에이전트 기반의 객체지향 소프트웨어 테스트 방안
(Agent Based Object Oriented Software Test Technique)

최 정 은 † 최 병 주 ††
(Jeongeun Choi) (Byoungju Choi)

요 약 컴퓨터 분야에서 에이전트의 개념은 전자 상거래, 정보 검색과 같은 많은 어플리케이션들에 응용되어 중요 시 되고 있다. 하지만, 아직까지 지능성을 가진 테스트 도구는 없었다. 이 논문에서 제안하는 테스트 에이전트 시스템은 에이전트의 특성을 가지고 테스터를 도와주는 테스트 도구이다. 테스트 에이전트 시스템은 객체지향 테스트 프로세스를 따라 테스터의 일을 대행해 주고, 테스터의 간섭을 최소화 시켜 준다. 이 시스템은 자동 생성된 많은 양의 테스트케이스에서 중복이 없고 일관성 있는 테스트케이스를 지능적으로 선택하여 테스트 시간을 단축시켜 준다. 테스트 에이전트 시스템은 3개의 에이전트 User Interface Agent, Test Case Selection & Testing Agent, Regression Test Agent로 구성된다. 특히 Test Case Selection & Testing Agent은 RE-Rule과 CTS-Rule을 통하여 중복이 없고 일관성 있는 테스트케이스를 지능적으로 선택하며, Regression Test Agent는 RRTIS-Rule을 통해 리그래션 테스트 항목을 지능적으로 선택한다.

Abstract The concept of an agent has become important in computer science and has been applied to the number of application domains such as electronic commerce and information retrieval. However, to our knowledge, no one has yet proposed a test tool using the intelligent agent technology. In this paper, we propose a Test Agent System (TAS) that employs intelligent agent techniques to provide active assistance to the tester. TAS carries out the testers job by employing the object-oriented test process and therefore minimizes tester interference. This system also reduces testing time by intellectually selecting redundant-free and consistent test cases from the many test cases automatically produced by the system. The test agent system consists of the User Interface Agent, the Test Case Selection & Testing Agent, and the Regression Test Agent. The Test Case Selection & Testing Agent intellectually selects redundant-free and consistent test cases through the RE-Rule and the CTS-Rule while the Regression Test Agent also intellectually selects regression test items through the RRTIS-Rule.

1. 서 론

소프트웨어의 신뢰성 향상과 품질 측정을 위해서 테스트 작업은 반드시 이루어져야 한다. 이 때 테스터가 하는 일로는 테스트 대상에 대해 테스트를 수행하기 위한 계획을 세우고, 테스트 선정 기준에 따라 테스트케이스를 선택하고, 테스트 결과보고서를 작성하는 일을 한다. 테스트 작업을 편리하게 하기 위해 많은 테스트 도구가 개발되어지고 있지만, 단위 테스트로부터 시스템 테스트까지 점진적으로 테스트를 할 수 있는 테스트 통합 환경은 많지가 않다. 또, 자동화된 테스트 도구라도 테스터가 테스트 수행에 간섭을 해야 하며, 많은 양의 테스트케이스가 생성되어 불필요하게 테스트 시간이 많이 걸리게 된다. 따라서 테스터의 간섭을 최소화하고, 효율적인 테스트케이스만을 선택함으로써 테스트 시간을 단축시킬 수 있는 테스트 도구가 필요하다.

컴퓨터 분야에서 에이전트의 개념[1]은 전자 상거래, 정보 검색과 같은 많은 어플리케이션들[2,3,4,5]에 응용되어 중요 시 되고 있다. 하지만, 아직까지 지능성을 가진 테스트 도구는 없었다. 이 논문에서 제안하는 테스트

· 이 논문은 2000년도 두뇌한국21사업에 의하여 지원되었음
† 학생회원 : 이화여자대학교 컴퓨터학과
982COG25@mm.ewha.ac.kr
†† 종신회원 : 이화여자대학교 컴퓨터학과 교수
bjchoi@mm.ewha.ac.kr
논문접수 : 2000년 2월 1일
심사완료 : 2000년 9월 25일

에이전트 시스템(TAS)은 에이전트의 특성을 가지고 테스터를 도와주는 테스트 도구이다. TAS는 사용자나 다른 에이전트의 직접적인 지시나 간섭 없이도 스스로 판단하여 행동하는 성질인 자율성(Autonomy), 사용자의 목적을 받아들여 계획을 세우고, 지식 베이스를 기반으로 하여 추론을 할 수 있는 지능성(Intelligence), 다른 에이전트간의 통신을 하는 사회성(Social ability)의 에이전트 특성을 갖춘 테스트를 위한 에이전트 시스템이다.

TAS는 객체지향 테스트 프로세스[6]를 따라 테스터의 일을 대행해 주고, 테스터의 간섭을 최소화 시켜 준다. 이 시스템은 자동 생성된 많은 양의 테스트케이스에서 중복이 없고 일관성 있는 테스트케이스를 지능적으로 선택하여 테스트 시간을 단축시켜 준다.

