University of Limerick

Understanding context and its role in software engineering tasks

or

Human Software Interaction

Dr. Chris Exton, Department of Computer Science and Information Systems

Motivation

- As software systems have grown, the granularity of the problem domain has changed with the realisation that in most software systems, complexity does not manifest itself as an algorithmic problem but as a problem of scale and interaction

- A problem of human software interaction!

Software Comprehension

- A person understands a program when able to explain the program, its structure, its behaviour, its effects on its operational context, and its relationships to its application domain in terms that are qualitatively different from the tokens used to construct the source code of the program (Biggerstaff et al., 1994).
- Given a program in a particular language, program comprehension is a process in which the programmer uses prior knowledge about programming and information present in the program to form a dynamic, evolving model of the program which can then be applied to a task (Good, 1999).

University of Limerick

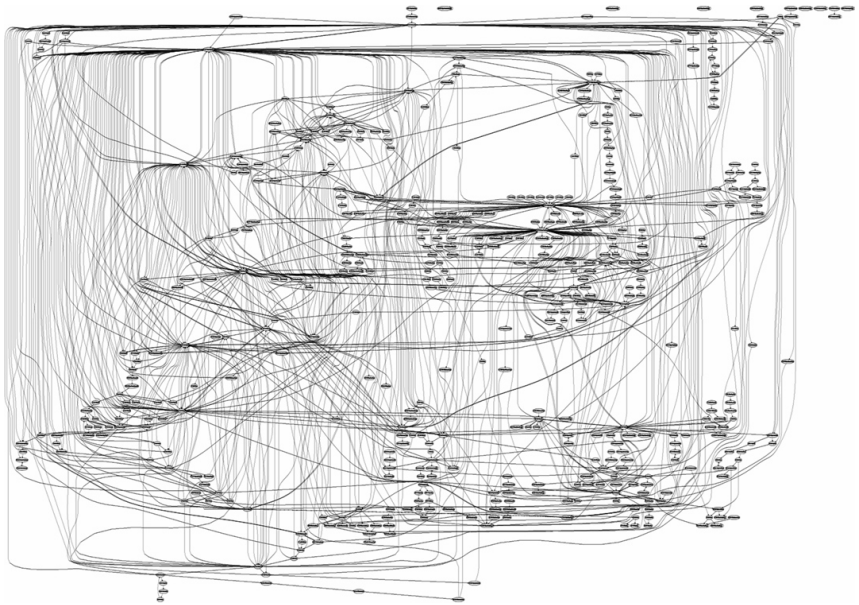
Human Software Interaction

- Although there is some research relating to how classical users interact with software systems there is a need to focus on how software maintainers and developers interact with their software environment
- Visualization takes advantage of the cognitive predisposition that people have to understand information visually
  - There are of course other approaches

University of Limerick

Reality of industrial code space

- A dense, structurally complex information space
- Fragmentation
  - Decomposition/Abstraction
  - **Control flow scatter**
  - Cross cutting concerns
- Dense arbitrary link topology
- Visually homogeneous
- N.B. Students are rarely exposed to such environments



Developers navigate within a complex information space to achieve their tasks

Weaknesses in existing approaches

- Program comprehension community
  - Provide abstract cognitive theories, not that much use to visualization tool developers
  - Cognitive theories often empirically assessed using unrealistic task sets
- Visualisation Tool Developers
  - Tools are often evaluated by their capacity to function correctly, rather than their value to the software engineer
  - Seldom ask/seek users for their requirements
  - Base them on introspection / abstract cognitive theories

The realities of programming

- Software Engineers are often overwhelmed by size of software projects
- Considering the size of most software systems Software Engineers have insufficient tools to enable them to
  - understand large existing bodies of code
  - understanding the dynamic behavior of their code when executing (especially under stress)
  - to isolation and locate bugs in an efficient manner

Context is key

- Ongoing context during exploration & development activities
- Overall context associated with a development task
- Code level context V's software architectural context

The problems with context

- Context is maintained in human memory
- An architectural level understanding whilst working at the code level takes a long time to develop
- Error prone, overloading, distraction, disorientation
- Fades over time & not easily transferable between developers
- Easily lost
  - Debugger example!

