Prolog and CHR in Finance

An easier way to occupy Wall Street?

Mike Elston, SecuritEase. ICLP 2012 Budapest, Hungary.

Aims

- Show how Prolog and CHR have been used to develop on "ordinary" application
- Demonstrate that the finance sector is ideally suited to use of Prolog and CHR

What have we done?

- Built SecuritEase, the dominant stock broking system in New Zealand
- Written in Prolog and CHR
- Grown a profitable business around the product

What does SecuritEase do?

- Stock broking back office + order interface
  - Orders -> Trades -> Contracts -> Payment/Delivery
  - Equities
  - Options
  - Futures
  - Bonds
  - Foreign Exchange
- Stock exchange settlement interfaces
  - CHESS (Australia)
  - OMX (New Zealand)
- Stock exchange order interfaces
  - FIX
  - HTML - Online trading

What are we doing now?

- Breaking into the Australian market
- In production at
  - National Australia Bank
  - Chi-X
  - Instinet
  - ABN AMRO
- Other installations in progress

Market Scale

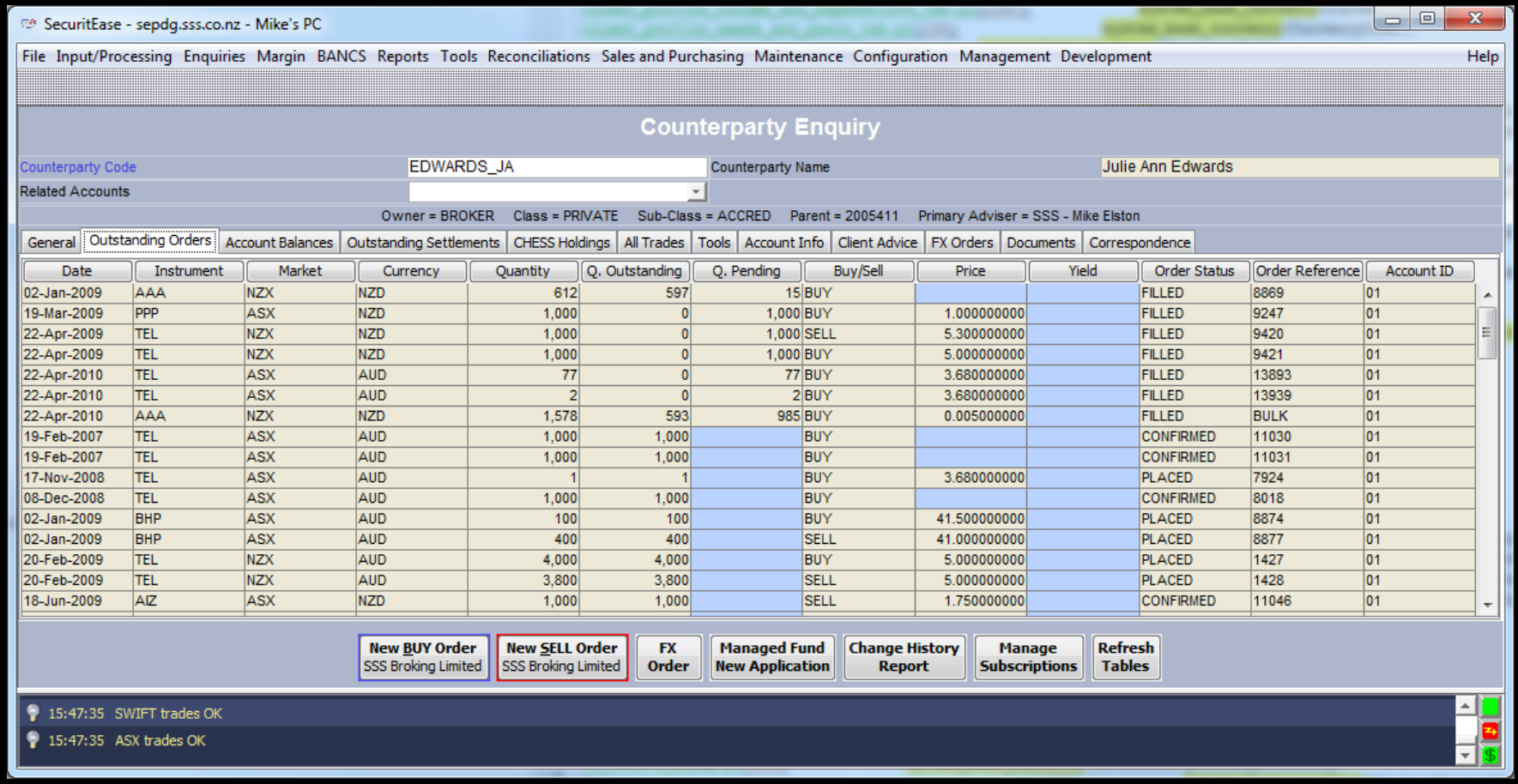
- New Zealand
  - 4.4 million (March 2012)
  - NZX
  - Market capitalization € 32 billion (June 2009)

- Republic of Ireland
  - 4.6 million (April 2011)
  - ISEQ
  - Market capitalization € 47 billion (Dec 2010)

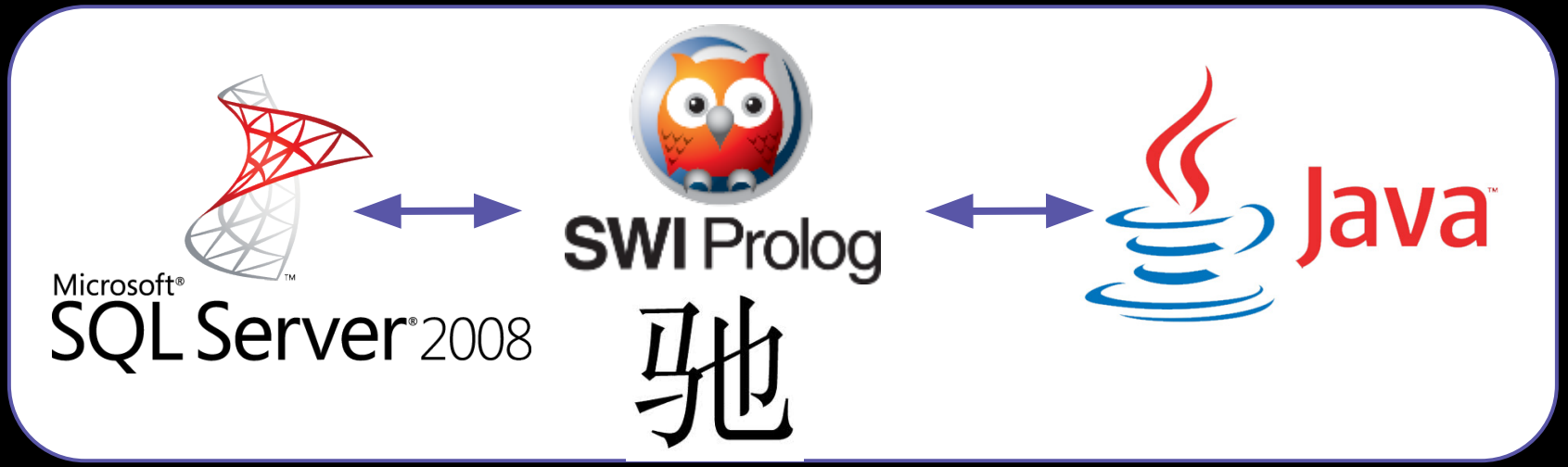
Australia

- Population 22.7 million (August 2012)
- Market capitalization € 1010 billion (Aug 2010)
- 5 times New Zealand's population
- 30 times market value

User Interface



Architecture



Sound Easy?

- High availability
- Auditability/Traceability
- Increasingly 24x6
- Settlement and order interfaces change every few years
- Large
  - 1648 modules
  - 863 forms
- Some logically complex modules
  - Corporate actions - complex time-based logic
- Reversibility
- Multi-million dollar systems

