using QuantLib from F#

Integrating quant libraries with F#

Cepheis

Introduction

- Background
- Tech-debt
- Thinking Functional
- Cephei
- Demo

Quantlib background

- It is Quantlib free/open-source (http://quantlib.org/index.shtml)
- Originated & sponsored by StatPro in Italy
- Quantlib is widely copied by quant-groups in some large investment banks
- Quantlib has wide Instrument coverage
- Quantlib has many wrappers mostly using SWIG + Excel & R

Quant library background

- Huge libraries developed over decades to cover wide range of quantitatively priced products in C++
  - Often using libraries only available to C++/Fortran (Nag, MKL, cuBLAS), often optimised for SIMD
- Always provided to Traders as Excel XLL addins
- Often have threading issues through cache
- Application API often very different from XLL

tech-debt

- Spreadsheets great for developing products
  - With Bloomberg, Reuters etc ticking market data
- Spreadsheets not good for
  - reliable price quoting
  - Monte carlo risk with thousands of paths
  - Multiple perturbations for Stress Test
- Many Managed code quant interfaces wrap Excel interface
  - Fragile production risk system

Why wrap an Excel interface for managed code

- Legacy assumptions about object ownership
  - Smart pointers not used
  - Assumptions that object not deleted
  - Fortran approach to memory management
  - Little RAII
- Wrapping a keyhole function less work than wrapping a class
- Need to avoid App dependency on specific quant library versions.
- Templates difficult to map.
- SWIG memory exceptions almost impossible to trace/debug

Thinking Functional

- Any pure function can be curried/wrapped in n-layers without changing semantics
- Any pure function can be executed in parallel without changing semantics
- Any pure function can be executed a number of times without changing the result
- Excel expressions are non-strict with no (visible) side-effects.

Cephei proposition

- Finite set of collections for Excel oriented library: Array, Matrix, Cube
  - Finite set of variations : values, pointer, shared_ptr, handle
- Finite set of template permutations
- Mutable code can be contained:
  - void functions mutate so lock objects in wrapper
  - void functions treated as fluent builder code

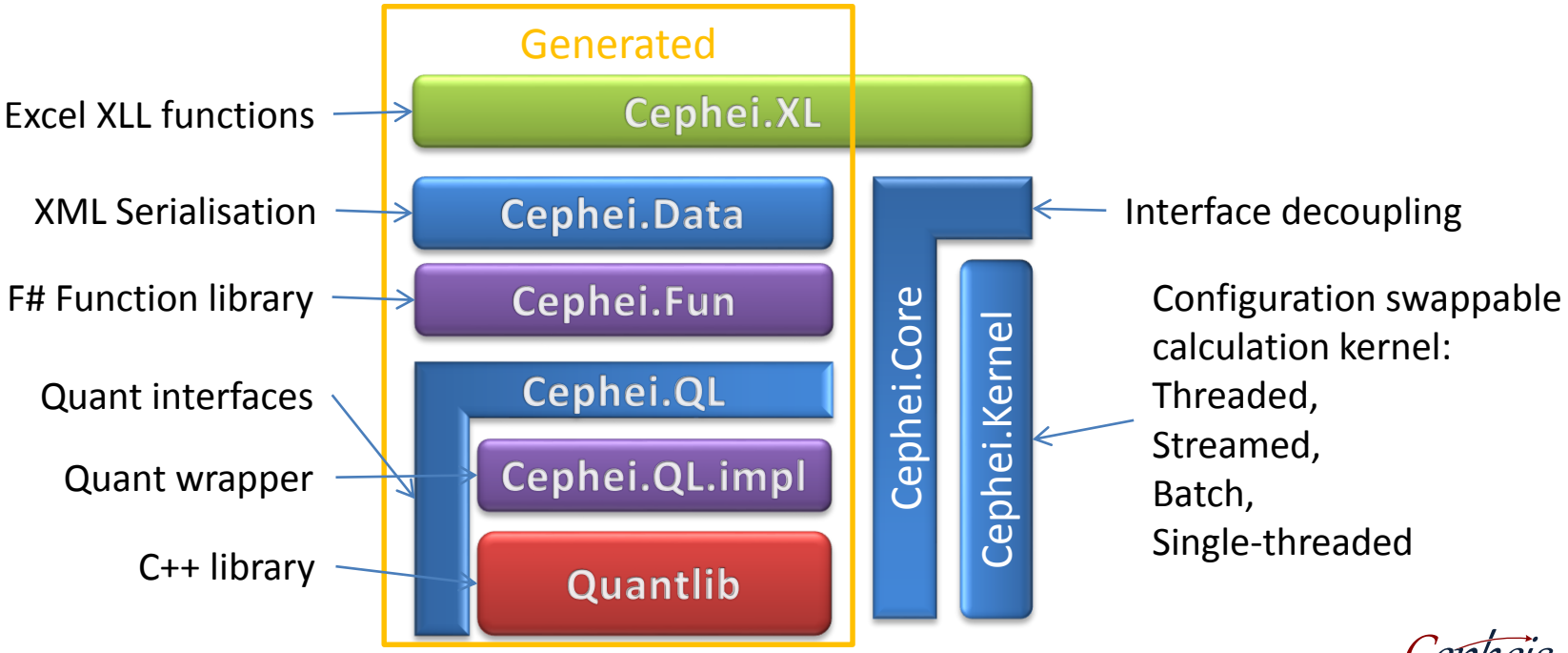
- Consistent wrapper layers allow for spreadsheets to be converted directly to code