본 논문의 구성은 다음과 같다. 2장에서는 에이전트의 개념, 에이전트에 대한 관련 연구와 기존의 테스트 도구에 대해 기술한다. 3장에서는 TAS를 구성하는 각 에이전트들의 구조, 지능성을 나타내는 규칙들을 기술하고, 4장에서는 TAS의 각 에이전트들이 에이전트로서 가지는 특성과 TAS를 분석한다. 마지막으로 5장에서 결론을 제시한다.

2. 관련 연구

2.1 에이전트 관련 연구

에이전트에 대한 정의는 사람마다 에이전트를 보는 시각에 따라 매우 다양[1,7,8]하지만 일반적으로 에이전트는 특정 목적에 대하여 사용자를 대신하여 작업을 수행하는 자율적 프로세스로 지식베이스와 추론 기능을 가지며 사용자, 자원(resource), 또는 다른 에이전트와의 정보 교환과 통신을 통해 문제를 해결하는 것이라고 정의한다. 에이전트는 자율성, 사회성, 지능성 등의 특성을 가지고 있다. 이러한 에이전트의 특성의 일반적인 정의[1]는 다음과 같다.

- 자율성(Autonomy) : 사용자로 하여금 상위단계의 목적에 집중을 하게 하고 그 목적을 달성하기 위한 세부 절차는 에이전트가 맡는다. 사용자나 다른 에이전트의 간섭 없이도 자율적으로 일을 수행하는 특성이다.
- 사회성(Social ability) : Standard language와 protocol을 이용하여 공통 목적에 도달하기 위해 에이전트들 간의 서로 협조/협력하는 특성이다.
- 지능성(Intelligence) : 사용자의 목적을 받아들여 계획을 세우고, 지식 베이스를 기반으로 하여 추론을 하며, 외부 환경과의 상호작용을 통하여 학습하는 능력이다.

한 에이전트가 가지고 있는 통신 구조는 그림 1과 같다. 통신부에서는 다른 에이전트 혹은 외부와의 통신을 하고, 처리부는 수행 엔진과 지식베이스를 가지고 있다. 에이전트 목적에 맞게 에이전트가 수행되는 것은 수행엔진 부분이다.

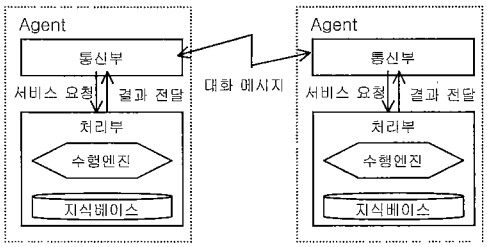

그림 1 에이전트 구조 모델

2.2 기존에 개발된 테스트 도구

테스터가 테스트를 수행하는 데에 이용되는 테스트 도구로써 테스트 기법에 따른 적정성 측정, 디버깅, 특정 테스트 기법을 적용하여 테스트케이스를 자동 생성하는 테스트 도구들이 많이 개발되어 지고 있다. 그 중에서도 대표적으로 특정 테스트 기법을 적용하여 테스트케이스를 선택하는 Proteum, 테스트 기법에 따른 적정성을 측정하는 ATAC, 그리고 디버깅과 리그래션 테스트에 효율적인 Visual Test도구들에 대해 알아본다.

(1) Proteum(Program Testing Using Mutants)[9]

Mutants analysis 기준을 지원하는 테스트 도구로 open windows 아래의 sun workstation 상에서 수행이 되는 테스트 도구이다. 특정 테스트 기법을 적용하여 테스트케이스를 자동 생성하는 테스트 도구에 속한다. 이 테스트 도구의 단점은 mutants의 종류는 많이 있으므로 mutants 연산자를 테스터가 어떻게 선택하느냐에 따라 테스트케이스가 생성되고 테스트의 성공 여부가 나타난다. 또, 테스트 대상 하나의 테스트가 완료 될 때까지 테스터는 테스트 도구 앞에서 작업을 해야 하므로 많은 양의 테스트케이스를 테스트 할 경우에는 불필요한 많은 시간을 소비하게 된다. 따라서 테스터가 테스트 진행 시간동안 테스트 도구 앞에 있지 않아도 테스트가 능동적으로 수행이 될 수 있는 즉, 에이전트의 자율성을 가진 테스트 도구가 요구되어진다.

(2) ATAC [10]

테스트 coverage를 분석하는 테스트 도구로 C언어로 만들어진 프로그램을 테스트하기 위해 개발이 되어졌고 coverage criteria에 대해서 측정되어진 수치가 report 되어진다. 이 테스트 도구는 테스트 기법에 따른 적정성

측정을 하는 테스트 도구에 속한다. 이 테스트 도구의 단점이라면 단위 테스트를 위한 테스트 도구이고 시스템 테스트, 통합 테스트에 대한 테스트 통합 환경이 구축되어 있지 않다. 또, 이 테스트 도구는 테스트 기법에 따른 적정성을 측정해 주어 적정성에 대한 수치만을 표시해 줄 뿐 어느 부분에서 오류가 생겼는지를 확실히 표현 해 주지 못한다. 따라서 리그래션 테스트 수행 여부를 판단 할 자료가 되는 오류가 생긴 지점을 표시해 주는 테스트 도구가 요구되어진다.