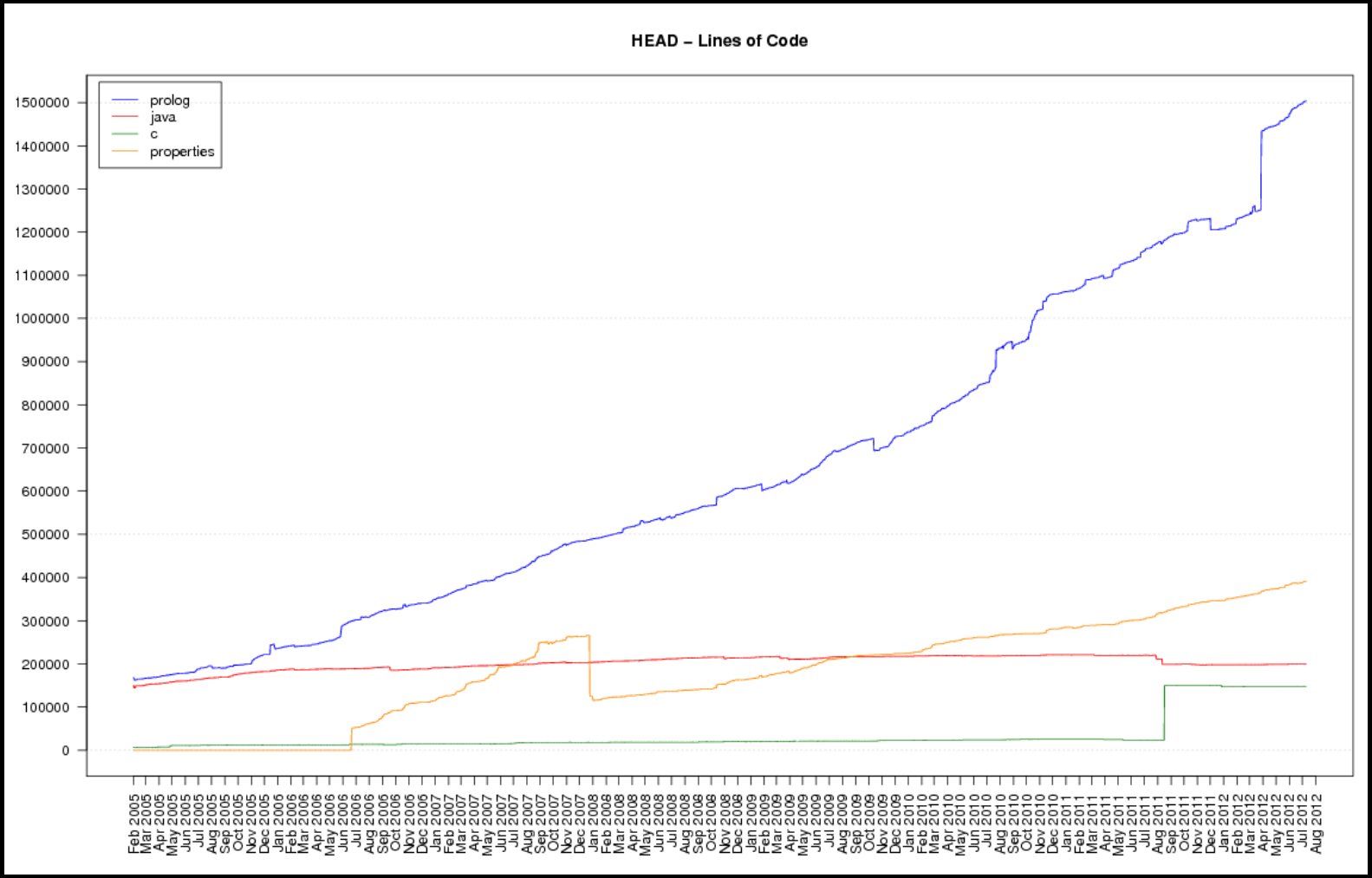
Evolution

- 2002
  - 1 entrepreneur
  - 1 developer
  - 1 analyst
  - 0 customers
  - 1 market

Evolution

- 2012
  - 9 developers
  - 6 analysts
  - 5 support staff
  - 11 customers
  - 2 settlement markets
  - Worldwide trading markets
  - Multi-product
    - Equities
    - Bonds
    - FX
    - Options
    - Futures
    - Margin lending

Statistics - Prolog



Statistics - CHR

- 1410 simplification/simpagation rules
- 205 propagation rules

Server Statistics

- 107 message queues
- 945,089 atoms
- 55,373 functors
- 145,842 predicates
- 2,527 dynamic predicates, 366,233 clauses
- 5,876 modules
- 83,864,789 VM-codes
- 73 threads

Interfaces

- AES
- SSL
- HTTP(S)
- Kerberos
- CSV
- SOAP
  - A generic SOAP interface would be nice
- SMI
- ODBC
- JNLP
- FIX
- SMTP
- SMS
- Excel
- CHESS
- Ajax
- CHESS
- IBM Message Queue
- Open Office
- Signal B
- Signal E
- XML

Why consider Prolog?

- Tried it
  - 1994 Soil fertility analysis system
- Personal preference
  - Own investment on the line
- Seemed a good match
  - Rules oriented
  - Relational database
  - Lots of string matching
  - Explicit representation of knowledge
  - Interactive development

Why SWI Prolog?

- Interfaces
  - ODBC
  - HTML
  - SSL
  - Sockets
  - XML
- Fast bug fixes
- Fast and cost effective feature extension
  - GMP
  - SSL
  - 5.9.x stack-shifter
- No runtime licence fee
  - Important in start-up phase

How to use Prolog?

- Can't hire ready-made Prolog developers
- Some graduates fearful of a non-standard career path
  - Other graduates seem happy to not be in the mainstream
- Lack of type checking a risk for large scale programs
- Create tools adapted to application
  - Use goal and term expansion

Roles

- Prolog experts and top programmers write tools
- Application programmers use the tools
- Analysts understand the tools

Prolog for Everyone?

Business specialist overheard:

“Do you think we should raise an exception here, or just let the predicate fail?”

Development Environment

- SWI Prolog and CHR
- Oracle Java/Swing
- Microsoft SQL Server ™
- MinGW (C)
- Emacs
- Bugzilla
- GIT
- Twiki
- Clear Reports, Jasper Reports
- SecuritEase
  - Form builder
  - Interactive development (Prolog console)

Key Application Characteristics

- Database intensive
- Large number of forms
- Complex message interfaces

Response

- CQL
  Integrates Prolog with an RDBMS
- SWIF
  Integrates logic (Prolog) with an OO graphical user interface
- Message Methodology
  Facts first, interface engines second

CQL - Interface

- Initially considered PL2SQL
  1992 Christoph Draxler
  https://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/lang/prolog/code/io/pl2sql/0.html
- Positional argument problem

```
order(_, _, _, _, _, _, _, _, _, _, MarketId, _, _, _, _, _, _)
```

  - Prolog-like, but
  - Every schema change would require source code modification
- No support for outer joins

CQL

- In an evolving system SQL text impractical
  - How to detect code invalidated by schema changes?

CQL

```
{[OrderId],
 se_order :: [order_id- OrderId,
              buy_sell_indicator- BuySellIndicator,
              quantity- QuantityNow,
              market_id- MarketId,
              instrument_code- InstrumentCode,
              cp_code- CpCode,
              cp_account_id- CpAccountId,
              order_type- OrderType]
 =*=
 se_cp_account :: [cp_code- CpCode,
                   cp_account_id- CpAccountId,
                   buy_trading_stopped_flag- BuyTradingStoppedFlag]
 =*=
 se_instrument :: [market_id- MarketId,
                   instrument_code- InstrumentCode]}
```

Generated SQL

```
SELECT
  si_15.instrument_code,
  si_15.market_id,
  sca_10.buy_trading_stopped_flag,
  sca_10.cp_account_id,
  sca_10.cp_code,
  so_9.order_type,
  so_9.cp_account_id,
  so_9.cp_code,
  so_9.instrument_code,
  so_9.market_id,
  so_9.quantity,
  so_9.buy_sell_indicator,
  so_9.order_id
FROM
  se_order so_9
  INNER JOIN se_cp_account sca_10 ON sca_10.cp_account_id=so_9.cp_account_id AND
                                     sca_10.cp_code=so_9.cp_code
  INNER JOIN se_instrument si_15 ON si_15.instrument_code=so_9.instrument_code AND
                                    si_15.market_id=so_9.market_id
WHERE
  so_9.order_id = ?
```

CQL

- Used CHR to implement the compiler
  - Suited the incremental addition of features
  - In Prolog extra arguments require many predicates to be updated
  - Argument -> Constraint
- ODBC for portability
  - 29 Aug 2011 Microsoft announces focus on ODBC
  - Dropping OLE DB
- Shared variables more concise than SQL
- Can support dialects of SQL for portability

Database Metadata

- All database metadata stored as Prolog facts
  - User defined data types (domains)
  - Table definitions
  - Views
  - Indexes
  - Constraints
  - Stored procedures
- Under code management (GIT)
- Can check and repair a database

Entity Definition

```
%%      entity(?EntityType:table/view, ?Schema:atom, ?EntityName:atom, ?Columns:list).
%
%       Term expansion translates these into database_entity/4 and database_attribute/8 facts.

entity(table,
       securitease,
       se_gl_movement,
       [ column(gl_movement_pk, domain(dom_se_primary_key), allows_nulls(false), is_identity(true), {null}),
         column(gl_company_code, domain(dom_se_gl_company_code), allows_nulls(false), is_identity(false), {null}),
         column(gl_division_code, domain(dom_se_gl_division_code), allows_nulls(false), is_identity(false), {null}),
         column(gl_department_code, domain(dom_se_gl_department_code), allows_nulls(false), is_identity(false), {null}),
         column(analysis_tag_attribution, domain(dom_se_user_id), allows_nulls(true), is_identity(false), {null}),
         column(inserted_, domain(dom_se_date), allows_nulls(true), is_identity(false), {null})]).
```

Plus other facts for constraints and indexes

Views

- Queries beyond CQL capabilities
- Access from other tools e.g. report engines
- Allow SQL experts to work in their native environment
- Recently started parsing views
  - DCG
  - Detect errors
  - Detect deviation from best practice

Views continued

- SQL view definitions are code
  - 765 views in SecuritEase
- Want them managed in GIT
- But want them accessible to Prolog
  - interface analysis
  - style analysis
  - module dependency analysis (make)

The "Long String" Problem

Prolog can handle long strings, but ...

