

Prolog and CHR in Finance

An easier way to occupy Wall Street?

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

SecuritEase - sepdg.sss.co.nz - Mike's PC

File Input/Processing Enquiries Margin BANCS Reports Tools Reconciliations Sales and Purchasing Maintenance Configuration Management Development Help

Counterparty Enquiry

Counterparty Code: EDWARDS_JA Counterparty Name: Julie Ann Edwards

Related Accounts: [dropdown]

Owner = BROKER Class = PRIVATE Sub-Class = ACCRED Parent = 2005411 Primary Adviser = SSS - Mike Elston

Date	Instrument	Market	Currency	Quantity	Q. Outstanding	Q. Pending	Buy/Sell	Price	Yield	Order Status	Order Reference	Account ID
02-Jan-2009	AAA	NZX	NZD	612	597	15	BUY			FILLED	8869	01
19-Mar-2009	PPP	ASX	NZD	1,000	0	1,000	BUY	1.000000000		FILLED	9247	01
22-Apr-2009	TEL	NZX	NZD	1,000	0	1,000	SELL	5.300000000		FILLED	9420	01
22-Apr-2009	TEL	NZX	NZD	1,000	0	1,000	BUY	5.000000000		FILLED	9421	01
22-Apr-2010	TEL	ASX	AUD	77	0	77	BUY	3.680000000		FILLED	13893	01
22-Apr-2010	TEL	ASX	AUD	2	0	2	BUY	3.680000000		FILLED	13939	01
22-Apr-2010	AAA	NZX	NZD	1,578	593	985	BUY	0.005000000		FILLED	BULK	01
19-Feb-2007	TEL	ASX	AUD	1,000	1,000		BUY			CONFIRMED	11030	01
19-Feb-2007	TEL	ASX	AUD	1,000	1,000		BUY			CONFIRMED	11031	01
17-Nov-2008	TEL	ASX	AUD	1	1		BUY	3.680000000		PLACED	7924	01
08-Dec-2008	TEL	ASX	AUD	1,000	1,000		BUY			CONFIRMED	8018	01
02-Jan-2009	BHP	ASX	AUD	100	100		BUY	41.500000000		PLACED	8874	01
02-Jan-2009	BHP	ASX	AUD	400	400		SELL	41.000000000		PLACED	8877	01
20-Feb-2009	TEL	NZX	AUD	4,000	4,000		BUY	5.000000000		PLACED	1427	01
20-Feb-2009	TEL	NZX	AUD	3,800	3,800		SELL	5.000000000		PLACED	1428	01
18-Jun-2009	AIZ	ASX	NZD	1,000	1,000		SELL	1.750000000		CONFIRMED	11046	01

