SEXTANT Software Exploration tool

발표자 : 장주연

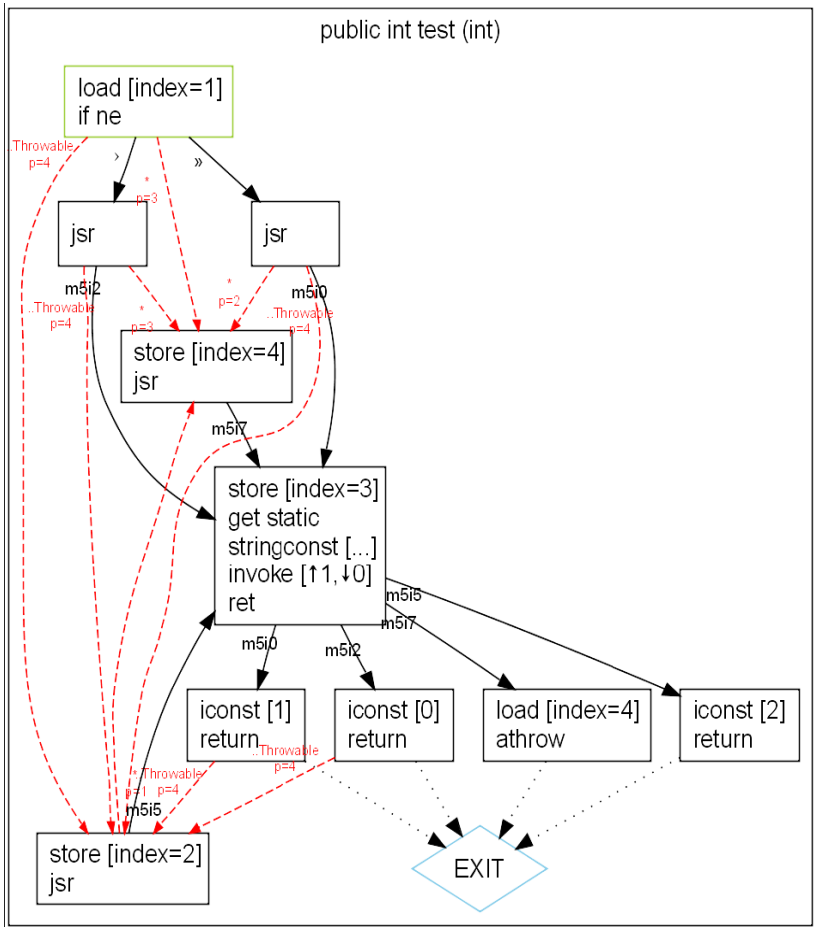
Contents

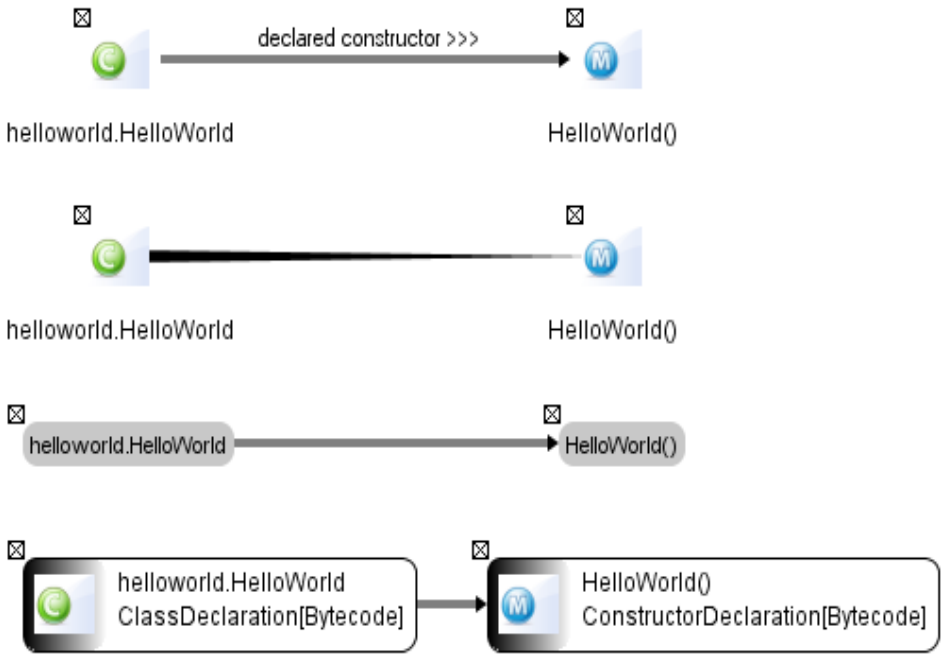
1. Introduction

Introduction

- software 유지보수의 중요성
  - 변화된 code에 대한 이해가 중요
- software tool의 기능
  - 기본적인 exploration 기능(searching과 browsing)
  - 이해하기 쉽게 하기 위한 visualization technique
- 이 논문의 포커스
  - visualization exploration tool의 5가지 functional 요구사항이 있음
  - 5개의 요구사항을 만족하는 software exploration tool인 SEXTANT을 보이고 개개의 요구조건에 어떻게 만족하는 지를 보여줌
  - SEXTANT와 다른 tool과의 비교
  - 논문에서 거론한 지식, functional/nonfunctional 요구사항과 연관된 SEXTANT를 거론

Introduction (cont.)





