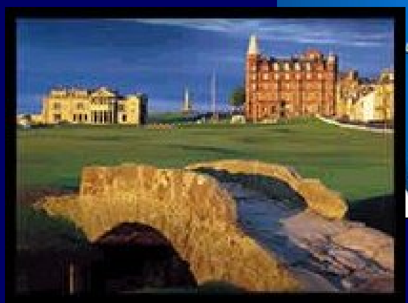
Design for failure: Software challenges of digital ecosystems

Prof. Ian Sommerville
School of Computer Science
St Andrews University
Scotland

St Andrews



- Small Scottish town, on the north-east coast of the UK
- Home of golf
- Scotland’s oldest university (founded in 1413)
- Small university focusing on research and teaching excellence

Trust and dependability

- Trust is fundamental to business dealings
- Trust
  - Reputation and recommendation
    - Companies establish trust through reputation and recommendation
  - Regulation
    - Organisations are trusted because they are externally regulated
  - Dependability
    - Positive experiences lead to trust. If users of a system find that it meets there needs, is available when required and doesn't go wrong then they trust the system.

What is dependability?

- System dependability is a critical factor in delivering a high quality of service
  - Availability. Is the system up and running?
  - Reliability. Does the system produce correct results?
  - Integrity. Does the system protect itself and its data from damage?
  - Confidentiality. Does the system ensure that information is only accessed by agents authorised agents?
  - Timeliness. Are the system responses produced within the required time frame?

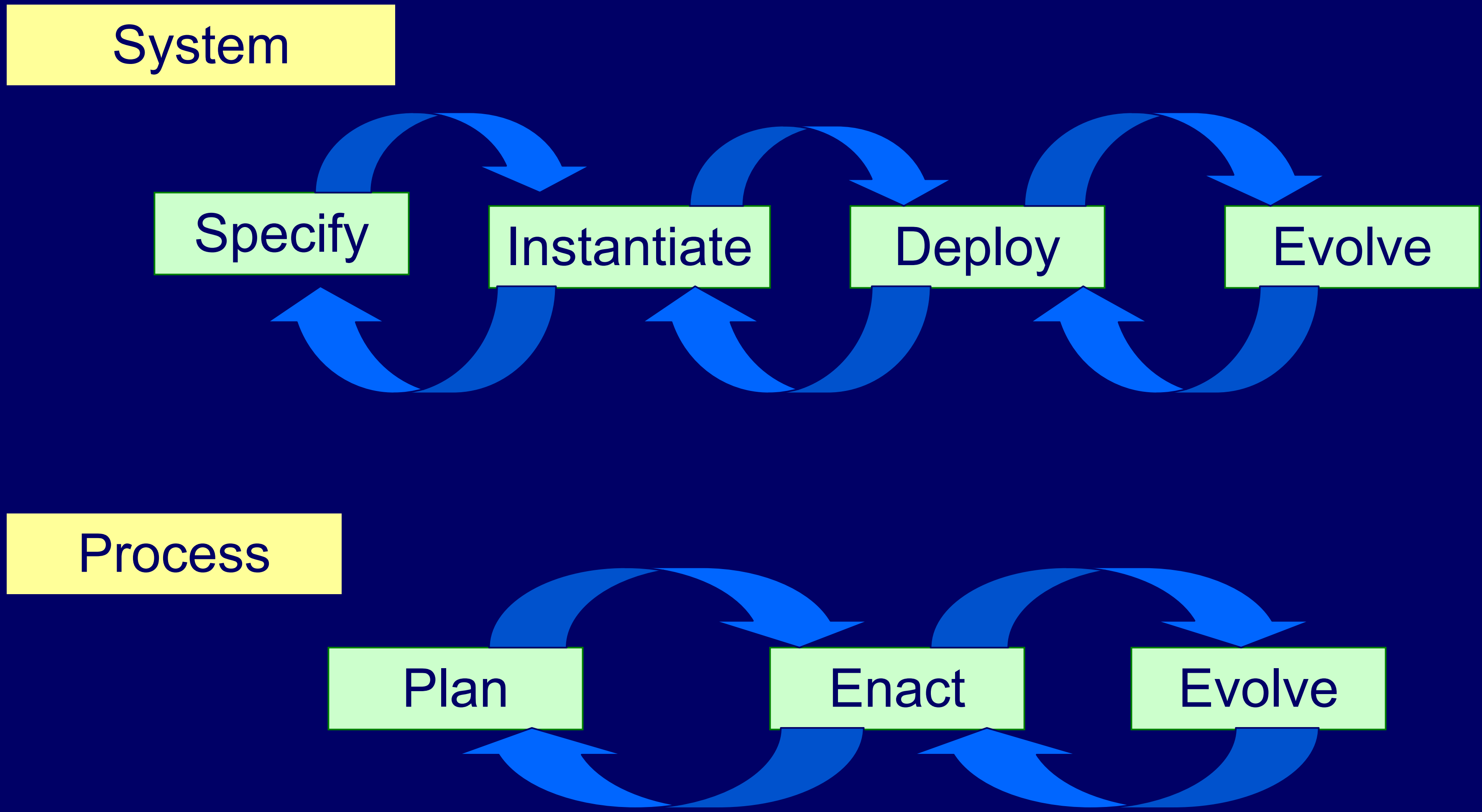
Why dependability?

