규칙 기반 시스템을 이용한 테스트 에이전트 시스템 구현

최정은 ° ,최병주
이화여자대학교 컴퓨터학과

Implementation of the Test Agent System with Rule-Based System

Jeongeun Choi, Byoungju Choi
Computer Science & Engineering, Ewha Womans University

요 약

테스터의 간섭 없이 테스트를 수행 할 수 있는 새로운 테스트 도구인 테스트 에이전트 시스템을 구현하였다. 이 시스템에서 테스터는 대스터 이름, 데스트 시간을 입력하고, 데스트 대상을 선택 후, 그 밖에 테스트에 필요한 정보를 입력하고 나면 테스트 에이전트 시스템이 스스로 테스트를 수행한 후 각 테스트 레벨에 대한 테스트 항목의 테스트케이스와 Quality 측정치를 테스트 결과로 받아 볼 수 있다 이 테스트 에이전트 시스템은 3개의 에이전트로 구성이 되어 있고 각 에이전트들은 에이전트의 특성인 자율성, 사회성, 지능성을 가지고 있다 특히 지능성을 나타내 주는 것을 규칙 기반 시스템을 이용하여 구현하였다. 'User Interface Agent'에는 '리그레션 테스트 대상 판단 규칙'과 'Test History 크기 관리 규칙'이 있고, 'Test Case Selection & Testing Agent'에는 테스트케이스를 선택하는 데에 적용하는 '중복성 제거 규칙'과 '일관성 있는 테스트케이스 선택 규칙'이 있다 'Regression Test Agent'에는 '리그레션 테스트 관련 항목 찾는 규칙'이 있어 각 에이전트들의 지능성을 뒷받침 해 준다 본 논문에서는 각 규칙들을 술어 논리로 표현하여 제시하였고, 구현한 테스트 에이전트 시스템의 Prototype을 기술한다.

1. 개 요

테스트 작업을 편리하게 하기 위하여 테스터가 테스트를 수행하는 데에 이용되는 테스트 도구로써 테스트 기법에 따른 적정성 측정, 디버깅, 특정 테스트 기법을 적용하여 테스트케이스 자동 생성하는 데스트 도구들이 있다. 하지만, 단위 테스트로부터 시스템 테스트까지 점진적으로 테스트를 수행할 수 있는 테스트 통합 환경은 많지가 않다.

또, 자동화된 데스트가 테스트에 도움을 주기는 하지만, 자동화된 테스트 도구는 여전히 테스터가 테스트 수행에 간섭을 해야한다. 그리고 자동화된 테스트 도구로 데스트케이스를 선택하여 테스트를 수행하는 데에 불필요한 시간이 걸리게 된다. 따라서 테스터의 간섭 없이도 스스로 판단하여 테스팅을 수행할 수 있는 테스트 도구가 소프트웨어 개발 현장에 필수적이다.

본 논문에서는 단위 테스트로부터 시스템 테스트까지 필요한 데스터의 작업을 대행 해 주고, 자율적인 판단 능력과 에이전트들 간의 통신을 하는 사회성 및 추론을 하는 지능성을 갖춘 데스트 에이전트 시스템[1]을 설계하고 구현하였다. 이 테스트 에이전트 시스템[1]은 입력된 데스트 대상 프로그램에 대한 적절한 테스트케이스 와 그 Quality 수치를 결과로 출력하고, 테스터와 정보를 주고받는 "User Interface Agent", 테스트케이스를 선택하는 "Test Case Select & Testing Agent", 리그래션 테스트를 담당하는 "Regression Test Agent"등 3개의 에이전트로 구성이 되어 있다. 각 3개의 에이전트들도 에이전트의 특성인 자율성, 사회성, 지능성을 갖추고 있다. 전체 테스트 에이전트 시스템의 구조 및 에이전트로써의 특성은 논문[1,6]에 기술되어 있으며 본 논문에서는 각 에이전트의 특성 중 지능성을 위한 규칙들을 중심으로 기술하겠다.

본 논문의 구성은 다음과 같다. 2장에서는 테스트 에이전트 시스템의 지능성을 나타내 주는 규칙들을 각 에이전트 별로 술어 논리[2]로 기술하였고, 3장에서 실제 테스트 에이전트 시스템을 구현한 것의 prototype을 기술하고, 4장에서 결론 및 향후 연구과제를 제시한다.