New BUY Order
SSS Broking Limited

New SELL Order
SSS Broking Limited

FX Order

Managed Fund New Application

Change History Report

Manage Subscriptions

Refresh Tables

15:47:35 SWIFT trades OK

15:47:35 ASX trades OK

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

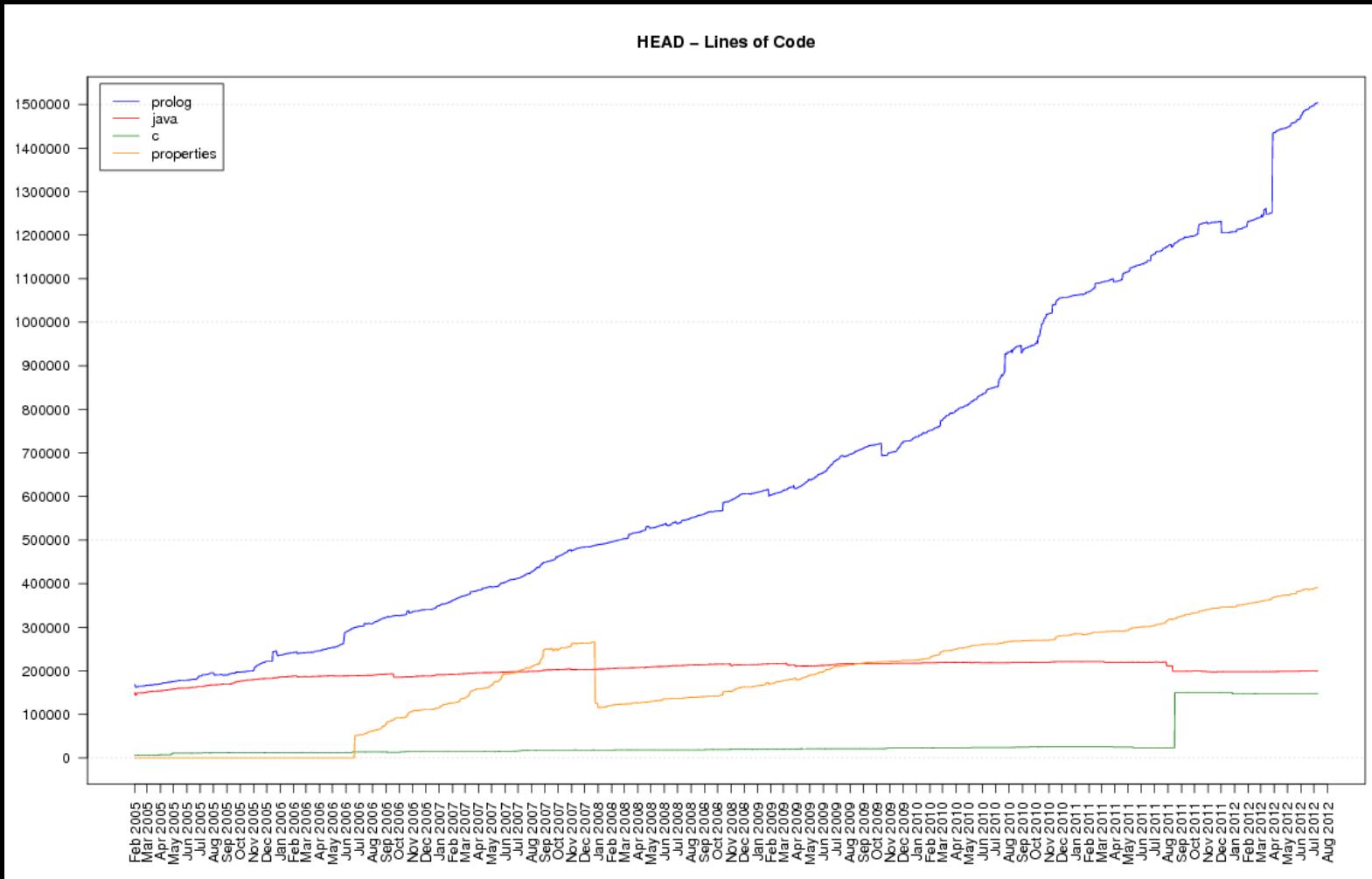
Evolution

- 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).
```

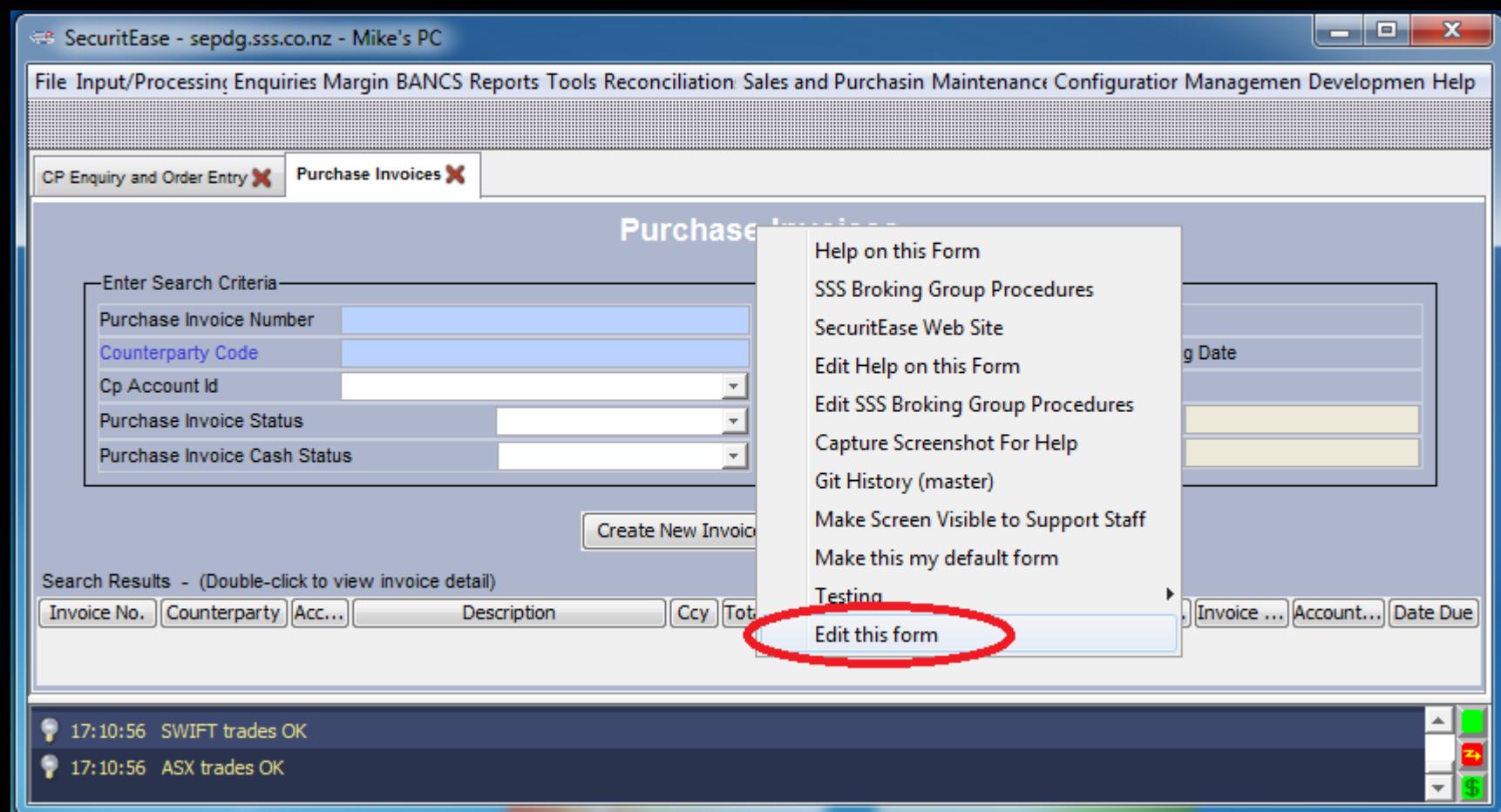
The "Long String" Problem Solved