- Dependability is a major factor in establishing reputation and brand.
- In e-business systems, undependability leads to loss of confidence, business and revenue.
- Dependability is necessary for a service to be trusted by its users.

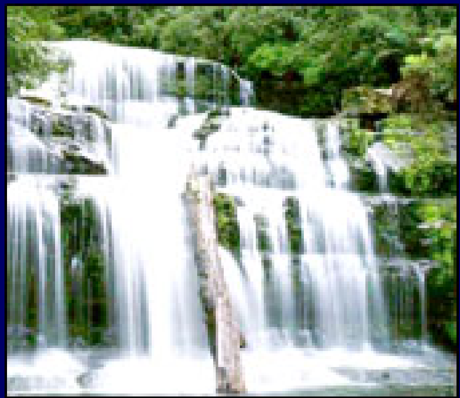
Achieving system dependability

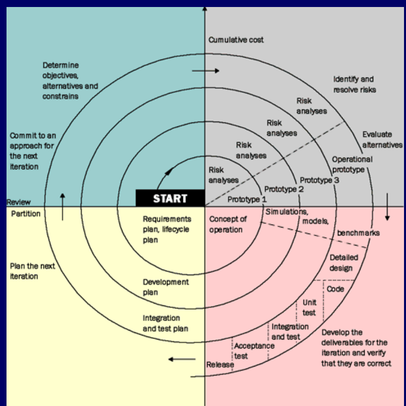
- Fault avoidance
  - Detailed analysis of specification
  - Extensive reviews and testing of system
  - Careful configuration control
- Fault tolerance
  - Redundancy
    - Additional capacity that can be used in the event of failure
  - Diversity
    - Different ways of doing things

Business system engineering



Top-down software engineering





- System vision
- Single specification
- Control of changes
- Complicated but not complex
- Client-contractor-sub-contractor relationships
- 'Clear' assignment of responsibilities
- Scope for whole-system analysis
- Trusted parties in collaboration

Ownership and control

- In top-down software engineering, a single organisation owns all parts of the system:
- Specification
  - Architecture and services offered can be controlled
- Instantiation
  - Engineering process can be controlled
- Deployment
  - Use can be controlled
- Evolution
  - Changes can be controlled

Ownership and dependability

- There is a close relationship between ownership (control) and dependability
- The more that is under the control of a single owner, the easier it is to produce dependable systems
  - Dependability through process
    - Fault avoidance
  - Dependability by design
    - Fault tolerance

Digital business ecosystems

- “A distributed environment that can support the spontaneous evolution and composition of software services, components, and applications”.
- DBEs are socio-technical entities that are not just populated by digital species
  - They include organisations, people, processes, regulations, etc.
  - Social, economic and political considerations are as important as technical issues.

Software engineering in a DBE

- System of systems.
- System instantiation involves cooperation and communication between entities in the ecosystem.
- Dynamic system re-configuration
  - The entities in the ecosystem evolve and become more/less suitable for some applications.
- Ecosystem evolution
  - The ecosystem itself exhibits a degree of self-organising behaviour. Applications may have to adapt to changes in the underlying environment.

Application ownership in a DBE

- Specification
  - Constrained by capabilities and entities of DBE
- Instantiation
  - Many owners of different parts of the system
  - The self-organising nature of the DBE means that the system owner has only partial control.
- Deployment
  - May be influenced by self-organising nature of DBE
- Evolution
  - Uncontrollable!

System failure

- Failure is inevitable.
- Failure is generally due to some conjunction of environmental effects which system designers have not considered.
- There are a huge number of possibilities and, eventually, if a system can fail, it will.
- Time to market pressures for new systems increase the chances of system failure.