2. 지능성을 나타내는 규칙

위에서도 언급하였지만 데스트 에이전트 시스템은 3개의 에이전트 즉, "User Interface Agent","Test Case Selection & Testing Agent","Regression Test Agent"로 구성이 되어 있다. 테스트 에이전트 시스템의 전체 구성은 그림 1과 같고, 각 에이전트들의 지능성을 나타내는 규칙들을 중심으로 기술하였다. 즉, User Interface Agent의 '리그레션 대상 판단 규칙(RTTD-Rule)'과 'Test History 크기 관리 규칙(THSM-Rule)', Test Case Selection & Testing Agent의 '중복 제거 규칙(RE-Rule)'과 '일관성 있는 테스트케이스 선택 규칙(CTS-Rule)', 그리고 Regression Test Agent의 '리그래션 데스트 관련 항목 찾는 규칙(RRTIS-Rule)'이 나타난다

이 장에서는 각 에이전트 별로 지능성을 나타내는 규칙들에 대해 기술하겠다

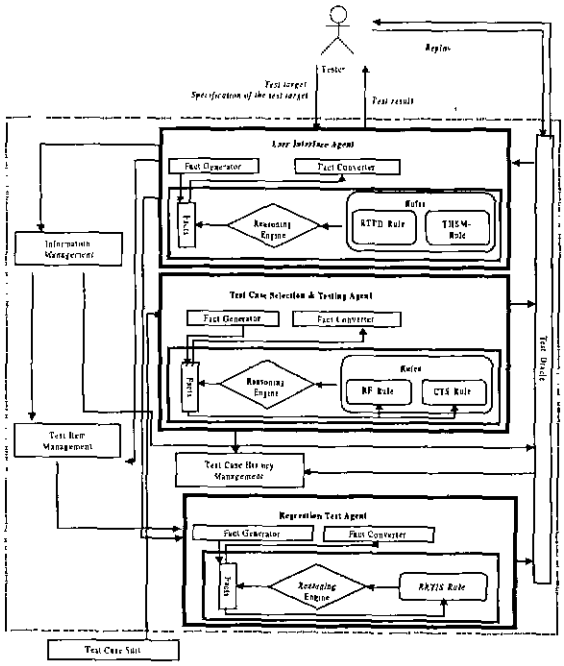


그림 1 Test Agent System

2.1 User Interface Agent

User Interface Agent에서 지능성을 나타내는 규칙은 리그래션 테스트 대상 판단 규칙과 Test History 크기 관리 규칙이 있고, 각 규칙을 술어 논리[2]로 표현하였다

2.1.1 리그래션 테스트 대상 판단 규칙(RTTD-Rule)

테스트 대상이 Test History로부터 리그래션 테스트 대상인지를 판단하는 규칙이다.

<술어 논리로 표현된 리그래션 테스트 대상 판단 규칙>

- $\forall x, y,$ Test-target $(x) \wedge$ History$(y) \wedge$ Member$(x,$ History$(y)) \supset$ Reg-target(x)
- $\forall x, y,$ Test-target $(x) \wedge$ History$(y) \wedge \neg$ Member$(x,$ History$(y)) \supset$ NReg-target(x)

2.1.2 Test History 크기 관리 규칙(THSM-Rule)

테스트 에이전트 시스템에서는 테스트 대상을 리그래션 테스트 발생률에 따라 Test History에 남겨지는 기간을 차별화한다. 그래서 메모리의 크기를 효율적으로 관리 할 수 있도록 한다. 예를 들면, 리그래션 테스트 발생률이 가장 높은 그룹은 Test History에 남겨지는 기간을 30일, 두 번째로 리그래이션 테스트 발생율이 높은 그룹은 20일, 그 외는10일로 할 수 있다. 이 경우 Test History 크기 관리 규칙을 규칙 기반 시스템으로 구현하는 것은 "termdecision.clp"에서 fact를 발생률에 따라 정렬시키고 정렬된 순서로 Test History에 남겨지는 기간을 추론하게 된다, 그리고 "deletetarget.clp"에서 Test History에 기록되어 있는 날짜와 정해진 Term과 현재 날짜를 비교하여 지난 것은 삭제를 해 준다. 규칙 기반 시스템으로 구현된 Test History 크기 관리 규칙을 술어 논리로 표현 한 것은 다음과 같다.

