

$M^{2.0}$

Finance

(Research Challenges in...)

Christian Nentwich

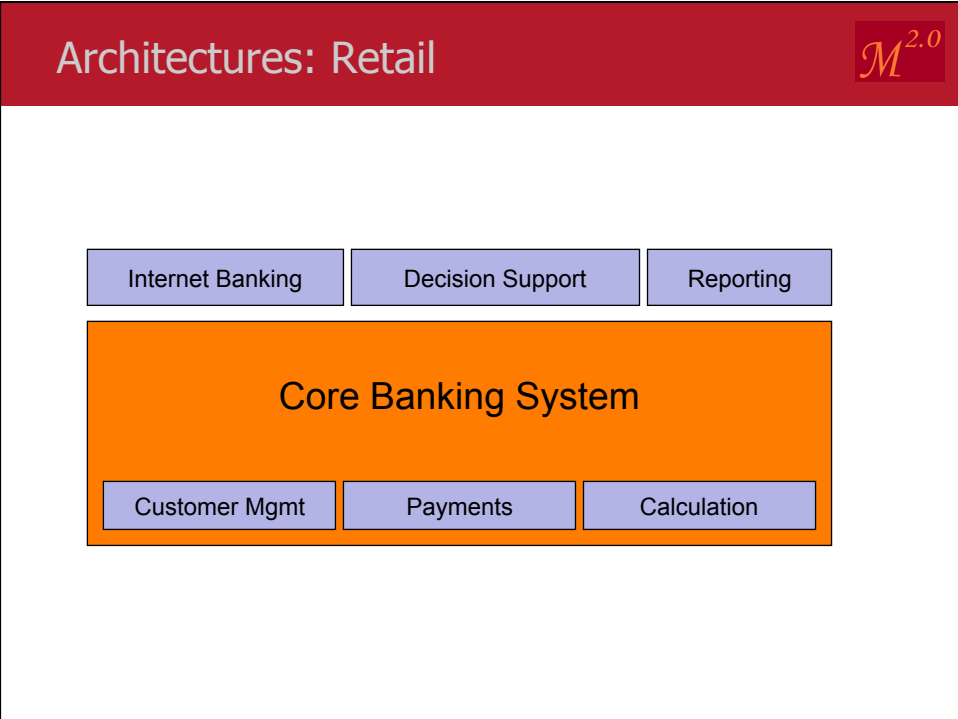
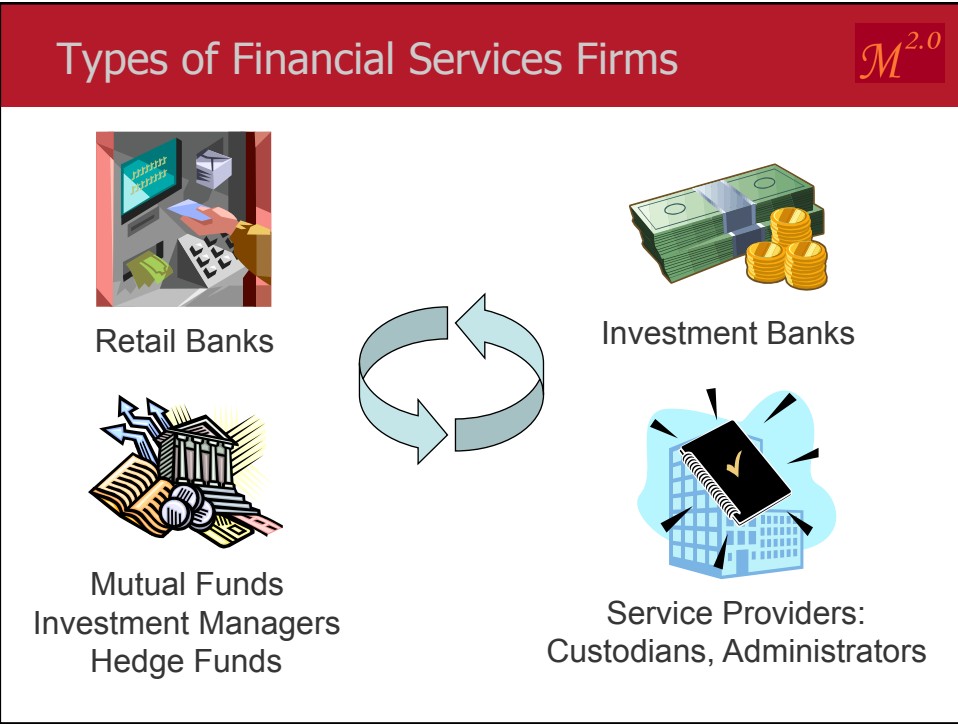
Model Two Zero

