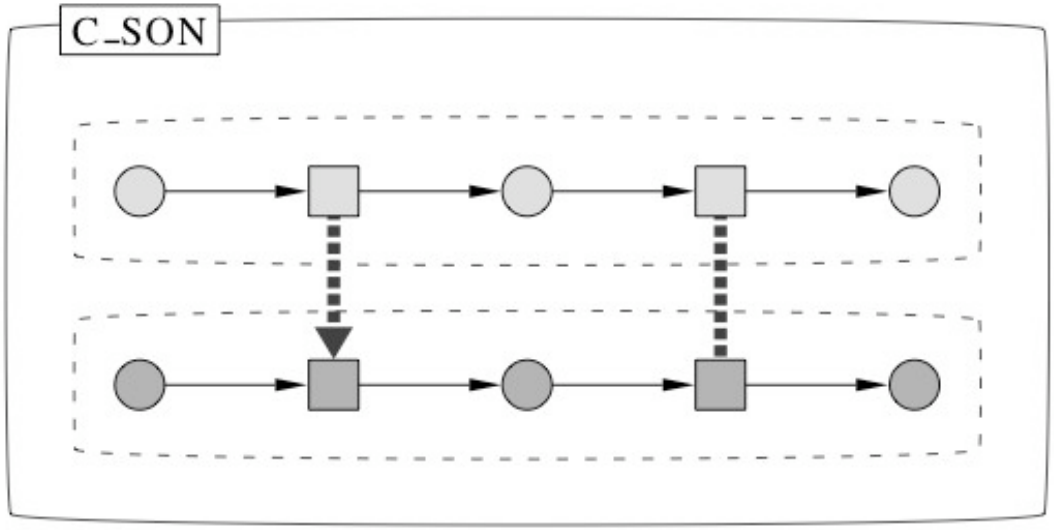
- If one could express even some of these ideas in a formal notation, this might facilitate:
  - the analysis of system failures.
  - the analysis and design of (fault-tolerant) systems themselves.
- The notation I've been experimenting with is that of *Occurrence Nets* (aka *Causal Nets, Occurrence Graphs, etc.*).
- ONs represent what (allegedly) happened, or might happen, and why, in a system – they model system behaviour, not actual systems.
  - Simple nets can be shown pictorially.
  - They can be expressed algebraically, and have a formal semantics.
  - Tools exist for their analysis (model-checking) and manipulation – and even for synthesizing systems from them (in simple cases).
- My thought experiments have concerned what might be called "Structured Occurrence Nets".
  - Their structure results from notions like the fundamental F-E-F chain.
  - This structure provides significant complexity reduction, and so could facilitate (automated) failure analyses, or possibly even system synthesis.

Occurrence Net Notation



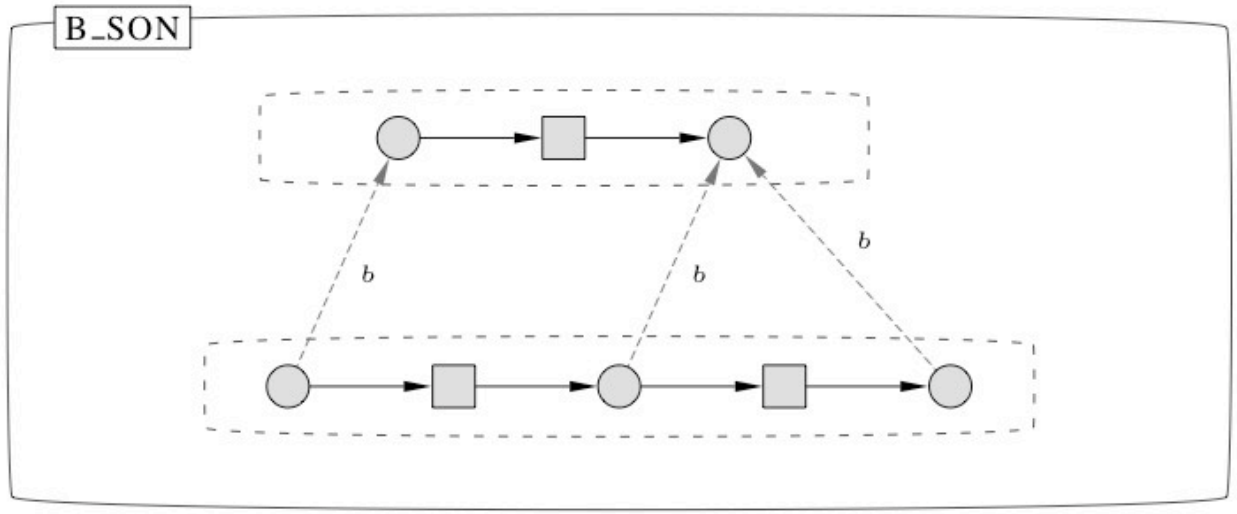
The (simple and perhaps new) idea is to introduce various types of (formal) relation *between* Occurrence Nets, and treat a set of such related ONs as a "Structured ON", hence much more manageable than a large unstructured (single level) ON describing the same complex situation.