2. Functional Requirements

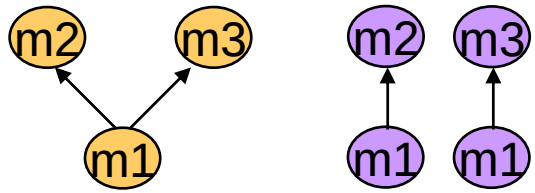
Functional Requirements

- R1 : Integrated Comprehension
  - 3가지 기본적인 software 분석 방법
    - bottom-up
    - top-down
    - mixed
  - 분석 방법 채택 방법
    - low level source code를 먼저 분석하려면 bottom-up
    - software의 초기 전제나 상위 단계로부터 분석하려면 top-down
    - bottom-up과 top-down 모두 사용은 mixed
    - 개발자의 경험, 프로그램의 특성에 따라 다른 분석 방법이 사용됨
    - 프로그램 분석을 top-down 방법으로 시작하여, bottom-up 방법으로 끝내기도 함
  - 위의 내용을 바탕으로 기존의 tool처럼 한가지 방법만 지원하는 것이 아니라, bottom-up과 top-down이 모두 지원되는 혼합된 방법을 제공

Functional Requirements (cont.)

- R2 : Cross-Artifact Support
  - 대부분의 exploration tool은 한 가지 artifact 만을 지원
  - 현대의 software 프로젝트는 source code, binary code, XML, script 등 다양한 artifact로 분석하는 기능이 제공되어야 함
  - 서로 다른 문서에 저장된 정보가 서로 밀접하게 관련이 되어 있을 수 있으므로, 다양한 artifact로 된 정보를 통합하여 고려하지 않으면 software 시스템을 정확하게 분석하기 어려움
  - 최근 여러 artifact를 지원하는 tool이 만들어짐
    - GSEE : multi source exploration을 지원

Functional Requirements (cont.)



- R3 : Explicit Representation
  - 대부분의 visualization tool은 mental map을 형성
  - software exploration tool은 exploration 경로에 따라 명확한 map을 만들어야 함
  - Eclipse
    - 하나의 element에서 다른 element로 exploration이 가능하도록 하이퍼링크를 사용할 수 있음
    - 클래스에서 그것의 superclass로 exploration이 가능하고, 함수가 어떻게 불리는지도 exploration 가능
    - 그러나 visual 적으로 그것에 대한 경로를 이해하기 힘듦
    - call hierarchy view같은 것을 사용함으로써, 간단한 관계에 대한 경로를 보는 것은 가능
    - 다른 종류의 관계를 보다 보면 이해하기 힘들 가능성이 큼
    - m1 함수가 m2와 m3에 의해 호출되면, m2로부터 m1, m3로부터 m1으로 경로가 m1이 중복되어 나타나도록 경로를 그림
  - software의 mental map을 생성할 때는, 참조 무결성(referential integrity)이 지원되도록 각 element는 한번만 나타나야 하고, 그것의 관계가 시각적으로 명확히 만들어져야 하며 하나의 통합된 뷰로 나타낼 수 있어야 함