- Adding the line continuation characters is tedious
- The continuation characters are visually distracting
- Might want to paste the code back to SQL

How not to do it

```
view(
'CREATE VIEW [dbo].[se_vw_gl_access]\n\c
AS\n\c
  SELECT u.user_id,\n\c
         glpb.gl_company_code,\n\c
         glpb.gl_division_code,\n\c
         glpb.gl_department_code\n\c
  FROM   dbo.se_gl_period_balance AS glpb\n\c
         INNER JOIN dbo.se_gl_access_company AS glac\n\c
           ON glpb.gl_company_code = glac.gl_company_code\n\c
         INNER JOIN dbo.se_user AS u\n\c
           ON glac.user_id = u.user_id\n\c
  WHERE  ( u.record_status = 'IN_USE' )\n\c
         AND ( glac.is_active = 1 )\n\c
...').
```


Better

- SWI Prolog already has a big string mechanism

```
%%      comment_hook(+Comments, +TermPos, +FileName, +Term)

:-dynamic
        user:process_sql_comments/4.
:-multifile
        user:process_sql_comments/4.

comment_hook(Comments, TermPos, FileName, Term) :-
        user:process_sql_comments(Comments, TermPos, FileName, Term).
```

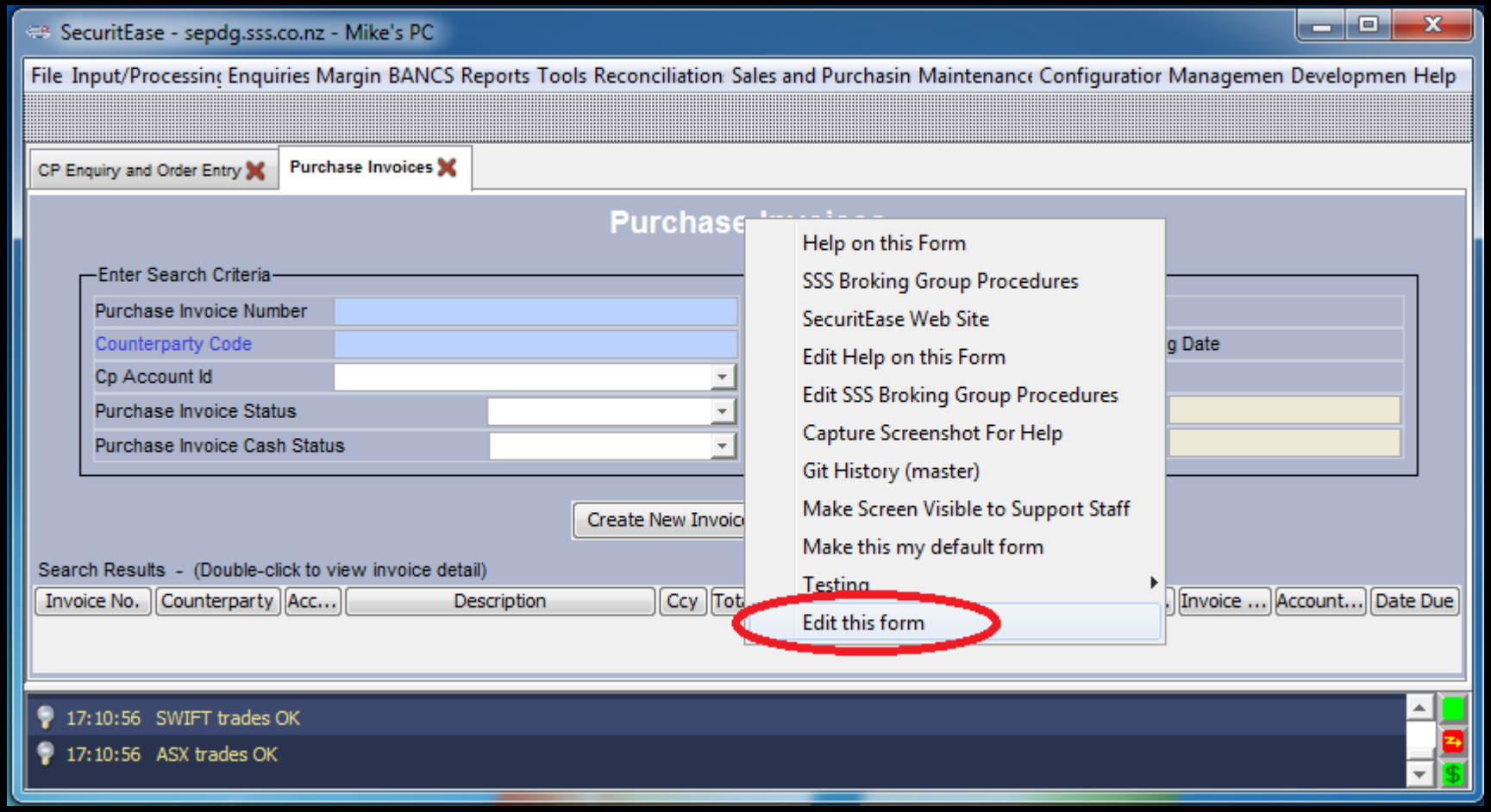

The "Long String" Problem Solved

```
/* $SQL$
CREATE VIEW [dbo].[se_vw_gl_access]
AS
  SELECT u.user_id,
         glpb.gl_company_code,
         glpb.gl_division_code,
         glpb.gl_department_code
  FROM   dbo.se_gl_period_balance AS glpb
         INNER JOIN dbo.se_gl_access_company AS glac
           ON glpb.gl_company_code = glac.gl_company_code
         INNER JOIN dbo.se_user AS u
           ON glac.user_id = u.user_id
  WHERE  ( u.record_status = 'IN_USE' )
         AND ( glac.is_active = 1 )
*/
```

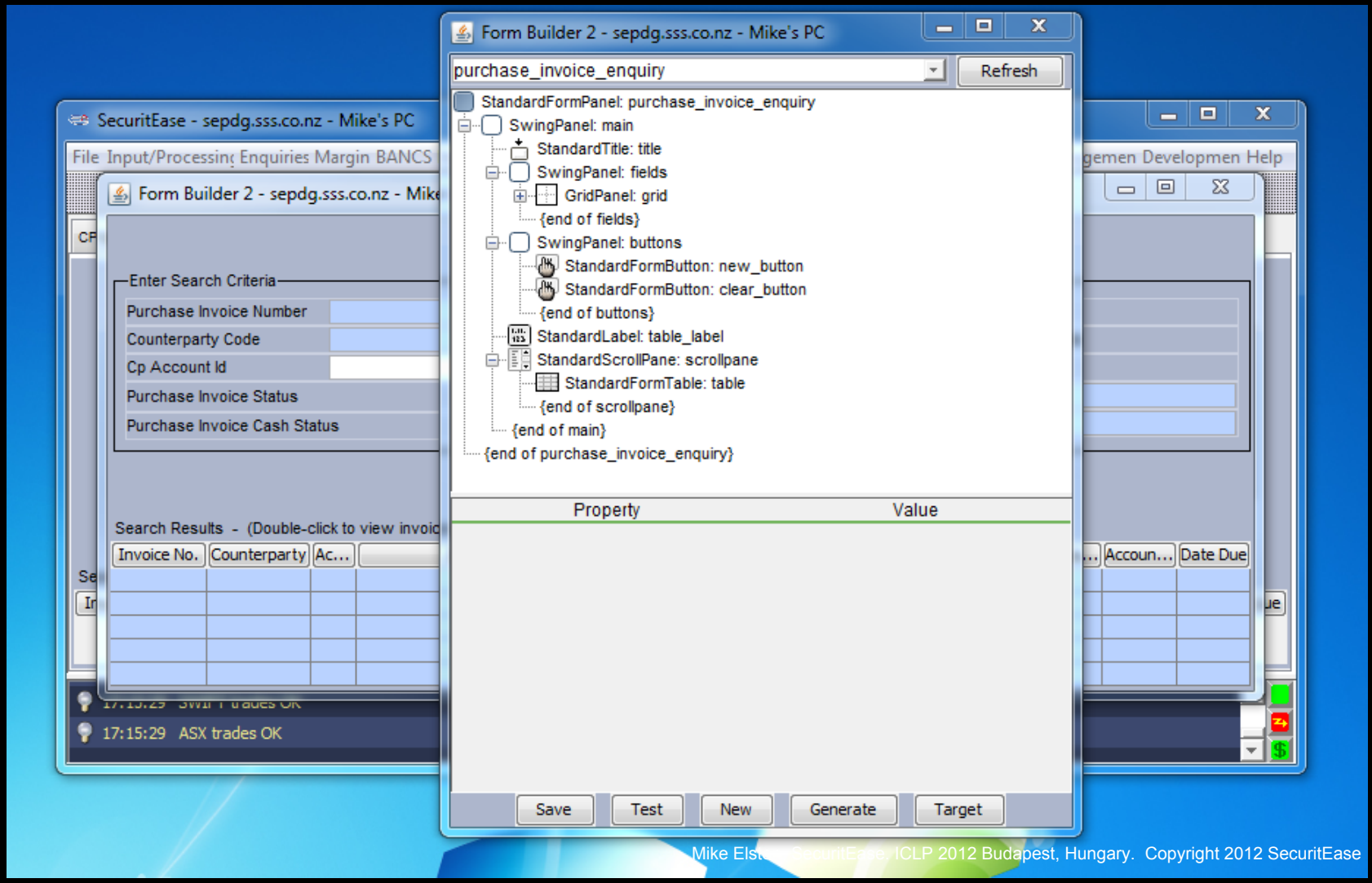

CQL details

- Collation to force Prolog (exact) string matching
- Reports
  - Reports cannot use CQL
  - Contain SQL query strings
  - Use pio to scan report source for invalid references
  - Possible because RDBMS schema captured as Prolog facts