(3) Visual Test [11]

32-bit Windows application, ActiveX Components, and DLLs의 테스트를 할 수 있는 테스트 도구로 테스트의 효율을 높이는 리그래션 테스트의 반복적인 일을 자동적으로 할 수 있고, 시간을 적게 들여 질 높은 application들을 만들 수 있는 기능성을 테스트하는 테스트 도구이다. 유용한 테스트 도구이지만 시스템 테스트에 중점을 맞춘 테스트 도구로 단위 테스트부터 테스트 프로세스를 따라 순차적으로 테스트가 수행되는 테스트 도구는 아니다. 따라서 단위 테스트부터 시스템 테스트까지 점진적으로 테스트를 수행하는데, 테스터의 간섭 없이도 테스트가 수행 될 수 있는 테스트 도구가 요구되어진다.

위에서 언급한 테스트 도구 외에도 많은 테스트 도구들이 테스트를 쉽고 자동적으로 수행 할 수 있어 많이 이용되고 있지만, 테스터가 요구하는 것에 대해 단순히 수동적으로 반응을 행하는 것이어서 스스로 판단하여 테스트 프로세스를 따라 테스트가 수행되는 테스트 도구는 아직 나오지 않고 있다. 따라서 단위 테스트로부터 시스템 테스트까지 점진적으로 테스트가 수행되고 테스트 프로세스에 따라 테스트 계획을 스스로 세워 자율적으로 테스트를 수행 할 수 있는 테스트 에이전트 시스템이 요구되어진다. 사용자의 작업을 대신해 주는 에이전트의 필요성은 우리 일상 생활 도처에서 찾아 볼 수 있다. 여행사, 회사, 비서, 도서관 사서, 판매 대리점 등의 역할이 에이전트가 적용될 수 있는 분야라고 할 수 있다. 정보화 사회의 진전으로 일상 생활의 복잡도와 다양성이 증가함에 따라 점차 모든 부분에서 에이전트의 비중이 커지고 있다. 이와 같이 에이전트가 적용될 수 있는 분야는 매우 넓지만 그 중에서도 인터넷 정보 검색[2], 전자상거래[3], 사용자 인터페이스[4], 이동 컴퓨팅[5] 분야에 대한 요구가 매우 많다고 할 수 있다. 일반적으로 테스터가 테스트 작업을 수행 할 때 테스트 도구 앞에서 많은 시간을 투자하여 테스트를 완료해야 하는 데, 요구되어지는 테스트 에이전트 시스템은 테스터가 테스트 대상을 지정해 주면 테스트가 완료 될 때까지 시스템 앞에 있을 필요 없이 다른 일을 할 수 있는 시간이 생길 수 있어 유용하다.

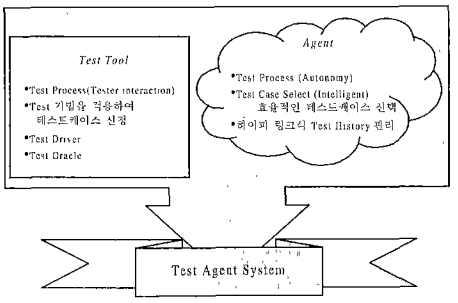

그림 2 2TAS 개념

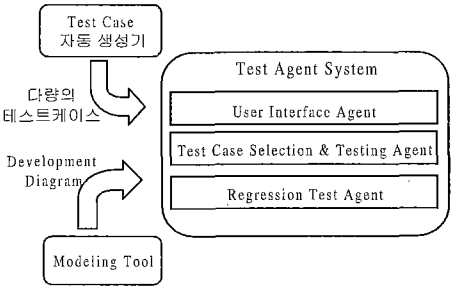

그림 3 3TAS의 구성도

테스트 과정은 테스트 정보 입력, 테스트 항목 선택, 테스트케이스 선택, 테스트, 리그래션 테스트로 나눌 수 있다. 이러한 테스트 과정에서 밀접한 관계가 있는 과정들끼리 묶어 이를 에이전트화 하면, 독립적으로 각자의 기능을 수행하면서, 서로간의 통신을 통해 원활한 테스트 작업을 진행시킬 수 있다. 에이전트 각각이 독립적으로 수행이 되면, 분산 환경에서도 쉽게 적응이 되어 테스트를 진행 할 수 있다. 또, 이들이 에이전트화 되면, 단순한 규칙 기반 시스템이 아니라, 지능성을 가짐으로써, 테스트를 수행하는 데에 있어 효율적으로 진행 할 수 있다. 따라서 본 논문에서는 이러한 점을 착안하여 에이전트를 User Interface Agent, Test Case Selection & Testing Agent, Regression Test Agent로 나누어 테스트를 수행하는 테스트 에이전트 시스템을 제안하였다. 본 논문에서 제안하는 이 시스템은 그림 2와 같이 테스트 도구로서의 기능에 에이전트의 특성이 결합되어 만들어진 시스템이다.

3. TAS의 특성 및 구조

본 논문에서 제안하는 TAS는 2장에서 기술한 에이전트의 정의를 기반으로 설계한 테스트 에이전트 시스템이다. TAS의 구성은 그림 3과 같이 테스터와 정보를 주고받는 'User Interface Agent', 중복이 없고 일관성 있는 테스트케이스를 선택하고 테스트를 수행하는 'Test Case Selection & Testing Agent', 그리고 리그래션 테스트를 담당하는 'Regression Test Agent'인 3개의 에이전트로 이루어진다. TAS는 외부로부터 다량의 테스트케이스와 테스트 작업에 필요한 개발 다이어그램들을 제공받는다. TAS를 구성하는 3개의 에이전트들의 구조와 특성은 다음과 같다.