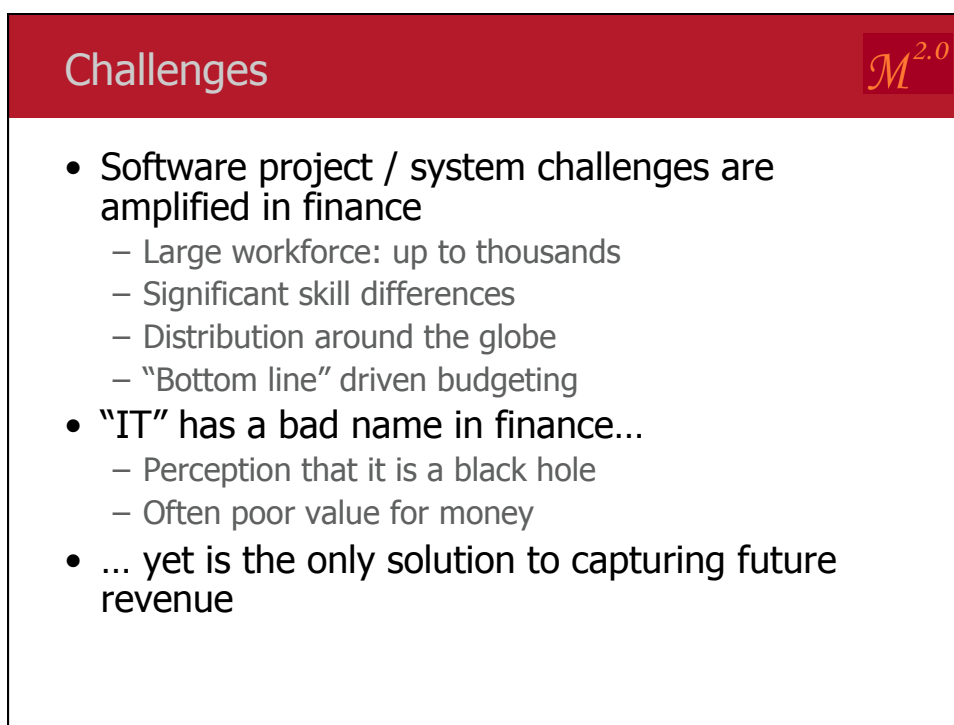
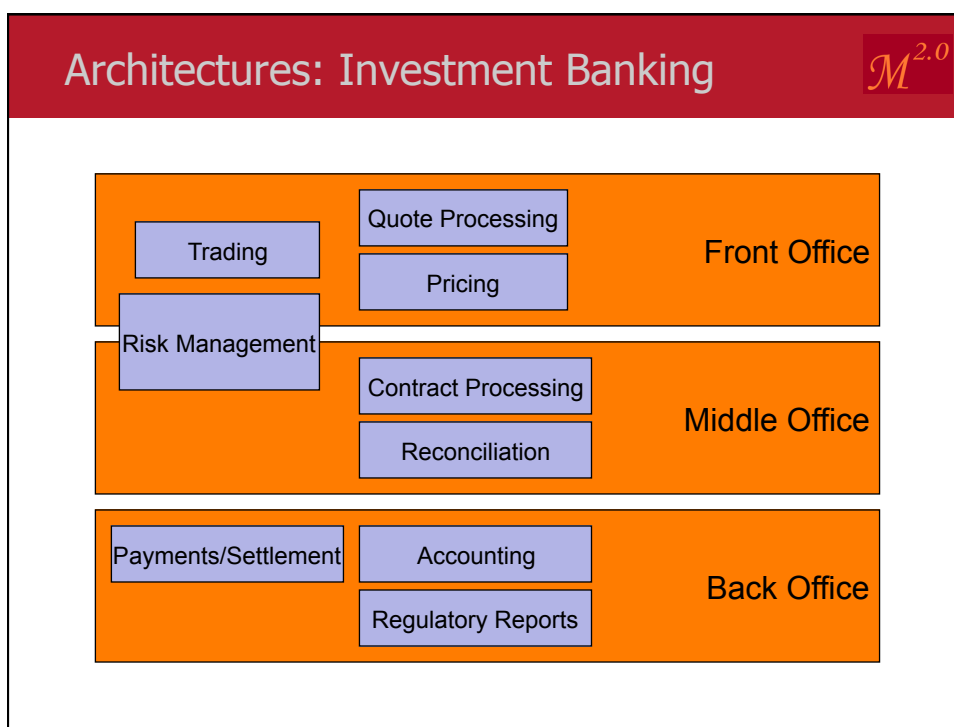
christian@modeltwozero.com