```
/* $SQL$  
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 )  
*/
```

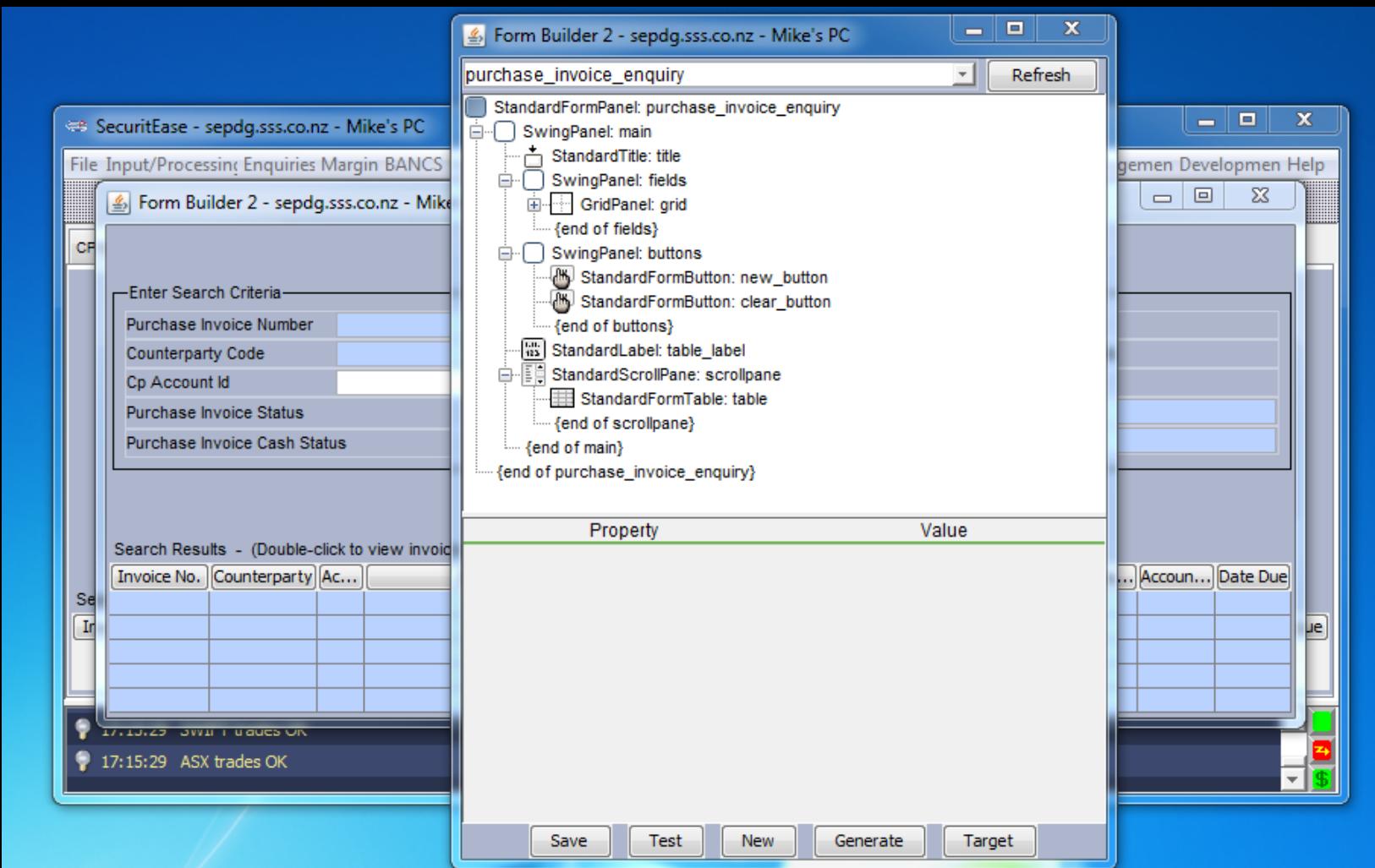
CQL details

- 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

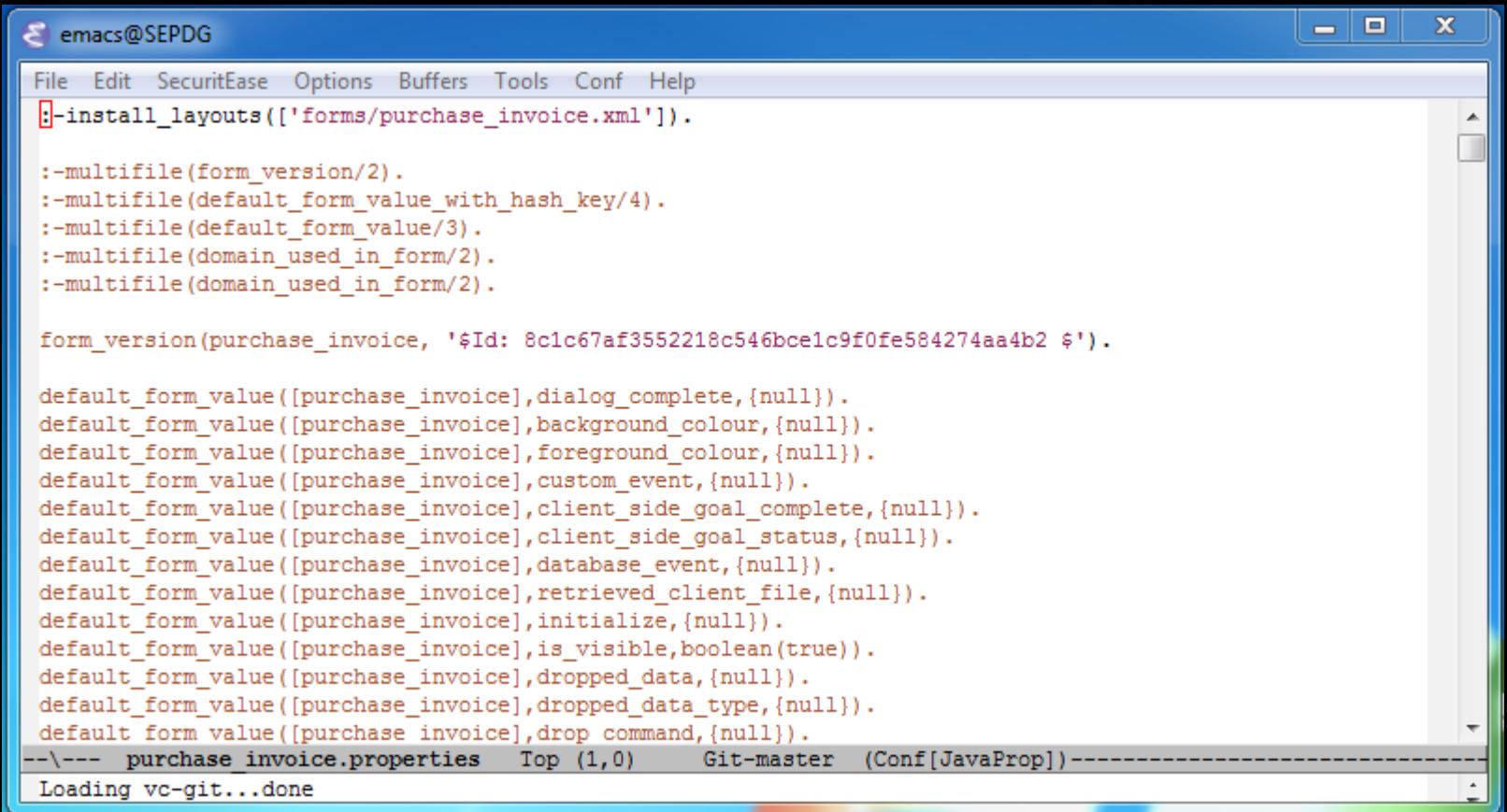
SWIF - SWI Forms



Form Builder



Form Builder



The screenshot shows an Emacs window with a blue title bar and a white buffer area. The title bar contains the text "emacs@SEPDG". The buffer area displays the following Prolog-like code:

```
:install_layouts(['forms/purchase_invoice.xml']).  
  
