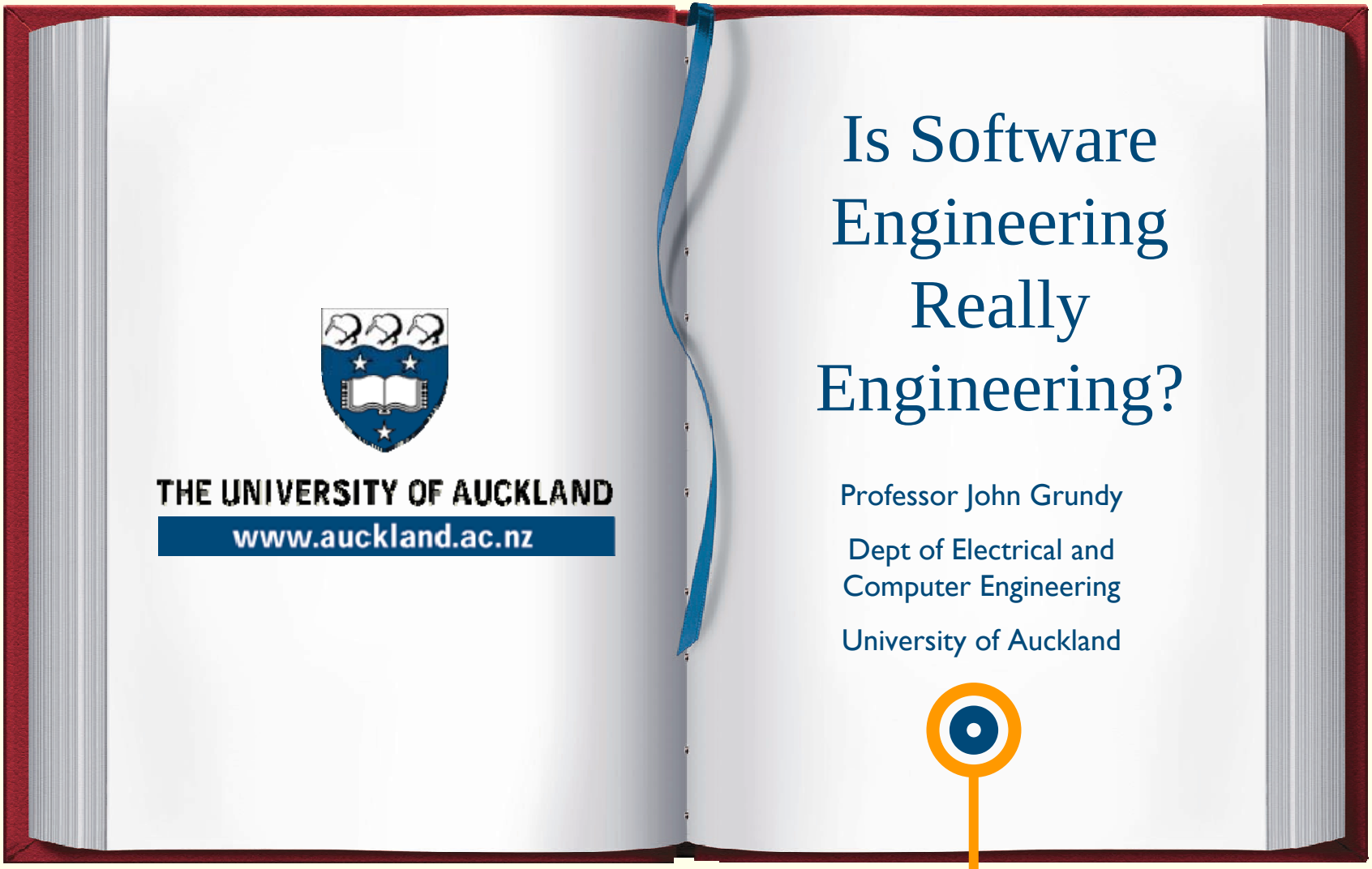


Is Software Engineering Really Engineering?

Professor John Grundy

Dept of Electrical and Computer Engineering

University of Auckland

Outline

- What is Engineering? My view…
- How SE is **like** other Engineering specialisations
- Particular challenges of Software Engineering
  - Nature of the product – "soft" vs "hard"
  - Lack of (some) theoretical foundations
  - Background of practitioners, academics
  - Maturity of processes, methods, tools
- Place of Engineering/SE in tomorrow's society?