<술어 논리로 표현된 Test History 크기 관리 규칙>

- $\forall x,$ Occurrence-order1$(x) \supset$ Due$(x, 30)$
- $\forall x,$ Occurrence-order2$(x) \supset$ Due$(x, 20)$
- $\forall x,$ Occurrence-order3$(x) \supset$ Due$(x, 10)$

2.2 Test Case Selection & Testing Agent

Test Case Selection & Testing Agent는 테스트케이스를 선택하여 테스트를 수행하는 에이전트이다. 이 에이전트의 지능성을 나타내는 규칙은 '중복 제거 규칙'과 '일관성 있는 테스트케이스 선택 규칙'이 있다. 객체지향 테스트 프로세스[3,4]에서 테스트케이스는 '메소드 순서에 의한 테스트케이스'와 실제 값에 해당되는 테스트 케이스로 2가지의 타입으로 나누어진다. 2가지 타입의 테스트케이스에 따라 각각 '중복 제거 규칙'과 '일관성 있는 테스트케이스 선택 규칙'이 적용되어 테스트케이스를 추론한다. 이 두 규칙을 술어 논리[2]로 표현한 것을 기술하겠다.

2.2.1 중복 제거 규칙(RE-Rule)

다량의 테스트 케이스에서 테스트를 하지 않아도 테스트 대상을 평가하는 데에 영향을 미치지 않는 경우도 생기게 된다. 이에 착안하여 메소드 순서에 의한 테스트 케이스 중에서 테스트케이스의 사이의 중복성을 제거하는 규칙을 세웠다.

'중복 제거 규칙'을 규칙 기반 시스템으로 구현한 것을 술어 논리로 표현 한 것은 다음과 같다.

<술어 논리로 표현된 중복성 제거 규칙>

- $\forall x, y,$ (Type1-testcase1$(x) \wedge$ Type1-testcase2 $(y) \wedge$ Subset(x, y)) $\supset$ Redundancy(x, y)
- $\forall x, y,$ (Type1-testcase1$(x) \wedge$ Type1-testcase2 $(y) \wedge$ Subset(y, x)) $\supset$ ReverseRedundancy(y, x)
- $\forall x, y,$ (Type1-testcase1$(x) \wedge$ Type1-testcase2$(y) \wedge \neg$(Subset(x, y)) $\wedge \neg$(Subset(y, x))) $\supset$ NRedundancy-testcase(x)

2.2.2 일관성 있는 테스트케이스 선택 규칙(CTS-Rule)

메소드 순서 테스트케이스에서 메소드들의 실제 값의 테스트 케이스를 일관성 있게 선택하여 주는 것이 중요하다. 서로 다른 테스트케이스의 값을 선택하였을 경우, 충분히 옳은 결과가 나올 수 있는데, 틀린 결과가 나와 오류가 있다고 표시 될 수도 있다. 이를 막기 위하여 메소드 사이의 실제 값의 해당하는 테스트케이스들을 일관성 있게 선택하여 주어야한다. 이에 착안하여 일관성 있게 테스트 케이스를 선택하는 방법을 규칙을 세워 표현하였다.

'일관성있는 테스트 케이스 선택 규칙'을 규칙 기반 시스템으로 구현한 것을 술어 논리로 표현한 것은 다음과 같다

<술어 논리로 표현된 일관성 있는 테스트케이스 선택 규칙>

- $\forall x, y,$ (Type2-testcase1$(x) \wedge$ Type2-testcase2(y) $\wedge$ Equal$(x,$ First$(y)) \wedge$ Subset$(x, y)) \supset$ Consistency-testcase(x, y)
- $\forall x, y,$ (Type2-testcase1$(x) \wedge$ Type2-testcase2(y) $\wedge$ Member$(x,$ Rest$(y)) \wedge$ Subset$(x, y)) \supset$ Consistency-testcase(x, y)
- $\forall x, y,$ (Type2-testcase1$(x) \wedge$ Type2-testcase2(y) $\wedge$ Equal$(y,$ First$(x)) \wedge$ Subset$(y, x)) \supset$ Consistency-testcase(x, y)
- $\forall x, y,$ (Type2-testcase1$(x) \wedge$ Type2-testcase2(y) $\wedge$ Member$(y,$ Rest$(x)) \wedge$ Subset$(y, x)) \supset$ Consistency-testcase(x, y)

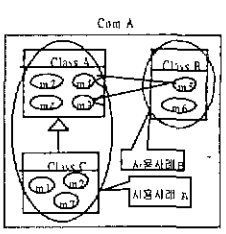
2.3 Regression Test Agent

*Regression Test Agent*는 리그래션 테스트를 수행하는 에이전트로 이 에이전트의 지능성을 나타내는 규칙은 '리그래션 테스트 관련 항목 찾는 규칙'(*RRTIS-Rule*)이 있다.