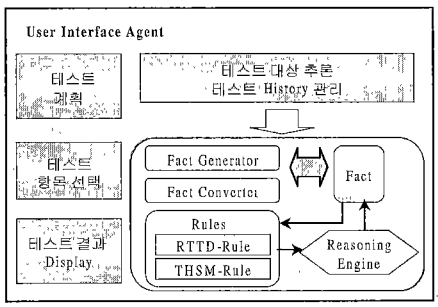

그림 4 User Inerface Agent

3.1 User Interface Agent

User Interface Agent는 테스터와 TAS와의 인터페이스 역할을 한다. 이 User Interface Agent는 테스터로부터 입력받은 테스트 계획서와 개발 다이어그램 등의 테스트 정보로부터 테스트 대상을 추론하고 테스트 결과를 보여 주는 것이 주요 목적이다. User Interface Agent는 그림 4와 같이 테스트 대상을 추론하여 테스트를 계획하는 부분, 테스트 대상이 리그래션 테스트인지를 추론하는 부분, 테스트 History의 메모리 관리를 위해 메모리 상황을 추론하는 부분과 테스트 결과를 화면으로 보여 주는 부분으로 구성되어 있다.

User Interface Agent는 리그래션 테스트 대상 판단 규칙(RTTD-Rule)에 의해 테스트 대상을 지능적으로 추론하며, 테스트 히스토리 크기 관리 규칙(THSM-Rule)에 의해 테스트 history 관리를 한다. 이 때 적용한 규칙(rules)은 다음과 같다.

(1) 리그래션 테스트 대상 판단 규칙(RTTD-Rule)

RTTD-Rule은 테스트 대상이 테스트 History로부터 리그래션 테스트 대상인지를 판단하는 규칙이다. 리그래션 테스트이면 테스트 History에 있는 테스트케이스를 가지고 테스트를 진행 되도록 하여, 테스트케이스를 획득하는 시간이나, 테스트 에이전트 시스템 내에서 테스트 계획을 세우는 일 없이 리그래션 테스트를 할 수 있도록 한다. 이 RTTD-Rule을 술어 논리[12]로 표현한 것은 다음과 같다.

<RTTD-Rule>

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

(2) 테스트 히스토리 크기 관리 규칙(THSM-Rule)

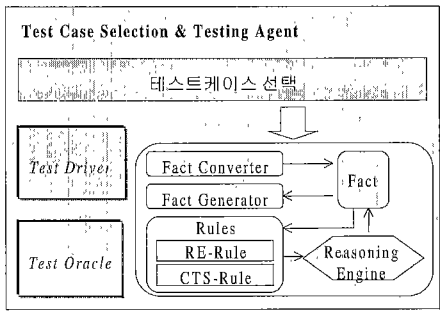

그림 5 Test Case Selection Method

THSM-Rule은 Test History 크기를 관리하는 규칙이다. User Interface Agent는 테스트 대상을 리그래션 테스트 발생률에 따라 History에 남겨지는 기간을 차별화 한다. 리그래션 테스트 발생률을 계산하여 그 비율에 따라 THSM-Rule에 의해 Test History에 테스트 대상이 남겨지는 기간을 정하게 된다. 그래서 Test History 메모리를 효율적으로 사용 할 수 있게 된다. 이 THSM-Rule을 술어 논리[12]로 표현한 것은 다음과 같다.

<THSM-Rule>

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

3.2 Test Case Selection & Testing Agent

자동화 도구에 의해서 생성된 수많은 테스트케이스로부터 테스트케이스 사이에 중복이 없고, 일관성 있는 테스트케이스를 선택함으로써 과도하게 테스트하는 것을 방지할 수 있다. 'Test Case Selection & Testing Agent'는 수많은 테스트케이스들 가운데 중복이 없고

일관성 있는 테스트케이스를 선택하여 테스트를 수행하는 것이 목적이다. 'Test Case Selection & Testing Agent'는 그림 5와 같이 테스트케이스를 선택하는 부분, 테스트를 수행하는 Test Driver, 그리고 수행 후 테스트 결과를 테스트 예상 결과와 비교하는 Test Oracle로 구성되어 있다.

Test Case Selection & Testing Agent는 테스트 실행 결과를 테스트 예상결과와 비교하여 테스트케이스의 품질 측정치(Measure)를 다음과 같이 나타낸다.

품질 측정치 = TRs / ITc

TRs는 Test Case Selection & Testing Agent가 테스트 실행 결과와 테스트 예상 결과를 비교해서 결과가 같으면 S로, 틀리면 F로 표현한 것 중 S로 표현된 테스트케이스 수이다. ITc는 테스트 항목 당 사용된 전체 테스트케이스 수이다.