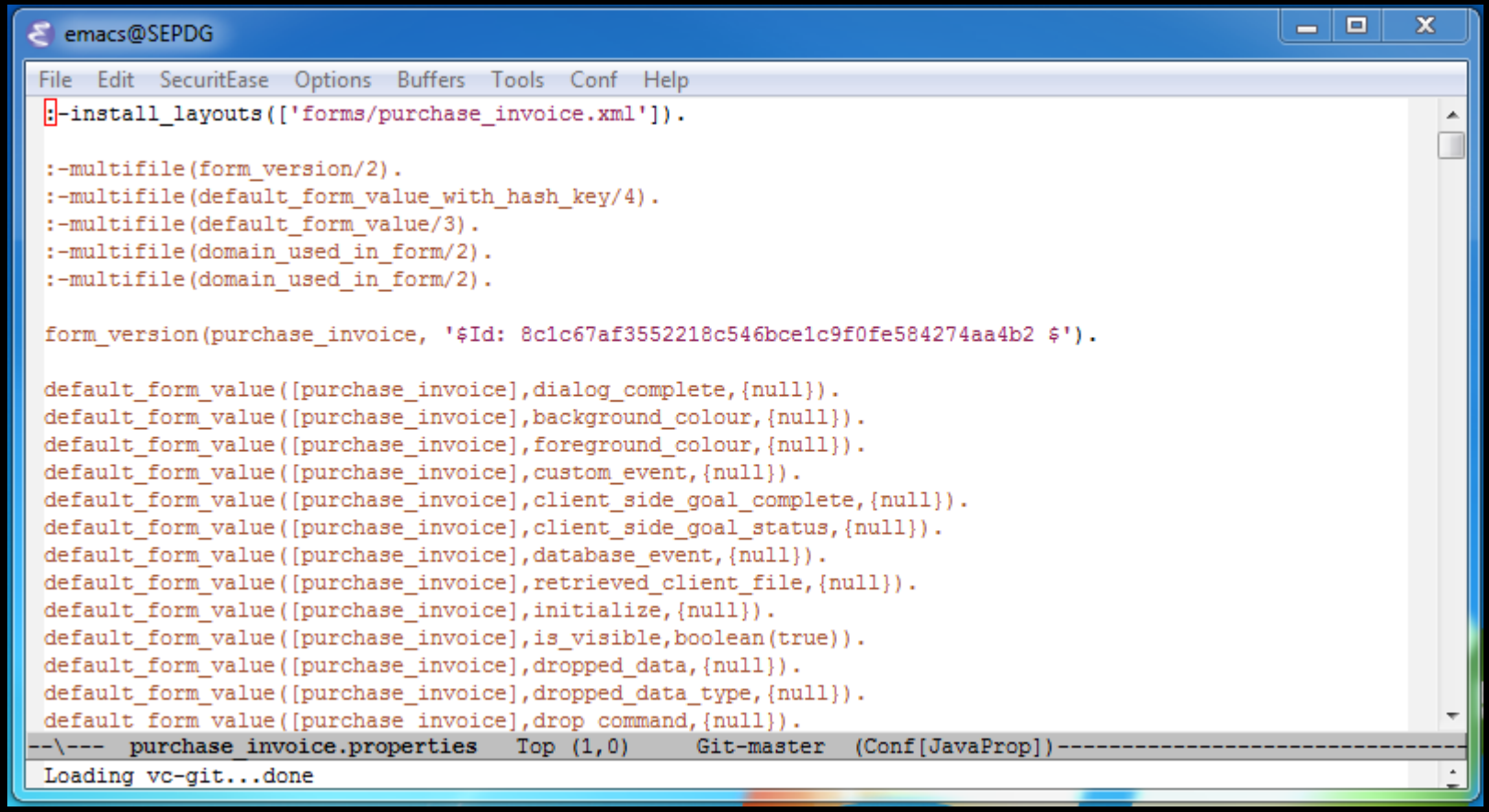
SWIF - SWI Forms



Form Builder



Form Builder



User Event Handling

- User interface properties are logical variables
  - `client_name :: value-ClientName`
  - `client_name:: is_visible-ClientNameIsVisible`
  - `client_name:: is_enabled-ClientNameIsEnabled`
  - `client_name:: is_right_clickable-IsRightClickable`

Formulas

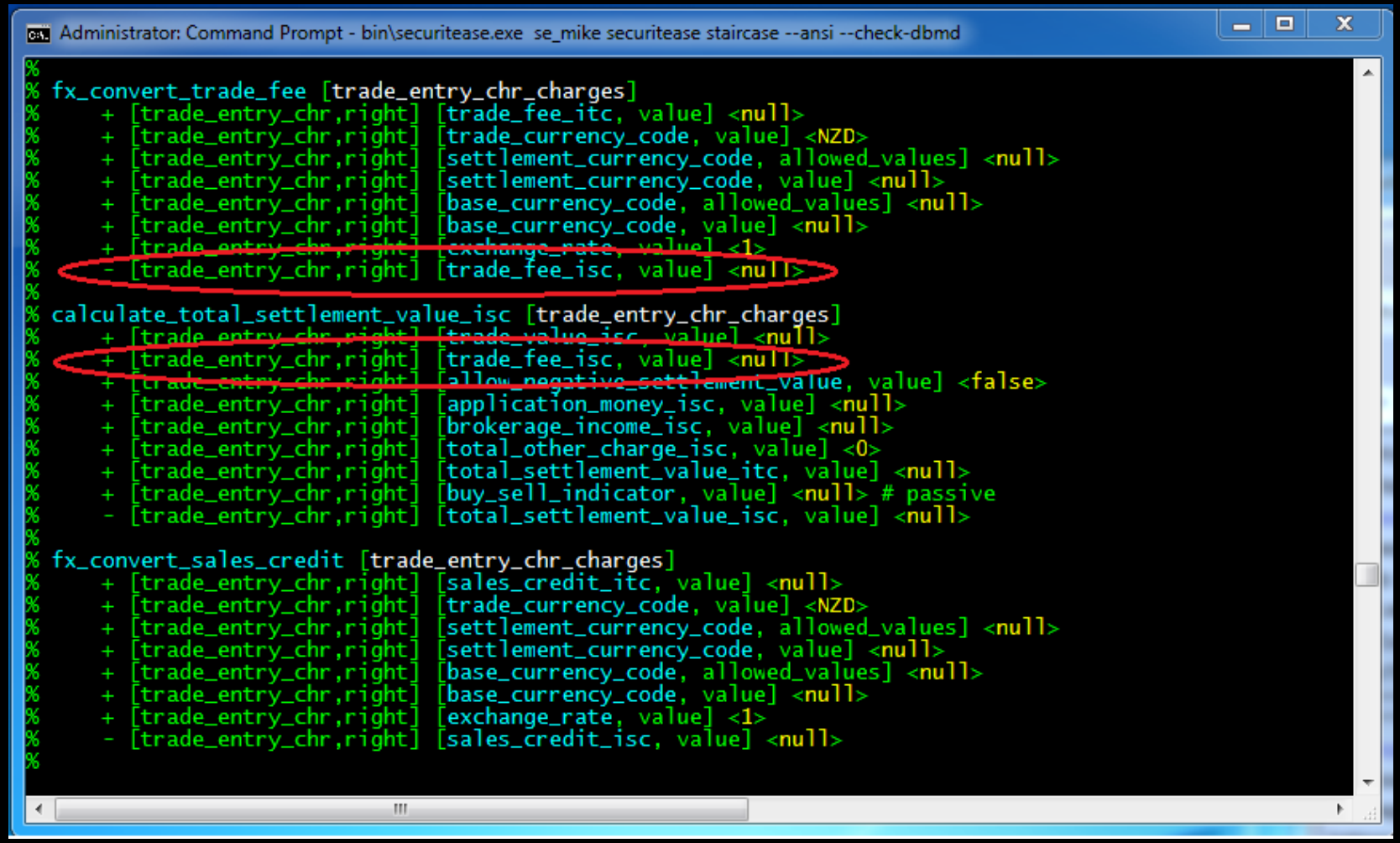
A logical model for user interface event processing inspired by CHR syntax

```
shareholder_number --
        cp_code :: value-CpCode,
        cp_account_id :: value-CpAccountId
        =>
        some_predicate(CpCode,
                       CpAccountId,
                       ShareholderNumber)
        |
       shareholder_number :: value-ShareholderNumber,
       shareholder_number :: is_visible-boolean(true).
```


Forward chaining cascade

- Computes all the consequences of a user interface property change
- GUI programming becomes ...
- Writing a set of true state transition statements

SWIF Forward Chaining



Logical GUI

```
rule_a --
        cp_code :: value-CpCode
        =>
        some_predicate(...)
        |
        shareholder_number :: is_visible-boolean(true).

rule_b --
        cp_code :: value-CpCode
        =>
        some_predicate(...)
        |
        shareholder_number :: is_visible-boolean(false).
```