The “Communicates” Relation



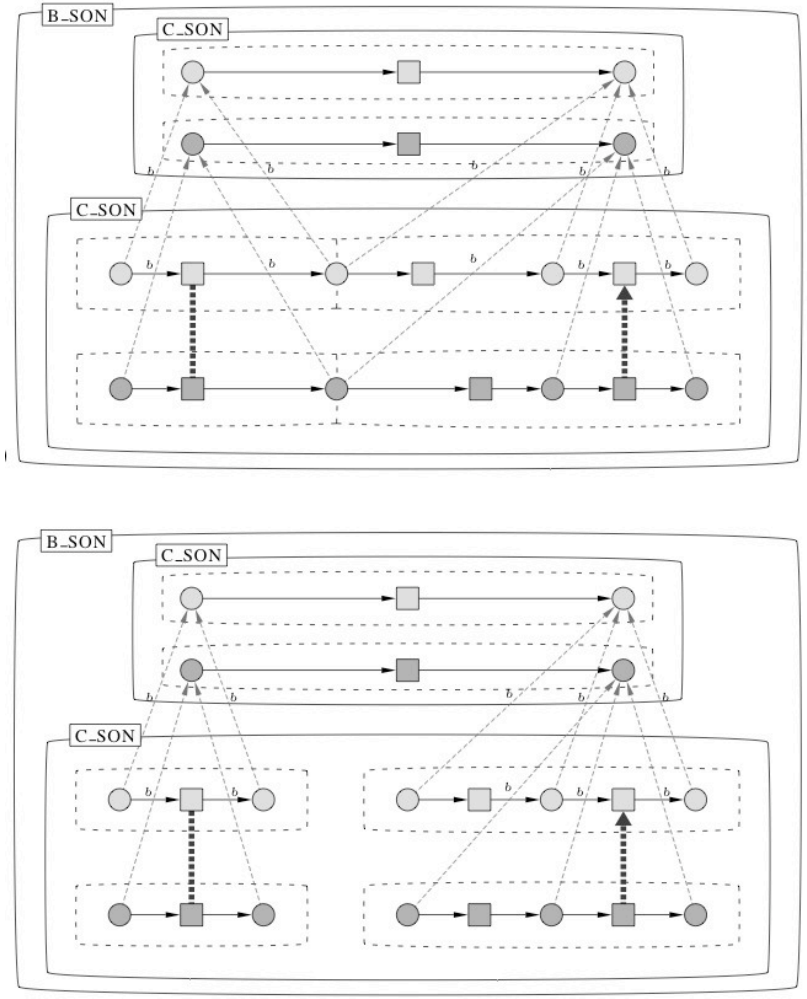
Thick dashed arcs represent interactions between systems (or rather system activities) as opposed to within an activity. Directed such arcs indicate that one event is a causal predecessor of another event (information flow was unidirectional), undirected ones that two events have been executed synchronously (information flow was bidirectional).