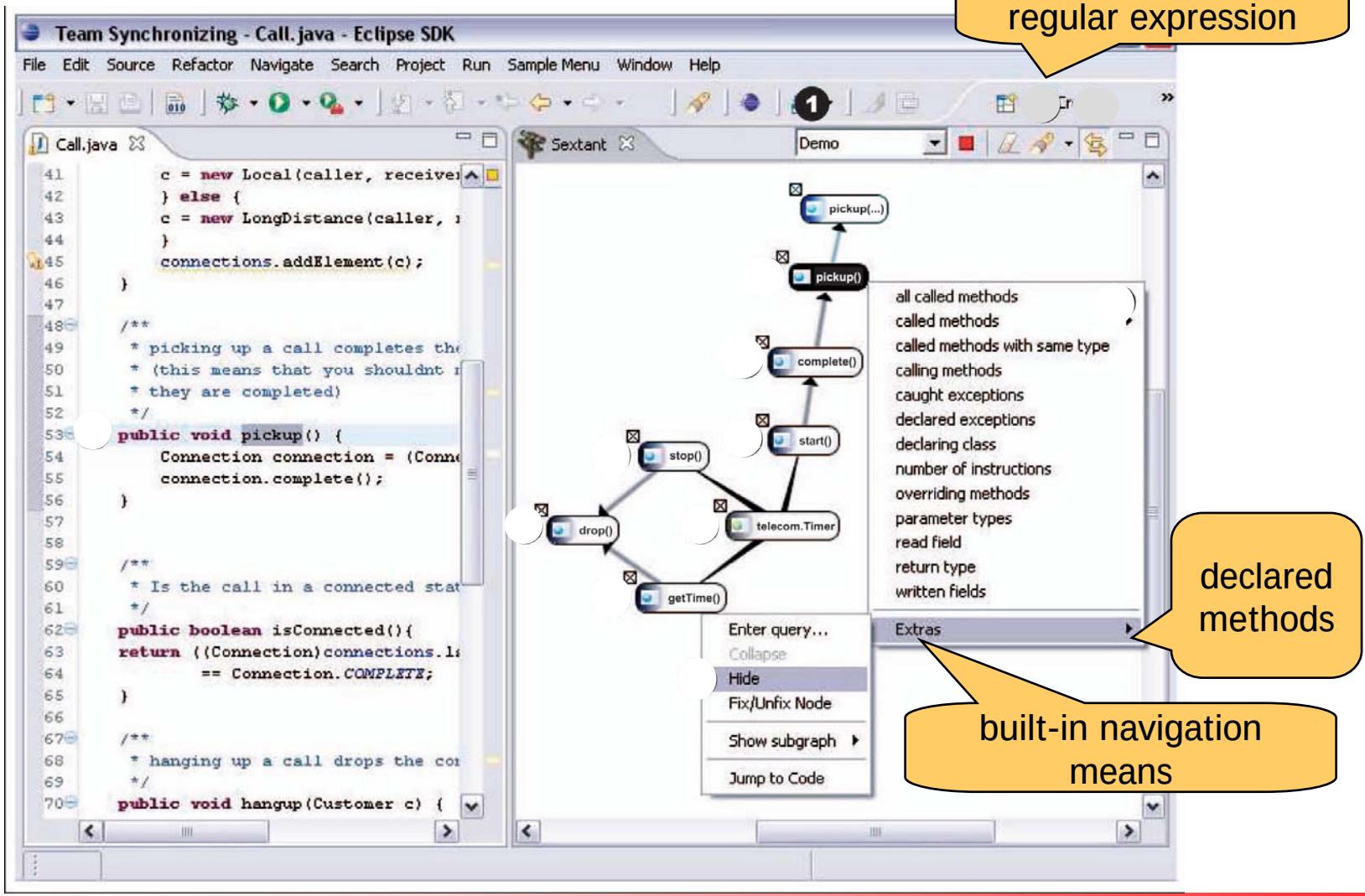
Functional Requirements (cont.)

- R4 : Extensibility
  - 개발자들은 보통의 relation을 따라 software element를 exploration
  - Java application인 경우, class, method, field로 exploration
  - 그러나, tool 개발자들은 이런 element를 모두 예측할 수 없기 때문에, 각 개발자가 확장할 수 있도록 extensibility가 보장되어야 함

- R5 : Traceability
  - source code traceability이나 수정이 필요
  - 가설의 확립이나 이해를 위해 source code를 읽는 것이 필요
  - source code와 대응되는 그래픽 주석이 잘 매칭된 traceability가 필요

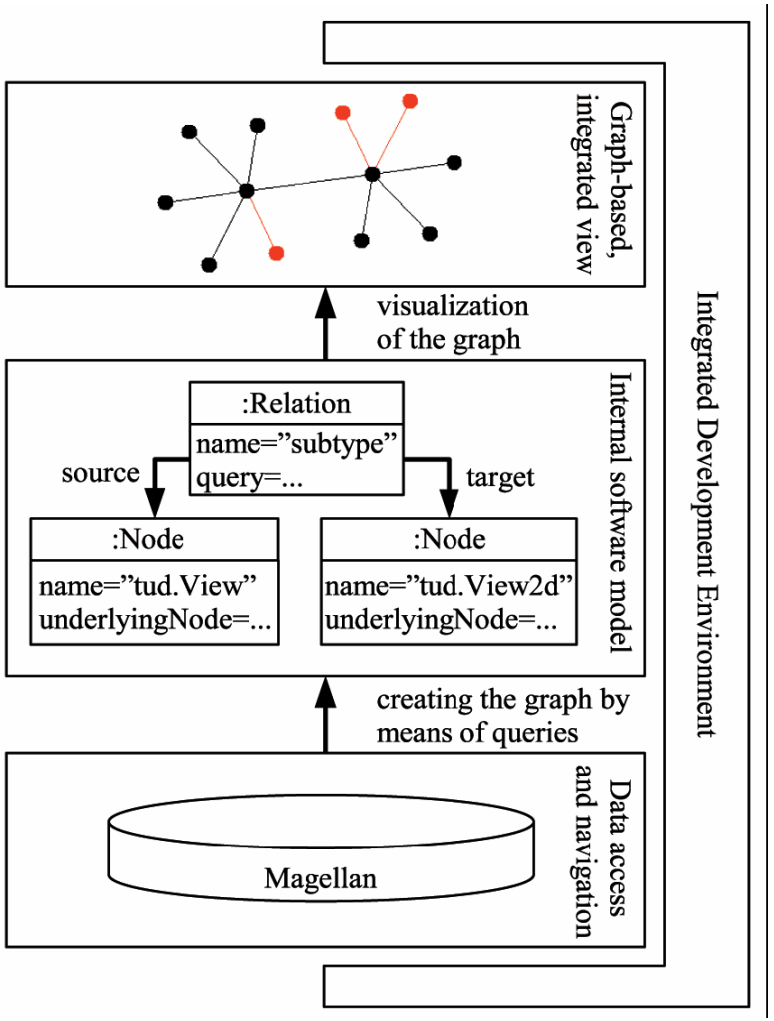
3. SEXTANT in Action

Example Scenario



4. Architecture

SEXTANT의 구조


Graph-based, integrated view
visualization of the graph
:Relation
name="subtype"
query=...
source
target
:Node
name="tud.View"
underlyingNode=...
:Node
name="tud.View2d"
underlyingNode=...
Internal software model
creating the graph by means of queries
Magellan
Data access and navigation
Integrated Development Environment

