Seminar session 3: Secure System Architecture – Flowchart Modelling and Distributed System Challenges

Outline:

- Announcements
- The importance of...
- Distributed System challenges
- Challenge 1: Deadlock and Live-lock
- Dining Philosophers
- Using Modelling for distributed systems
- Seminar 4 Questions
- QA and Next Session

Announcements:

- Forums and Posts
- What did you think of the AD tool?
- Informal feedback
- → Approaches from both teams are good.
- →Limitations
- →Probabilities

Distributed Systems:

- Two properties that characterise a distributed system:
- → Need for Synchronisation
- → Determinism

- Synchronisation
- → Messages
- → Token Passing
- → Broker / Leader /Server

- Determinism:
- → The NP Problem (Nonterministic Poly.??) → Problem is increasing with increasing user
- → Deadlock / Livelock
- → Spinlocks & Semaphores

Distributed System Challenges: Examples

- Kernels on multi-core systems
- → Pre 2.0: Linux Kernel was designed for uniprocessor and not threaded – deterministic → Enables the calculation of the performance of the system.
- → Post 2.0: Big Kernel Lock implemented spinlock mechanism → Only one process can be carried out at the same time.
- → Modelled by the Dining Philospohers Problem
- Networks
- → Original Ethernet used CSMA/CD
- → Wireless Networks use CSMA/CA → Contention: Messages collide on the cable.
- → Contrast with Token Bus Networking – deterministic → Token passing, better use of throughput but felt under misuse.
- NP (Non-deterministic Polynomial) Problems → As the number increases the problem solving process getting more difficult.
- → Same problem with CSMA/CA and Wi-Fi.
- → While recognise what happens it is difficult to figure out where it happens.

The Importance of:

- Non-Determinism:
- → a nondeterministic algorithm represents a method of thinking of computer programs as being in part governed, not by efficient causes (causes which precede their effects) but by final causes (goals; causes for the sake of which their effects are carried out) Floyd (1967).
- --> "Nondeterminism form an important basis for the definition and analysis of the behaviour of parallel processes" (Broy, Gantz, and Wirsing, 1978, p. 554)
- → Multi-core processors
- → Message passing networks

Dining Philosophers Model:

- Dining Philosophers: classic model for deadlock and live-lock illustration
- Featured in many papers (original model by Dijkstra (1965))
- Many modelling systems provide solutions
- → Problems: Deadlock & Live-lock

Mitigation:

- Basic Spin-Lock

How to deal with ND & Synchronisation:

- Non Determinism
- → The problem is scale and complexity – how do we reduce these?
- Synchronisation
- → As discussed multiple methods available – which is best?
- Modelling
- → The answer is to build a simplified, scaled down model of the system
- → Allows ND to be reduced or removed
- → Allows multiple sync methods to be tested and evaluated
- → Mandy different types of model

Formal Methods/ Modelling:

-"Formal methods are techniques used to model complex systems as mathematical entities. By building a mathematically rigorous model of a complex system, it is possible to verify the system's properties in a more through fashion than empirical testing."

- "While Rigorous description promise to improve system reliability, design time and comprehensibility, they do so at the cost of an increased learning curve; the mathematical disciplines used to formally describe computational systems are outside the domain of a traditional engineering education. In addition, the metamodels used by most formal methods are often limited in order to enhance provability. There is a notable tradeoff between the need for rigor and the ability to model all behaviours." (Collin, M. (1998) CMU.

Formal Methods Tools and Approaches:

- Abstraction – choose WHICH behaviours to model
- Methods:
- → Z Notation – used to describe models in a formal language/ notation
- → MBSE – Model Based Systems Engineering – using FM to design engineering systems using AADL, Arcadia and SysML
- → Functional languages such has Haskell, Lisp and declarative languages such as Prolog and Maude
- Model checkers such as NuSMV, SPIN, FDRx
- Applications: CPU desing (VHDL); Aircarft and Transport Systems Design, Simulation

MBSE Example Arcadia:

- Supporting Efficient Collaboration in Engineering
- Validating/Justifying solution against Operational Need, Easing Impact Analysis

Non-determinism and Cyber Security:

- Formal Methods and Modelling are also applicable to cyber security systems and solutions.
- Examples:
- → Threat-Modelling: very much an ND activity (why?) We have already explored AD Trees – what about other approaches?
- Networking: How can we use FM in secure network design and investigation?
- Static and Dynamic code checkers?
- Any other examples?

Task:

➔ Discuss in the team a model using SysML and post it in the module.
➔ How to simplify based on the model.

Assessment Part 2:

- Checklist for this section of the assignment:
- → Produce a prototype in Python that models a smart device (such as a light actuator, heating valve) as well as the home controller/ interface.
- → Demonstrate interactions between the two simulated systems, illustration how your solution addresses some of the common problems of distributed systems (ND issues such as latency, power consumption, lost messages, reliability and security)
- Provide evidence of the above (via demos and / or test outputs)
- Comprehensive testing (of style, code security and functionality/ operation)
- Documentation both as code comments/ markup and a supplementary README file

➔ Maybe use MQTT
➔ Instead of build the system and mitigate the vulnerabilities choose the top 5 threats and build a system that shows how to mitigate

Next Session:

- Unit 6: Submit assessment – Application Code
- Group Review / Check-up
- Next Seminar: distributed system future directions
- → Prepare some slides of the future directions.
- Any Questions?