Behavioural Abstraction



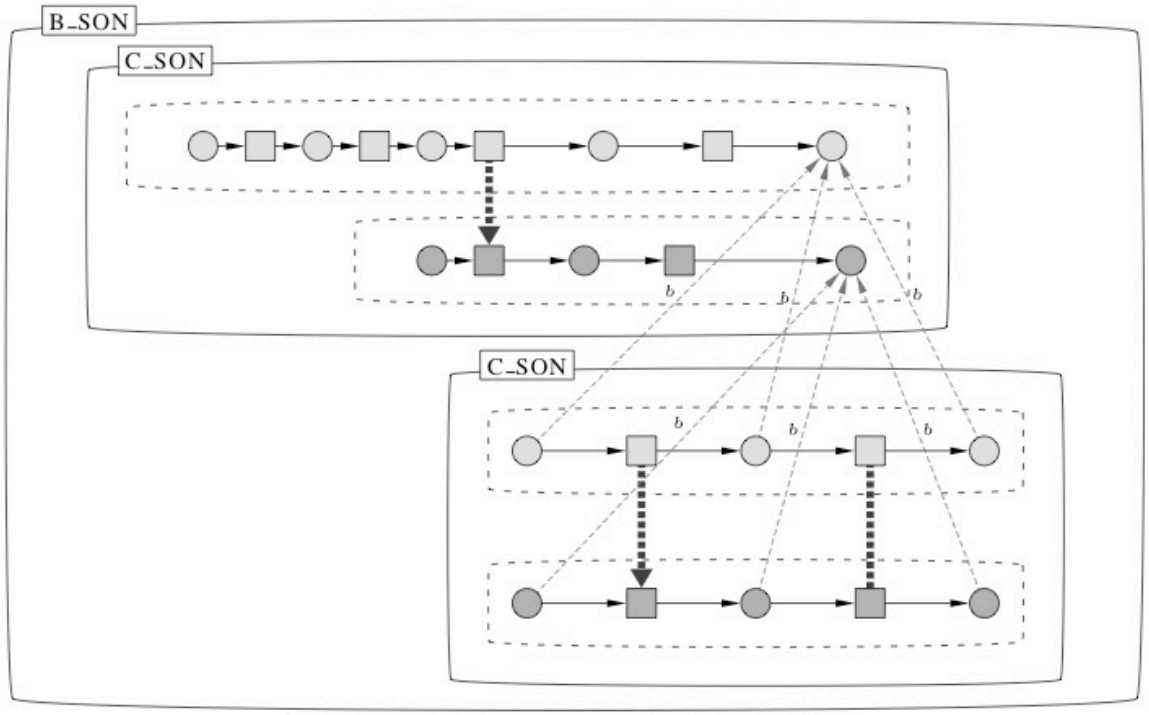
Any condition can be viewed either as a state (of some system), or as a system itself (that presumably has its own states) - just which is a matter of abstraction. Thus one can have two related occurrence graphs, one showing what has happened in terms of a system and its evolution, the other showing the behaviour of the various versions of this system.

System Modification



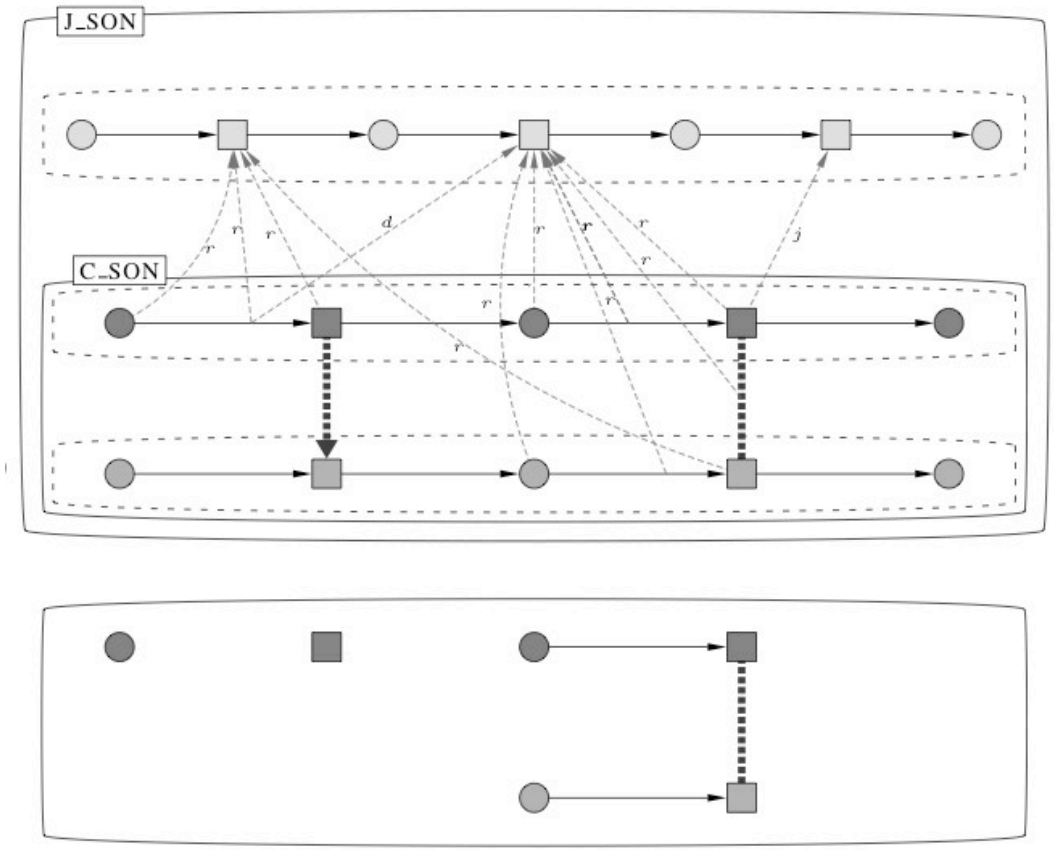
This shows (above) the history of an online modification of two systems. In one case the modified systems carry on from the states that had been reached by the original systems, the other a modification that is followed by re-initialisation.