:-multifile(form_version/2).  
:-multifile(default_form_value_with_hash_key/4).  
:-multifile(default_form_value/3).  
:-multifile(domain_used_in_form/2).  
:-multifile(domain_used_in_form/2).  
  
form_version(purchase_invoice, '$Id: 8c1c67af3552218c546bc1c9f0fe584274aa4b2 $').  
  
default_form_value([purchase_invoice], dialog_complete, {null}).  
default_form_value([purchase_invoice], background_colour, {null}).  
default_form_value([purchase_invoice], foreground_colour, {null}).  
default_form_value([purchase_invoice], custom_event, {null}).  
default_form_value([purchase_invoice], client_side_goal_complete, {null}).  
default_form_value([purchase_invoice], client_side_goal_status, {null}).  
default_form_value([purchase_invoice], database_event, {null}).  
default_form_value([purchase_invoice], retrieved_client_file, {null}).  
default_form_value([purchase_invoice], initialize, {null}).  
default_form_value([purchase_invoice], is_visible, boolean(true)).  
default_form_value([purchase_invoice], dropped_data, {null}).  
default_form_value([purchase_invoice], dropped_data_type, {null}).  
default_form_value([purchase invoice], drop_command, {null}).  
--\--- purchase_invoice.properties Top (1,0) Git-master (Conf[JavaProp])-----  
Loading vc-git...done
```

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

```
Administrator: Command Prompt - bin\securitease.exe se_mike securitease staircase --ansi --check-dbmd

% fx_convert_trade_fee [trade_entry_chr_charges]
% + [trade_entry_chr,right] [trade_fee_itc, value] <null>
% + [trade_entry_chr,right] [trade_currency_code, value] <NZD>
% + [trade_entry_chr,right] [settlement_currency_code, allowed_values] <null>
% + [trade_entry_chr,right] [settlement_currency_code, value] <null>
% + [trade_entry_chr,right] [base_currency_code, allowed_values] <null>
% + [trade_entry_chr,right] [base_currency_code, value] <null>
% + [trade_entry_chr,right] [exchange_rate, value] <1>
% - [trade_entry_chr,right] [trade_fee_isc, value] <null>
%
% calculate_total_settlement_value_isc [trade_entry_chr_charges]
% + [trade_entry_chr,right] [trade_value_isc, value] <null>
% + [trade_entry_chr,right] [trade_fee_isc, value] <null>
% + [trade_entry_chr,right] [allow_negative_settlement_value, value] <false>
% + [trade_entry_chr,right] [application_money_isc, value] <null>
% + [trade_entry_chr,right] [brokerage_income_isc, value] <null>
% + [trade_entry_chr,right] [total_other_charge_isc, value] <0>
% + [trade_entry_chr,right] [total_settlement_value_itc, value] <null>
% + [trade_entry_chr,right] [buy_sell_indicator, value] <null> # passive
% - [trade_entry_chr,right] [total_settlement_value_isc, value] <null>
%
% fx_convert_sales_credit [trade_entry_chr_charges]
% + [trade_entry_chr,right] [sales_credit_itc, value] <null>
% + [trade_entry_chr,right] [trade_currency_code, value] <NZD>
% + [trade_entry_chr,right] [settlement_currency_code, allowed_values] <null>
% + [trade_entry_chr,right] [settlement_currency_code, value] <null>
% + [trade_entry_chr,right] [base_currency_code, allowed_values] <null>
% + [trade_entry_chr,right] [base_currency_code, value] <null>
% + [trade_entry_chr,right] [exchange_rate, value] <1>
% - [trade_entry_chr,right] [sales_credit_isc, value] <null>
```

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

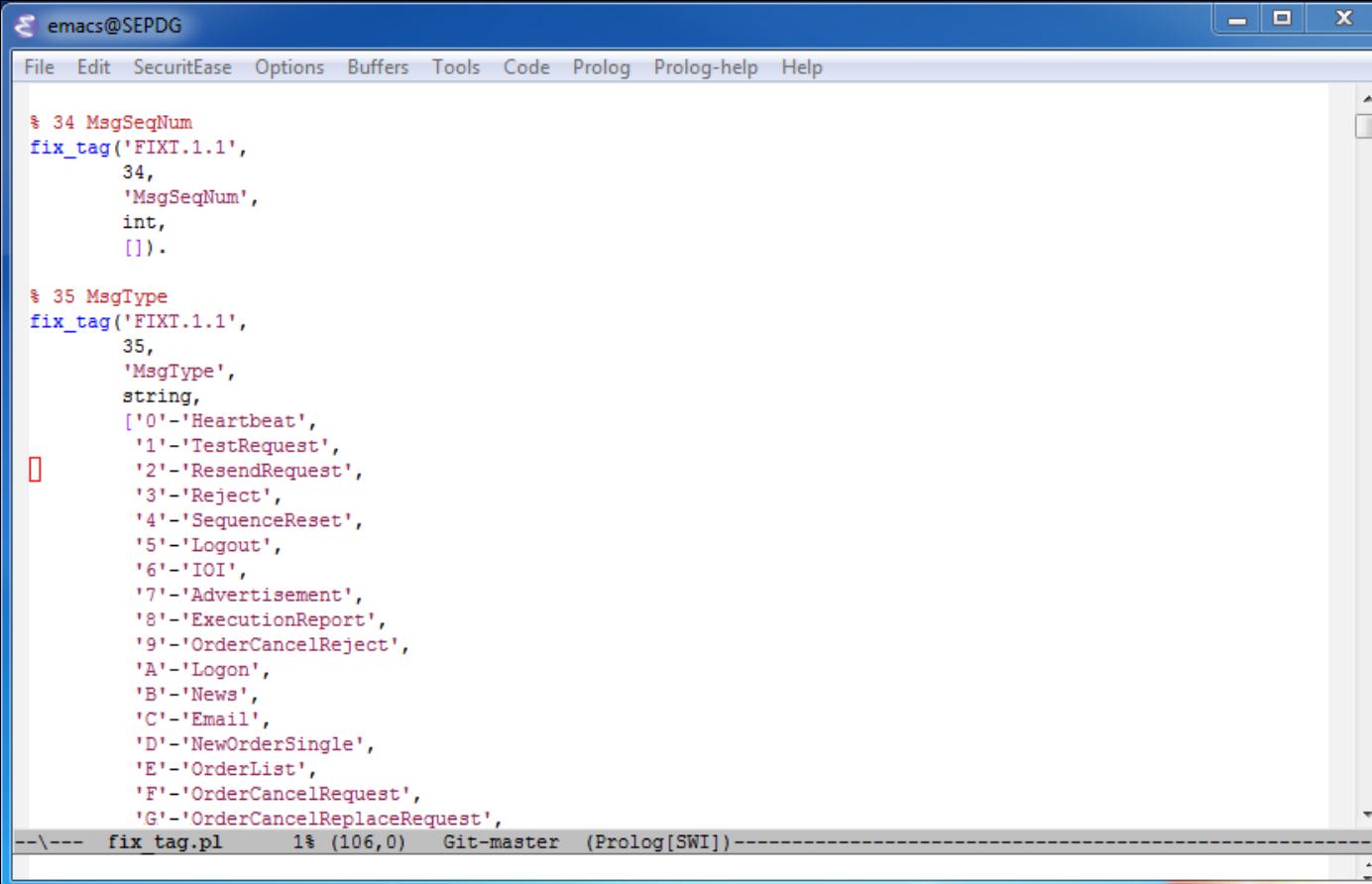
Prolog at the Front End

- 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



The screenshot shows an Emacs window titled "emacs@SEPDG" displaying Prolog code. The code defines two facts, fix_tag/2 and fix_tag/3, which map integer and string values to FIXT message tags. The fix_tag/2 fact maps integers 34 and 35 to 'MsgSeqNum' and 'MsgType' respectively. The fix_tag/3 fact maps strings from '0' to 'G' to their corresponding FIXT message tags. A red rectangular box highlights the first few entries of the fix_tag/3 fact.

