IT-driven innovation: From computer science to next-generation enterprise and financial systems

Fritz Henglein
Datalogisk Institut, Københavns Universitet (DIKU)

DIKU virksomhedsklub, 2010-05-06
Outline

- A bit about DIKU and myself
- 3gERP: 3d generation enterprise resource planning
  - POETS: Process-oriented event-driven transaction systems
  - Illustration: An executable contract specification
- The high-productivity high-performance computing challenge
- DIKU and me, COME, from topps-description-5a.pptx
- POETS, from Northwestern.pptx
  - 3gERP
  - General architecture
  - POETS contract code
  - POETS UI
  - NAV contract code
  - Summary: simplicity by …, productivity by domain-orientation, predictability by static analysis
- Financial apps, from Simcorp.pptx
- R&I cooperation for innovation, from topps-description-5a.pptx
• University of Copenhagen (KU)
  • ca. 45,000 students (Bachelor, Master’s)
  • 8 faculties, no engineering school
  • Member IARU, ranked Top-20 Europe, Top-100 World
    (Shanghai and Financial Times university rankings)
• Faculty of Science (SCIENCE)
  • 8 departments (Physics, Biology, Chemistry, Geosciences,
    Maths, Sports/Physiology, CS, Didactics of Science, plus
    museums
• Department of Computer Science (DIKU)
  • ca. 25 faculty, 50 Ph.D. students, 800 B.S. & M.S. students in
    computer science
  • Ranked Top-100 World (Shanghai ranking of CS departments)
  • Bachelor i datalogi (B.S.)
  • Copenhagen Master of Excellence in Computer Science (M.S.)
  • involved in 4 other IT programs: Communication & IT (B),
    Linguistics & IT (M), Health & IT (B), Science & IT (B)
DIKU researcher groups

- **Algorithms and Programming Languages (APL)**
  - Theory and practice of programming languages (TOPPS)
  - Algorithms and optimization (ALGO)
  - Performance engineering
- **Human-Centered Computing (HCC)**
  - Usability, organizational aspects
- **Image group**
  - High-performance systems, eScience
  - Computer vision, graphics and animation
Programming languages: A Scandinavian specialty

- Scandinavian tradition in programming languages and software development
  - Turing Awards:
    - Dahl/Nygaard, Oslo U. (2002), Simula;
    - Naur, DIKU (2005), Algol 60
  - Danish PL (co-)designers:
    - Algol 60 (the "Latin" of modern programming languages),
    - Turbo/Delphi Pascal,
    - C++,
    - Standard ML,
    - C#,
    - PHP
    - Ruby on Rails
    - ...
Fritz Henglein

- Head of Algorithms and Programming Languages Group at DIKU
- Background:
  - Bachelor studies CS & Mathematics, TU München (1980-83)
  - M.S. and Ph.D. in CS from Rutgers University (1983-89)
  - Academic positions: IBM Research, New York University, Utrecht University, IT University of Copenhagen, DIKU (1989-now)
- Research:
  - Algorithmic, logical and semantic aspects of programming languages
  - Program analysis and type systems
  - Algorithmic functional programming
  - **Domain-specific languages**
  - Applications of programming language technology; e.g. to
    - **Enterprise Resource Planning (3gERP.org)**
    - Trustworthy eHealth care (TrustCare.eu)
    - ...
Enterprise Resource Planning (ERP) Systems

- Electronically manage everything in a company: sales and purchase orders, warehouse, bookkeeping, production etc.
- Store relevant data.
- Perform analysis on business data.
Functional View of present-day ERP System Architecture

Source: http://www.bluedzine.com/images/erp_diagram.gif
Centralized database

Configurations on data (tables)
User data (tables)

Code units
Form specifications
Developer’s view of ERP Systems

Large integration, customization and maintenance costs!