And system A begat system B . . .



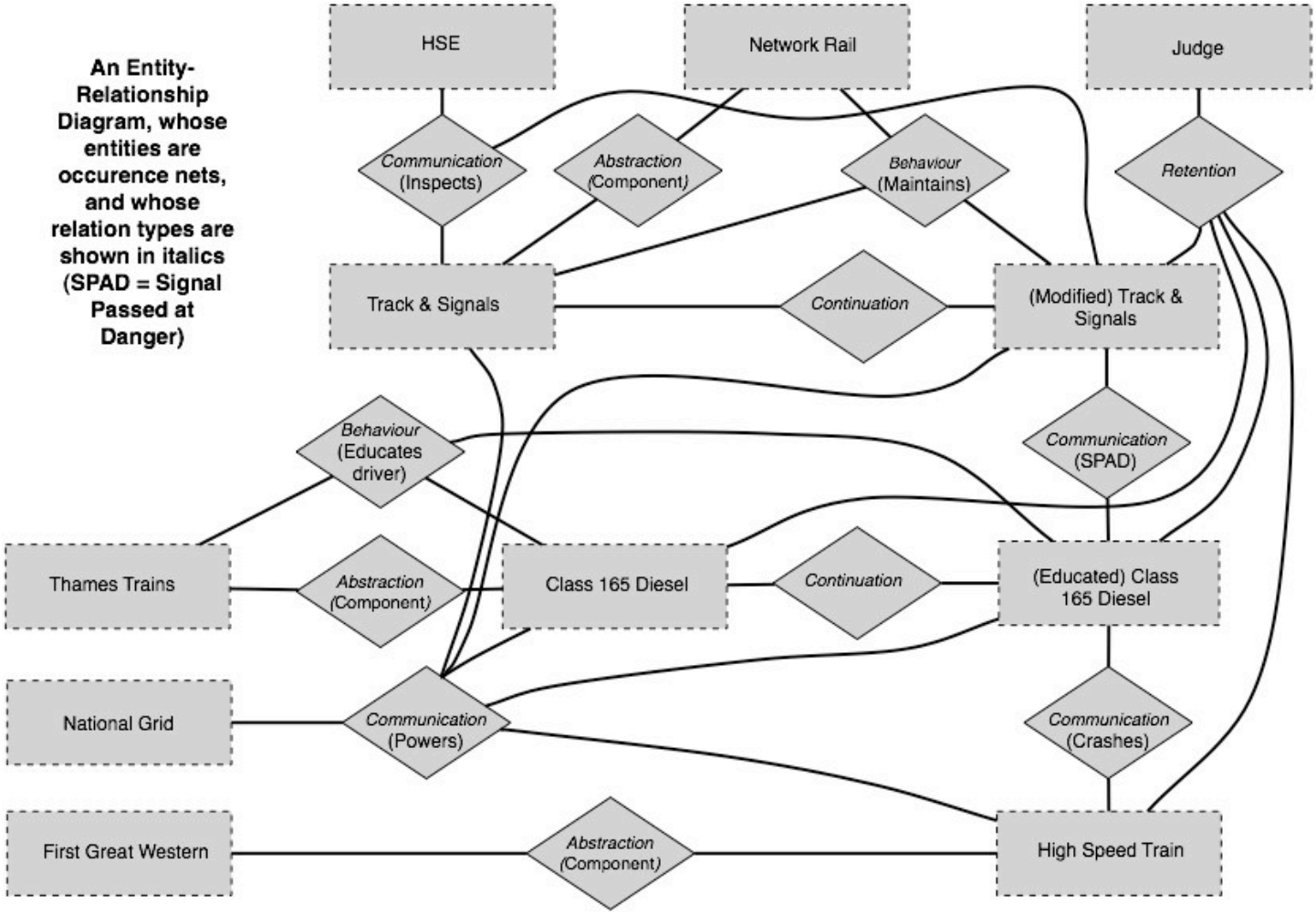
This shows that one system has spawned another system, and after that both systems went through some independent further evolutions - and indicates how the latest versions of these systems have interacted.