University of Limerick

Some approaches to software context

- **Inline source code exploration**
- Degree of interest
- Thumsoft
- Software Reconnaissance
- Reflextion
- Novel requirement gathering approaches
- Other work and conclusion

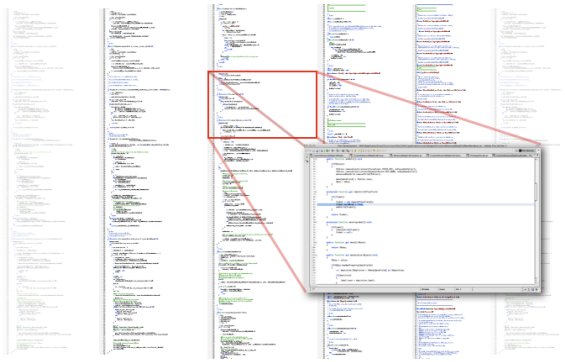
Control flow scatter

- An ever growing problem?
  - *"Go To Statement Considered Harmful"*
  - Polymorphism
  - Inheritance
  - Runtime type identification
  - Increasing use of XML and mixed language environments
    - e.g. Andriod development
  - Aspect oriented programming

University of Limerick

IDE provides a keyhole view

- The keyhole property
- A lack of visual momentum
  - Source code explored as a sequence of discrete source code displays
- Programmer maintains navigation and task context in memory
- Problems
  - Getting lost and disoriented
  - Pursuit of digressions
  - Thrashing between displays/ Interface management



University of Limerick

Embedded annotations

Java - JHotDraw7/src/org/jhotdraw/draw/DefaultDrawing.java - Eclipse SDK

File Edit Navigate Search Project Fluid Run Tasks Window Help

JSheet.java | Main.java | AbstractDrawing.java | DefaultDrawing.java

```
        figure.addFigureListener(figureHandler);
        invalidateSortOrder();
    }

    public void basicRemove(Figure figure) {
        figures.remove(figure);
        figure.removeFigureListener(figureHandler);
        invalidateSortOrder();
    }


    public void draw(Graphics2D g) {
        synchronized (getLock()) {
            ensureSorted(
            ArrayList<Fig                    ist<Figure>(figures.size());
            Rectangle cli                    );
            for (Figure f
                if (f.get                    (clipRect)) {
                    toDra
                }
            }
            draw(g, toDraw);
        }
    }

    public java.util.List<Figure> sort(Collection<Figure> c) {
        HashSet<Figure> unsorted = new HashSet<Figure>();
        unsorted.addAll(c);
        ArrayList<Figure> sorted = new ArrayList<Figure>(c.size());
        for (Figure f : figures) {
            if (unsorted.contains(f)) {
                sorted.add(f);
                unsorted.remove(f);
            }
```

<<Expand>>

Method
void ensureSorted()
Declaring Type
org.jhotdraw.draw.DefaultDrawing

Press 'F2' for focus.

Writable | Smart Insert | 80 : 5

University of Limerick

Expand inline declarations

Java - JHotDraw7/src/org/jhotdraw/draw/DefaultDrawing.java - Eclipse SDK

File Edit Navigate Search Project Fluid Run Tasks Window Help

JSheet.java | Main.java | AbstractDrawing.java | DefaultDrawing.java

```
        figure.addFigureListener(figureHandler);
        invalidateSortOrder();
    }

    public void basicRemove(Figure figure) {
        figures.remove(figure);
        figure.removeFigureListener(figureHandler);
        invalidateSortOrder();
    }


    public void draw(Graphics2D g) {
        synchronized (getLock()) {
            ensureSorted();
```

DefaultDrawing.java

```
            /**
             * Ensures that the figures are sorted in z-order sequence from back to
             * front.
             */
            private void ensureSorted() {
                if (needsSorting) {
                    Collections.sort(figures, FigureLayerComparator.INSTANCE);
                    needsSorting = false;
                }
            }

            ArrayList<Figure> toDraw = new ArrayList<Figure>(figures.size());
            Rectangle clipRect = g.getClipBounds();
            for (Figure f : figures) {
                if (f.getDrawingArea().intersects(clipRect)) {
                    toDraw.add(f);
                }
            }
            draw(g, toDraw);
```

Writable | Smart Insert | 80 : 5

University of Limerick

Nested expansion

Java - JHotDraw7/src/org/jhotdraw/draw/DefaultDrawing.java - Eclipse SDK

File Edit Navigate Search Project Fluid Run Tasks Window Help

JSheet.java | Main.java | AbstractDrawing.java | DefaultDrawing.java

```
            }
        }
        return null;
    }
    public Figure findFigureBehind(Point2D.Double p, Collection<Figure> figures) {
        int inFrontOf = figures.size();
        for (Figure f : getFiguresFrontToBack()) {
```

DefaultDrawing.java

```
            /**
             * Returns an iterator to iterate in
             * Z-order front to back over the figures.
             */
            public java.util.List<Figure> getFiguresFrontToBack() {
                ensureSorted();
```