리그래션 테스트를 수행시 리그래션 테스트 항목을 테스터가 한 항목을 선택하면 선택한 항목과 관련된 항목까지도 찾아 리그래션 테스트를 수행 할 수 있게 한다. 즉, 관련된 테스트 항목을 찾는다는 것은 입력된 리그래션 테스트 항목이 속한 클래스, 컴포넌트, 사용사례를 찾아 입력된 것 뿐만 아니라, 속해 있는 것에 대한 테스트도 행하는 것이다.

예를 들면, 그림 2에서 리그래션 테스트 항목으로 메소드 m1이 입력되었다. 그림 3과 같이 찾아가면서 메소드 m1이

속해 있는 클래스 Class A, 메소드 m1이 Class C에 상속되어 있으므로 Class C, 메소드 m1을 호출하는 메소드 m5, 메소드 m1이 속해 있는 컴포넌트 Com A, 사용사례 A가 관련된 테스트 항목으로 검색된다



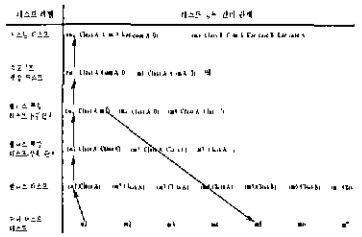


그림 2 Class Diagram Example　　그림 3 테스트 항목간의 관계

입력된 리그래션 테스트 항목과 관련된 항목을 하이퍼 링크식[5]으로 관련되어 있는 것을 검색하는 리그래션 테스트 관련 항목 찾는 규칙을 규칙 기반 시스템으로 구현한 것을 술어 논리로 표현한 것은 다음과 같다.

<술어 논리로 표현된 리그래션 테스트 관련 항목 찾는 규칙>

Intra-method test level, Class-test level
- $\forall x, y, (\text{RegtestItem}(x) \wedge \text{Relatedlist}(y) \wedge \text{Subset}(x, \text{Relatedlist}(y)) \supset \text{Related-regtestItem}(y)$
- $\forall x, y, (\text{RegtestItem}(x) \wedge \text{Relatedlist}(y) \wedge \text{Subset}(\text{Relatedlist}(y), x)) \supset \text{Related-regtestItem}(y)$

Inter-class test level-inheritance
- $\forall x, y, z, (\text{Regtestitem}(x) \wedge \text{Related-regtestitem}(y) \wedge \text{Relatedlist}(z) \wedge \text{Equal}(\text{Last}(z), \text{Seconedelement}(y)) \supset \text{Related-regtestItem}(z)$

Inter-class test level-association, Component integration-test level
- $\forall x, y, z, (\text{RegtestItem}(x) \wedge \text{Related-regtestItem}(y) \wedge \text{Relatedlist}(z) \wedge \text{Equal}(\text{First}(z), \text{Last}(y)) \supset \text{Related-regtestItem}(z)$

System-test level
- $\forall x, y, z, (\text{RegtestItem}(x) \wedge \text{Related-regtestItem}(y) \wedge \text{Relatedlist}(z) \wedge \text{Equal}(\text{Fourthelement}(z), \text{Last}(y)) \supset \text{Related-regtestItem}(z)$

4 Prototype

본 논문에서 구현한 테스트 에이전트 시스템은 Windows환경에서 JDK1.2.1(Java Development Kit)로 구현하였고, 에이전트의 지능성을 나타내는 User Interface Agent에서의 리그래션 테스트 대상 판단 규칙과 Test History 크기 관리 규칙, Test Case Select & Testing Agent에서의 중복 제거 규칙과 일관성 있는 테스트케이스 선택 규칙, 그리고 Regression Test Agent에서의 리그래션 테스트 관련 항목 찾는 규칙은 자바와 연동이 용이한 JESS4.4(Java Expert System Shell) 전문가 시스템을 이용하여 구현하였다. 자바에서 데이터베이스 접속 방법으로는 JDBC (Java DataBase Connectivity)를 사용하였다

본 절에서는 구현된 테스트 에이전트 시스템의 Prototype을 보이겠다 그림 4는 Test Agent System의 흐름을 나타낸 그림이다. 그림 5는 그림 4에 맞추어 테스트 에이전트 시스템이 수행되는 과정의 일부 예를 화면으로 보인 것이다.