본 테스트 에이전트 시스템의 테스트 대상은 객체지향 프로그램이다. 객체지향 프로그램[6]을 위한 테스트 케이스는 '메소드 순서 형태와 테스트 데이타 형태'의 두 가지 테스트케이스 종류가 있다. 테스트 에이전트 시스템은 이 두 가지 형태의 테스트케이스에 따라 각각 중복 제거 규칙(RE-Rule)과 일관성 있는 테스트케이스 선택 규칙(CTS-Rule)을 통하여 중복됨이 없고 일관성 있는 테스트케이스를 지능적으로 선택한다.

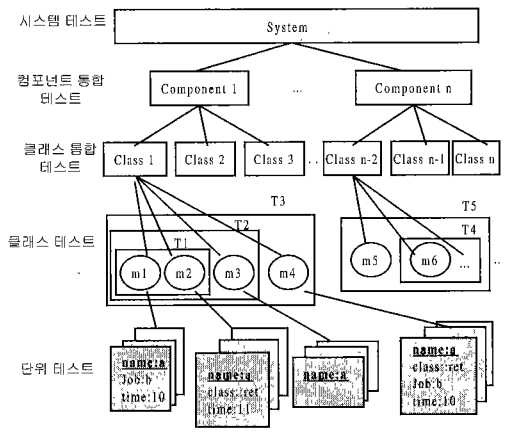

그림 6 테스트케이스 선택 방법

(1) 중복 제거 규칙(RE-Rule)

이 규칙은 메소드 순서 형태의 테스트케이스들에서 중복된 테스트케이스를 제거하는 규칙이다. 예를 들면, 그림 6에서 메소드 순서 형태의 테스트케이스인 T1, T2, T3 중에서 T1과T2를 포함하고 있는 테스트케이스 T3를 선택한다. 즉, T1=(m1,m2), T2=(m1,m2,m3), T3=(m1,m2,m3,m4)이므로 T3를 선택할 경우, T1과 T2의 메소드도 포함하고 있으므로 m1, m2, m3에 대해 중복되게 테스트를 실행 할 필요가 없다. 다음은 RE-Rule을 술어 논리[12]로 표현한 것이다.

<RE-Rule>

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

(2) 일관성 있는 테스트케이스 선택 규칙(CTS-Rule)

하나의 테스트케이스가 여러 개의 메소드로 이루어졌을 때, 각 메소드의 테스트 데이타를 일관성 있게 선택하는 것은 중요하다. 예를 들면 그림 6에서 메소드 m1, m2, m3, m4의 테스트케이스가 각각 {(name:a, Job:b, time:10), (name:c, Job:bj time:10), (name:u, Job:eb, time:10)}, {(name:a, class:ret, time:11), (name:wq, class:gg, time:8), (name:zx, class:rt, time:5)}, {(name:a), (name:tr), (name:ww)}, and {(name:a, class:ret, Job:b, time:10), (name:h, class:ret, Job:l, time:9), (name:ee, class:fd, Job:b, time:12)}이라 하자. 이 예제에서 name이라는 변수는 m1, m2, m3, m4 의 메소드에 있음을 알 수 있다. 따라서 name이라는 변수에 'a'라는 값이 테스트 데이타로 선정되었을 경우, m1의 테스트케이스는 (name:a, Job:b, time:10), m2의 테스트케이스는 (name:a, class:ret, time:11), m3의 테스트케이스는 (name:a), m4의 테스트케이스는 (name:a, class:ret, Job:b, time:10)를 선택함으로써 테스트케이스 사이의 일관성을 유지해야 한다. Test Case Selection & Testing Agent는 CTS-Rule을 통해 일관성 있는 테스트케이스를 선택 할 수 있다. 다음은 이 CTS-Rule을 술어논리[12]로 표현한 것이다.

<CTS-Rule>

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

3.3 Regression Test Agent

'Regression Test Agent'는 리그래션 테스트 수행을 위해 리그래션 테스트 항목을 선택하는 기능을 가진 에이전트로서 선택된 리그래션 테스트 항목과 관련된 모든 테스트 항목에 대한 리그래션 테스트를 수행하는 것이 목적이다. Regression Test Agent는 그림 7과 같이 리그래션 테스트 관련 항목 찾는 규칙(RRTIS-Rule)이 수행되는 추론 엔진을 중심으로 구성되어 있다. 'RRTIS-Rule'은 선택된 리그래션 테스트 항목과 관련된 항목을 하이퍼 링크식[13]으로 검색하는 규칙이다.

(1) 리그래션 테스트 관련 항목 찾는 규칙(RRTIS-Rule)