DefaultDrawing.java

```
                /**
                 * Ensures that the figures are sorted in z-order sequence from back to
                 * front.
                 */
                private void ensureSorted() {
                    if (needsSorting) {
                        Collections.sort(figures, FigureLayerComparator.INSTANCE);
                        needsSorting = false;
                    }
                }

                return new ReversedList<Figure>(figures);
            }

            if (inFrontOf == 0) {
                if (f.isVisible() && f.contains(p)) {
                    return f;
                }
            } else {
```

Writable | Smart Insert | 73 : 37

University of Limerick

Polymorphism

Java - JHotDraw7/src/org/jhotdraw/draw/DefaultDrawing.java - Eclipse SDK

File Edit Navigate Search Project Fluid Run Tasks Window Help

JSheet.java | Main.java | AbstractDrawing.java | DefaultDrawing.java

```
public void draw(Graphics2D g, Collection<Figure> figures) {
    Rectangle2D clipBounds = g.getClipBounds();
    if (clipBounds != null) {
        for (Figure f : figures) {
            if (f.isVisible() && f.getDrawingArea().intersects(clipBounds)) {
                f.draw(g);
            }
        }
    } else {
        for (Figure f : figures) {
            if (f.isVisible()) {
                f.draw(g);
```

Figure.java

```
void SVGPathFigure.draw(Graphics2D g)
void SVGGroupFigure.draw(Graphics2D g)
void AbstractAttributedDecoratedFigure.draw(Graphics2D g)
void AbstractCompositeFigure.draw(Graphics2D g)
void GraphicalCompositeFigure.draw(Graphics2D g)
void SVGImageFigure.draw(Graphics2D g)
void LabeledLineConnectionFigure.draw(Graphics2D g)
void SVGAttributedFigure.draw(Graphics2D g)
void AbstractAttributedFigure.draw(Graphics2D g)
```

```
            }
        }
    }
}

public Figure findFigure(Point2D.Double p)
{

    for (Figure f : getFiguresFrontToBack()) {
```

Writable | Smart Insert | 135 : 1

Inline source code exploration

- Explore source code in context
  - Maintains a representation of navigation history
  - Supports the pursuit of digressions in context
  - Facilitates comprehension of fragmented code
- Eclipse Editor extension
  - Method, type, field, variable expansion
  - Unlimited nesting
  - Extensible extension architecture - supports inline hyperlink via embedded web browser & image expansion

Results

- 8 participants over a series of 8 exploration tasks
  - Participants completed the exploration tasks 14 % faster on the inline interface
  - Participants performed 31% less ‘navigation’ using the inline interface
- Effective
  - A visual representation of navigation history
  - Support for digressions in context
  - Support for comprehending fragmented code
- Getting ‘lost in expansion space’

University of Limerick

Inline source code exploration Demo

- Basics of the fluid editor

The Fluid Source Code Editor
Part #1 Basics

- Advanced features of the fluid editor such as polymorphic method exploration and navigation
- Available at sourceforge
- http://fluideditor.sourceforge.net/

The Fluid Source Code Editor
Part #2 Advanced

University of Limerick

Some approaches to software context

- Inline source code exploration
- **Thorr (Degree of interest)**
- Thumsoft
- Software Reconnaissance
- Reflextion
- Novel requirement gathering approaches
- Other work and conclusion

Thorr

- Another attempt at addressing the Context v's Detail problem
- Based on the following observations
  - Programmers will often lose context when stepping through code
  - Good programmers are aware of inheritance topology

Focus + context techniques

- One of the biggest difficulties in visualizing large quantities of information is the lack of screen space in which to visualize it
- Users can become disoriented or even 'lost' in a visualisation where complex navigation is required to obtain information.
- Focus + context techniques were introduced as means of displaying huge quantities of information on one screen
- A distortion algorithm is used to magnify a certain part of the screen while de-emphasizing the rest
  - As a result, the user can have a constant overview of the information while focusing in on one particular area.