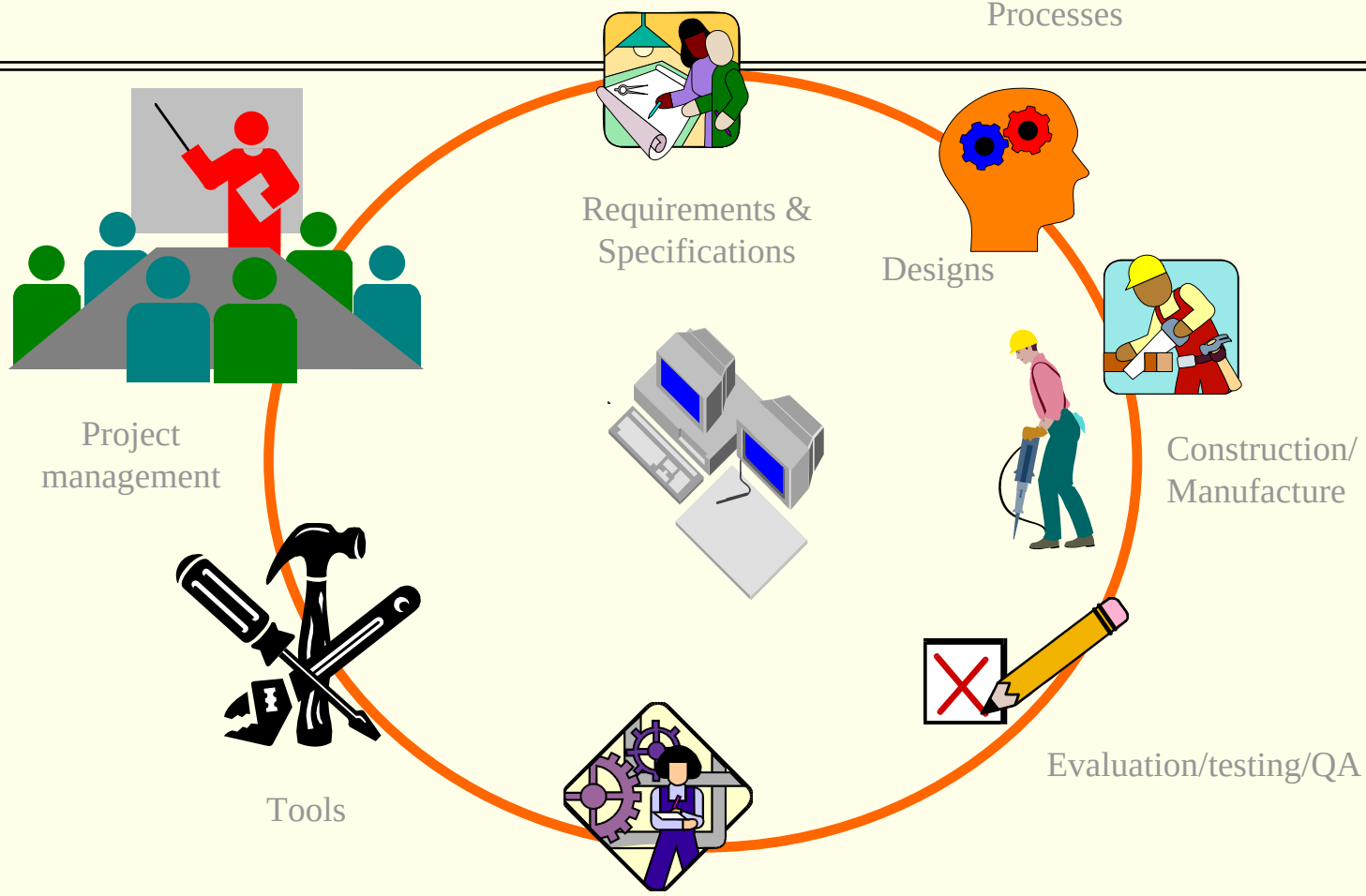
What is Engineering?