Building Blocks과 지원되는 요구사항

- R1: Integrated Comprehension
- R2: Cross-Artifact Support
- R3: Explicit Representation
- R4: Extensibility
- R5: Traceability

Building block / Requirement	R1	R2	R3	R4	R5
Data access and navigation	*	*		*	
Internal software model			*	*	
Graph-based, integrated view	*		*		
IDE integration					*

Data Access와 Navigation

- 역할
  - exploration하기 위한 software를 표현하는 raw data로의 접근을 제공하기 위함
  - query 기능을 제공
- cross-artifact information engineering platform Magellan에서 구성
- data 저장과 기본적인 접근
  - Magellan은 프로젝트의 모든 source를 변환
    - Java code, configuration files, XML descriptors 등을 XML 형식으로 변환하여 데이터베이스에 그것을 저장
- XML로 변환된 정보를 잘 이용하기 위하여 XQuery 사용
  - Search query, context-dependant query, query for derived information

Internal Software Model

- 내부적인 표현을 위해, raw data로부터 추상화된 일반적인 그래프 기반 모델을 사용
  - Java code -> XML : 트리 기반 구조
  - SEXTANT : 트리 표현 -> 그래프 기반 모델

- 각각의 변환
  - node
    - data store에 저장된 software element이거나 query 실행으로부터 반환된 요구된 정보
  - edge
    - query를 실행함으로써 표출되는 두 개의 node 사이의 relationship
  - element
    - data store에 대응되는 element에 대한 참조
    - multiple query에 의해 선택될 지라도 데이터베이스의 각 element가 그래프의 하나의 node에 표현되도록 함

Graph-Based Integrated View

- 통합된 뷰
  - element들은 node로, relation들은 edge로서 표현
  - 이는 searching과 browsing 사이의 switching을 가능케 함

- 통합된 뷰의 특징
  - 개발자가 software를 통해 browsing하고 새로운 프로그램 element를 발견한다면, exploration 그래프는 그것들을 나타내고 relation의 종류를 구체화
  - 그래프 기반인 것은 참조 무결성(referntial integrity)를 지키게 함

IDE Integration

- SEXTANT는 Eclipse IDE와 밀접하게 통합됨
  - software 프로젝트의 source의 synchronization에 매우 중요
  - Eclipse에 artifact가 추가/제거/변경됨에 따라 리소스 변경 이벤트를 보내고, Magellan 데이터베이스가 그에 상응하게 업데이트 됨
  - 그래픽 주석과 source code 표현 사이에 끊기지 않게 switching 되도록 의미를 제공함에 따라 visual layer에 한해 영향을 미침

- tool을 IDE와 통합화 함으로서 장점
  - 비슷한 모양새와 느낌의 사용
    - 새로운 tool을 학습하는 시간과 노력을 감소
  - IDE 내부에서 다른 tool 사이의 교환은 IDE와 외부 tool 사이의 교환보다 더 안정적

5. Case Studies

Case Studies

- 각각의 case study
  - 5명의 참가자에게 4개의 연구 사례를 증명하게 함
    - 컴퓨터 공학의 박사 3명과 석사 2명
    - 5년 이상 경력의 프로그래밍, 3년 이상의 Java 관련 지식, Eclipse를 사용한지 1년 이상, exploration tool을 사용하지 않음

- 각 연구 사례에서 토론된 요구사항

Case study / Requirement	R1	R2	R3	R4	R5
1: Analyzing a bug	*	*		*	
2: Refactoring			*		*
3: Concern exploration	*		*		*
4: Verifying a hypothesis	*			*	