(Thorr) Degree of Interest

- Developed at XEROX-PARC (Card and Nation, 2002) the DOI (degree-of-interest) tree is an example of an attention reactive interface, used for visualising hierarchical information
- The focus + context method, which it utilises, differentiates it from other similar visualisations.
- Other focus + context visualisations apply distortion algorithms
  - fisheye views and perspective walls
- DOI tree applies transformations and magnifications to the tree to illustrate interest in specific node

(Thorr) Degree of Interest

- The Thorr programming environment hopes to overcome some of the shortcomings currently associated with similar visualisation projects, in terms of balance between level of detail and context, by using 'Degree-of-Interest‘

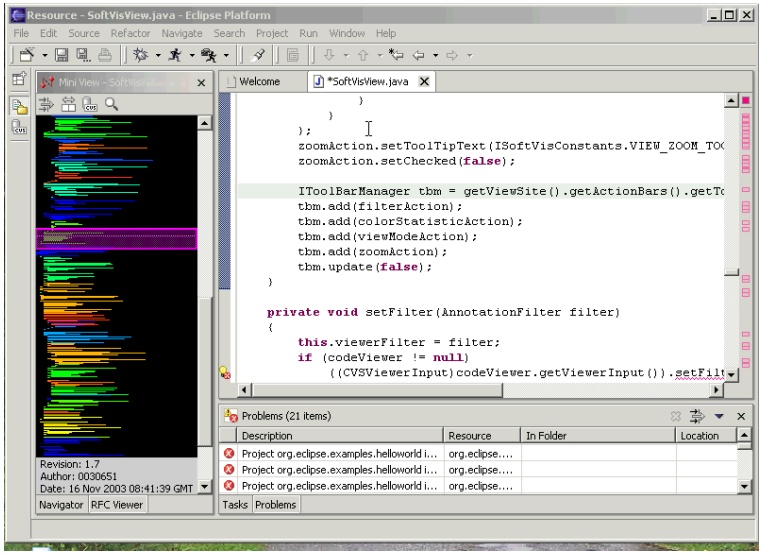
University of Limerick

Some approaches to software context

- Inline source code exploration
- Thorr (Degree of interest)
- **Thumsoft**
- Software Reconnaissance
- Reflextion
- Novel requirement gathering approaches
- Other work and conclusion

ThumbSoft

- Observations
  - Programmers often look for the most recently changed lines of code
  - Programmers are aware of program topology
  - Programmers will often seek advice from the original author



University of Limerick

Some approaches to software context

- Inline source code exploration
- Thorr (Degree of interest)
- Thumsoft
- **Software Reconnaissance**
- Reflextion
- Novel requirement gathering approaches
- Other work and conclusion

University of Limerick

Software Reconnaissance

- Another attempt at addressing the Context v's Detail problem
- Observations
  - Programmers will segment large systems using domain language
  - Programmers seek information that links Software to Domain details
  - Line based debuggers are fine for detailed code understanding but do not work for architectural level investigations

Concept Assignment

- Assisting Concept Assignment using Probabilistic Classification & Cognitive Mapping
- Intent
  - What developers intend when they write software (Simonyi, 2005).
- Conceptual Gap
  - Results from constraints on the sets of concepts expressible in the language in which the intent is described.
- Concept Assignment Problem
  - The problem of discovering individual human oriented concepts and assigning them to their implementation oriented counterparts (Biggerstaff et al., 1993).

Tackling Concept Assignment

- Manual/ Tool assisted techniques (Biggerstaff, 1989) (Robillard, 2003),
- Lexical Analysis (Anquetil and Lethbridge, 1999) (Marcus et al., 2004)
- Dynamic Code Analysis (Wilde and Scully, 1995) (Eisenbarth et al., 2003)

University of Limerick

Our Approach

- Apply probabilistic classification to the concept assignment problem
- Incorporate expert knowledge
- Encapsulate approach in eclipse plug-in
- Evaluate performance of technique against related work

Cognitive Mapping

- Cognitive mapping is a quantitative text analysis technique that systematically extracts and analyses the links between words in a text in order to model the authors mental or cognitive map as networks of words (Lewis et al., 2001) (Diesner and Carley, 2004) (Popping, 2000).
- This map is hypothesised to approximate a portion of the mental model of the texts author at the time the text was composed (Carley, 1997).

Cognitive Mapping

- A *concept* is a single idea or ideational kernel represented by a single word, a composite phrase consisting of multiple words or more complex phrases (Carley, 1997b).
- A *relationship* is the tie that links two concepts together (Diesner and Carley, 2004b).
- A *statement* consists of two concepts and the relationship that exists between them (Lewis et al., 2001).
- If two statements share a concept then those statements are linked, the union of all linked statements per text forms a map or a network of concepts (Carley, 1997a).