Post-hoc Judgement



This portrays a situation in which a judgemental system has obtained only incomplete evidence of the systems' states and events, and even of the causal relationships between conditions and events. (Here the relations employed are "retains" and "discards".)

Modelling the Ladbroke Grove Crash



And the Formal Modelling?

(An example of what you've been spared!)

Definition 9 (temporal abstraction SON). *A* temporal abstraction structured occurrence net *is* $\mathcal{TASON} = (\mathcal{ION}, \mathcal{ION}', \xi)$ *where* $\mathcal{ION}$ *is as in Def. 3,* $\mathcal{ION}' = (\mathcal{ON}'_1, \ldots, \mathcal{ON}'_k, \kappa', \sigma')$ *is an interaction occurrence net with* $\mathcal{ON}'_i = (C'_i, E'_i, F'_i)$ *(for* $i \leq k$*), and* $\xi : \mathbf{C}' \cup \mathbf{E}' \rightarrow \mathbf{C} \cup \mathbf{E}$*; 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* $e \in E_i$*; and* $|\xi^{-1}(c)| = 1$*, 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*
- $\sigma = \{(e, f) \mid (\xi^{-1}(e) \times \xi^{-1}(f)) \cap \sigma' \neq \varnothing\} \cup$
  $\{(e, f) \mid (((\xi^{-1}(e) \times \xi^{-1}(f)) \cap \kappa' \neq \varnothing) \wedge ((\xi^{-1}(f) \times \xi^{-1}(e)) \cap \kappa' \neq \varnothing))\}$.

Failures: Their Definition, Modelling and Analysis,
Randell, B. and Koutny, M. ICTAC-2007, Macao, LNCS 4711, pp. 260-274.

Structured ONs - summary

- Structured ONs – based on Occurrence Nets, a well-established formal notation with good tool support – 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 Structured ON typically involves following (possibly in both directions) causal arrows *within* ONs, and relations *between* ONs – and could be largely automated, through extensions of existing tools.
- The envisaged forms of structuring have various potential benefits:
  - They allow fairly direct representation of what happens in various complex situations, such as dynamic system evolution, infrastructure failures, etc.
  - They provide “divide-and-conquer”-style complexity reduction, compared to the use of ordinary (one-level) occurrence nets.
  - And so could prove very useful regarding tool performance issues - whether for use in system model ckecking, system synthesis, or failure investigations.
- The theoretical foundations are in place, but much remains to be done to establish the utility of Structured ONs.

Avoiding Unnecessary Complexity

- Structure, abstraction, formal models, tool support, etc., are all approaches to *coping* with complexity, and hence to improving system dependability.
- Their role in technical systems is well-established - but less so in *socio-technical systems.*
- In such systems, one of the most important issues is that of determining whether the technical sub-system is unnecessarily ambitious and as a result *overly complex*, so putting the dependability of the overall system at risk.
- The example I'll use to illustrate this point in fact comes from the world of healthcare, rather than that of transportation - a world that was much investigated by DIRC.
- (DIRC is a recently-ended large six-year five-university "interdisciplinary research collaboration" led by Newcastle on *The Dependability of Computer-Based Systems* - http://www.dirc.org.uk/.)