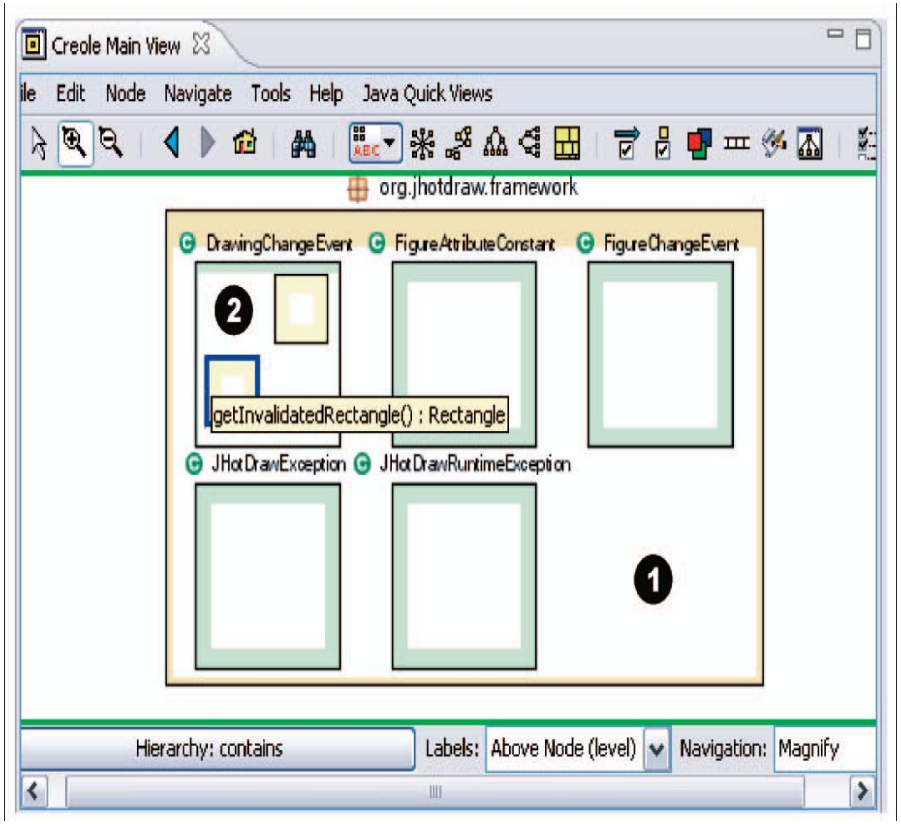
Setup

- case study할 때 참가자에게 할당한 tool
  - (C: CREOLE, J: JQUERY, S: SEXTANT)

- tasks와 subject system
  - 가장 작은 application(1000라인의 code)
  - 가장 큰 application(가장 작은 것의 200배)

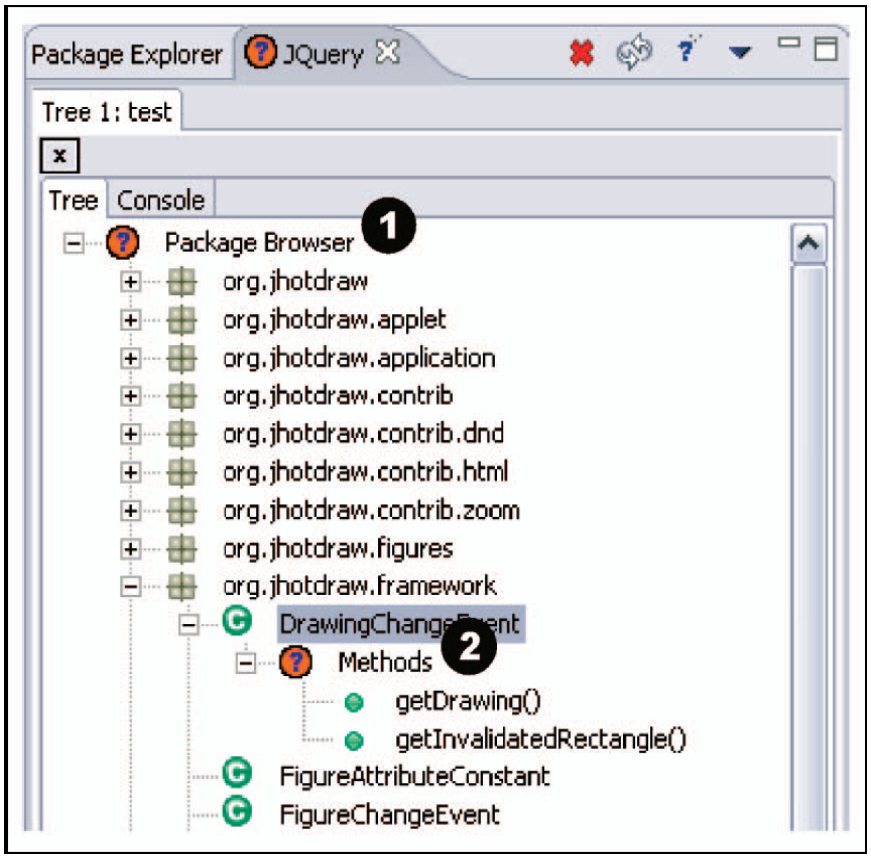
Case study / Participant	P1	P2	P3	P4	P5
1: Analyzing a bug	S	S	C	C	J
2: Refactoring	J	J	S	S	C
3: Concern exploration	C	C	J	S	S
4: Verifying a hypothesis	J	S	C	J	C

Creole



- Eclipse IDE로 SHriMP tool과 통합된 plug-in
- 중첩된 element가 중첩된 node로 보여짐(index : 1)
- 부모 node를 접거나 펼칠 수 있음 (index : 2)
- 상위 레벨의 추상화로 software를 그래픽화 하는 것과 하위레벨의 세부사항을 나타내는 것 모두 가능
- fish-eye view, pan, zoom과 같은 그래픽 기술을 이용 가능

JQuery



- query 기반의 장점과 계층적 browser tool의 장점을 혼합
- query로 시스템에서 element를 찾고, 미리 정의된 view를 생성하거나 다양한 형태의 relationship에 의하여 code를 exploration하기 위해 사용됨
- 그 결과가 tree 기반 계층적 표현으로 그래픽화 됨(index :1)
- 각 결과 element는 source element의 subtree로 구축되어 새로운 query의 source로서 사용될 수 있음(index : 2)

Setup (cont.)