Form properties are logical variables

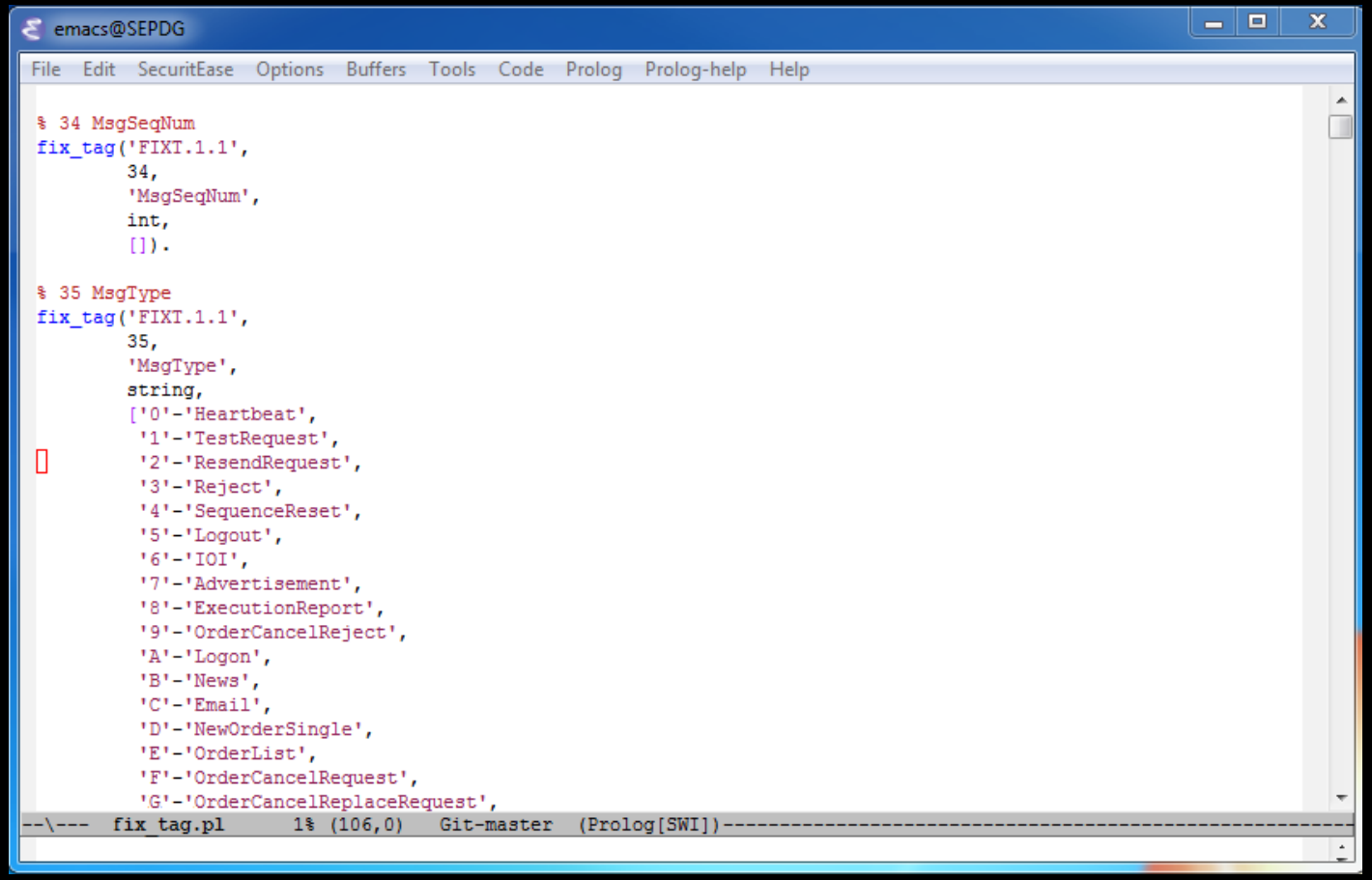
Prolog at the Front End

- GPJ - GNU Prolog for Java

```
quote_cell_formatter(BuySell, Result) ::-
        ( BuySell == 'B' ->
            Result = [background-'UIColour'(0,0,255)]
        ; BuySell == 'S' ->
            Result = [background-'UIColour'(255,0,0)]
        ).
```


Data Driven Interface Development

Capture interface specification as facts



Data Driven Interface Development

- Worry about interpretation of the facts later
- Develop interface engines incrementally

Result

Application programmers have to know basic Prolog concepts

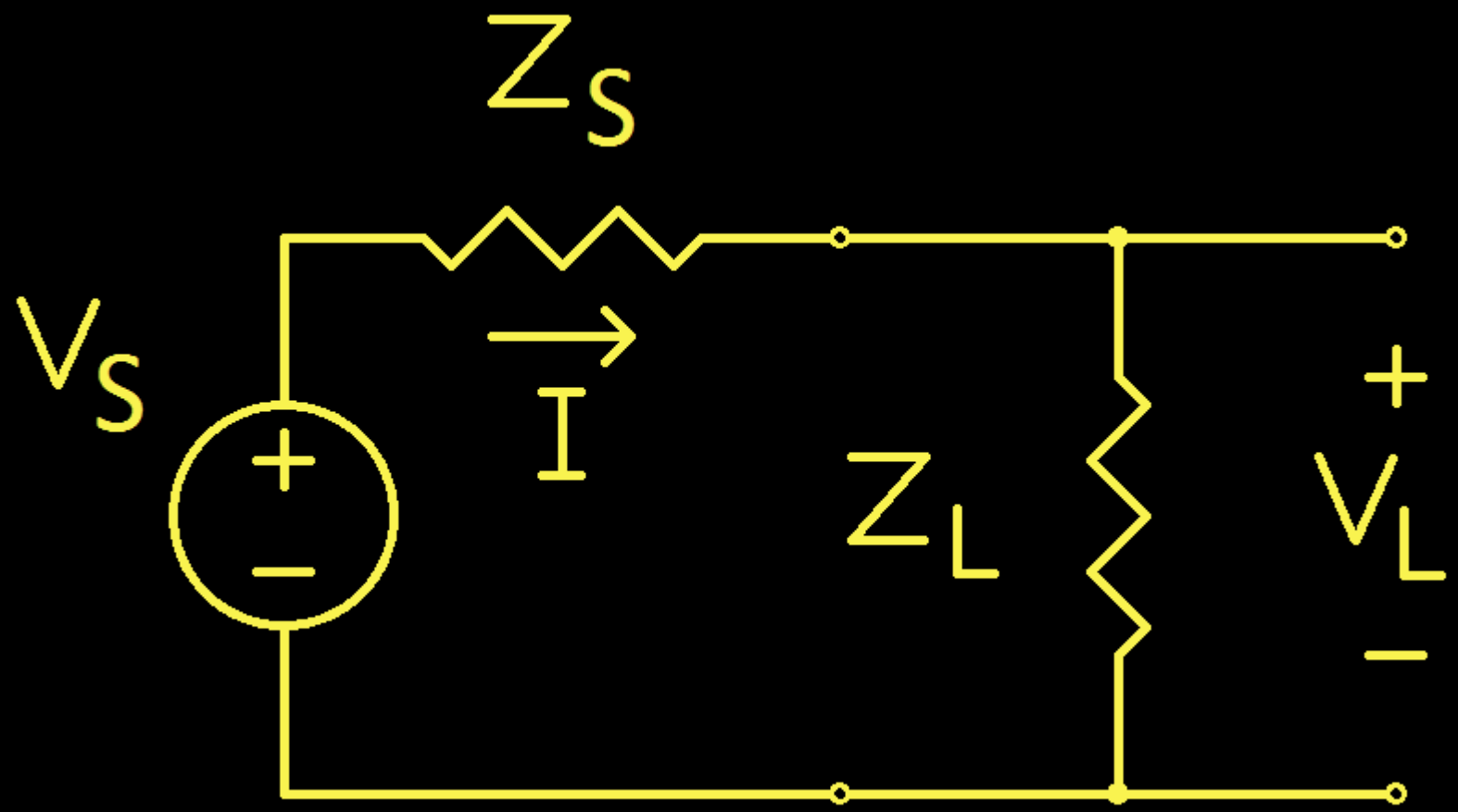
- syntax
- logical variables
- unification
- recursive list processing
- non-determinism
- setof/3 and friends

but don't have to be Prolog experts

But there is hope

- Its an object oriented world, but ...
- Finance is built on relational databases

Impedance Mismatch



Impedance Mismatch

- According to Wikipedia

  A set of conceptual and technical difficulties when an RDBMS is being used by a program written in an object-oriented programming language.

- Not qualified to comment on the various workarounds and philosophical discussions

- There are many!

- With Prolog no such problems

Impedance Match

- In Prolog
  - Tables = Predicates
  - Success = Commit
  - Failure = Rollback
  - Exception = Rollback
- Great benefit when majority of processing is RDBMS operations

Data Type Mapping

Database	Prolog	Notes
String	Atom	Case sensitive collation used
Int	Integer	
Decimal	rdiv/2	GMP infinite precision for all monetary amounts. $12,345,678,901.90 + $0.01
Timestamp	t7(Y, M, D, H, Min, S, Ms)	t7(Y, M, D, 0, 0, 0, 0) for dates
Bit	boolean(true) boolean(false)	
Blob	Atom	Exploits atom GC to clean up
null	{null}	

Prolog as a source code tool

- library(pio)
  - " ...processing input streams from the outside world using pure predicates, notably grammar rules (DCG)"
- Used to reformat entire codebase in a few minutes

library(pio)

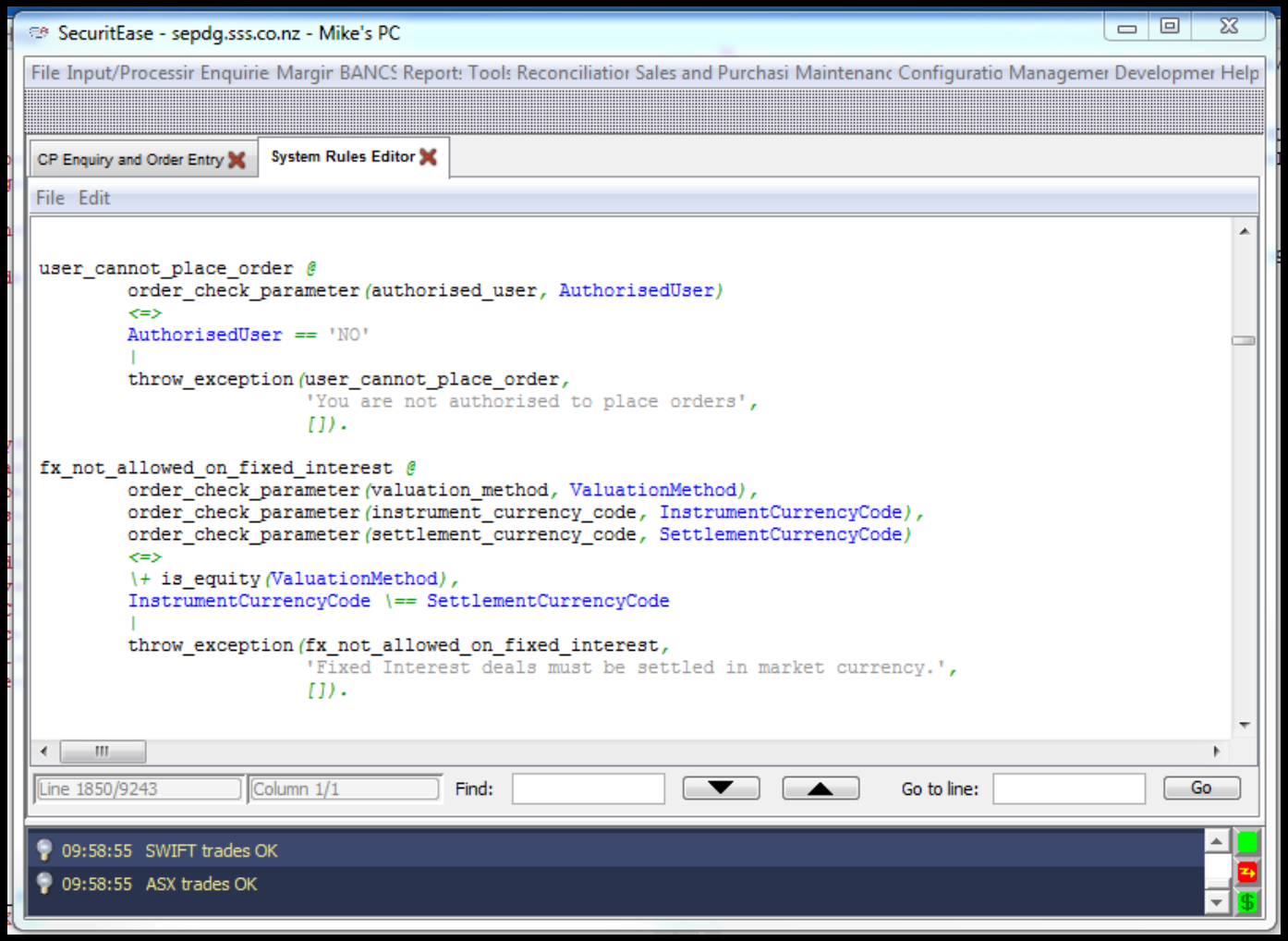
```
%%      convert_domain_constants
%
%       Convert Domain.DOM_XXX to "DOM_XXX" in every Java file

map(Stream) -->
        "Domain.", domain(DomainCodes), !,
        {format(Stream, '"~s"', [DomainCodes]),
        atom_codes(DomainMC, DomainCodes),
        downcase_atom(DomainMC, Domain),
        ( domain(Domain) ->
              true
          ; otherwise ->
              console('Warning: Unknown domain: ~w~n', [Domain])
          )},
        map(Stream).
map(Stream) -->[C], !, {format(Stream, '~c', [C])}, map(Stream).
map(_) --> [].
```