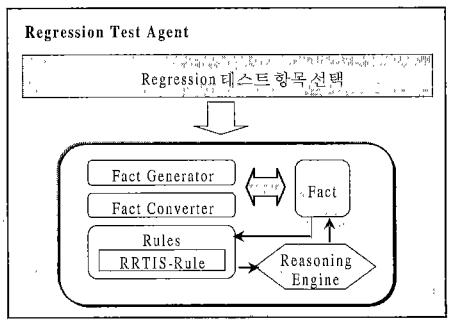

그림 7 Regression Test Agent

RRTIS-Rule은 리그래션 테스트의 관련 항목을 찾는 규칙(Regression test Related Test Item Search Rule)이다. 객체지향 테스트 프로세스[6]에서는 테스트 레벨을 다섯 레벨로 나누고 있다. 단위 메소드 테스트 레벨, 클래스 테스트 레벨, 클래스 통합 테스트 레벨, 컴포넌트 통합 테스트 레벨, 시스템 테스트 레벨로 나눈다. 클래스 통합 테스트 레벨에서는 상속관계와 연관관계로 나누어 통합 테스트를 한다. 테스트 레벨에 따라 테스트 항목은 다르다. 단위 메소드 테스트 레벨에서는 단위 메소드, 클래스 테스트와 클래스 통합 테스트 레벨에서는 클래스, 컴포넌트 통합 테스트 레벨에서는 컴포넌트, 시스템 테스트 레벨에서는 사용 사례이다. 리그래션 테스트를 위한 관련된 테스트 항목이란 입력된 리그래션 테스트 항목 중 메소드 호출 관계에 의해 상호 호출되는 메소드 혹은, 그 테스트 항목이 속한 클래스, 컴포넌트, 사용 사례이다. 리그래션 테스트 관련 항목을 찾는 것은 입력된 것뿐만 아니라, 속해 있는 것에 대한 테스트도 행하기 위함이다. 본 논문에서는 입력된 리그래션 테스트 항목과 관련된 항목을 하이퍼 링크식[13]으로 표현하였으며, 이로부터 리그래션 테스트 관련 항목 찾는 규칙을 술어 논리[12]로 표현한 것은 다음과 같다.

<RRTIS-Rule>

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

- $\forall x, y, z, (\text{Regtestitem}(x) \wedge \text{Related-regtestitem}(y) \wedge \text{Relatedlist}(z) \wedge \text{Equal}(\text{Fourthelement}(z), \text{Last}(y)) \supset \text{Related-regtestitem}(z)$

3.4 구 현

본 논문에서 제시한 테스트 에이전트 시스템은 Windows환경에서 JDK1.2.1(Java Development Kit)로 구현하였다. 에이전트의 지능성을 나타내는 User Interface Agent에서의 '리그래션 테스트 대상 판단 규칙'과 'Test History 크기 관리 규칙', Test Case Select & Testing Agent에서의 '중복 제거 규칙'과 '일관성 있는 테스트케이스 선택 규칙' 그리고 Regression Test Agent에서의 '리그래션 테스트 관련 항목 찾는 규칙'은 자바와 연동이 용이한 Jess4.4(Java Expert System Shell) 전문가 시스템을 이용하여 구현하였다. 그리고, TAS의 데이타베이스는 테스트케이스 데이타베이스, 테스트 대상에 있는 구성요소들의 리스트가 저장되어 있는 리스트 데이타베이스, 테스트 항목이 저장되어 있는 데이타베이스, 테스트 예상 결과가 저장되어 있는 데이타베이스, 테스트 실행 후 테스트 결과를 저장하는 데이타베이스, 테스트 실행에 사용된 테스트케이스들이 저장되어 있는 데이타베이스, 테스트 대상이 저장되는 데이타베이스, 테스트 항목간의 소속 관계를 저장 해 놓는 데이타베이스가 있다.

4. 분 석

TAS를 구성하는 3개의 에이전트들은 2절에서 기술한 에이전트의 특성인 자율성, 사회성, 지능성을 갖고 있다. 각 에이전트별로 자율성, 사회성, 지능성을 분석하면 다음과 같다.

(1) User Interface Agent가 갖는 에이전트로서의 특성

■ 자율성(Autonomy) : 에이전트의 특성인 자율성은 사용자로 하여금 상위단계의 목적에 집중을 하게 하고 그 목적을 달성하기 위한 세부 절차 등은 에이전트가 맡는다. 테스트 에이전트 시스템에서 제시하고 있는 User Interface Agent에서 사용자 즉, 테스터의 상위 목적은 테스트 결과를 보는 것이고 테스트 결과가 나오기까지의 세부 절차는 User Interface Agent가 맡아서 한다. 즉, 테스트 수행 순서를 계획하고, 테스트 항목을 찾는 일을 스스로 알아서 행동을 하므로 자율성(autonomy)이 있다.

■ 사회성(Social ability) : User Interface Agent는 테스트케이스를 선택하는 것과 테스트를 수행하는 기능은 없다. 대신 그 기능을 하는 에이전트와 통신을 하여 테스트 수행을 계속 진행 되도록 한다.

■ 지능성(Intelligence) : User Interface Agent는 입력된 테스트 대상이 리그래션 테스트 대상인지를 판단하는 규칙 'RTTD-Rule로 추론을 하여 지능성을 나타낸다. 또, User Interface Agent의 지능성을 나타내는 규칙으로 THSM-Rule있다. 이 규칙은 리그래션 테스트 발생률에 따라 테스트 대상이 테스트 History에 남겨지는 기간을 차별화 하여 메모리의 크기를 지능적으로 관리한다.

(2) Test Case Selection & Testing Agent가 에이전트로서 갖는 특성

■ 자율성(Autonomy) : Test Case Selection & Testing Agent는 외부의 자극이 없이도 선정한 테스트케이스로써 테스트를 자율적으로 테스트를 execute한다. 특히 테스트 항목에 대한 중복이 없고 일관성 있는 테스트케이스를 테스트케이스 풀에서 추론을 통하여 스스로 판단하여 중복이 없고 일관성 있는 테스트케이스를 선택한다. 또, 지식베이스를 통하여 테스트케이스를 선택하는데 필요한 정보를 자율적으로 인식하므로 자율성(autonomy)이 있다.