Source: http://www.bluedzine.com/images/erp_diagram.gif
The 3gERP Project (www.3gERP.org): 3d generation ERP Systems
ERP market size

- SAP, Oracle, Microsoft Dynamics, many more (mostly regional)
  - Inclusive definition: “enterprise systems” (ERP, CRM, HRM, SCM, etc.)
- Multibillion dollar industry
- Global annual revenues (2009, Gartner Group projections):
  - Videogames: 42 billion dollars
  - Enterprise Software: 223 billion dollars
Goal

- Reduce TCO for SMEs by 50%
  - Alternatively, double TBO
  - easier to distribute globally at relatively low cost,
  - easier to implement (configure, customize and deploy)
  - easier to localize to different markets, industries and enterprises,
  - easier to maintain/update (preserve customizations)
  - easier to integrate with other systems and ERP systems of other enterprises in the value chains/value networks
  - easier to provide better business insight for managing enterprises

- 3gERP I:
  - Evolutionary track: Building on top of MS Dynamics NAV
  - Revolutionary (“tabula rasa”) track: New architecture (POETS) based on declarative domain-specific languages

- 3gERP II: Focus on POETS
3gERP II: POETS (revolutionary track)

- **POETS**: Process-oriented Event-driven Transaction Systems

- **Objectives**:
  - Easy, rapid and reliable mapping of requirements to specifications
  - Safe and predictable customization
  - Light-weight continuous change

- **Method: Simplicity**
  - No coding: Specifications = code
    - Requirements architecture ~ specification architecture = software architecture
  - Domain orientation: Problem-oriented instead of computer-oriented “code”
    - What, not how
  - High performance, predictability and reliability
    - Automatic, provably correct optimization; static type systems
  - Start from scratch

---

"The cheapest, fastest, and most reliable components are those that aren't there."

Gordon Bell
POETS: Ontology for ERP system requirements modeling

- **Data**: Resources, events ("transactions"), agents, documents (basic information such as invoices)

- **Reporting**: Interpretation of all base data by selection, aggregation, correlation, transformation etc.

- **Processes**: Specifications of *expected* sequences of events, in particular (commercial) contracts

- **Rules**: Legal and business constraints on how things are to/may be done, e.g. VAT or customs rules

- **Interfaces**: Specification of interactions between system components, and between system and users (roles).
POETS: Process-oriented Event-driven software architecture
1. Receive 3 iPhones and 2 MacBooks from supplier X
2. Receive 2 iPhones and 1 MacBooks from supplier Y
3. Receive an invoice from X for 3 iPhones (3 * 400 USD incl. VAT) and 2 MacBooks (2 * 2000 USD incl. VAT) and rush delivery charge (20 USD – VAT exempt)
4. Receive invoice from Y for 3 iPhones (3 * 420 USD incl. VAT) and 2 MacBooks (2 * 1940 USD incl. VAT) and shipping (100 USD incl. VAT)
5. Deposit 5220 USD into X’s bank account
6. Send check to Y to the amount of 5240 USD
7. Observe on our bank account that check has been cashed
8. Receive order from A of 1 MacBook and 1 iPhone priced at 3000 USD incl. VAT
9. Deliver 1 MacBook and 1 iPhone to A
10. Receive from A 3000 USD into our bank account
11. Pay VAT due
12. A year passes
13. Deliver, invoice, and receive payment for 1 MacBook worth $800 incl. VAT to Z
<table>
<thead>
<tr>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Receive 3 iPhones and 2 MacBooks from supplier X</td>
</tr>
<tr>
<td>2 Receive 2 iPhones and 1 MacBooks from supplier Y</td>
</tr>
<tr>
<td>3 Receive an invoice from X for 3 iPhones (3 * 400 USD incl. VAT) and 2 MacBooks (2 * 2000 USD incl. VAT) and rush delivery charge (20 USD – VAT exempt)</td>
</tr>
<tr>
<td>4 Receive invoice from Y for 3 iPhones (3 * 420 USD incl. VAT) and 2 MacBooks (2 * 1940 USD incl. VAT) and shipping (100 USD incl. VAT)</td>
</tr>
<tr>
<td>5 Deposit 5220 USD into X’s bank account</td>
</tr>
<tr>
<td>6 Send check to Y to the amount of 5240 USD</td>
</tr>
<tr>
<td>7 Observe on our bank account that check has been cashed</td>
</tr>
</tbody>
</table>