- Go to www.google.com and enter "What is Engineering?". Some results:
  - The practical application of science to commerce or industry
  - The process or organization responsible for the skillful design, construction, maintenance and enhancement of complex or sophisticated systems of hardware, software, processes, etc.
  - Engineering is the principled application of science, methods, tools, and experience to the production of designed objects.
  - There are a number of types of engineering (chemical, civil, mechanical, electrical) which apply to different areas of design and construction. All engineering work is regulated by safety standards, and issues of patents and design protection may also arise.
  - The art and science by which the mechanical properties of matter are made useful to man in structures and machines.
  - The discipline dealing with the art or science of applying scientific knowledge to practical problems; "he had trouble deciding which branch of engineering to study"
  - a room (as on a ship) in which the engine is located

What is Software Engineering?

- Search on the above:
  - The system of applying of an engineering discipline to the design, implementation and maintenance of software systems.
  - Software Engineering is an engineering discipline which is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.
  - A collection of theories, techniques, and tools which enable fallible humans to design, construct and maintain large software products in a reliable and cost effective manner.
  - A disciplined and standardized approach to program development, both in its managerial and technical aspects
- My personal definition (that I tell students):
  - Disciplined application of formal, semi-formal and informal techniques to make large-scale software development more effective and efficient

An Outline of Software Engineering



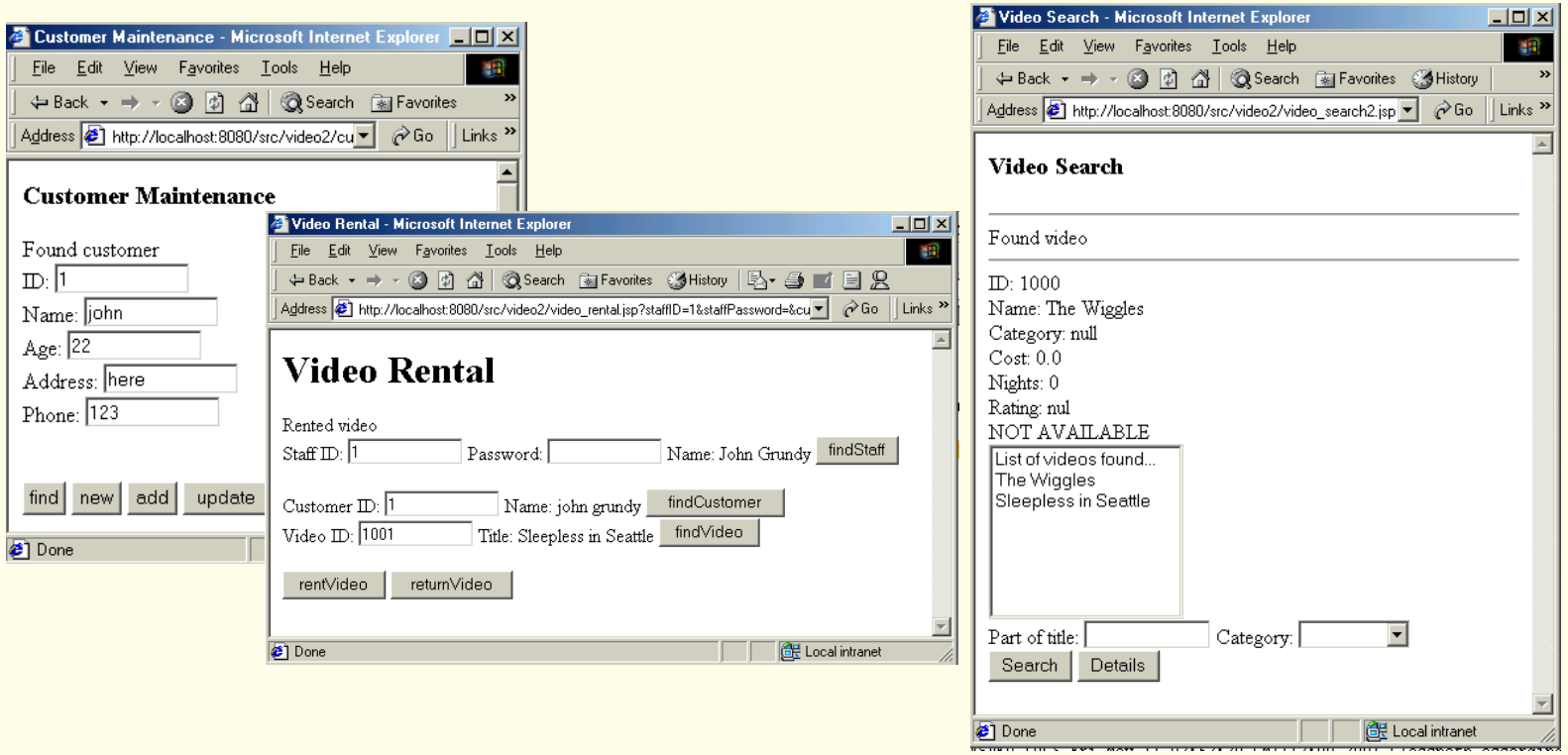
Software Engineering vs Bridge Engineering