A Quick Introduction

$M^{2.0}$

- Financial services firms are some of the largest users of computing
- The business lends itself well to computers
 - Large amounts of non-tangible assets
 - Big customer-base to be maintained
 - Modern finance = number manipulation!





Urgent Issues for the Next 10 Years



- “Hard science” problems
 - System integration
 - High performance computing
- “Soft science” problems
 - Improved project management
 - Procurement technique
 - Knowledge transfer
- There are others – these are amongst the ones that hurt most or offer most opportunity

System Integration



- IT departments of banks spend almost all their time integrating systems
- Reasons include
 - Purchase of off-the-shelf systems
 - Mergers/acquisitions
 - Retiring out of date systems (mainframe systems, COBOL, etc.)
 - Consolidation onto single platforms
 - Development of new financial products
- Scale
 - Mutual funds: tens of systems, few sites; a few projects per year
 - Investment bank: thousands of systems, globally deployed; hundreds of projects per year

Problems

M^{2.0}

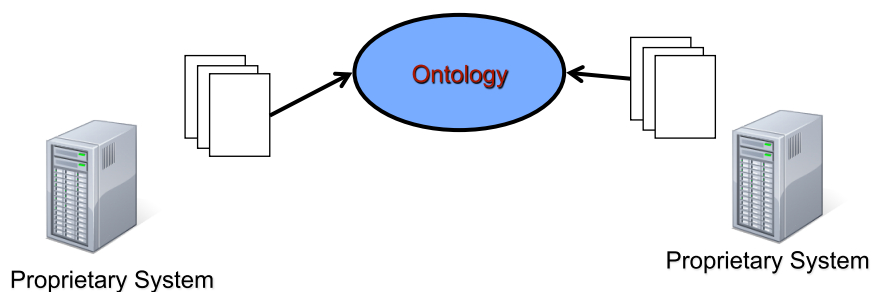
- Dependency on expensive specialists
- Off-the-shelf systems are poorly documented
- Role overlaps
- Avoidable communication overhead with multi-site projects
- Improvements in practice are **not** keeping pace with increased integration needs
- This is a billion-dollar-a-year problem



Possible Solutions

M^{2.0}

- Semantic Integration
 - Establishes a business ontology
 - Relates system data to ontology
 - Transformations are inferred
 - Unproven except for toy examples
 - Too data centric – ignores workflow; What do you map flows to?



Possible Solutions



- Natural Language / Model-driven techniques
 - Raise level of abstraction so that specialists can understand and write rules
 - Helps with handover between teams
 - 2.5 years of work in this area with HSBC
 - 3 projects completed
 - Conclusion outstanding

Context: RelativeDateOffset

Rule "shared-4"

```
If dayType is equal to DayTypeEnum.Business then
  businessDayConvention is equal to
  businessDayConventionEnum.NONE
```

Integration



- This space is crowded by tool vendors (from small companies to IBM)
- Their solutions plug gaps but are very far from sufficient
 - Somewhat akin to trying to save the Titanic using a handful of buckets
- The challenge remains



High-Performance Computing

M^{2.0}

- High-performance computing in banks has many forms
 - High transaction volumes – e.g. payment processing, automated trading
 - Low-latency applications – e.g. responding to quote requests, pricing complex derivatives, risk management