```
% 34 MsgSeqNum
fix_tag('FIXT.1.1',
        34,
        'MsgSeqNum',
        int,
        []).

% 35 MsgType
fix_tag('FIXT.1.1',
        35,
        'MsgType',
        string,
        ['0'-'Heartbeat',
         '1'-'TestRequest',
         '2'-'ResendRequest',
         '3'-'Reject',
         '4'-'SequenceReset',
         '5'-'Logout',
         '6'-'IOI',
         '7'-'Advertisement',
         '8'-'ExecutionReport',
         '9'-'OrderCancelReject',
         'A'-'Logon',
         'B'-'News',
         'C'-'Email',
         'D'-'NewOrderSingle',
         'E'-'OrderList',
         'F'-'OrderCancelRequest',
         'G'-'OrderCancelReplaceRequest',
         ]).

--\--- fix tag.pl    1% (106,0)  Git-master  (Prolog[SWI])---
```

Data Driven Interface Development

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

Result

Application programmers have to know basic Prolog concepts

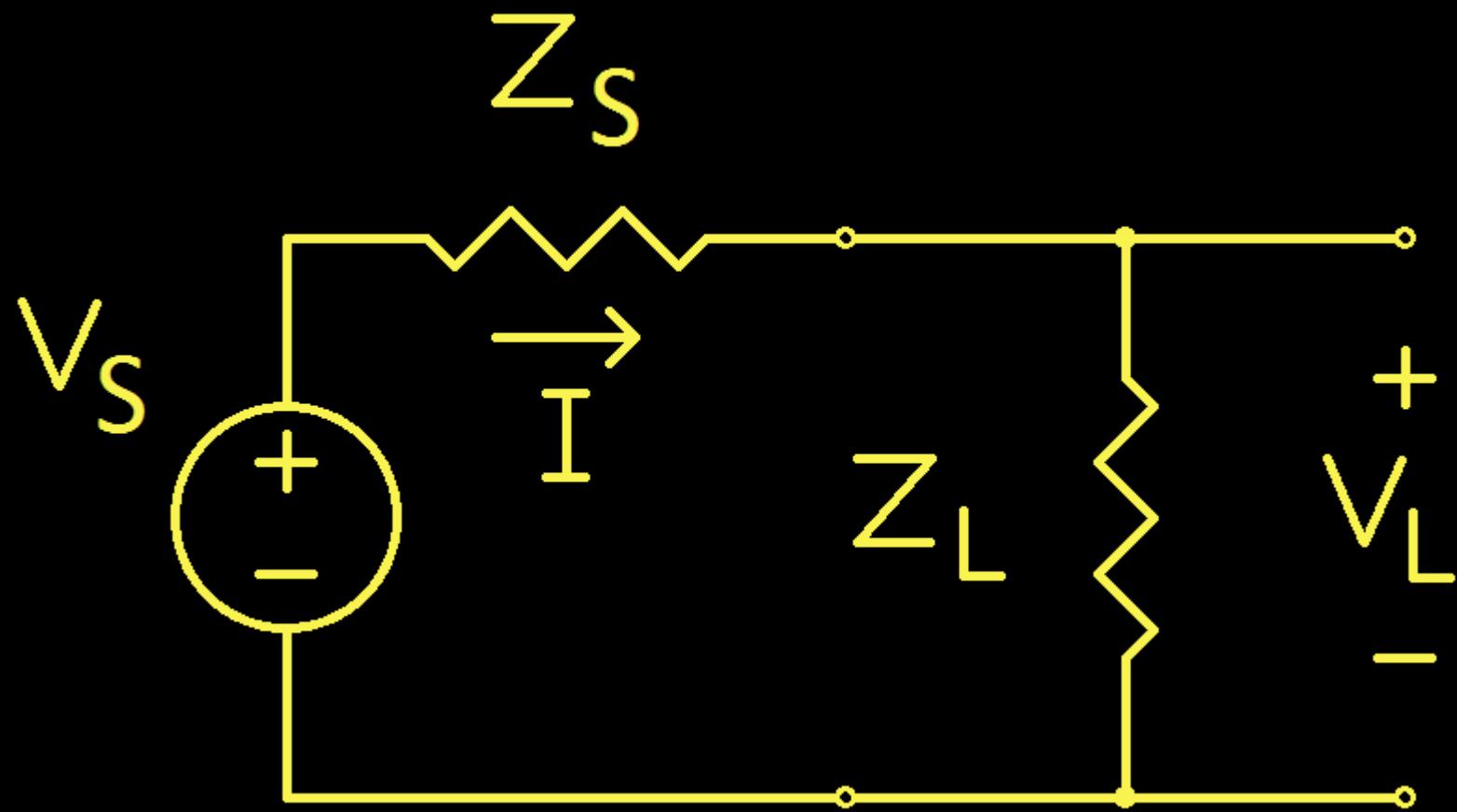
- syntax
- logical variables
- unification
- recursive list processing
- non-determinism
- setof/3 and friends

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

The screenshot shows a window titled "SecuritEase - sepdg.sss.co.nz - Mike's PC". The window has tabs for "CP Enquiry and Order Entry" and "System Rules Editor", with "System Rules Editor" being active. The main area contains the following Groovy-like code:

```
user_cannot_place_order @
    order_check_parameter(authorised_user, AuthorisedUser)
    <=>
    AuthorisedUser == 'NO'
    |
    throw_exception(user_cannot_place_order,
                    'You are not authorised to place orders',
                    []).

fx_not_allowed_on_fixed_interest @
    order_check_parameter(valuation_method, ValuationMethod),
    order_check_parameter(instrument_currency_code, InstrumentCurrencyCode),
    order_check_parameter(settlement_currency_code, SettlementCurrencyCode)
    <=>
    \+ is_equity(ValuationMethod),
    InstrumentCurrencyCode \== SettlementCurrencyCode
    |
    throw_exception(fx_not_allowed_on_fixed_interest,
                    'Fixed Interest deals must be settled in market currency.',
                    []).
```

The status bar at the bottom shows two log entries: "09:58:55 SWIFT trades OK" and "09:58:55 ASX trades OK".

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

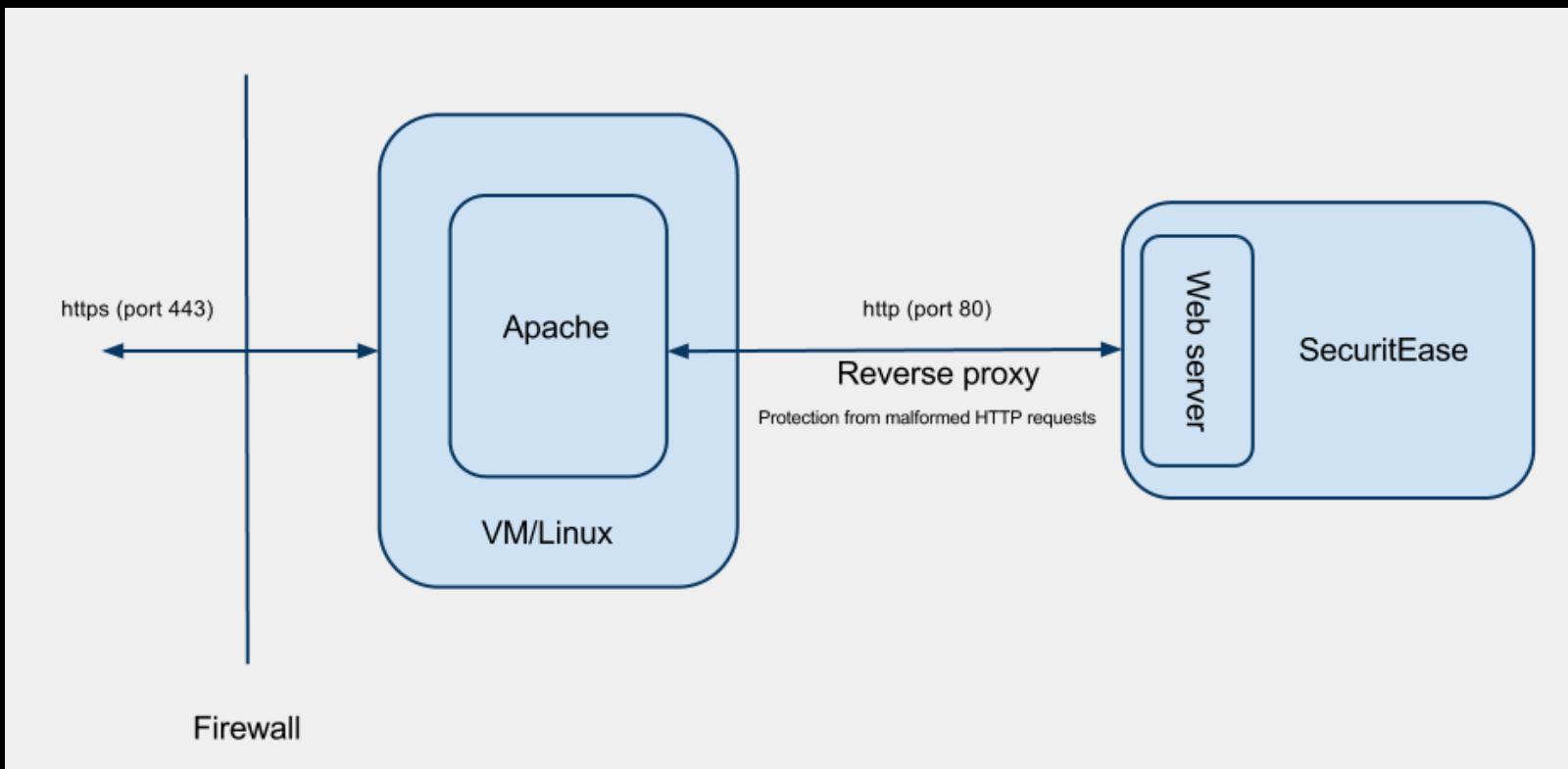
Document Management System

The screenshot illustrates a Document Management System interface. The main window is titled "Counterparty Enquiry" and displays details for a counterparty named Julie Ann Edwards. The counterparty code is EDWARDS_JA. The interface includes tabs for General, Outstanding Orders, Account Balances, Outstanding Settlements, CHESS Holdings, All Trades, Tools, Account Info, Client Advice, FX Orders, Documents, and Correspondence. The "Documents" tab is currently selected, showing a list of files including "A large PDF" which is highlighted. Below this list is a contact list for John Dealer and Gary Edmund Williams, each with a list of associated documents. The bottom of the main window shows two buttons: "New BUY Order" and "New SELL Order". A log at the bottom indicates successful trades: "15:39:20 SWIFT trades OK" and "15:39:20 ASX trades OK".