University of Limerick

Cognitive Mapping Procedure

- Corpus Definition
  - Design Documentation, emails, interview transcripts
- Concept Set Definition
  - System expert uses content analysis to identify a set of problem domain concepts related to the SUS

Cognitive Mapping Procedure

- Cognitive Map Generation
  - Text Pre-processing is performed over the corpus, including the application of a delete list, stemming and a generalisation thesaurus
  - A co-occurrence analysis is then performed over the corpus filtered by the expert defined concept set that creates edges between concepts which co-occur within a window in the corpus
  - Edges of the cognitive map are annotated with the frequency of co-occurrence of concept pairs

Classifier Construction

- Define Concept Map related to concept being assigned.
- For Each Software Element
  - **Identify the subset of the concepts contained in the active concept map that also occur in the software element**
  - **Identify the section of the cognitive map that this subset defines.**
  - **Over this map subset construct a maximum spanning tree based on the weights assigned to the edges.**
  - **Finally we get the product of the conditional probabilities and assign the element its classification score.**

Evaluation

- Participants
  - 4 Participants
  - average 12 months commercial development experience
  - Average 3 years academic development experience
- Evaluation System
  - Chive
  - 15 KLOC
  - Approximately 300 methods

Chive

- The CHIVE framework supports user definable datasets, multiple graph layouts and an environment in which users can interact

- Visualisations can created from applying a graph layout to a any dataset

University of Limerick

Experiment Procedure

- Participants briefed on experiment objectives and protocol
- Participants then received in the cognitive assignment plug-in
- Participants then received a 10 minute introduction to the SUS
- Participants were then presented with 4 tasks which they were asked to perform in series
- Participants were asked for each task to identify the set of method which they considered to be important to the tasks they were asked to perform.

University of Limerick

Results and Analysis

- Precision and Recall Performance
  - Not as good as we had anticipated
  - We were especially disappointed with recall
  - However tan based classifier does appear to differentiate between instances better than naïve Bayes.
- Lowest Element Investigated
  - Our hypothesis was confirmed, investigation of results was confined across participants to the top few entries.

University of Limerick

Some approaches to software context

- Inline source code exploration
- Thorr (Degree of interest)
- Thumsoft
- Software Reconnaissance
- **Novel requirement gathering approaches**
  - **Content analysis**
- Other work and conclusion

University of Limerick

Content analysis

- The application of a content analysis scheme to Java program summaries extracted from online open source developer mailing lists
- Example research Question:
  - Can we ascertain the degree and speed of adoption of software Design Patterns within the software development community?

University of Limerick

Content analysis

- We carried out an investigation using data from a number of open source mailing lists
- A psychology based methodology was used on the program summaries posted to mailing lists by experienced programmers as well as the type of analysis involved
- When compared many other empirical studies, an advantage is gained allowing stronger ecological valid insights into programmer comprehension

Method

- 12 Patterns chosen from 23 GOF patterns
- 4 from each of Structural, Behavioural and Creational
- Criteria for choice - most popular patterns with most references to design patterns

Selected Patterns

- Factory Method
- Abstract Factory
- Prototype
- Singleton
- Composite
- Decorator
- Façade
- Flyweight
- Template Method
- Mediator
- Memento
- Visitor