■ 사회성(Social ability) : 테스트 항목에 대한 중복이 없고 일관성 있는 테스트케이스를 선택하는데 필요한 정보, 즉 테스트 항목과 테스트케이스 풀을 획득하기 위하여 다른 에이전트와 통신을 하므로 사회성(social ability)이 있다.

■ 지능성(Intelligence) : Test Case Select & Testing Agent는 두 가지 형태의 테스트케이스를 각 규칙에 따라 추론하여 선택 할 수 있는 지능성이 있다. 즉, 메소드 순서 형태의 테스트케이스를 중복이 없도록 선택하기 위해 중복 제거 규칙(RE-Rule)에 따라 추론하여 테스트케이스를 선택한다. 테스트 데이타 형태의 테스트케이스를 일관성 있게 선택하기 위해 일관성 있는 테스트케이스 선택 규칙(CTS-Rule)에 따라 추론을 하여 테스트케이스를 선택한다.

(3) Regression Test Agent가 에이전트로서 갖는 특성

■ 자율성(Autonomy) : Regression Test Agent는 리그래션 테스트를 위해 필요한 테스트 항목을 외부의 간섭 없이 스스로 인식한다.

■ 사회성(Social ability) : Regression Test Agent는 테스트 항목을 획득하기 위해 User Interface Agent와 통신을 하고, 리그래션 테스트 항목으로 검색되어진 테스트 항목의 리그래션 테스트를 수행 하기 위해 Test Case Selection & Testing Agent와 통신을 한다.

■ 지능성(Intelligence) : Regression Test Agent은 테스터가 선택한 리그래션 테스트 항목으로부터 관련된 리그래션 테스트 항목을 규칙 RRTIS-Rule'에 의해 추론하므로 지능성이 있다. 즉, Test Case History에서 검색된 테스트 항목과 기존의 테스트케이스 전부를 리그래션 테스트하는 것이 아니고, 리그래션 테스트 관련 항목 찾는 규칙에 의해 테스트 항목간의 관련성을 지능적으로 추론한다.

(4) TAS의 에이전트들간의 상호 작용

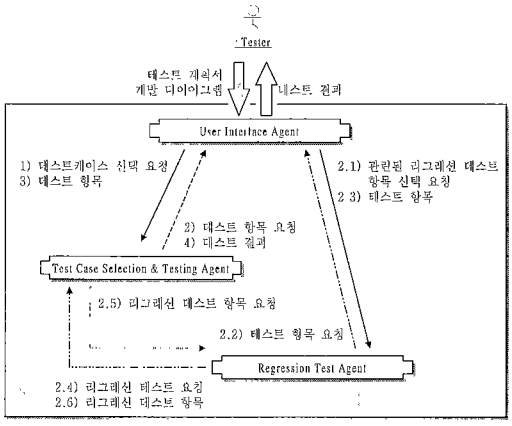

그림 8 TAS의 에이전트들간의 상호작용

그림 8은 TAS를 구성하는 3개의 에이전트들간의 상호 작용을 표현한 것이다. 예를 들면, 그림 8에서 User Interface Agent와 Test Case Selection & Testing Agent와의 상호 작용을 보면, User Interface Agent는 테스트 항목 다음의 테스트 수행 행동인 테스트케이스 선택을 위해 이 기능을 가지고 있는 Test Case Selection & Testing Agent에게 1) 테스트케이스 선택

요청을 한다. Test Case Selection & Testing Agent는 테스트케이스를 선택하기 위해 필요한 2) 테스트 항목을 요청한다. 요청을 받은 User Interface Agent는 3) 테스트 항목들을 Test Case Selection & Testing Agent에게 전달 해 준다. 테스트 항목을 받은 Test Case Selection & Testing Agent는 테스트 항목에 맞는 테스트케이스를 선택하고, 테스트를 실행 후 4) 테스트 결과를 User Interface Agent에게 전달한다.

TAS는 에이전트의 특성인 자율성, 사회성, 지능성을 갖춘 시스템으로 TAS의 특성을 다음과 같이 정리 할 수 있다.

- 규칙에 따라 추론을 할 수 있는 지능성(Intelligence)이 있다.
- 테스터나 다른 에이전트의 직접적인 지시나 간섭 없이도 스스로 판단하여 테스트를 수행 할 수 있는 자율성(Autonomy)이 있다.
- 다른 에이전트간의 통신을 하는 사회성(Social ability)을 가지고 있다.
- 단위 테스트로부터 시스템 테스트까지 객체지향 테스트 프로세스에 따라 점진적으로 테스트를 수행 할 수 있는 테스트 통합 환경을 제공한다.
- 중복이 없고, 일관성 있는 테스트케이스를 선택하여 테스트 시간을 단축한다.
- 테스터의 간섭을 최소화한다.
- 지능적으로 리그래션 테스트 항목 선택할 수 있도록 한다.