System Rules

- Load site specific code at run time
- Achieve more flexibility than is possible with configuration tables
- Special purpose editor

System Rules Editor



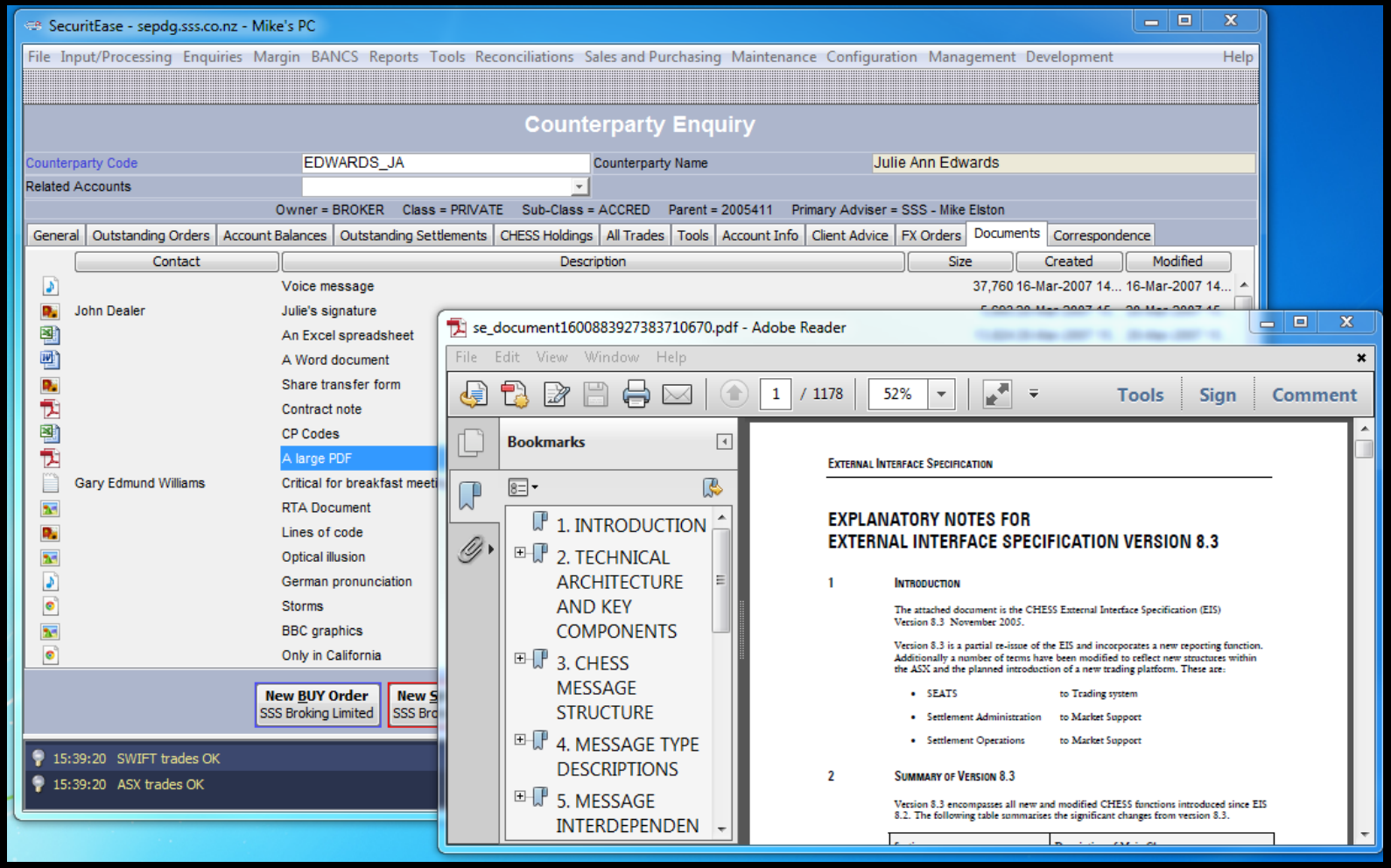
Hotfixes

- Part of SWI Prolog
- System does not need to be shut down
- Patches load:
  - On start up
  - On demand

Document Management System

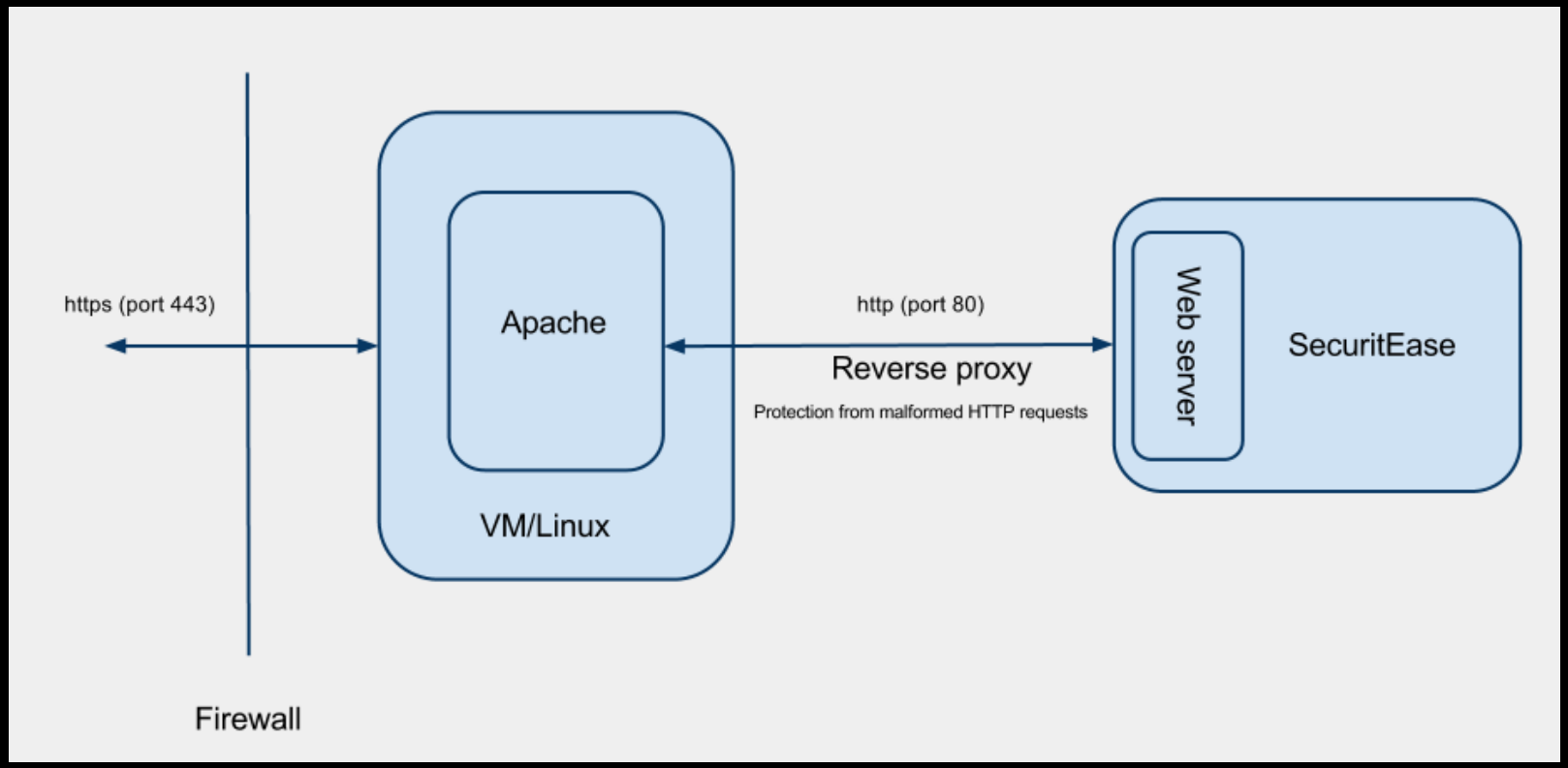
- Drag and Drop
- Large documents handled in Prolog
  - Streams
  - Avoid large atoms