...
<table>
<thead>
<tr>
<th></th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Receive order from A of 1 MacBook and 1 iPhone priced at 3000 USD incl. VAT</td>
</tr>
<tr>
<td>9</td>
<td>Deliver 1 MacBook and 1 iPhone to A</td>
</tr>
<tr>
<td>10</td>
<td>Receive from A 3000 USD into our bank account</td>
</tr>
<tr>
<td>11</td>
<td>Pay VAT due</td>
</tr>
<tr>
<td>12</td>
<td>A year passes</td>
</tr>
<tr>
<td>13</td>
<td>Deliver, invoice, and receive payment for 1 MacBook worth $800 incl. VAT to Z</td>
</tr>
</tbody>
</table>

Registered events
Contract language (preliminary design):
Example of complete contract specification

```
saleOfGoods (deliveryDeadline : Time,
            payment : ValueDesignator,
            goods : Resource,
            deliveryLocation : StationaryLocation,
            damageCompensation : Resource) : Contract =

  // First the goods must be transferred and delivered
  TransferAndDeliver(r <- resource, l <- location) by one @ t
  where r = goods and
      l = deliveryLocation
  before deliveryDeadline
  then
  do
  {
    // The goods must be payed for
    payment : Payment(v <- value) by two
    where v = payment
    before deliveryDeadline,

    // After receiving the goods, the customer has 14 days to notify the seller
    // of any claim for damages, and subsequently the seller has 30 days to
    // compensate
    claim : claimForDamages(goods,
                            t + 14 days,
                            damageCompensation,
                            30 days,
                            deliveryLocation)
  }
```

• Executable: no coding!
• Domain-oriented: no code for database access, memory allocation, etc.
Contract language (preliminary design):
Example of complete contract specification

(Demo of graphical user-interface for user-defined contracts)
HIPERFIT:
Functional High-Performance Computing for Financial Information Technology

(joint initiative with Brian Vinter, DIKU, head of eScience Center at U. Copenhagen)
The parallel programming challenge

Typical system specification anno 2010:

<table>
<thead>
<tr>
<th>System unit</th>
<th>Parallel processing units</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>240 cores</td>
</tr>
<tr>
<td>CPU</td>
<td>8 cores</td>
</tr>
<tr>
<td>Large cluster</td>
<td>40,000 servers * (4-8) cores</td>
</tr>
</tbody>
</table>

2010: How to exploit ~100 cores
2020: How to exploit ~3000 cores
(+ how to allocate and transfer data)
Technology trend: Parallelism

- Moore’s Law: Doubling of transistors on IC every 2 years
  - Earlier: Also increased clock frequency
  - Now: Only doubling of on-chip processing units (cores)

Source: Top500.org, 2010-01-28
General-purpose graphics processor units

- Earlier:
  - Closed (graphics programs only)
  - No data integrity

- Now/soon:
  - Programmable (CUDA C, OpenCL)
  - Data integrity (ECC)

Example: Nvidia’s Fermi

512 hyper-threaded cores
## Supercomputing

<table>
<thead>
<tr>
<th>Year</th>
<th>Rank</th>
<th>Computer</th>
<th>CPUs</th>
<th>Rpeak</th>
</tr>
</thead>
<tbody>
<tr>
<td>1993</td>
<td>1</td>
<td>CM-5</td>
<td>1024</td>
<td>131</td>
</tr>
<tr>
<td>2002</td>
<td>500</td>
<td>Power3</td>
<td>132</td>
<td>198</td>
</tr>
<tr>
<td>2006</td>
<td>n/a</td>
<td>CELL</td>
<td>1 (9)</td>
<td>205 (110)</td>
</tr>
<tr>
<td>2010</td>
<td>n/a</td>
<td>Fermi</td>
<td>1 (512)</td>
<td>1248 (624)</td>
</tr>
</tbody>
</table>
Parallel programming challenge: What will no longer work

Parallelization of existing sequential code ("dusty deck" optimization)

<table>
<thead>
<tr>
<th>Technique</th>
<th>Parallization</th>
<th>Known since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software pipelining</td>
<td>2-4</td>
<td>70s</td>
</tr>
<tr>
<td>Hardware pipelining</td>
<td>5</td>
<td>80s</td>
</tr>
<tr>
<td>Speculative computation (trace sched. + VLIW)</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Dynamic instruction scheduling, hyperthreading</td>
<td>3-5</td>
<td></td>
</tr>
</tbody>
</table>