시작 화면에서 테스터 이름과 테스트 시간을 입력하고 테스트 대상을 선택하는 것까지가 테스터가 할 일이다.(그림 5의 ①, ②) 그 후의 행동은 테스트 에이전트 시스템에 있는 에이전트들이 스스로 테스트를 진행시킨다 (그림 5의 ④, ⑤, ⑧) 테스터는 최종적으로 테스트 결과를 테스트 에이전트 시스템으로부터 받는다.(그림 5의 ⑨) 테스트 결과 화면에서는 각 테스트 레벨에 대한 테스트 항목의 테스트케이스와 테스트 수행의 Quality 측정치를 제공한다.

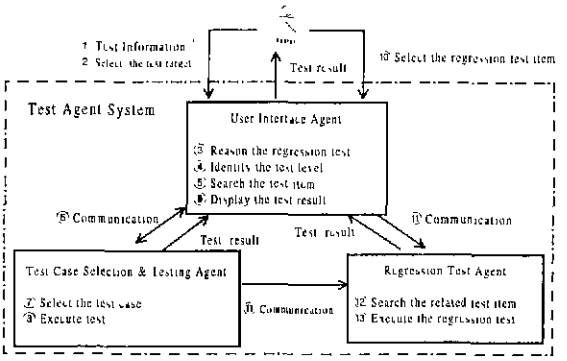


그림 4 *Flow of the Test Agent System*

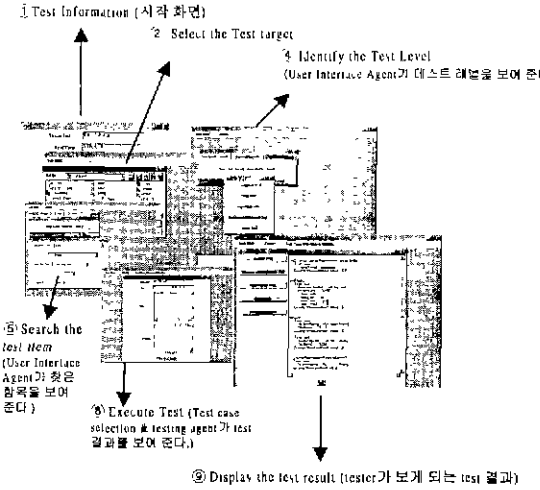


그림 5 테스트 에이전트 수행 화면

5 결론 및 향후 연구 과제

에이전트의 자율성, 사회성, 지능성을 갖춘 새로운 테스트 도구인 테스트 에이전트 시스템을 구현하였다. 본 논문에서 제시한 테스트 에이전트 시스템은 테스터의 간섭 없이 단위 테스트로부터 시스템 테스트까지 테스팅에 필요한 작업을 스스로 판단하여 수행하는 테스트 도구이다.

현재 테스트 에이전트 시스템의 장점을 부각 시킬 수 있는 실험이 진행중이다.

[참고문헌]

[1]최경은, 최병주,"테스트 에이전트 시스템 설계", 한국 정보과학회 '99 봄 학술발표논문집((26, 1), pp 614-616
[2]Thomas Dean, James Allen, Yiannis Aloimonos, Artificial intelligence Theory and Practice, The Benjamin/Cummings publishing company, pp71-119, 1995
[3]객체지향 소프트웨어 개발을 위한 체계적 테스트 프로세스, 노미나, 석사학위 논문, 이화여자대학교 컴퓨터학과, 1997
[4]Test Process in the Object-oriented Software Development Life Cycle bases on the Test Standards, Mina Rho, Byoungju Choi, Proceedings of Asia-Pacific Workshop on Software Process Improvement. pp17-32. 1997
[5]Thorsten Joachims, Tom Mitchell, Dayne Freitag, and Robert Armstrong, WebWatcher: Machine Learning and Hypertext, May, 1995.
[6]J.Choi & B.Choi. "Test Agent System Design", 8th IEEE Int Fuzzy Systems Conf. Proceedings, pp326-331, 1999 8