Document Management System

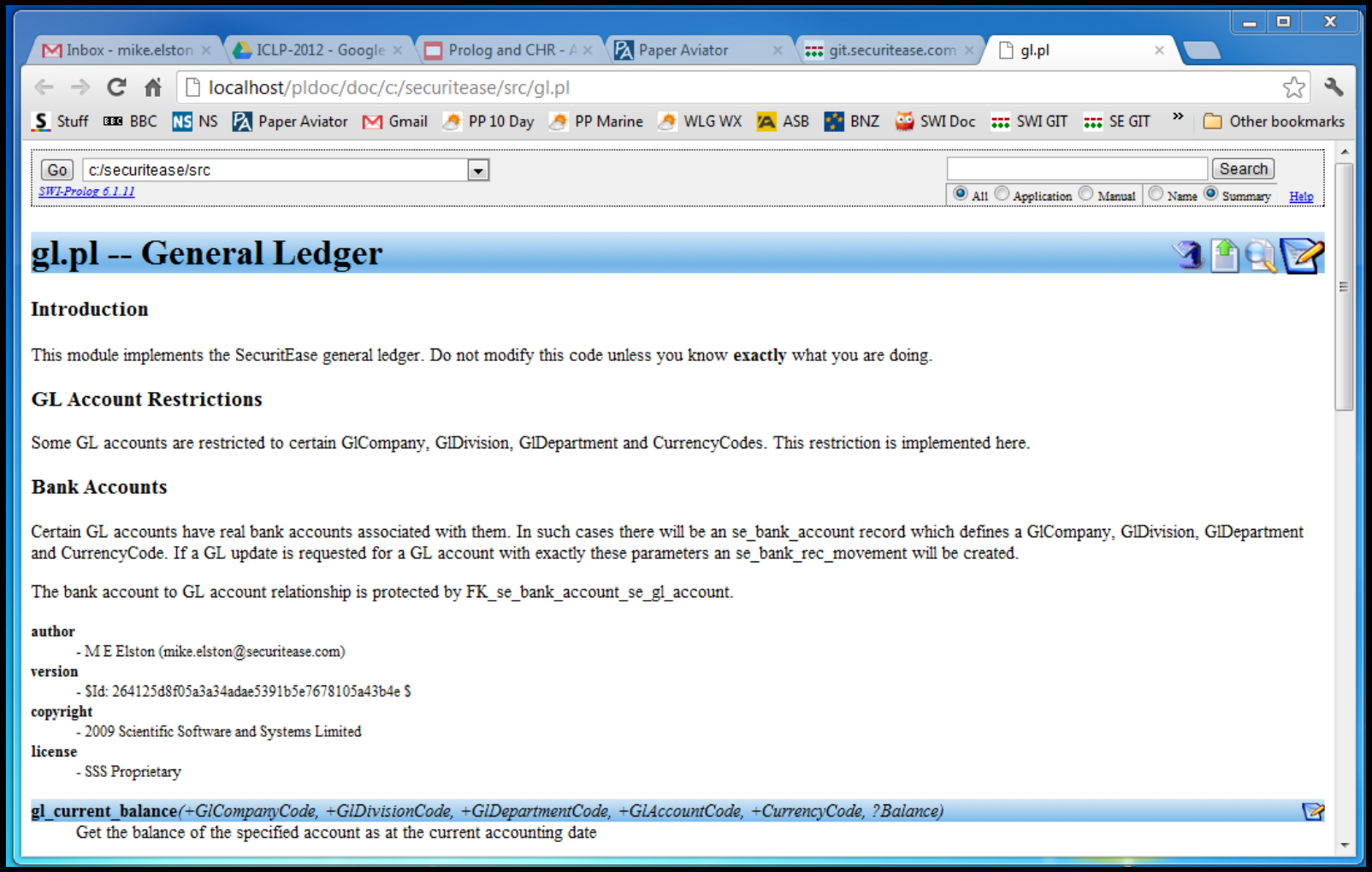


The Web

- PWP via reverse proxy
- Used to collect client profile data



Literate Programming - pldoc



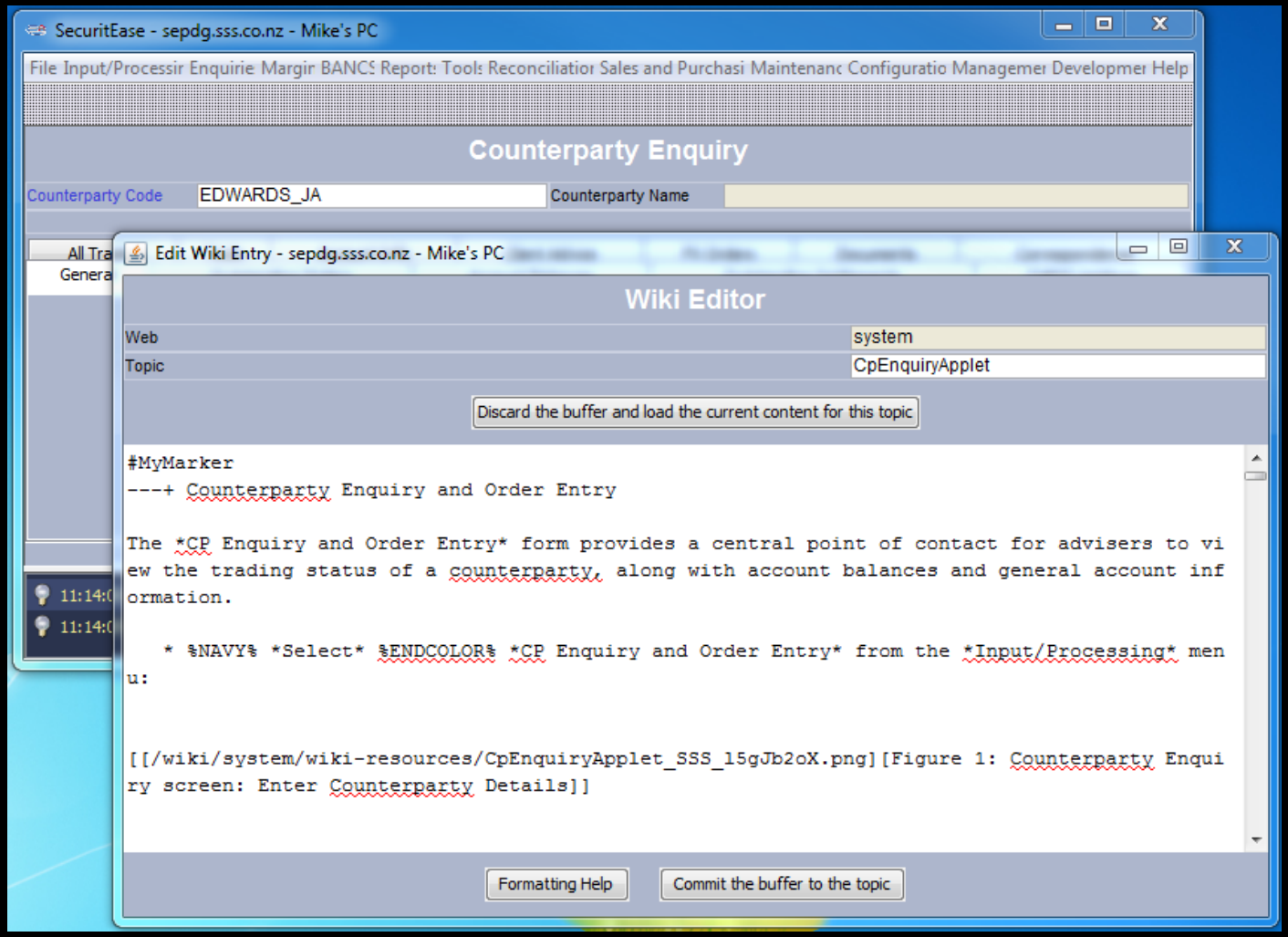
Unit Testing - plunit

- Nightly regression tests
- Goal expansion to generate coverage tests

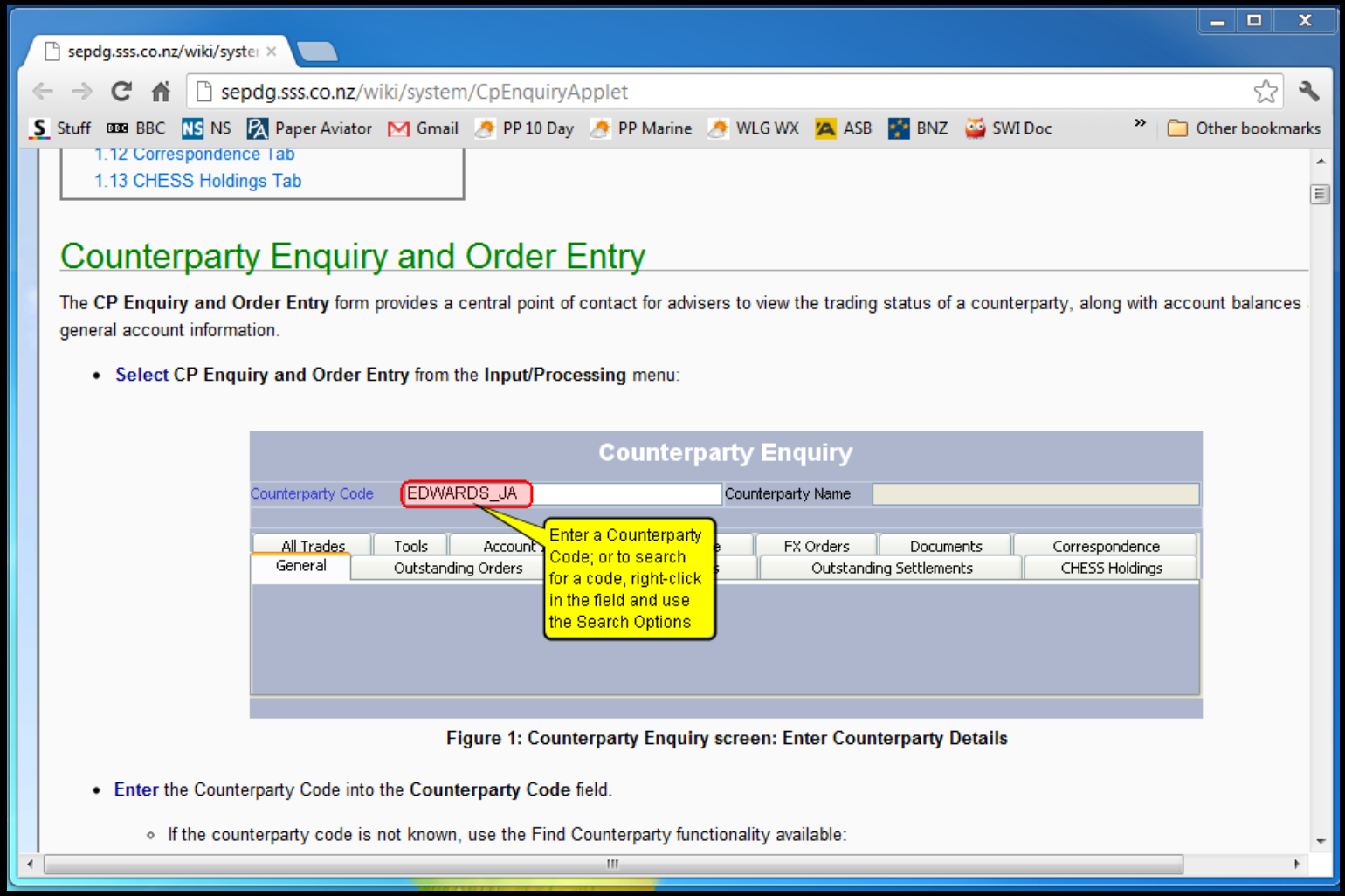
User Documentation

- Right click on the title of any form to
  - Write help text in twiki syntax
  - Capture and annotate screen shots
- Back end entirely in Prolog

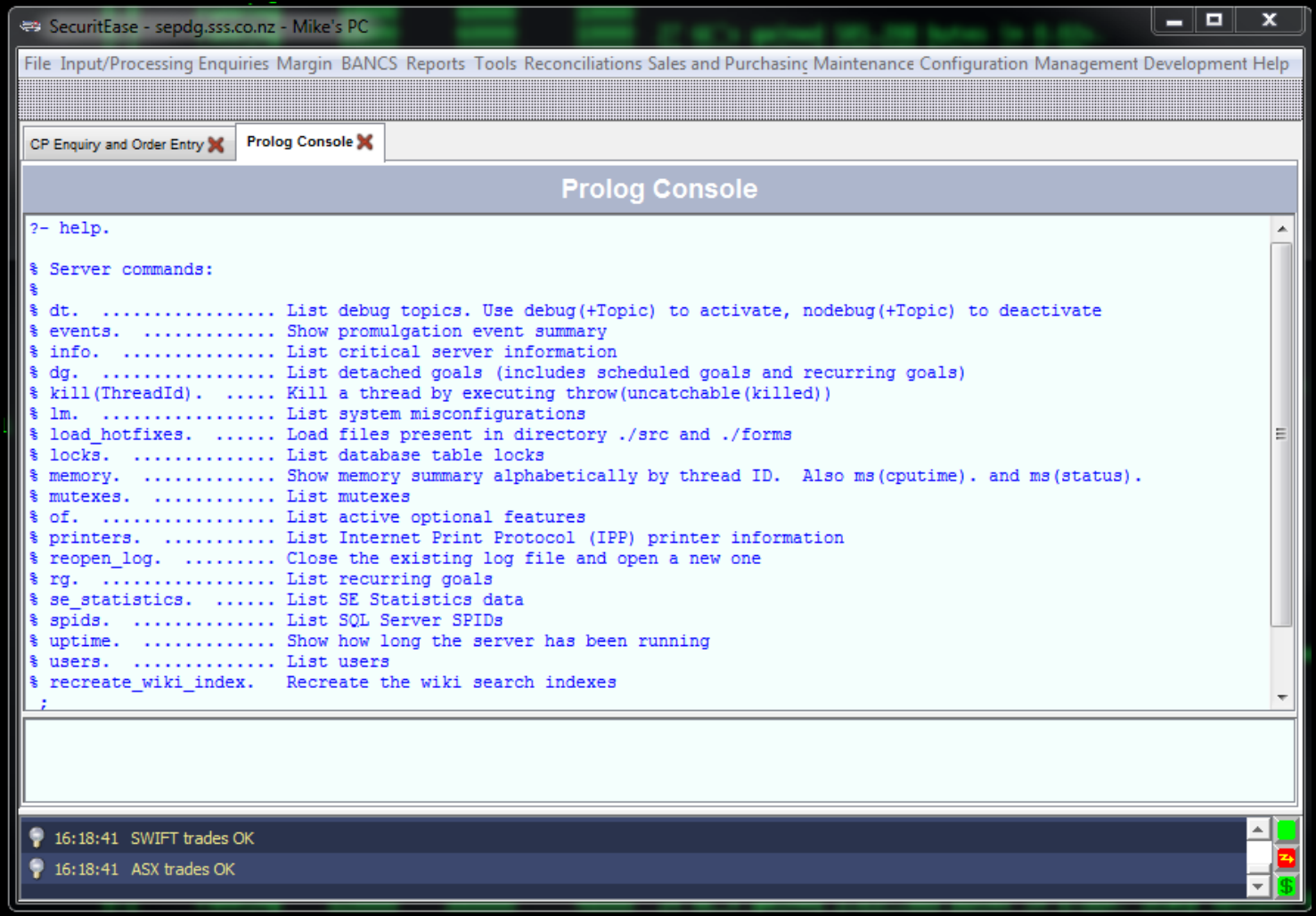
User Documentation - Wiki Editor



User Documentation - Browser



Console Comes for Free



Prolog Database

- Mostly the Prolog database is not used
- Except where speed is critical
  - Market Depth
  - Database table caching

Ports

Experimentally ported from Windows to:

- OSX™
- Linux
- PostgreSQL

Security

- AES
  - Stored data encryption
  - Client-server message encryption
- SSL
  - Web server
- Kerberos
  - User authentication
- Independent security analysis
  - entire system by major bank
  - web interface by major stock broker

Prolog Reasoning at Every Level

- Application logic
- Database metadata
- Interfaces
- Unit tests
- User interface

"Reason in the application
and
about the application"

The Big Question

- If you're so smart, why aren't you rich?

  Why Prolog? Justifying Logic Programming for Practical Applications. G. Lazarev, Prentice Hall, 1989

- Lazarev's answer

  Prolog is rich in features and possibilities

- 2012 answer

  Prolog + CHR have allowed a new business to grow in a harsh economic climate despite well-entrenched competitors

Where are the Prolog applications?

?-

From Possible to Practical

- 1962
  Robert N. Hall demonstrates the first laser diode
- 1982
  Sony releases the CDP-101, the first affordable Compact Disc player
- 20 year wait

From Possible to Practical

- 1936
  Hungarian engineer Kálmán Tihanyi describes the principle of "plasma television" and conceives the first flat-panel display system.
- 1997