- 각 tool과 만족되는 요구사항

Tool / Requirement	R1	R2	R3	R4	R5
Creole			*		
JQuery	*			*	*
SEXTANT	*	*	*	*	*

- 각각의 tool의 focus
  - 둘 다 software exploration을 지원
  - 필수적인 기능을 모두 제공하는 것이 아니라 소수만 제공
  - Creole은 representation
  - Jquery는 querying
  - GSEE를 포함시키려고 했으나 저작권법 상 하지 못함

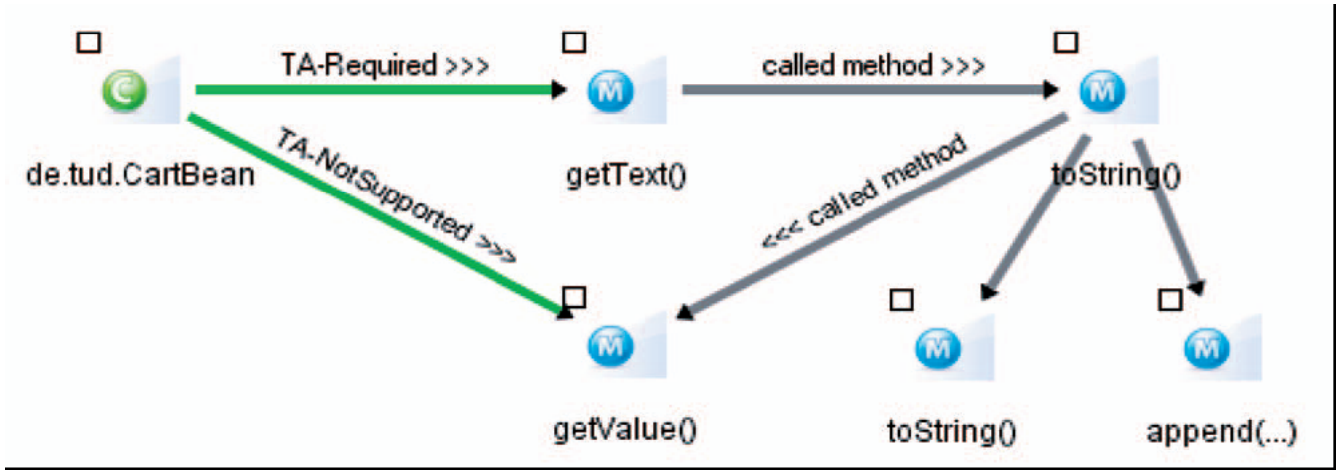
Analyzing a Bug in an EJB Project

- task
  - transaction을 지원하지 않는 method를 호출하도록 bug 생성
  - bug를 찾기 위해 다음을 이해해야 함
    - component의 구현의 control flow
    - 호출된 method의 transaction 속성

- observations
  - P5를 제외한 참가자가 bug를 발견
  - 모든 참가자가 component의 구현에서 bug 발견
  - domain-specific query가 EJB name에 기초되어 SEXTANT의 사용자가 더 쉽게 찾음
  - Creole과 JQuery는 EJB-specific query에 지원이 부족

Analyzing a Bug in an EJB Project (cont.)

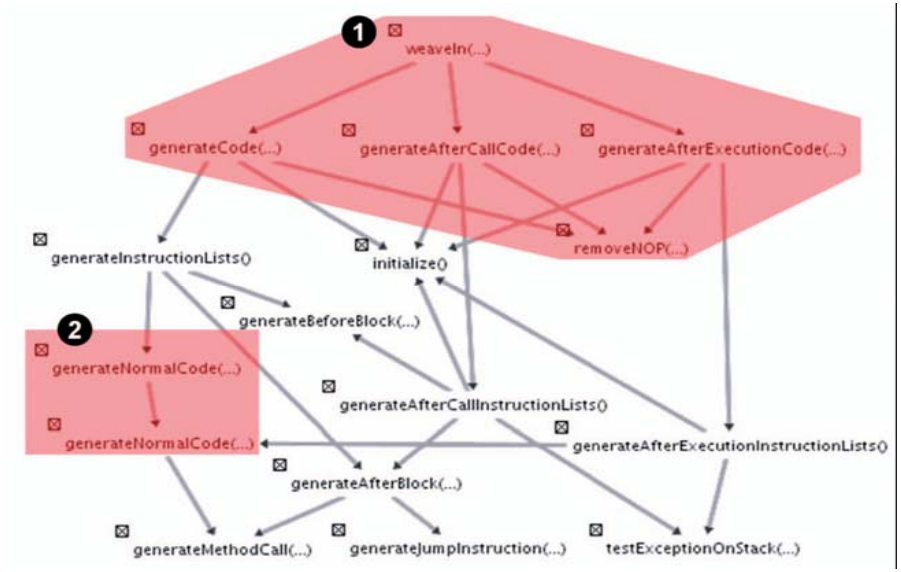
- source의 bug를 보여주는 exploration graph



- result
  - JQuery는 확장성이 있지만, R2에 대한 지원 부족
  - JQuery, Creole 둘 다, 분석 방법을 switching하는 것을 지원하지 않음
    - call graph의 exploration 동안 method의 transaction 설정을 찾는 것
  - SEXTANT는 call graph에서 high-level의 정보로 통합하는 것이 가능