Cephei.QL

- Code generated from Sparx EA model of C++ source code.
  - C# interface definitions
  - C++/CLI wrappers
  - F# module.
- Proof of concept of code generator available as open source http://ql.codeplex.com
- Mixed-mode assemblies for 32/64 bit
  - Portable to C++/CX (when MS gets around to non-store deployment)
- Part of Wider Framework

Cephei Framework

- Cephei Cell framework replicates Excel event calculation
- Cephei.XL saves Excel model as F# code



Cepheis

Cell Model (from EWI)

- Model presented to C# as a class with properties
- Source indicators seeded with default values

```
type EWISample (model : IModel) as this =
    inherit Cephei.Kernel.Model (model)
    // cell definitions
    let _hsbc_px_last = cell.Value (Fun.load "hsbc_px_last" 609.1)
    let _hsbc_history = cell.Value (Fun.load "hsbc_history" (Fun.defaultTS 260 Double.NaN))
    let _hsbc_CDS_3month = cell.Value (Fun.load "hsbc_px_last" 33.1)
    let _hsbc_CDS_9month = cell.Value (Fun.load "hsbc_px_last" 43.1)
    let _hsbc_px_close = cell { return Fun.first _hsbc_history.Value }
    let _hsbc_three_nine_diff = cell { return _hsbc_CDS_3month.Value - _hsbc_CDS_9month.Value }
    let _hsbc_30avg  = cell { return Fun.avg30day _hsbc_history.Value }
    let _hsbc_30high = cell { return Fun.max30day _hsbc_history.Value }
    let _hsbc_30low = cell { return Fun.max30day _hsbc_history.Value }
    let _hsbc_px_last_rag = cell {let! red = _hsbc_px_last.Value < _hsbc_30low.Value
                                  let! amber = _hsbc_px_last.Value < _hsbc_30avg.Value
                                  return if red then "R" elif amber then "A" else "G"}
    let _hsbc_CDS_3month_rag = cell {let! red = (_hsbc_CDS_3month.Value - _hsbc_three_nine_diff.Value) /
                                                _hsbc_CDS_3month.Value > 0.95
                                     let! amber = (_hsbc_CDS_3month.Value - _hsbc_three_nine_diff.Value) /
                                                  _hsbc_CDS_3month.Value > 0.90
                                     return if red then "R" elif amber then "A" else "G"}
    do this.Link ()
    // access properties
    member this.hsbc_px_last = _hsbc_px_last
```

Cepheis

Cell closures (from EWI)