DBE technology stack

E-business applications

Business ‘services’

Domain/business knowledge

Shared business data

Implementation infrastructure (SOA, P2P…)

RAD support

- Construction
- Communication
- Organisation
- Dependability

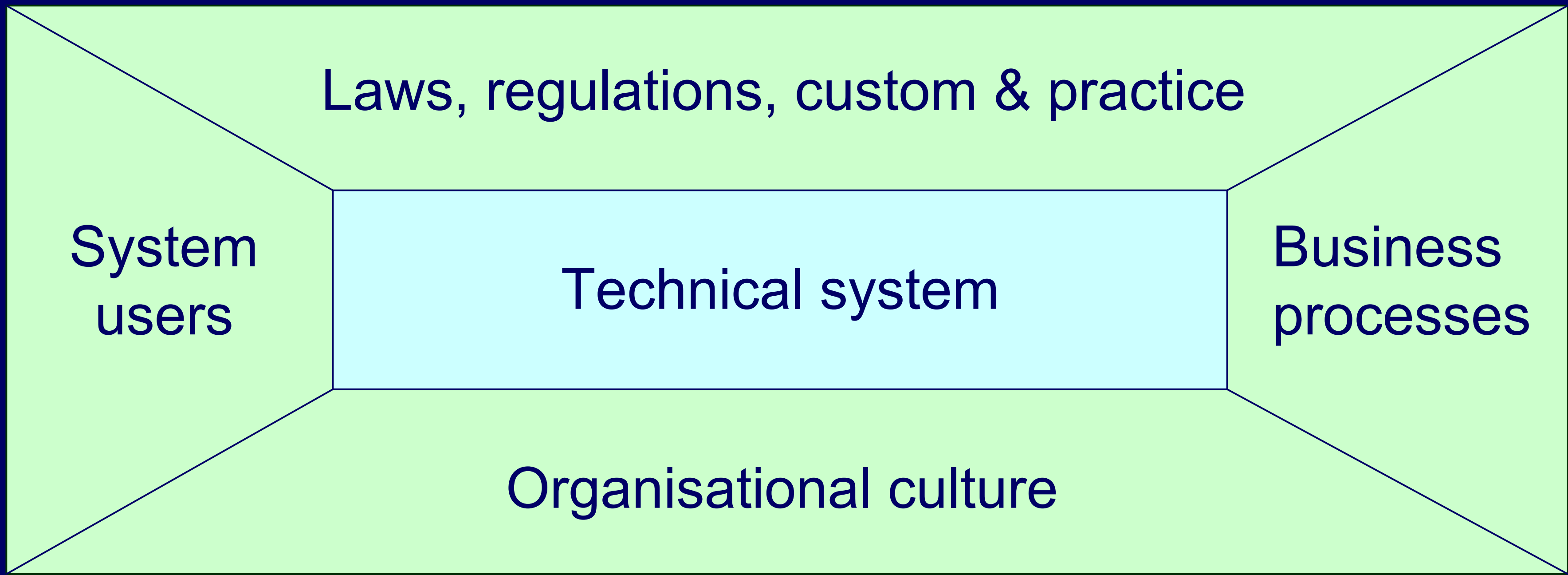
Technical failures in DBEs

- Infrastructure failure
  - Technology infrastructure is unavailable/corrupt
- Data failure
  - Required data is incorrect or unavailable
- Knowledge failure
  - Required knowledge does not exist, is unavailable, is incomplete or is incorrect
- Service failure
  - DE components are faulty/unavailable
- RAD support failure
  - RAD run-time system is faulty
  - Application composition mechanism is faulty
  - Application composition is faulty

Security failures in DBEs

- Malicious component
  - Deliberate interference with the functioning of the application system
- Malicious data and knowledge
  - Deliberate introduction of incorrect data/knowlege
- Insecure infrastructure
  - DBE infrastructure is compromised by malicious components
- Insecure component
  - Digital 'species' is compromised by malicious code