Finding Refactoring Opportunities

- task
  - weaveIn method의 내부적 module 구조를 이해
- observation

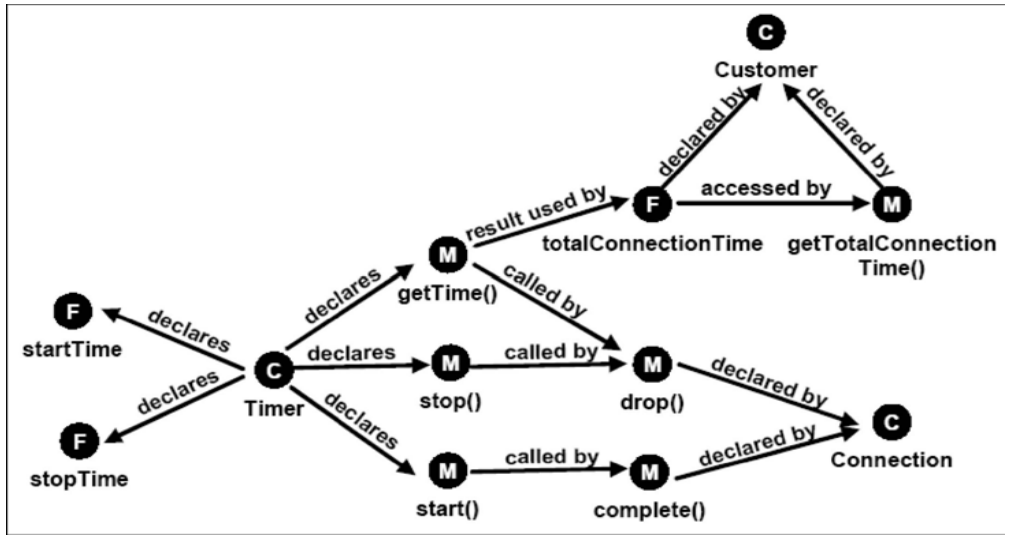


- result
  - bad smell
  - 목적
    - mental map을 포함하고 생성
    - exploration된 element와 relation에 대한 설명
  - JQuery와 SEXTANT만이 graphical 표현과 code 사이의 switching에 오류가 없음

Concern Exploration

- task
  - 연관된 프로그램 element와 relationship을 분석

- observation

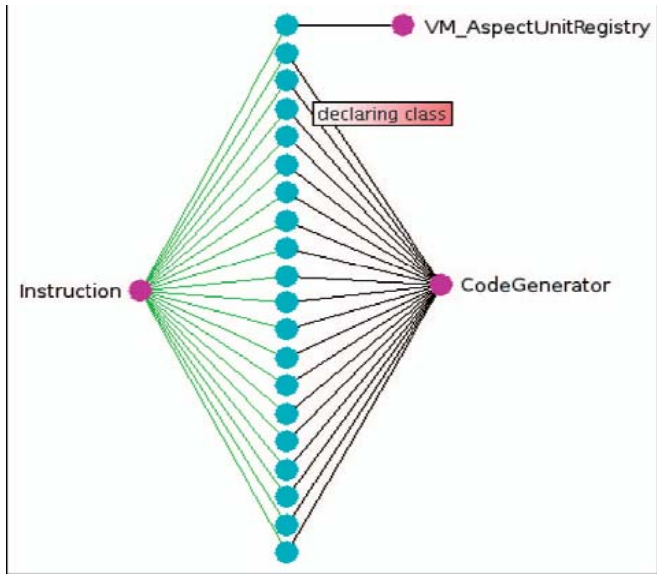


- result
  - searching과 browsing이 둘 다 필요하다는 것을 보여줌
  - source code로 graphical 표현을 보충하는 것이 필요

Verifying a Hypothesis

- task
  - 명령어들이 생성된 위치와 지역성 제한이 지켜지고 있는지를 분석

- observation



- result
  - 다른 추상화 레벨로 통합된 분석이 필요
  - Creole – 선언된 클래스와 생성된 instruction을 통합하는 것이 불가능
  - JQuery, SEXTANT – *Instruction* object를 생성함으로 통합화 가능

Synthesis and Threats to Validity

- 4가지 case study로 5개의 요구사항을 조사
- SEXTANT와 같은 5가지 요구사항을 모두 만족시키는 분석 tool이 다양한 분석 작업을 하는데 효과적임
- 현재 이 논문의 focus는 software system의 overview를 생성하는 것이라기 보다는 exploration을 지원하는 데 있음
  - Creole과 같은 경우, SEXTANT보다 전체적인 시스템의 구조를 이해하는 데 효과적임

6. Comparison to Other Requirements

Tool-Specific Cognitive Design Elements

- Bottom-up comprehension
  - 3가지 주요 역할
    - software element와 그것들의 relation의 증명
    - 위치적 특성을 제거한 플랜에서의 code browsing
    - 낮은 단계의 정보를 높은 단계 추상화로 덩어리화 함

- Top-down comprehension
  - 가설을 빠르게 입증해야 할 경우 사용됨
  - SEXTANT는 다양한 search query를 써서 가설을 입증하거나 가설이 입증되지 않음을 증명하기 위하여 단서 찾는 것을 가능하게 함

- Integrated comprehension
  - 통합된 분석으로 인해 개발자가 “서로 참조된 다양한 mental model을 표현하는 링크된 뷰”를 구성하는 것을 가능케 함

Tool-Specific Cognitive Design Elements (cont.)

