Background

- Who we are
  - Quantitative Strategies: Modelling & Quantitative Analysis team in Credit Suisse
  - Previous incarnation known as Global Modelling & Analytics Group
- What we do
  - Build valuation, pricing, risk and market analysis models and tools
  - Work with trading desks to analyse (potential) trades and risk exposure
- How we do it
  - C++: Computationally intensive numerical algorithms/model building
  - F#: Composition of C++ building blocks to value specific products
  - Excel: UI, data munging, additional higher-level logic/analytics
  - C#: IT systems for data-processing and orchestrating risk analysis processes
  - Legacy use of proprietary/other technologies

Motivation for Eden

- Objective is to replace Excel for UI
- Richer, more dynamic UI – free from constraints of grid-like layout
- Better structured and more maintainable code

What we need

1. Laziness and partial recalc 2. Caching 3. Asynchronous result production 4. Automatic parallelization 5. Optional manual calculation 6. Cancellation	Dependency graph-based approach
7. Fully debuggable	Pure F# code
8. Wide selection of rich UI controls 9. Separation of business logic and view	WPF MVVM pattern

DEMO

After two years of usage…

- What works
  - Built several large-scale applications
  - UI well received by the users
  - Responsiveness and performance are good
  - Easy to add commonly used services (i.e. undo, persistence)

- What can be improved
  - Difficult to code in functional side effect free way for some people
  - Some need for side effects still
  - Hard to debug asynchronous and parallel computations
  - Difficult if you go outside the framework

Summary

- Eden is an async parallel calculation graph implemented with F# async
- It works well with the declarative nature of WPF
- Experience with it is overall positive with some areas of improvement