

## Neurosphere Lab: A case study in simulation software development

Mark Wakabayashi

## Seminar overview

Introduction  
Demonstration  
Software Design

GUI  
Software process  
Conclusion

- Introduction
- Demonstration
- Software design
- GUI development and design
- Software process
- Conclusion

## About me

Introduction  
Demonstration  
Software Design

GUI  
Software process  
Conclusion

- Bachelor of Information Technology and Bachelor of Arts (Cognitive Science), 2005
- ACCS Summer Research Assistant, 2005-2006
- Now: IT honours
  - Special Topics

## Project background

Introduction  
Demonstration  
Software Design

GUI  
Software process  
Conclusion

- Neurospheres:
  - In vitro growths of neurons into sphere of several thousands
- Kristin Hatherley (Rod Rietze's lab, QBI), is conducting lab studies and developing mathematical models
- Important in understanding generation of new neurons in adult nervous system
- Why write software?
  - A platform for implementation and validation of mathematical models

## Requirements analysis

Introduction  
Demonstration  
Software Design

GUI  
Software process  
Conclusion

- No formal requirements
  - Face-to-face meetings
  - No formal documentation
- Main requirements
  - Model specification (format unknown)
  - Cell lineage visualisation
  - 3D cell structure visualisation
  - Interaction between visualisations
  - Mac-compatible

Introduction  
Demonstration  
Software Design

GUI  
Software process  
Conclusion

Software demonstration

## Software overview

Introduction  
 Demonstration  
 Software Design  
 GUI  
 Software process  
 Conclusion

- Language and extensions
  - Java 1.4
  - SWT (GUI library)
  - OpenGL (3D rendering)
  - JiBX (XML binding)
  - BIRT (graphing)
- Project size
  - ~2.5 months full time
  - 7137 lines of functional code
  - 12024 total lines of code
  - 90 classes

## Design goals

Introduction  
 Demonstration  
 Software Design  
 GUI  
 Software process  
 Conclusion

- Extensibility and maintainability
  - Software lifetime beyond the scope of employment
- Rapid development and extension
- Explore software designs in computational modelling

## Software design patterns

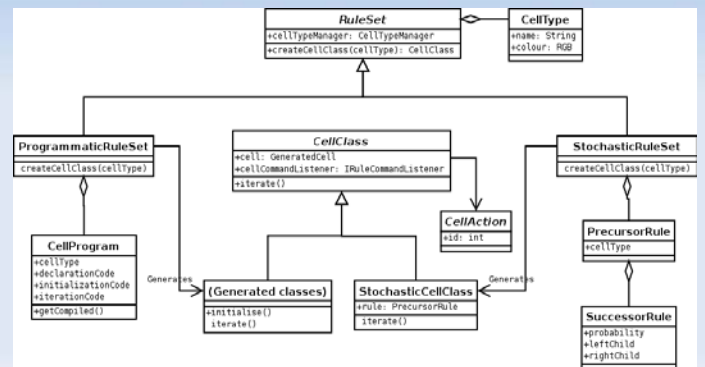
Introduction  
 Demonstration  
 Software Design  
 GUI  
 Software process  
 Conclusion

- Repeatable solutions to recurring problems in programming
- Applications to
  - Creation
  - Structure
  - Behaviour
- Support design goals
  - Extensibility and maintainability
  - Rapid development
  - Explore software designs

## RuleSet structure

Introduction  
 Demonstration  
 Software Design  
 GUI  
 Software process  
 Conclusion

- Specifies the mathematical model of neuron growth



## RuleSet design

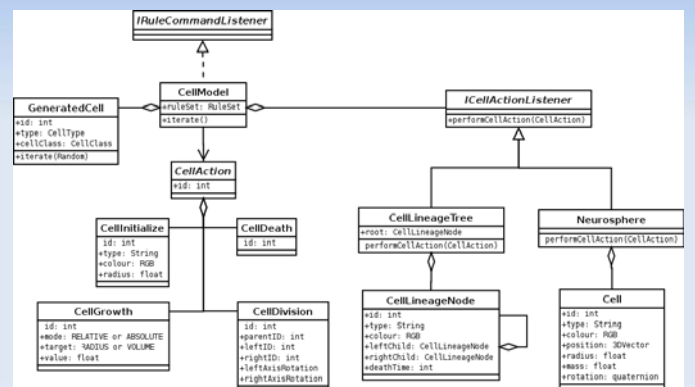
Introduction  
 Demonstration  
 Software Design  
 GUI  
 Software process  
 Conclusion

- Goal: to encapsulate the specification of the behaviour of individual cells
- Factory pattern
  - = Methods to return objects with required properties
  - RuleSet is a CellClass factory
- Command pattern
  - = Capture operations in objects
  - CellAction encapsulates an operation in the model

## Cell model structure

Introduction  
 Demonstration  
 Software Design  
 GUI  
 Software process  
 Conclusion

- CellModel encapsulates the behaviour of the entire neurosphere



## Cell model design

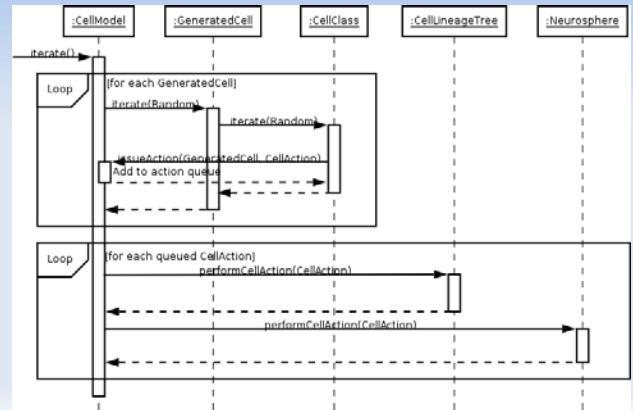
Introduction  
Demonstration  
Software Design  
GUI  
Software process  
Conclusion