- Requirements/Specification
  - What do users want? How complex are the user requirements vs the design? How do we capture this? Can we change it?
- Architecture/Design
  - Models we need to build? How verify the models? What physical vs other laws help us do this? Constraints on designers? Can it be changed? Where does design end/construction begin?
- Construction/Manufacture
  - "Building" the product. What is the product?? Can it be changed?
- Quality Assurance
  - How? When? By who? Compliance?
- Methods used? Tools used? Processes used?

Challenge #1: Nature of the Product

- Software is essentially bunch of bits in computer memory or on Tape/Hard Disk/CD/DVD/…
- It is inherently changeable (but this doesn't mean it *should* be changed, just because we can do so!)
- "Manufacture" phase = Write code? Or copy the bits (from anywhere!)?
- Is "programming" design or construction?
- Complexity is incredible – to what level do software engineers need to know about?
- Quality assurance – how do we do it?

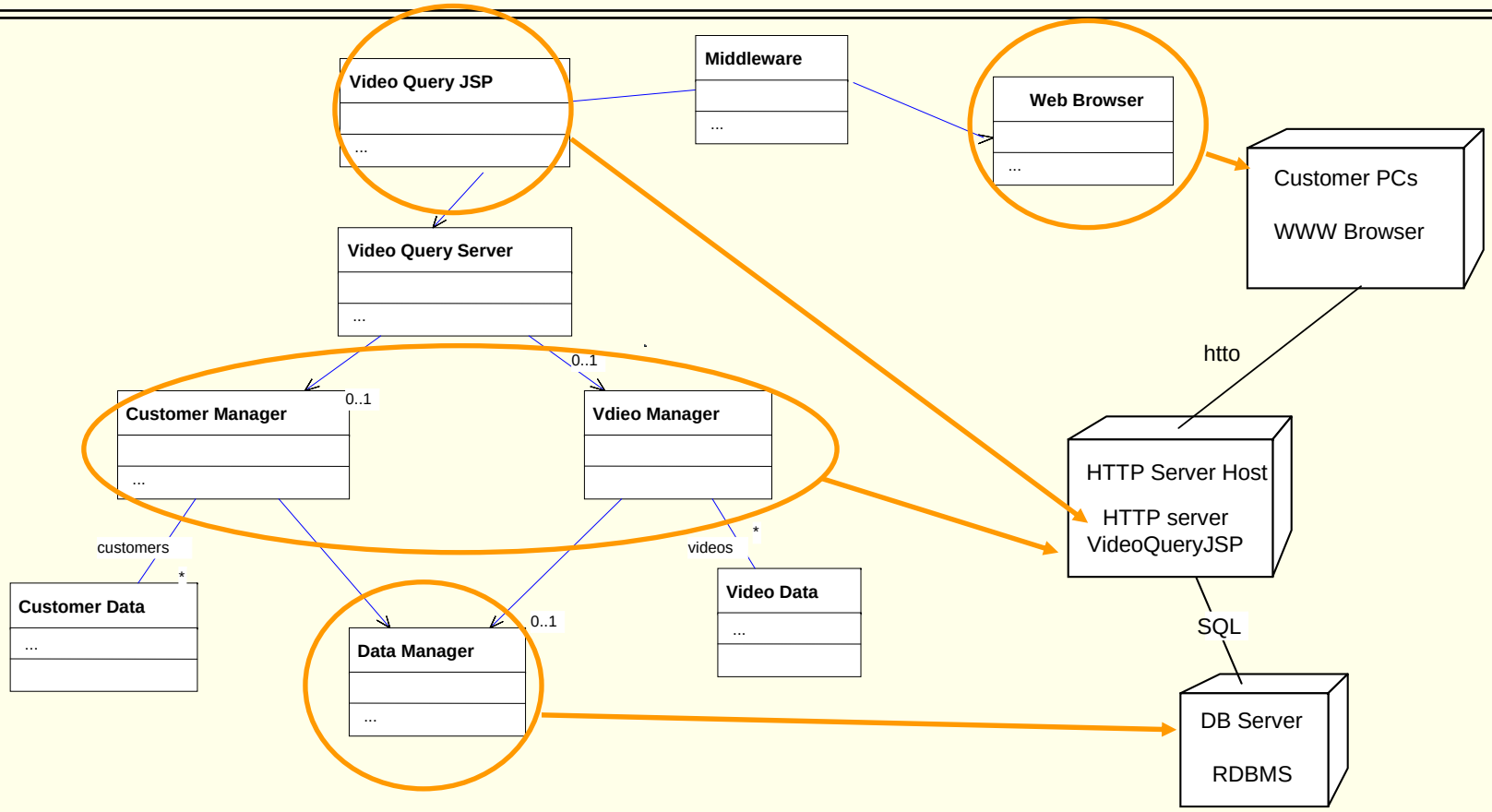
Example: A Video Store On-line Library



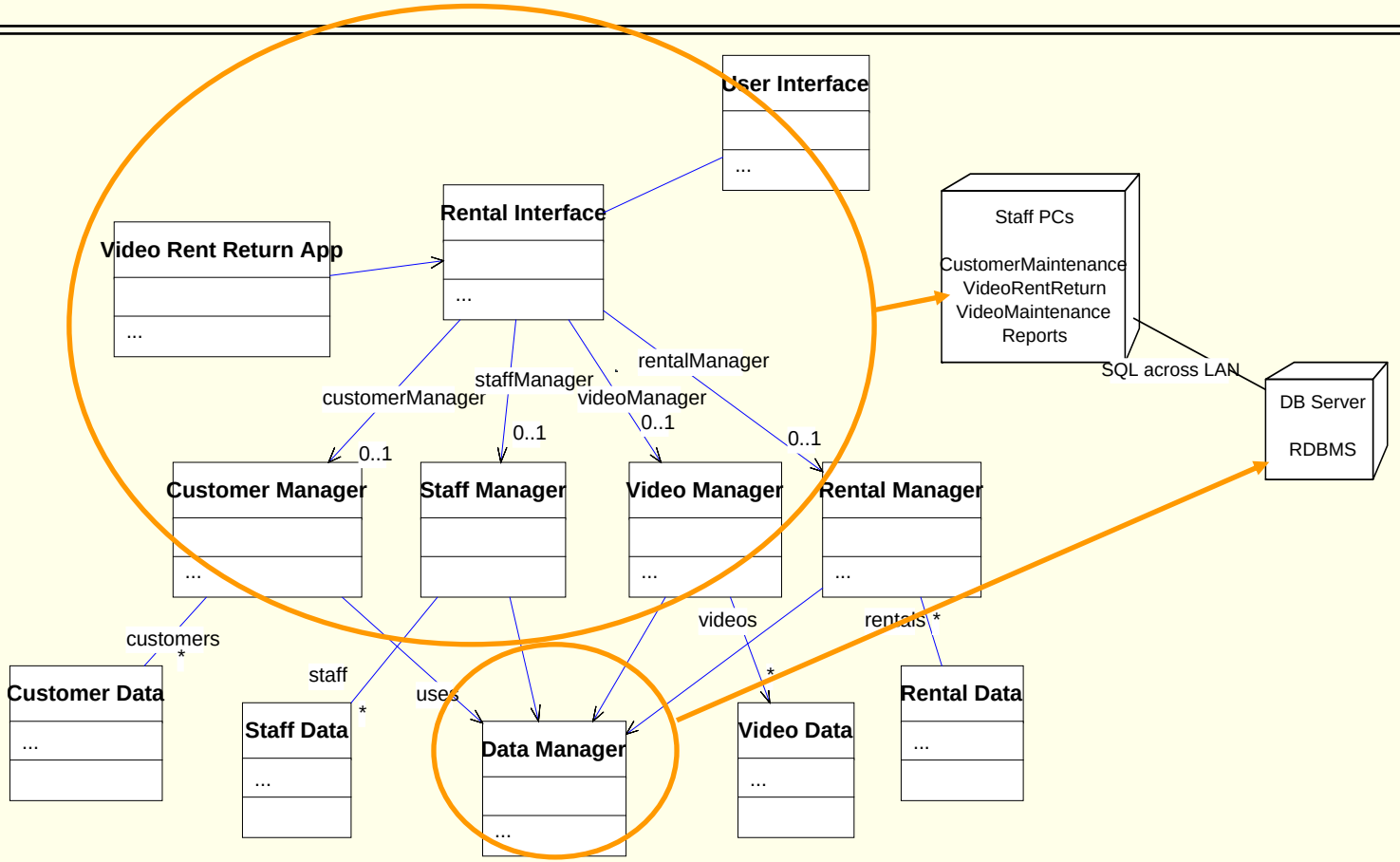
Challenge #2: Lack of theories

- Much of Software Engineering foundations come from:
  - Computer Science: computers, data structures, networks, some aspects of design, Human-Computer Interaction, theoretical CS
  - Computer Systems Engineering: structure of computers
  - Information Systems: requirements, design, applications, processes
- Unlike most other Engineering specialisations, no physical laws constrain us (or do they?)!
- Nature of the end product is thus rather different…
- Design, quality assurance are areas of particular weakness in the theoretical foundations of the discipline
- Movements in these areas: model-driven architectures; test-first development; automated software engineering. Are these Engineering or not??

Video Store Example – Part of a Design



An Alternative Design…