5. 결 론

본 논문에서는 테스터의 간섭 없이도 스스로 판단하여 테스트 프로세스를 따라 테스트가 수행 될 수 있는 테스트 에이전트 시스템인 TAS를 제안하였다. 이 시스템은 단위 테스트로부터 시스템 테스트까지 점진적으로 테스트를 수행 할 수 있는 테스트 통합 환경을 지원한다.

TAS는 에이전트의 특성인 자율성, 사회성, 지능성을 갖추고 있어 기존의 테스트 도구에 비하여 테스터의 작업을 대행해 주는 새로운 개념의 테스트 도구이다. TAS는 테스터와 정보를 주고 받는 'User Interface Agent', 중복이 없고 일관성 있는 테스트케이스를 선택하고 테스트를 수행하는 'Test Case Selection & Testing Agent', 그리고 리그래션 테스트를 담당하는 'Regression Test Agent'인 3개의 에이전트로 이루어졌다. 특히 'User Interface Agent'는 'RTTD-Rule'과 'THSM-Rule', 'Test Case Selection &Testing Agent'는 'RE-Rule'과 'CTS-Rule', 그리고 'Regression Test Agent'는 'RRTIS-Rule'을 통하여 에이전트의 지능성을 갖고 있다.

이 시스템에서 테스터는 테스트 수행의 상위 목적인 테스트 결과를 보는 것에만 집중하면 된다. 테스트를 수행하는 세부 절차는 TAS의 에이전트들이 대행해 준다. 즉, 테스트 계획을 세우고, 테스트 항목을 추출하고, 테스트케이스를 선택하고, 그리고 테스트의 수행과 리그래션 테스트 수행 등의 모든 실제적인 테스트 작업을 대행한다.

TAS를 구성하고 있는 3개의 에이전트들은 필요한 정보를 획득하기 위하여 서로서로 통신을 하며, 테스트 진행을 원활히 해 나간다. TAS는 객체 지향 소프트웨어에만 적용되는 테스트 도구는 아니고, 객체 지향 소프트웨어가 아닌 구조적 프로그램, 컴포넌트 기반 프로그램에도 적용 될 수 있는 테스트 도구이다.

현재 TAS의 구현과 TAS가 규칙을 RE-Rule과 CTS-Rule을 통하여 효과적으로 테스트케이스를 선택해 줌으로써 테스트 시간을 단축시킬 수 있다는 것을 증명하는 실험이 진행되고 있다. TAS 는 객체지향 프로그램 테스트만을 대상으로 구현되고 있는데, 확장시켜 컴포넌트 기반 프로그램 테스트에 관한 연구가 필요하다.

참 고 문 헌

[1] Hyacinth S.Nwana, Software Agent: An Overview, Knowledge Engineering Review, vol11, No3, pp1-40, Sept 1996.

[2] Knoblock C. and Arens Y., An architecture for information retrieval agents, Working Notes of AAAI Spring Symposium on Software Agent, pp49-56, 1994.

[3] Yuh-Jong Hu, Intelligent Agent & Electronic Commerce web page (http://www.cs.nccu.edu.tw/~jong/agent/agent.html

[4] Etzioni, O. and Weld D., A softbot based interface to the internet, Comm. ACM, Vol.37, No.7, pp72-79, 1994.

[5] Gifford D. and Stamos J., Remote evaluation, ACM Trans. on Programming Language and Systems, Vol.12, No.4, pp537-565, 1990.

[6] Mina Rho, Byoungju Choi,Test Process in the Object-oriented Software Development Life Cycle bases on the Test Standards, Proceedings of Asia-Pacific Workshop on Software Process Improvement, pp17-32, 1997.

[7] Franklin S. and Graesser A. Is it an agent, or just a program?: A taxonomy for autonomous

agents, Proc. of Third International Workshop on Agent Theories, Architect Theories, Architectures, and Languages. 1996
[8] Nicholas R. Jennings, Intelligent Agents: Theory and Practice, Michael Wooldridge, Knowledge Engineering Review January 1995.
[9] Proteum-A Tool for the Assessment of TestAdequacy for C Programs User's Guide, by Marcio Eduardo Delamaro and Jose' Carlos Maldonado, SERC-TR-168-P, April, 1996. (http://hesperus.oboe.com/serc/TechReports/abstracts/catagory/Testing.html)
[10] http://www.clark.net/pub/dickey/atac/atac.html.
[11] http://www.rational.com/products/visual_test/prodinfo/whitepapers.
[12] Thomas Dean, James Allen, Yiannis Aloimonos, Artificial intelligence Theory and Practice, The Benjamin/Cummings publishing company, pp71-119, 1995.
[13] Thorsten Joachims, Tom Mitchell, Dayne Freitag, and Robert Armstrong, WebWatcher: Machine Learning and Hypertext, May, 1995.

최 정 은
1998년 2월 서울산업대학교 전자계산학과 학사. 1998년 3월 ~ 현재 이화여자대학교 대학원 컴퓨터학과 석사과정. 관심분야는 소프트웨어 공학, 객체지향 소프트웨어 테스팅, 분산 컴포넌트 테스팅, 에이전트, 객체지향 소프트웨어 개발 프로세스.

최 병 주
정보과학회논문지 : 소프트웨어 및 응용 제 27 권 제 2 호 참조