Current State

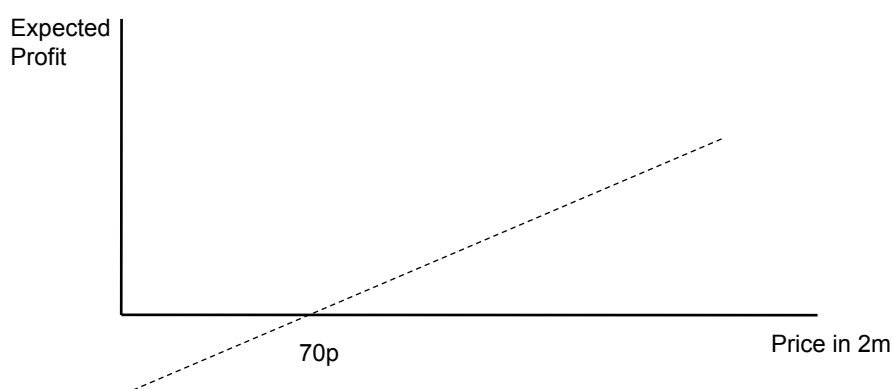
M^{2.0}

- Volumes keep increasing dramatically
 - Derivatives market has seen >100% volume increases year after year for 5 years
 - Margins are being eroded: need to trade *more* to make the same profit
- Power is running out!
 - Traditional implementation techniques are very power-inefficient
 - At least two major investment banks have had requests for further power connections turned down by local councils

Example: Options

 $\mathcal{M}^{2.0}$

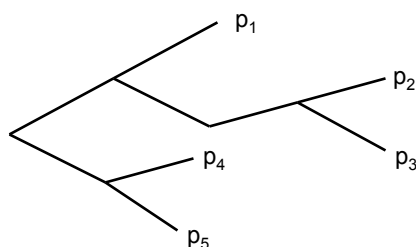
- I want to acquire the right to purchase 10,000 RBS shares in 2 months for 70p
- They are currently trading at 57p



Example: Options

 $\mathcal{M}^{2.0}$

- Assume you want to sell me the option
- How much should you charge for it?
 - Answer depends on how the shares might behave
 - Stochastic simulation commonly used



Increasing Speed



- **Increasing efficiency will pay off**
 - Can trade more
 - Can respond faster than competitors
 - Ultimately, makes the market more efficient
- **Parallel programming is changing fast**
 - Intel multi-core CPU architectures
 - IBM Cell processor
 - Programmable GPUs (e.g. NVIDIA Cuda); 240 parallel processing units instead of 2!
 - FPGAs are back in fashion

Challenges and Opportunities



- **Massively Parallel programming is too hard**
 - Primitives are too low-level
 - Analysts (generally very smart) have to fight the paradigm as well as invent financial models
 - Early state of evolution
 - High-level massively parallel language primitives are needed, for languages that are also good at mathematical manipulation

Challenges and Opportunities



- **Framework lock-in is unacceptable**
 - Portability across platforms is standard for existing languages
 - Cannot move between FPGA, Cell, Multi-Core and GPU architecture without complete rewrite
 - What should the base platforms for massively parallel computing look like?
 - What higher-level languages can be made available on top of the base platform to help with specific tasks?

Project Management



- **Project management skill varies between good and abysmal**
- **Largely a problem of knowledge transfer**
 - Up to date techniques not being adopted
- **Outstanding questions**
 - How should agile processes work with globally distributed teams?
 - How should management structures change to facilitate this? e.g. conflict between part-committed resources and practices like sprints or pair-programming

Procurement



- Computer science / software engineering programmes emphasize requirements management...
 - To collaborate with users to ascertain their needs
 - Widely recognized as the single most crucial step to project success
- ... yet nobody teaches good procurement practice to users
 - How does a non-expert go about buying a system?
 - What is the best process for management to communicate with IT and hold it accountable?
- *Current practice is shockingly ad-hoc*

Knowledge Transfer



- Banks (as large organisations) are extremely poor at “learning”
 - Techniques are very backwards: e.g. use of waterfall process
- How can good practice be effectively transferred to thousands of people?
 - Need to work around different skill levels
 - Cannot assume absence of political barriers
- How can performance of good practice be predicted and subsequently measured?

Summary and Hints



- Research of long-term value in finance needs to solve typical “big company” problems
 - Many of the big problems are “soft science” problems
- Never ignore the real world in research if you want your work to be applicable in practice
 - Catch-22: politics
 - Scale, scale, scale (and then some!)
- The art of successful large organisations is to be able to work with average people
- Challenges are plenty: both for short term and long-term work