- Goal: complete encapsulation of the neurosphere model and behaviour
- Composite pattern
  - = Tree structure
  - Lineage tree uses a tree data structure
- Observer / Listener pattern
  - = Objects are registered to listen to events, and are notified by the subject
  - In an iteration, events are propagated from the CellModel to the lineage and neurosphere models

## Iteration sequence

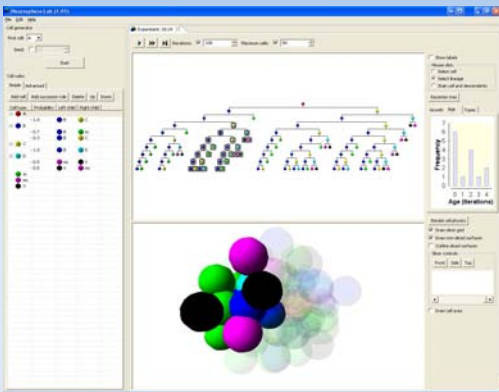
Introduction  
Demonstration  
Software Design  
GUI  
Software process  
Conclusion

- Simplified flow of control in an iteration



## GUI

Introduction  
Demonstration  
Software Design  
GUI  
Software process  
Conclusion



## GUI development is hard

Introduction  
Demonstration  
Software Design  
GUI  
Software process  
Conclusion

- Time consuming
  - Large API
  - Complex interfaces require manual coding
  - OpenGL
- Requires parallel threads
- Users are unpredictable
  - Need to account for a large number of possible interactions

## GUI design guidelines

Introduction  
Demonstration  
Software Design  
GUI  
Software process  
Conclusion

- Have most of the application visible at all times
  - Let the user centre their attention on what's relevant
- Keep buttons and switches near what they control
- Provide immediate feedback for user actions
- Emphasise mouse-based interactions
  - More intuitive
  - More efficient for interaction with visualisation

## GUI software patterns

Introduction  
Demonstration  
Software Design  
GUI  
Software process  
Conclusion

- Observer / Listener:
  - User interaction initiates listener code
- Facade / Provider
  - = Define a class that acts as a simplified interface to a larger set of classes or functions
  - Interfaces between the models and the views:
    - CellLineageProvider
    - NeurosphereCellProvider

## Threading and synchronization

Introduction  
Demonstration  
Software Design

GUI  
Software process  
Conclusion

- Constant UI reads of model states
- Threaded model iterator
- Simultaneous UI reads and iterator thread writes
- Solution: Locking mechanism implementation in model data providers

## Software process

Introduction  
Demonstration  
Software Design

GUI  
Software process  
Conclusion

- Agile programming
  - Emphasises adaptive rather than predictive methods
  - Different from 'cowboy coding'
- My development method
  - Iterative implementation of features
  - Continual refactoring

## Speculation

Introduction  
Demonstration  
Software Design

GUI  
Software process  
Conclusion

- Dangers of:
  - Speculative generality
  - Premature optimisation
- First make it work, then make it work well
  - Don't always know what will need optimisation
  - Only works if you actively seek to improve
    - Refactoring
    - Execution profiling

## Tools

Introduction  
Demonstration  
Software Design

GUI  
Software process  
Conclusion

- Development environment: Eclipse
  - + TPTP profiler plugin
- Version control: Subversion
  - Implement without fear
  - Project portability
- Build tool: ANT
  - Implemented out of a need to manage:
    - Platform-specific libraries
    - Java version-specific compiler library
    - JiBX bytecode modification after compilation

## Documentation

Introduction  
Demonstration  
Software Design

GUI  
Software process  
Conclusion

- Explicit written explanation of software code, design, and motivations
- Agile software development is not conducive to comprehensive documentation

### Documentation in parallel

- Too much work
- Quickly outdated
- Bad documentation is worse than no documentation
- Disincentive to refactor

### Documentation at completion

- Big task
- Not fun
- No time
- Knowledge lost

## Documentation – my approach

Introduction  
Demonstration  
Software Design

GUI  
Software process  
Conclusion

- Use descriptive names, and keep them accurate
  - Modern IDEs make this feasible
- Document in the code
- Don't over-comment
  - Code is obscured
  - Danger of inconsistency

## Overall evaluation

Introduction  
Demonstration  
Software Design

GUI  
Software process  
Conclusion

- As a software product:
  - Project handed over on time
  - All required features were implemented
  - No major bugs reported
- As an exercise in software development:
  - Refined a personal software engineering methodologies
  - Used and appreciated software patterns
  - General software development experience

## Take home messages

Introduction  
Demonstration  
Software Design

GUI  
Software process  
Conclusion

- Good software design makes a difference:
  - Modularity and encapsulation
  - Software patterns
  - Continual refactoring

## Acknowledgements

Introduction  
Demonstration  
Software Design

GUI  
Software process  
Conclusion

- ARC Centre for Complex Systems
- ARC grant to Janet Wiles
  
- Rod Rietze, Kristin Hatherley
- Janet Wiles, David Carrington
- David Woolford, James Watson, Stefan Maetschke

## The good and the bad

Introduction  
Demonstration  
Software Design

GUI  
Software process  
Conclusion

- The good
  - CellAction method of specifying cell growth and division
    - Complete encapsulation of growth model
  - Real-time interactive visualisation
    - There's something about interactivity that's not possible in static images or movies
- The bad
  - Underestimated porting time
  - Some instances of overgeneralising