**Corollary:** We need new languages, tools and code!

Combined effect: < 10 CPU cores
Financial systems trends

- Exponential data growth
  - Doubling every ~3 years
- Increasing real-time requirements
  - Low-latency algotrading
  - RDBMS bottlenecks
  - Real-time business intelligence
- Increasing computing requirements
  - Monte-Carlo simulations
  - Quantitative credit risk modeling
- Increased flexibility and functionality
  - Custom OTC contracts
  - Custom business intelligence reports and real-time data
The Fundamental Challenge

- Parallelism, *and*
- Productivity, *and*
- Hardware independence, *and*
- Correctness
The Goal

Current workflow

Target workflow

Did not work/Too slow

Idea → Prototype → Full version → Production

Too slow

Did not work

Idea → Full version → Production

<Undfylde sidefod-oplysninger her>
Why functional programming (languages)?

- Domain-oriented modeling of “functional” problems:
  - Mathematical problems (linear algebra, partial differential equations)
  - Domain-specific problems (contract monitoring and valuation)
  - Data (base/warehouse) querying, reporting, mining

- Scalable data-parallelism and asynchronous parallel execution:
  - Bulk data processing operations (sequences, multisets, sets; matrices)
  - Exploitation of vector processing hardware incl. GPGPUs, FPGAs
  - Functional skeletons for distributed processing (MapReduce, etc.)

- Protection of investment:
  - Specification of functional objectives (what?), not system/implementation particulars (how?)
  - Proven technology (lambdas and closures since 60s, garbage collection and polymorphism since 70s, modules and parallel execution since 80s, whole-program optimization since 90s, )
  - Safe software evolvability (compactness/genericity, type abstraction, checking, modularity, embeddable domain-specific languages)
Results so far (data parallelism)

![16M samples Monte Carlo Pi](image)

- **Toolbox**: Shortest time
- **Std Python**: Longest time

![Speedup](image)

- **Nvidia**
- **CPU**

*Speedup vs. Number of Cores*
Results so far

From a sequential application

Distributed Memory: 60.5 speedup on 64 CPU
National network: Processes and IT

- **InfinIT: Innovationsnetværk for IT**
  - [www.infinit.dk](http://www.infinit.dk)
- **Interest group on *Processes and IT***
  - Kontakt:
    - Thomas Hildebrandt, [hilde@itu.dk](mailto:hilde@itu.dk)
    - Fritz Henglein, [henglein@diku.dk](mailto:henglein@diku.dk)
    - Or infinit.dk secretariat
- **Kick-off event: April 27, 2010**
International academic network

- Carnegie Mellon University (Blelloch/Harper)
- University of Chicago (Manticore)
- English university consortium (under formation)
- Scottish university consortium (pH)
- Chalmers University of Technology (Obsidian)
- University of New South Wales (Data parallel Haskell)
- Microsoft Research, Cambridge/UK and Redmond/WA
- Princeton University
- Barcelona Supercomputing centre
- FORTH/Greece
What do I believe makes for a good university-industrial *research* cooperation?

- Every party brings something to the table and needs the other parties to get its own goals accomplished
- Clear agreement on goals, priority, time line, knowledge exchange/ internal deliverables and success criteria
  - Success criteria for academics involved (“research quality”, “recognition”, “publication”, etc.)
  - Success criteria for companies involved (“cost/benefit”, “ROI”, “IP”, etc.)
- Acceptance and explicit commitment (of resources and priority) from senior managements
  - Commitment to agreements even as management and/or workers involved in the companies get replaced
- Parallel track: research activities and development activities
  - *Not* placing the cooperation’s goals/deliverables on a company’s critical path (okay to do so for low-risk, low-return *innovation* projects, but not for high-risk, high-return *research* projects)
  - Synchronization points for evaluating and transferring results between research and development tracks
- (Industry) Preferably 1 Ph.D. at each company who can act as communication/cultural bridge and who has
  - something to gain from a successful cooperation
  - something to say (power) at the company to facilitate activities.
- (University) Recognition that cooperations require up-front personal investment (without academic pay-off) and are not “free”, even when funding is provided by 3d party, since time and effort expended disallow pursuing alternative activities.