  Fujitsu and Philips sell first "affordable" plasma TV displays
- 61 year wait

Where are the Prolog applications?

- 1972
  First Prolog system
- 2004
  SecuritEase project becomes financially self sustaining
- 32 year wait

Big Business Clients

- Pro-what?
  - Technical evaluation staff
    "I did an AI paper years ago and thought Prolog was cool"
- Company reputation more important
  - Reference sites
  - First customer is the hardest one to get
- Escrow
  - Except for the *Clear Reports* report writer SecuritEase is compilable with open source tools

The Competition

- Large brokerage has twice before attempted to change their broking system
- Millions of euros per attempt
- So far, so good

Software Engineering

- The smaller the team, the faster the progress
  - Productive tools required
- Fewer languages means fewer internal interfaces
- Lower the barriers to understanding of domain experts

Challenges

- Reliability
- Training
  - Particularly CHR
- Refactoring

If you want it, pay for it!

- GMP
- SWI Prolog stack shifter
- SSL
- CHR performance improvements
- library(aggregate)

Mistakes

- Name-Value pair lists
  - Concise
  - Compiler can't help detect silly mistakes
- Unwanted choice points
  - Probably better if we had to declare non-deterministic predicates

Where has Prolog worked best?

- Platform for database and GUI server tools
- General application logic

Where has CHR worked best?

- Tools
  - CQL compiler
  - SWIF forms compiler
- Why?
  - Concise
  - Well suited to incremental development
  - Flexible, extensible definition of exception cases

Lessons

- Build application specific tools
- Use term and goal expansion

Take home messages

- Prolog and CHR work in the financial world
  - Productive
  - Flexible
  - Reliable enough
  - Fast enough
  - Secure enough
  - Integrated enough

What do we need?

- More static analysis
- Automatic refactoring
- But not much else

The Future

- Now we have a platform we can do interesting things
  - reasoning about the market
  - reasoning about the clients
  - realise the promise of Prolog and CHR

