Functional Programming In Real Life

Dr. Erik Stenman
Director of Engineering
Kreditor Europe AB
Introduction

• I will talk about KREDITOR, a company that bet it's future on Erlang a functional programming language

• Conventional wisdom...
  choose proven technology

• ... the KREDITOR way ...
  choose ERLANG

• ... are they the same?

• I will tell you what KREDITOR does, how we do it, why we do it this way, and whether it worked out or not... at least, so far.
• The business model:
  – Bring trust to Internet shopping.
  – Bring old style billing into the new IT-economy.

• Brief background:
  – With < $100,000 in venture capital.
  – Live system in March 2005.

• The company vision:
  – “Be the coolest company in Sweden.”
The Problem

- Internet shopping is a question of **trust**.
  - The **shop** has to **trust** the **customer** to get paid.
  - The **customer** has to **trust** the **shop** to send the right stuff.
- Many **customers** are **uncomfortable** using credit card over the Internet.
- Many banks are actually **worried about the security** of Internet **shops** handling credit card information.
- Also, doing a **partial return** when using credit card is a **hassle**, both for the **customer** and the **shop**.
The Solution

• Bring in a trusted party, i.e., KREDITOR.
• Send an invoice with the goods to the customer.
• The customer pays after receiving the goods and takes no risk. The customer does not have to trust anyone.
• The shop is guaranteed (by contract) to get money from KREDITOR. The shop only have to trust KREDITOR with whom they have a written contract.
Added benefits

- The **customer** gets credit.
- The **customer** can pay using familiar methods.
- Returning goods is easy.
- Better fraud detection.
- Advanced credit assessments.
- Easy to add similar features like *pre-pay*, *subscriptions* and *installment plans*.
The Implementation

A customer surfs a webshop. The webshop uses xmlrpc to communicate with KREDITOR for a kredit check. KREDITOR provides credit information.
The Implementation

A customer surfs a webshop, which uses xmlrpc to communicate with KREDITOR for credit information. KREDITOR stores this information in a database (DB).
The Implementation

A customer surfs to a webshop with invno=42, which checks credit information with KREDITOR and KREDITOR DB.
The Implementation

A customer

A webshop

Buy ok

Pack goods, print invoice, ship.

Credit information

KREDITOR

DB
The Implementation

A customer

A webshop

Credit information

KREDITOR

Bank

Pay

DB

Pay
The Implementation

A customer

A webshop

Credit information

Reminder

Print & mail

Bank

KREDITOR

DB
The Implementation

A customer

A webshop

Credit information

Web-UI

KREDITOR

DB

Bank

Print&mail
The system is a combination of fault tolerant servers, lots of protocol- and glue-code, persistent storage, and a web-UI.
Some details

- The system is built from scratch using LYME (Linux, Yaws, Mnesia, and Erlang).
- So far we operate in Sweden and Norway.
- We have a distributed system with multiple servers to provide a fault tolerant, high availability solution.
- We aim for less than 5 minutes down-time per year, in a setting where we introduce new features in the system every week.
- The problem fits Erlang really well.
Why not use Erlang?

• The main reasons that I have heard of are:
  1. Politics – Erlang is not C/Java, company policy.
  2. One provider – Concern that Ericsson will stop supporting Erlang.
  3. Lack of programmers – Erlang is still not mainstream how can we ensure we get qualified staff?

• When starting a new company, 1 is (usually) not a problem.
• I can't see 2 happening, and Erlang is open source anyway.
• When setting up in Stockholm, 3 is not a problem.
Why Use Erlang?

• Easy to build fault-tolerant systems.
• Fast development.
• Easy to maintain and upgrade.
• Can utilize multi core technology.
• Good way to get great programmers.
Easy to Build Fault-tolerant Systems

- **Erlang** was designed from the ground up with the purpose of making it easy to develop fault-tolerant systems.
- **Erlang** was developed by [Ericsson](#) with the telecom market in mind.
- **Erlang** supports processes, distributed systems, advanced exception handling, and signals.
- **Erlang** comes with [OTP-libraries](#) (Open Telecom Platform), e.g. *supervisors* and *generic servers*.
• **Erlang** has a number of features to support rapid prototyping and fast development:
  – Automatic memory management.
  – Symbolic constants (atoms).
  – An interactive shell.
  – Dynamic typing.
  – Simple but powerful data types.
  – Higher order functions and list comprehensions.
  – Built in (distributed) database.
Easy to Maintain and Upgrade

- **Erlang** has a number of features that makes it easy to maintain and upgrade:
  - Hot code loading.
  - Distribution.
  - Interactive shell.
  - Simple module system.
  - No shared state.
  - Virtual machine.
Multi Core Technology

• The concept of processes is an integral part of Erlang.
• The Erlang Virtual machine (BEAM) has support for symmetric multiprocessing.
• No shared memory -- easier to program.
Great Programmers

- **Nice paradox**: The lack of Erlang programmers makes it easier for us to find great programmers.
- There are many great C and Java programmers, I’m sure, but they are hidden by hordes of mediocre programmers.
- Programmers who know a functional programming language are often **passionate** about programming.
- Passionate programmers makes great programmers.
Did it work?

• **Erlang** has been a great help in providing rapid development with maintained high availability.

• **KREDITOR** has introduced new services several times per year.

• We have signed with over **1700** internet shops. (Dustin, Discshop, SIBA, Webhallen, Gymgrossisten)

• The system has never been down. Not even while upgrading the system hardware.
Did it work?

- Using **Erlang** has meant low development costs.
- We have been able to grow with our own money.
- In order to finance the new installment plans we recently sold a large part of the company to the very reputable investment firm Öresund.
Did it work?

- The business model **has been sound**.
- Total investment ~600,000 SEK
- Number of connected stores:
- Number of employees ~40.
- Turnover:
  - 2004: ~0 SEK
  - 2005: 1.5 million SEK
  - 2006: 13.5 million SEK, (500 million SEK in invoices)
  - 2007: 30 million SEK (1 billion SEK in invoices)
- Profit 2007: 4-5 million SEK.
KREDITOR took a bet on Erlang, and we are winning.

Questions?