The second window, titled "se_document1600883927383710670.pdf - Adobe Reader", displays the contents of a PDF document. The document is titled "EXTERNAL INTERFACE SPECIFICATION" and contains "EXPLANATORY NOTES FOR EXTERNAL INTERFACE SPECIFICATION VERSION 8.3". It includes sections for "INTRODUCTION", "TECHNICAL ARCHITECTURE AND KEY COMPONENTS", "CHESS MESSAGE STRUCTURE", "MESSAGE TYPE DESCRIPTIONS", and "MESSAGE INTERDEPENDENCIES". The "INTRODUCTION" section notes that Version 8.3 is a partial re-issue of the EIS and incorporates a new reporting function, along with structural changes within the ASX and the introduction of a new trading platform. It also lists new reporting functions: SEATS to Trading system, Settlement Administration to Market Support, and Settlement Operations to Market Support.

The Web

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



Literate Programming - pldoc

The screenshot shows a web browser window with the URL `localhost/pldoc/doc/c/securitease/src/gl.pl`. The browser has several tabs open at the top, including "Inbox - mike.elston", "ICLP-2012 - Google", "Prolog and CHR - A", "Paper Aviator", "git.securitease.com", and "gl.pl". The main content area displays the `gl.pl -- General Ledger` documentation. The page includes sections for **Introduction**, **GL Account Restrictions**, and **Bank Accounts**. It also lists metadata such as author (M E Elston), version (\$Id: 264125d8f05a3a34adae5391b5e7678105a43b4 \$), copyright (2009 Scientific Software and Systems Limited), and license (SS Proprietary). At the bottom, there is a code snippet for `gl_current_balance` and a note about getting the balance of a specified account.

This module implements the SecuritEase general ledger. Do not modify this code unless you know **exactly** what you are doing.

GL Account Restrictions

Some GL accounts are restricted to certain GICompany, GIDivision, GIDepartment and CurrencyCodes. This restriction is implemented here.

Bank Accounts