- Cell encapsulates calculation and records dependencies through profiling
- Change events trigger re-calculation

```
type EWISample (model : IModel) as this =
    inherit Cephei.Kernel.Model (model)
    // cell definitions
    let _hsbc_px_last = cell.Value (Fun.load "hsbc_px_last" 609.1)
    let _hsbc_history = cell.Value (Fun.load "hsbc_history" (Fun.defaultTS 260 Double.NaN))
    let _hsbc_CDS_3month = cell.Value (Fun.load "hsbc_px_last" 33.1)
    let _hsbc_CDS_9month = cell.Value (Fun.load "hsbc_px_last" 43.1)
    let _hsbc_px_close = cell { return Fun.first _hsbc_history.Value }
    let _hsbc_three_nine_diff = cell { return _hsbc_CDS_3month.Value - _hsbc_CDS_9month.Value }
    let _hsbc_30avg  = cell { return Fun.avg30day _hsbc_history.Value }
    let _hsbc_30high = cell { return Fun.max30day _hsbc_history.Value }
    let _hsbc_30low = cell { return Fun.max30day _hsbc_history.Value }
    let _hsbc_px_last_rag = cell {let! red = _hsbc_px_last.Value < _hsbc_30low.Value
                                  let! amber = _hsbc_px_last.Value < _hsbc_30avg.Value
                                  return if red then "R" elif amber then "A" else "G"}
    let _hsbc_CDS_3month_rag = cell {let! red = (_hsbc_CDS_3month.Value - _hsbc_three_nine_diff.Value) /
                                                _hsbc_CDS_3month.Value > 0.95
                                     let! amber = (_hsbc_CDS_3month.Value - _hsbc_three_nine_diff.Value) /
                                                  _hsbc_CDS_3month.Value > 0.90
                                     return if red then "R" elif amber then "A" else "G"}
    do this.Link ()
    // access properties
    member this.hsbc_px_last = _hsbc_px_last
```

Cepheis

Conditional Cell closures (from EWI)

- Conditional expressions are profiled using monadic let! Verb

```
type EWISample (model : IModel) as this =
    inherit Cephei.Kernel.Model (model)
    // cell definitions
    let _hsbc_px_last = cell.Value (Fun.load "hsbc_px_last" 609.1)
    let _hsbc_history = cell.Value (Fun.load "hsbc_history" (Fun.defaultTS 260 Double.NaN))
    let _hsbc_CDS_3month = cell.Value (Fun.load "hsbc_px_last" 33.1)
    let _hsbc_CDS_9month = cell.Value (Fun.load "hsbc_px_last" 43.1)
    let _hsbc_px_close = cell { return Fun.first _hsbc_history.Value }
    let _hsbc_three_nine_diff = cell { return _hsbc_CDS_3month.Value - _hsbc_CDS_9month.Value }
    let _hsbc_30avg  = cell { return Fun.avg30day _hsbc_history.Value }
    let _hsbc_30high = cell { return Fun.max30day _hsbc_history.Value }
    let _hsbc_30low = cell { return Fun.max30day _hsbc_history.Value }
    let _hsbc_px_last_rag = cell {let! red = _hsbc_px_last.Value < _hsbc_30low.Value
                                  let! amber = _hsbc_px_last.Value < _hsbc_30avg.Value
                                  return if red then "R" elif amber then "A" else "G"}
    let _hsbc_CDS_3month_rag = cell {let! red = (_hsbc_CDS_3month.Value - _hsbc_three_nine_diff.Value) /
                                                _hsbc_CDS_3month.Value > 0.95
                                     let! amber = (_hsbc_CDS_3month.Value - _hsbc_three_nine_diff.Value) /
                                                  _hsbc_CDS_3month.Value > 0.90
                                     return if red then "R" elif amber then "A" else "G"}
    do this.Link ()
    // access properties
    member this.hsbc_px_last = _hsbc_px_last
```

Cepheis

Demo

- steve.channell@cepheis.com

- http://www.cepheis.com
- http://ql.codeplex.com
- http://quantlib.org/index.shtml