- Navigation support
  - 사용자가 여러 종류의 관계를 browsing 할 수 있도록 방향성 exploration을 제공
  - application이나 사용자 정의 링크를 따라 도달하지 않아도 위치로 exploration할 수 있는 “arbitrary navigation”을 지원

- Orientation clues
  - 개발자가 focusing하고 있는 node를 highlight
  - 그래프는 exploration process의 map과 통신

- Reduction of disorientation
  - 통합 뷰를 사용하여 방향성을 잃는 것을 감소
  - SEXTANT는 현재 두 가지의 다른 레이아웃을 제공하고, node와 edge를 customize하여 표현 가능

General Cognitive Design Requirements

- Zayour와 Lethbridge는 reverse engineering tool의 낮은 채택 문제를 조사

- Short-term memory utilization
  - artifact를 쉽게 이용할 수 있어야 함
  - 새로운 정보가 기존 정보로 의미 있게 링크되어야 함
  - exploration 동안 불확실한 것은 제거되어야 함

- Short-term memory fading
  - STM에 artifact들이 포함되는 시간을 최소화함으로써 STM fading을 감소
  - 대부분의 시간은 하나의 단계에서 관련된 element를 찾아오는데 소비됨
  - SEXTANT에서는 모든 exploration된 element가 그래프에 명확히 나타나 있고, 그 exploration 그래프로부터 관련된 element를 재구성 할 수 있디 때문에 이것이 해결됨

Functional and Nonfunctional Requirements

- J. Singer는 “An Examination of Software Engineering Work Practices” 논문에서 3개의 functional requirement와 몇 개의 nonfunctional requirement를 포함하는 software exploration tool을 위한 핵심 요구사항의 목록을 생성

- Functional requirements
  - search 능력이 exploration의 중요한 element
  - 주어진 관련된 모든 element들 사이에 명확한 관계가 이루어져야 함
  - 지속적인 search history가 요구됨

- Nonfunctional requirements
  - 대부분의 query가 알아챌 수 없을 정도로 적은 delay가 있어야 함
  - 다양한 프로그래밍 언어로 쓰여진 source code를 수용할 수 있어야 함
  - 다른 tool로의 상호이용이 가능해야 함

Comparison to Other Requirements

- 이와 같이, SEXTANT가 일반적인 분석 tool을 위한 요구사항을 대부분 만족

- SEXTANT의 기능
  - bottom-up, top-down, integrated comprehension을 지원
  - query 기반 접근이 exploration을 기능화하고, 하나의 통합화된 뷰를 사용 함으로서 정보 손실 없이 exploration 그래프를 명확하게 나타냄
  - 요청이 있을 때마다 exploration 접근을 하기 때문에 software의 모든 element를 보여줄 경우, short-term memory load와 fading을 최소화 할 수 있음
  - 대부분의 functional/nonfunctional 요구사항을 지원

7. Discussion and Future Work

Discussion and Future Work

- Scalability
  - 대형 크기의 프로젝트의 수용을 위해 필요
- Extensibility
  - 현재 Java software만을 지원, 다른 언어로의 지원도 필요
- Storing Explorations
  - exploration을 한 정보를 저장할 수 있어야 함
- Flexible Visualization
  - graph나 tree, layout, node와 relationship의 모양을 사용자가 결정할 수 있어야 함
- Improving Comprehension Support
  - SEXTANT의 query 기반 접근이 분석 기술을 더 증가시키리라 생각됨

8. Related Work

RELATED WORK

- Rigi
  - 모든 subsystem이 계층화되어 나타남
  - 접근하는데 multiwindow를 사용
- SHriMP, Creole
  - fish-eye view, pan, zoom과 같은 유명한 시각적 tool로 하나의 view를 지원
- Hy+, Ciao
  - advanced query mechanism을 지원
- Feat
  - 모든 연관된 element와 관계가 그래프 표현으로 잘 추상화됨
- JQuery
  - SEXTANT와 비슷한 tool
  - tree 기반 표현

9. Summary

SUMMARY

- software exploration tool을 위한 5가지 기능적인 요구사항
- 이 요구사항이 개발자들이 software를 이해하기 쉽도록 효과적으로 도움을 준다는 것을 보임. 이 증명에 따라, 5가지 요구사항이 지원되는 tool인 SEXTANT를 보임
- 다른 분석 tool에 의해 지원되는 개개의 요구사항이 있으나, SEXTANT에 의해서는 기능이 혼합되어 제공됨
- 4개의 case study를 통해 요구사항의 부분만을 지원하는 tool과 비교