Certain GL accounts have real bank accounts associated with them. In such cases there will be an `se_bank_account` record which defines a GICompany, GIDivision, GIDepartment and CurrencyCode. If a GL update is requested for a GL account with exactly these parameters an `se_bank_rec_movement` will be created.

The bank account to GL account relationship is protected by `FK_se_bank_account_se_gl_account`.

author
- M E Elston (mike.elston@securitease.com)
version
- \$Id: 264125d8f05a3a34adae5391b5e7678105a43b4 \$
copyright
- 2009 Scientific Software and Systems Limited
license
- SSS Proprietary

gl_current_balance(+GiCompanyId, +GiDivisionId, +GiDepartmentId, +GlAccountId, +CurrencyCode, ?Balance)

Get the balance of the specified account as at the current accounting date

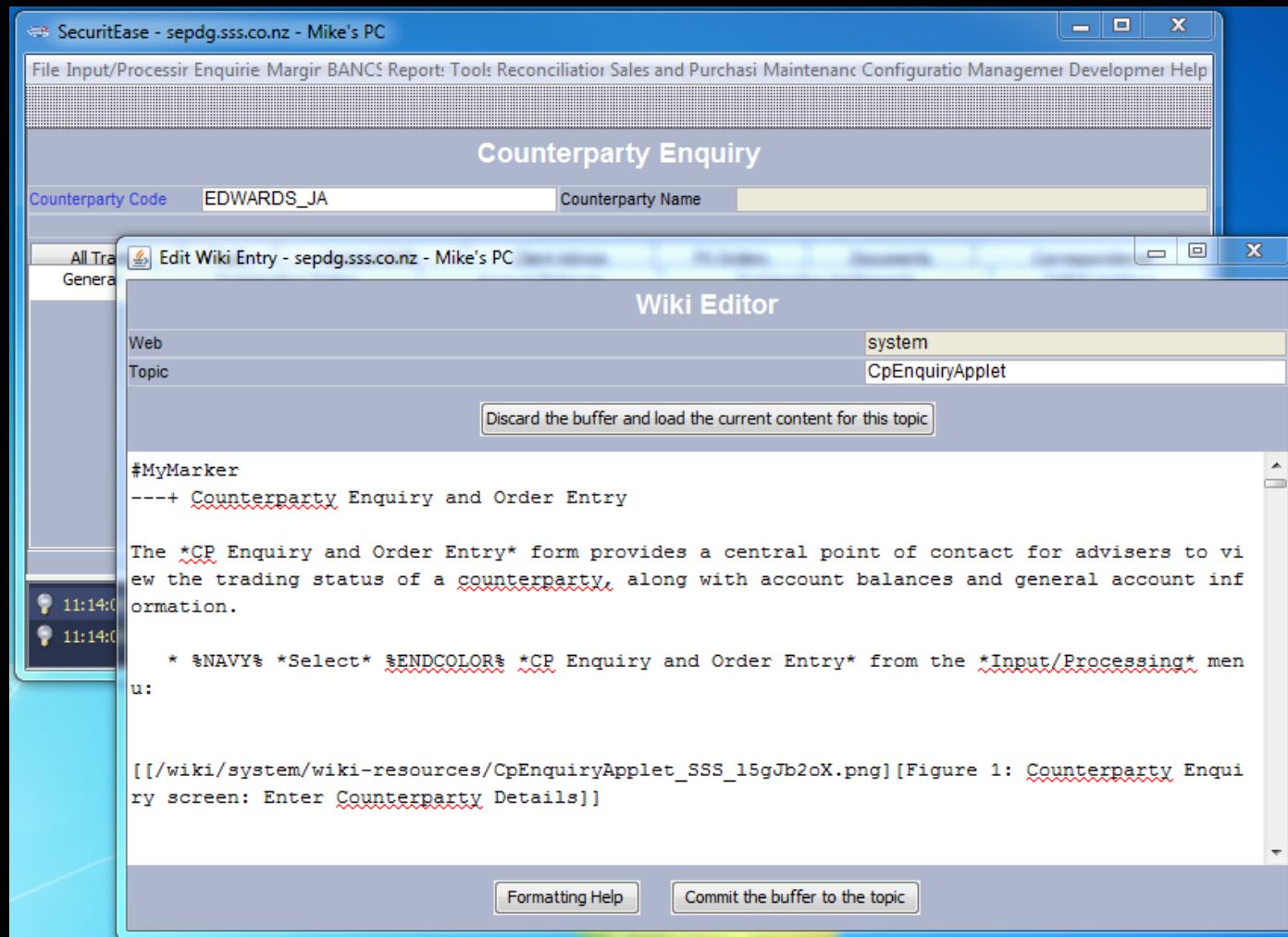
Unit Testing - plunit

- 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

The screenshot shows a web browser window with a blue header bar. The address bar contains the URL sepdg.sss.co.nz/wiki/system/CpEnquiryApplet. Below the address bar is a toolbar with various icons and links. The main content area displays a green header "Counterparty Enquiry and Order Entry". Below the header is a paragraph of text: "The CP Enquiry and Order Entry form provides a central point of contact for advisers to view the trading status of a counterparty, along with account balances general account information." A bulleted list follows: "Select CP Enquiry and Order Entry from the Input/Processing menu".

Counterparty Enquiry

Counterparty Code **EDWARDS_JA** Counterparty Name

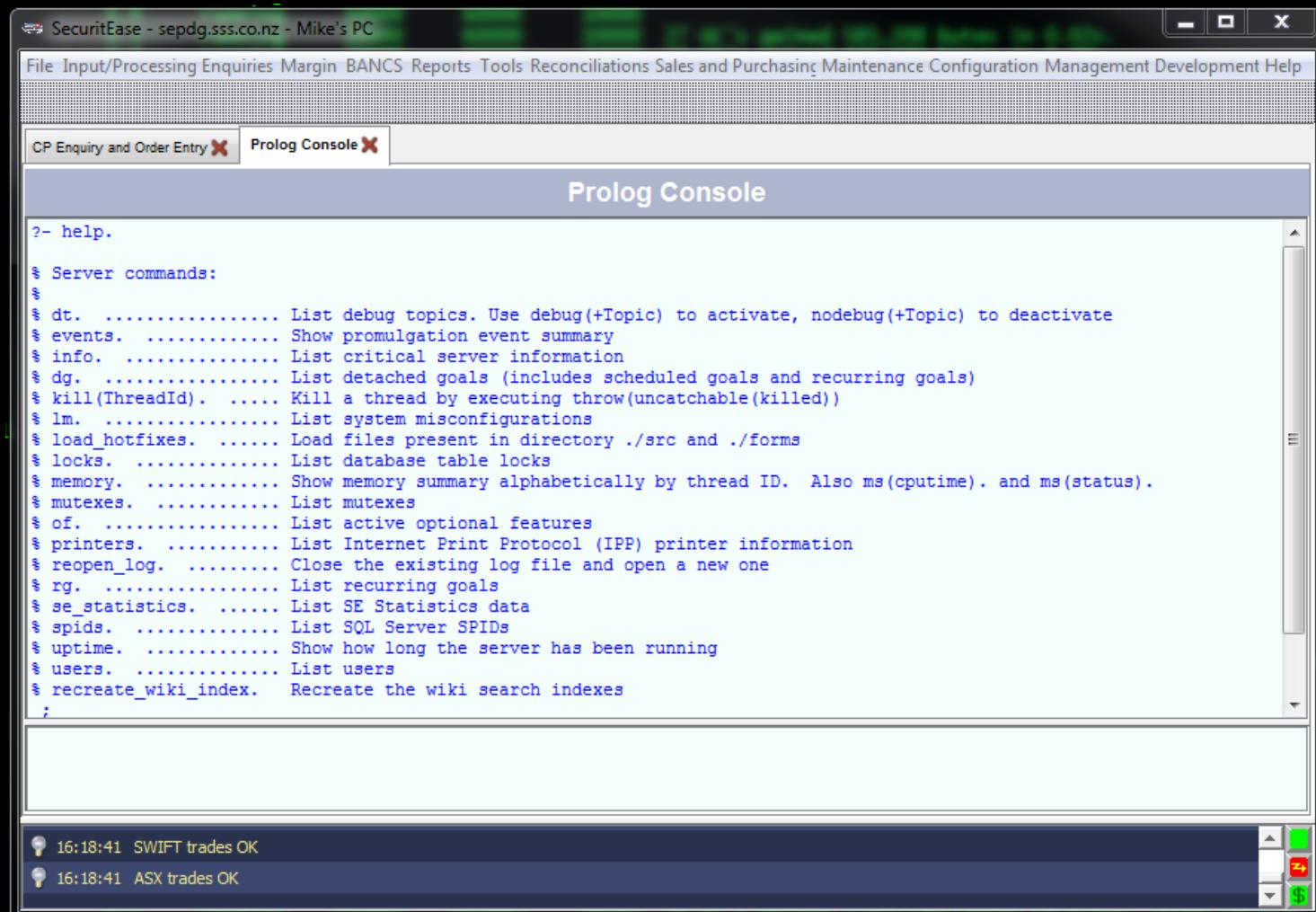
All Trades Tools Accounts
General Outstanding Orders FX Orders Documents Correspondence
Outstanding Settlements CHESS Holdings

Enter a Counterparty Code; or to search for a code, right-click in the field and use the Search Options

Figure 1: Counterparty Enquiry screen: Enter Counterparty Details

- Enter the Counterparty Code into the Counterparty Code field.
 - If the counterparty code is not known, use the Find Counterparty functionality available:

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