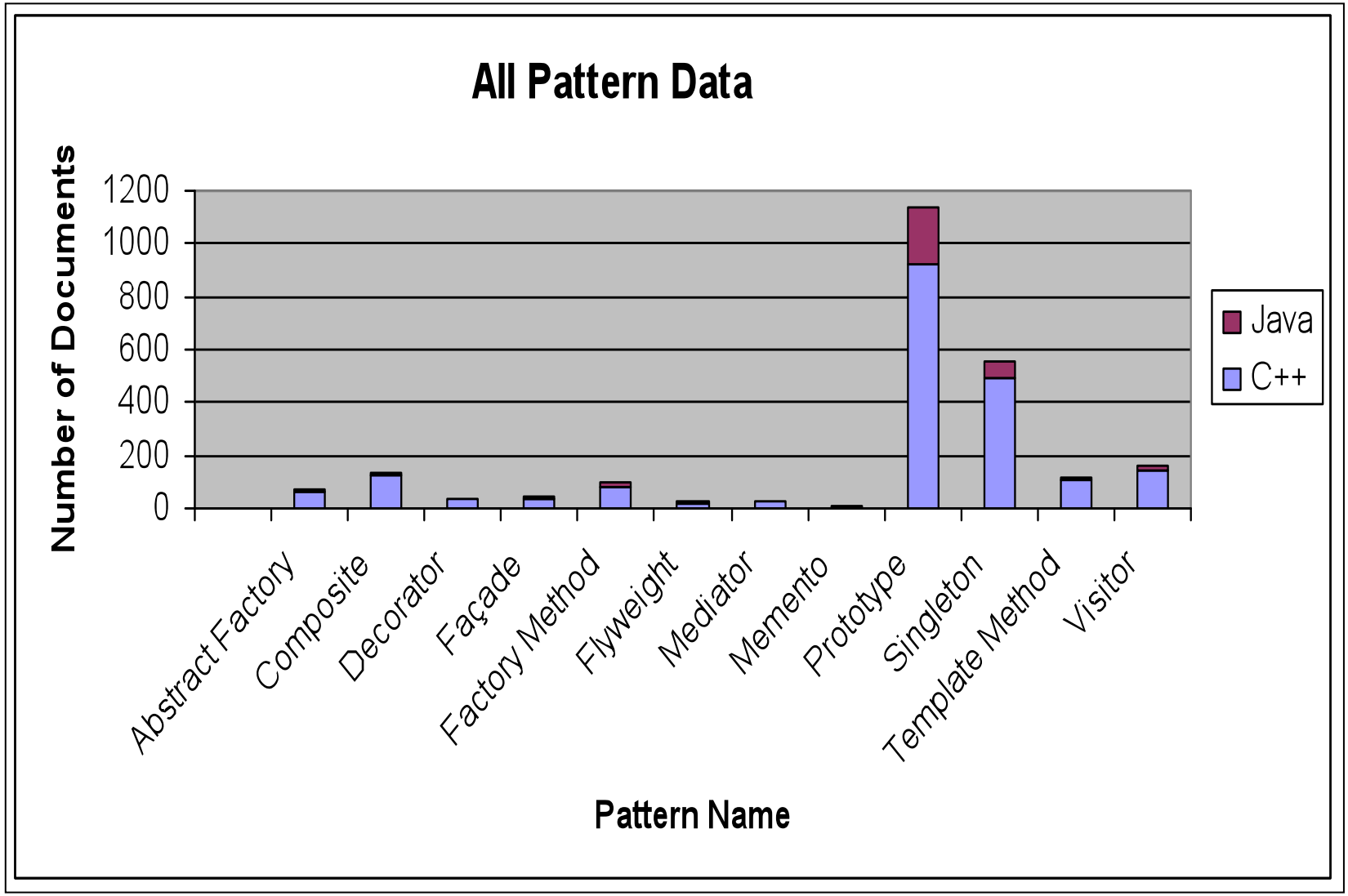
University of Limerick

Graph - all pattern data

- 2422 documents downloaded for all 12 patterns
- 2058 for c++
- 364 for Java
- Hit rate
  - Highest Number of Documents - Prototype
  - Lowest Number of Documents - Memento