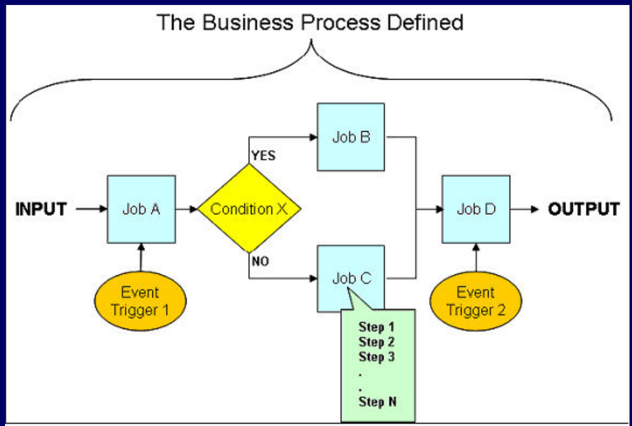
Socio-technical systems

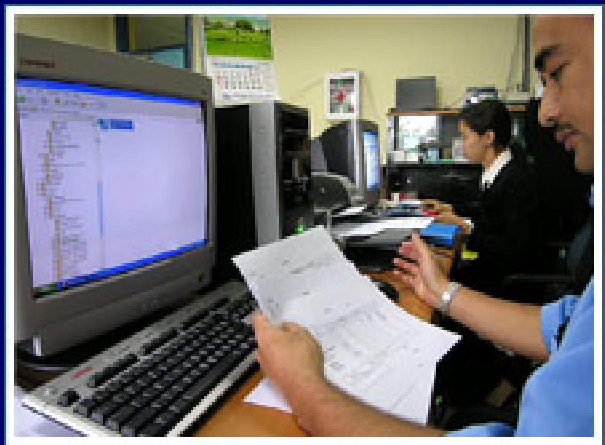


Coping with failure

- Socio-technical systems are remarkably robust because people are good at coping with unexpected situations when things go wrong.
  - We have the unique ability to apply previous experience from different areas to unseen problems.
  - Processes are designed to recognise and deal with exceptions.
  - We often have channel redundancy ie email, phone, walk up and talk.
  - Information is held in diverse forms (paper, electronic). Failure of software does not mean that information is unavailable.
- Coping with failure often involves ‘breaking the rules’.

Consequences of automation





- Increasing automation reduces minor human error but makes it more difficult to cope with serious failures
- Rules enforced by system
  - Lead to dependability by catching failures and errors.
  - But it makes it harder to break the rules.
- Information redundancy is minimised
  - There is a single copy of information, maintained by the system and inaccessible in the event of failure.

What's different about DBEs

- Many rules enforced in different ways by different systems.
- No single manager or owner of the system
  - Who do you call when failures occur?
- Information is distributed - users may not be aware of where information is located, who owns information, etc..
- Probable blame culture
  - Owners of components will blame other components for system failure. Learning is inhibited and trust compromised.

Dependability challenges

- Trust and confidence
- Reasoning about DBEs
- Fault tolerance and recovery
- Self-organisation
- Socio-technical reconfiguration

Trust in technology

- Provenance
  - Who are the suppliers of the technology? What business environment do they operate in?
- Transparency
  - What information is available about the operation, structure and implementation of the technology?
- Predictability
  - Does the technology behave in the way we expect each time that we use it? Is it dependable?

Trusting systems of systems

- What mechanisms do we need to convince ourselves that DBEs and application systems in these DBEs are trustworthy and dependable
  - New approaches to constructing dependability arguments because existing approaches are designed for top-down software engineering
  - Methods and tools for testing DBE infrastructures and configurations
  - Self-aware systems that make information about their operation and failure available for scrutiny and use
  - Regulatory and social mechanisms to ensure that undependable and untrustworthy elements of the system are excluded from the DBE

Reasoning about DBEs

- We need to be able to reason about DBE configurations to convince ourselves that they are 'good enough'
  - What abstractions should be used to represent DBEs?
  - How do we express assumptions about DBE instances and how do we monitor the DBE to ensure that these assumptions remain valid?
  - How do current approaches to risk analysis need to evolve to reason about system risks?

Fault tolerance

- The DBE has the potential to be a fault-tolerant execution environment as it may contain multiple diverse instances of the same service.
  - What mechanisms are required to create fault-tolerant configurations?
  - How are faults automatically detected?
  - How do we recognise redundant and diverse services?
  - How do we handle partial computations and compensating actions?

Self-organising DBEs

- It has been suggested that DBEs will have some degree of self-organisation where the system will organise itself without human intervention.
- How do we know that each possible reorganisation is trustworthy?
- Does the reorganisation optimise service to the community or to an individual?
- How do we ensure that QoS to a community member is not unacceptably degraded?
- How do we know that each possible instance of the DBE conforms to regulations?

Socio-technical reconfiguration

- To cope with failure, DBEs must have the capacity to dynamically reconfigure themselves to replace automated with non-automated components.
  - How do we describe failures that might be solved by socio-technical reconfiguration? How do we recognise the symptoms of these failures?
  - How do we find a person with the appropriate knowledge to address the problem?
  - How do we ensure that they are provided with the necessary information and access to resources to solve the problem?

Conclusions

- DBEs offer an opportunity to radically change the business environment for SMEs.
- Their adoption is dependent on users trusting the resultant socio-technical systems.
- Failure by researchers and practitioners to design for failure will inevitable lead to the failure of the vision of digital business ecosystems.