The National Programme for IT (NPfIT)

- NPfIT is a huge 'system-of-systems' being developed for the UK National Health Service's "Connecting for Health" organisation, at an estimated total overall cost of £20B!
  - It is believably claimed to be the world's biggest civil IT Project, intended to serve "40,000 GPs, 80,000 other doctors, 350,000 nurses, 300+ hospitals, 50m+ patients, and 1.344m healthcare workers."
  - NPfIT is largely the responsibility of a set of so-called Local Service Providers - CSC, BT, Fujitsu and (until Sep 2006) Accenture, each dealing with a population equivalent to that of a medium-sized EU member state.
- One central aim has been to enable controlled online access throughout England to what is conceptually a central database containing summary electronic health records for the entire (>50m) English population.
- This aspect of the Programme is at least two years behind schedule, and is highly controversial, e.g. regarding patient confidentiality and privacy.
- I am one of 23 CS professors who have been trying, for over two years, to persuade the Government to commission an independent assessment of the Programme's basic technical viability - see http://nhs-it.info/

NPfIT Security and Privacy Issues

- NPfIT's planned approach to safeguarding the confidentiality of 50M patient records involves:
  - system user authentication using smart-cards and pass-codes, issued to all several hundred thousand medical personnel.
  - recording of "legitimate relationships" (e.g. between doctors and patients).
  - role-based access control using such relationship information.
  - (cryptographically) sealed envelopes for particularly sensitive patient data.
  - pseudonymization of data that is collected for secondary uses, such as research and statistical analyses.
- The practicality of each of these mechanisms, as proposed for NPfIT, has been questioned - e.g.
  - There have been reports of smart card sharing, e.g. in hospital accident and emergency wards, due largely to usability and performance problems.
  - Managing RBAC in such a huge system will be 'challenging'.
  - The sealed envelope scheme has yet to be fully specified, leave alone implemented.
  - Pseudonymization is rarely if ever effective against determined attacks.

Newcastle University

Avoiding a Dilemma

- The dilemma is that is possible (with difficulty) to achieve any *two* of (a) high security, (b) sophisticated functionality, and (c) great scale – but achieving all *three* is (and may remain) beyond the state of the art.
- But 95% of all NHS data usage is local, and patients willingly trust their GPs and local hospital doctors to protect the confidentiality of their records.
- So a claimed way out is to provide only (truly) local data repositories, for individual hospitals and groups of doctors:
  - without complex data compartmentalisation within such repositories
  - with controlled/audited means by which distant clinicians can request information from individual data owners (patients or their doctors)
  - and to associate this with full devolution of IT system specification, acquisition, and management to individual hospitals and sets of GPs.
- i.e., a set of much simpler technical systems could suffice, and form the basis of a much more dependable (and acceptable) overall socio-technical system.

A Computer Scientist's Reactions to NPfIT, Randell, B. *Journal of Information Technology* , 22, 3, pp 222-234 Palgrave Macmillan, 2007.

From Connecting for Health

- Recommendations:
  - It is desirable to leave to the local systems those things best handled locally, while specifying at a national level those things required as universal in order to allow for exchange among subordinate networks.
  - Avoid 'Rip and Replace': Any proposed model for health information exchange must take into account the current structure of the healthcare system...
  - Separate Applications from the Network: ... The network should be designed to support any and all useful types of applications...
  - Decentralization: Data stay where they are... [this] leaves judgments about who should and should not see patient data in the hands of the patient and the physicians and institutions that are directly involved with his or her care.

From Connecting for Health

- Recommendations:
  - It is desirable to leave to the local systems those things best handled locally, while specifying at a national level those things required as universal in order to allow for exchange among subordinate networks.
  - Avoid 'Rip and Replace': Any proposed model for health information exchange must take into account the current structure of the healthcare system...
  - Separate Applications from the Network: ... The network should be designed to support any and all useful types of applications...
  - Decentralization: Data stay where they are... [this] leaves judgments about who should and should not see patient data in the hands of the patient and the physicians and institutions that are directly involved with his or her care.
- *Unfortunately, these recommendations are not from the British, but from the American, Connecting for Health organization!*

Concluding Remarks

- Complexity (and excessive simplicity) lead to undependability.
- This applies to computers (hardware and software), and especially to *computer-based systems* (socio-technical systems involving people as well as computers).
- People are potent sources of faults, especially in the presence of complex computer systems, though they also can provide effective means of coping with the faults of such systems.
- But it is much better to avoid complexity than to have to cope with it!
  - The most successful highly-critical large IT systems, such as the worldwide VISA payments system (studied by DIRC), achieved their success through ruthless control of their complexity, as well as through high levels of hardware/software reliability, and attention to socio-technical issues.
- Echoing the Tony Hoare lament with which I started, how does one persuade government organizations that are already enmeshed in huge system procurements to curb their naively complex ambitions?