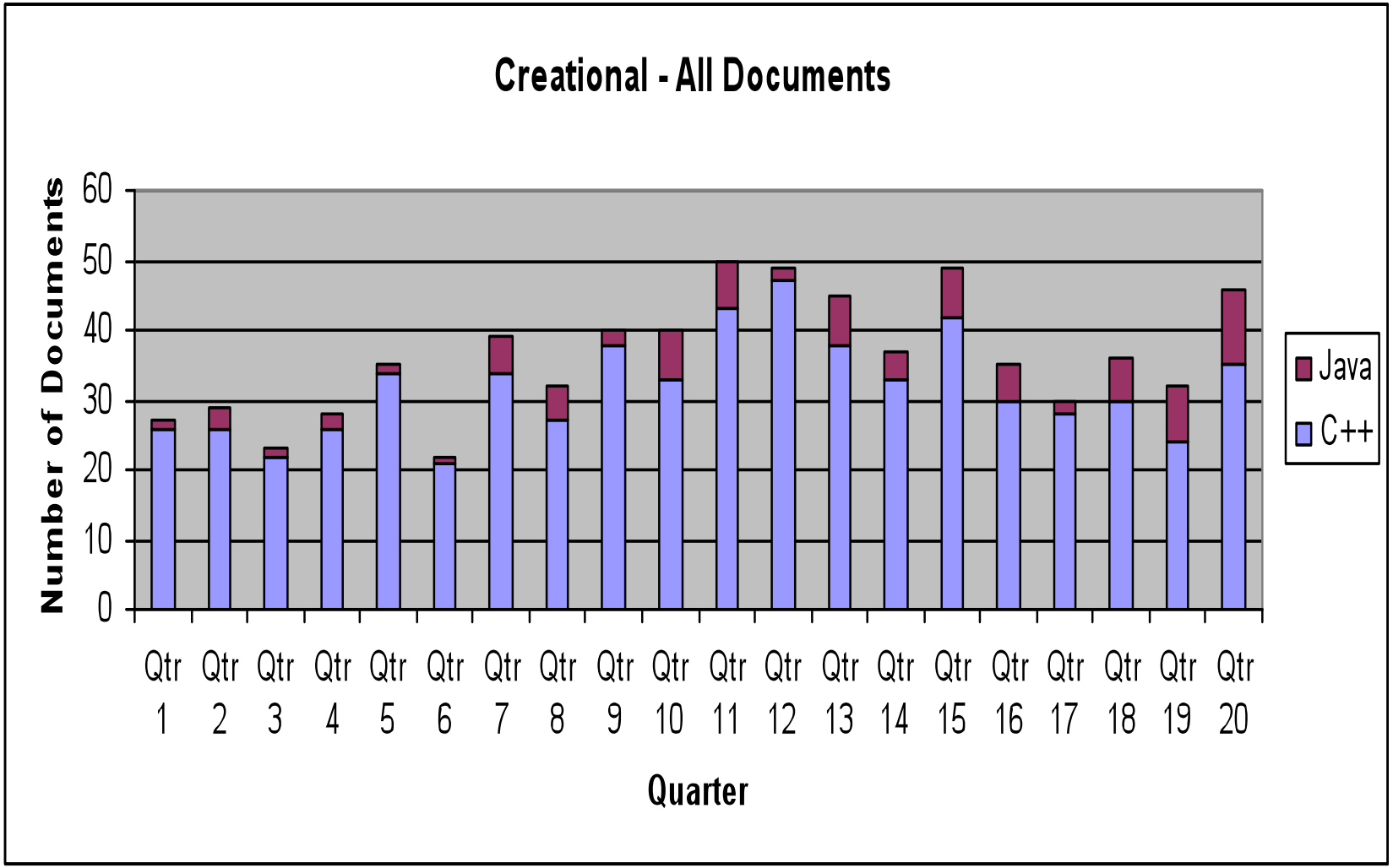
University of Limerick

Graph - all pattern data



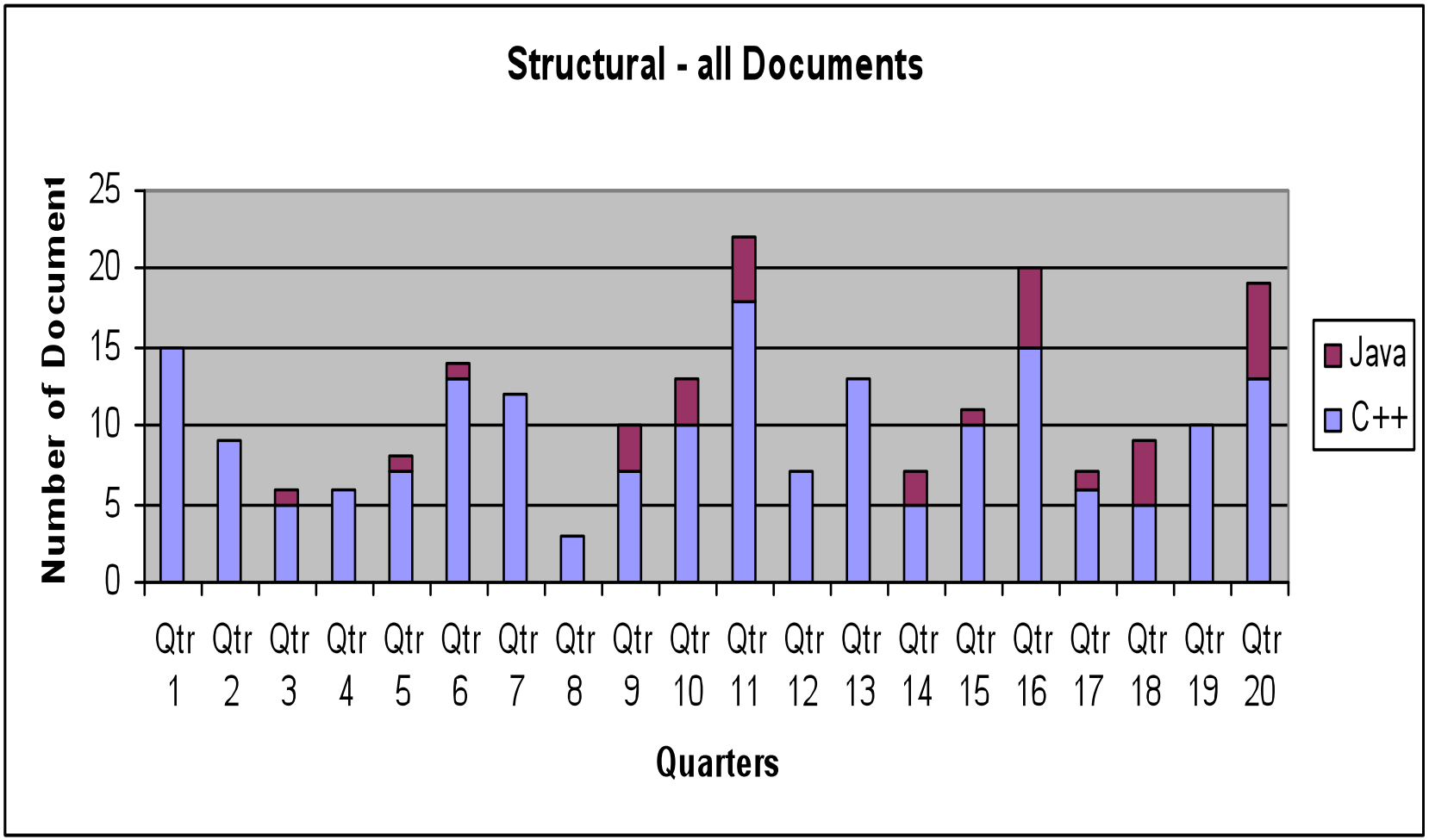
University of Limerick

Distribution over 5 Years (Behavioural)



University of Limerick

Distribution over 5 Years (Structural)



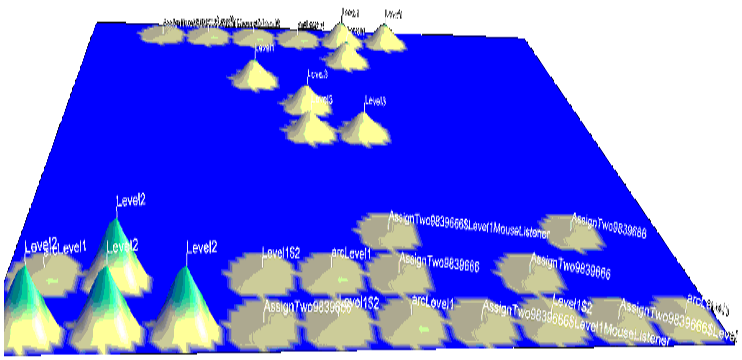
University of Limerick

Discussion

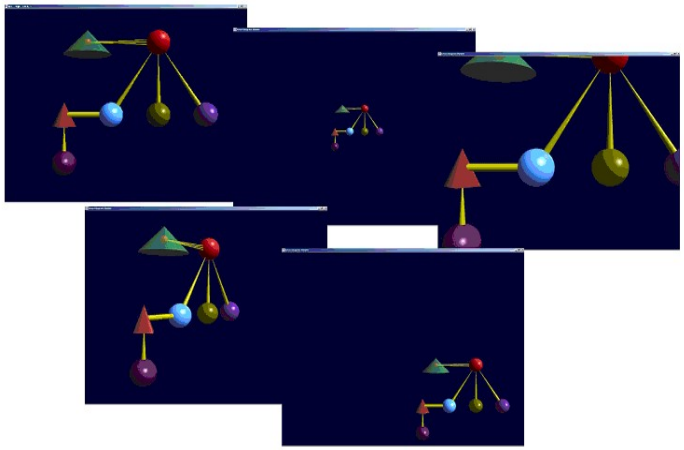
- Some growth in use of Design Patterns in comp.lang.c++ and comp.lang.java over the five years
- A lot more references found in comp.lang.c ++ than java
- More references for some patterns namely Composite, Visitor and Prototype
- We have used this method with other work
  - Code beacon identification
  - Task type differences

Some other past projects

- Cohesion and coupling using terrain maps



- Genome representation of software architecture



University of Limerick

University of Limerick

Problem domain characteristics

- Program understanding is estimated at 50% - 60% of the maintenance effort
- The challenges that need to be addressed
  - Scale (Context v's Detail)
  - Complexity
  - Dynamics
  - Multivariate data
  - Software/Domain requirements linkage
- These challenges are not unique to the Software Engineering community

University of Limerick

On going questions

- We need to better understand the behaviour of Software Engineers
- Requirements gathering
  - Identify the information they seek
  - Identify the information they obtain
  - Identify the (in)consistencies
    - Information sought and found
    - Information sought but not found
- Develop better methods of evaluation

University of Limerick

Questions?