Challenge #3: Background of the People

- Most practitioners at present:
  - Do not have Engineering degree/background
  - From Computer Science/Information Systems background
  - Do not belong to professional societies; do not need accreditation from professional body to practice
  - Are not acting as "Engineers", even if they have the job title
- Much the same could be said of most academics teaching into Software Engineering programmes
- Turf wars
- Professional responsibility – see any software license agreement…

Video Store - Development Team

- Store manager has MBA
- Project manager has BE(Engineering Science)
- Team leader has BCom in accounting
- Architects have BSc(Comp Sci)
- Developers have BSc(CS), BCom (MSIS)
- Documentation expert has BA (Linguistics)
- Development process – "evolutionary prototyping"
- Clients - who are they?

Challenge #4: Maturity of the Discipline

- Very new discipline – computability theory from the 1930's vs civil engineering over thousands of years!!
- Technology changes incredibly quickly:
  - Software processes e.g. "Agile Processes"
  - Software methodologies e.g. "Component-based SE"
  - Software tools e.g. MS Visual Studio, IBM Rational Rose
- Very rapid time-to-market drives software companies
- Huge user base in society
- Complexity growth of software systems incredible
- Need to "network", distribute software

Video Store - Technology

- Move platform from Windows 2000 to LINUX
- Change target technology from Windows/ASP to J2EE/JSPs
- Change development method from evolutionary prototyping to eXtreme Programming
- Must integrate with Time/Warner video database, EFTPOS network, MYOB Accounting Suite
- ALL during development of the software!

Current Practice of Software Engineering

- Wide mix of processes, methods, tools used
- Many problems with software faults after production and release to consumers
  - London Ambulance Service
  - Ariane 5 rocket
  - NZ Police, Health Waikato
- BUT – many successes too:
  - Microsoft product suite
  - LINUX operating system, Apache web browser
  - Games – see Harry Potter, Dungeon Seige, Xbox/PlayStation…
- Often rapid organisational/context changes

Video Store – Organisational Change

- Store is bought by major franchiser as can't get software into use in time to increase market share
- Franchiser has own new software development
- Old store's software development adandoned
- Most of development team made redundant

- Get hired by bank developing next generation Internet Banking system…

The Future: for Software Engineering

- Component-based Software Engineering – the "build it from pieces" model
- Agent-based Software Engineering – extension of CBSE – autonomous agents/emergent behaviours
- Automated Software Engineering; Model-Driven Architecture – generate systems from high-level models
- System integration – legacy and new systems – very high distribution/peer to peer systems/collaborative work
- Ubiquitous computing systems – software in everything…
- Professional responsibility – drive to make "Software Engineers" accountable for their product development

The Future: for Engineering

- Engineering in the 21st Century – what are the key issues?
- More and more "entrepreneurial" , wealth-creating engineering vs "infrastructural", wealth-consuming engineering – how educate/train appropriately?
- More and more demand for Accountability, Ethical practice, Professionalism – how achieve?
- Life-long learning for all Engineers – how do we support as a profession? As a society?
- Holistic view of Engineering – personal example of myself and my father's career…

Conclusions

- Software Engineering is Engineering, just not as we know it
- Product vastly different to "traditional" Engineering products
- Still maturing – theories, methods and personnel
- Area of continuing rapid technological changes – have to plan for these in all software projects
- Doesn't stand alone – Systems Engineering