열차제어 S/W 변경영향 분석을 위한 방법 연구

The Method of Change Impact Analysis for Railway Signaling S/W

조현정† 황종규*

Jo, Hyun-Jeong Hwang, Jong-Gyu

ABSTRACT

Recent advances in computer technology have brought more dependence on software to railway signaling systems. Hence, the safety assurance of the vital software running on the railway signaling system is very critical task and yet, not many works have been done. While much efforts have been reported to improve electronic hardware's safety, not so much systematic approaches to evaluate software's safety. In this paper, we suggested an automated analysis tool for S/W change impact in railway signaling system, and presented its result of implementation. The analysis items in the implemented tool had referred to the international standards in relation to the software for railway signaling system, such as IEC 61508 and IEC 62279. In these international standards, 'change impact analysis' for railway signaling system S/W has to be required mandatorily. The proposed tool can be utilized at the assessment stage and also the software development stage.

1. 서 론

철도시스템 특히, 열차제어시스템은 최근 기존의 기계적 장치로부터 컴퓨터시스템으로 전환되고 있으며, 소프트웨어에의 의존성이 급격하게 증가하고 있다. 컴퓨터 기술의 발달에 따라 지능화 및 자동화를 위해 소프트웨어가 더욱 복잡해지게 되면서, 열차제어시스템에서 소프트웨어가 차지하는 비중이 더욱 증대되고 있다. 열차제어시스템 소프트웨어의 크기와 복잡도는 하드웨어의 발달 속도보다는 느리지만, 점차적으로 규모가 커지며 복잡도도 증가할 것을 예상된다. 이에 따라 <그림 1>에서와 같이 임베디드화 된 열차제어시스템 소프트웨어의 신뢰성과 안전성을 검증하는 것이 중요한 문제로 대두되기 시작했다. 철도시스템 소프트웨어 안전성 요구사항들이 최근 들어 IEC 61508과 IEC 62279에 의해 국제표준화 되었고[1] [2], 또한 국내에서도 철도안전법이 제정되어 이러한 철도시스템 관련 국제표준에서 요구하는 각종 소프트웨어 테스팅 및 검증활동을 요구하는 분위기가 조성되고 있다. 하지만 아직까지 소프트웨어 검증은 개발과정에 대한 문서에 주로 의존하고 있으며, 극히 일부만 테스팅에 의한 정량적인 분석이 이루어지고 있다. 또한, 국내에서의 국제표준에 따른 철도시스템 소프트웨어 테스팅 및 검증을 위한 기준이나 이에 부합하는 기술에 대한 연구는 이제 막 시작하는 초기단계에 불과한 실정이다[3-5].

따라서 철도시스템 소프트웨어관련 국제표준에서 요구하고 있는 안전성 활동에 대한 문서 검증뿐만 아니라 소프트웨어 테스팅을 통한 분석 및 평가에 대응하기 위한 구체적인 기술 개발이 매우 필요한 상

† 책임저자 : 정회원, 한국철도기술연구원, 열차제어통신연구실, 주임
E-mail : hjjo@krri.re.kr
TEL : (031)460-5458 FAX : (031)460-5449
* 정회원, 한국철도기술연구원, 열차제어통신연구실, 선임, 공학박사

황이다. 특히, 국제표준에서 요구하고 있는 철도 소프트웨어 검증 항목 중 하나인 변경영향 분석(Change Impact Analysis)은 대부분 SIL(Safety Integrity Level) 등급이 3 또는 4로 분류되는 전자연동장치와 같은 바이탈한 열차제어시스템 소프트웨어 검증의 경우, 관련 국제 표준에서 'M : Mandatory' 조건이며 ATO(Automatic Train Operation) 같은 SIL 1 또는 2 등급으로 분류되는 설비는 'HR : Highly Recommend' 조건으로 규정되어 있다[1][2]. 만일 열차제어시스템과 같이 소프트웨어 규모가 큰 임베디드 시스템에서의 소프트웨어 일부가 변경되면, 다른 부분에까지 영향을 끼쳐서 예측하지 못한 에러를 유발하여 안전성까지 위협하게 될 수 있으므로 국제표준에서도 이러한 소프트웨어 변경영향 분석을 의무사항 또는 권고사항으로 정해놓은 것이다. 게다가 소프트웨어 개발 초기에 소스 코딩단계에서도 변경영향 분석도구를 활용한다면, 새로운 변경 사항 발생 시 변경의 영향이 미치는 범위를 미리 분석하여 디버깅의 효율을 최대화시킬 수도 있다.

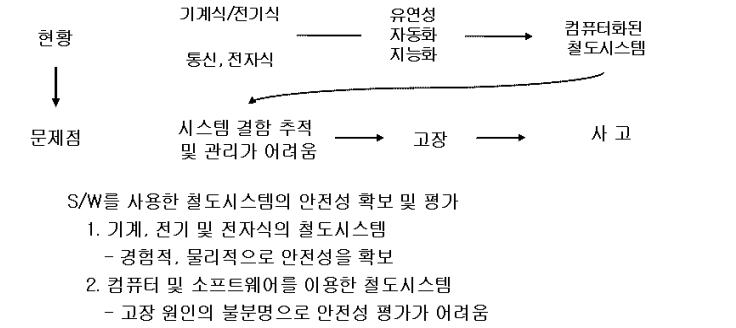

그림. 1 철도시스템 소프트웨어 안전성 평가 및 검증의 중요성

이와 같은 소프트웨어의 변경영향은 문서의 검증이나 정성적인 검증방법 보다는 자동화된 도구를 통해서 정확한 분석이 가능할 것이나, 철도시스템 소프트웨어의 변경영향 분석을 가능하게 해주는 자동화 도구는 국내외에 아직 개발된 적이 없다. 따라서 본 논문에서는 이러한 철도시스템 소프트웨어의 검증을 위해서 변경영향 분석 자동화 도구를 설계 및 개발하였다. 본 논문은 2장에서는 철도시스템 소프트웨어 테스팅 개요를, 3장은 철도 소프트웨어 변경영향 분석 자동화 도구의 설계 및 개발에 대한 구현결과를 보여주고, 마지막으로 4장에서 결론을 맺는 것으로 구성한다.

2. 열차제어시스템 소프트웨어 테스팅

열차제어시스템 임베디드 소프트웨어는 개발초기부터 테스팅 과정을 통해 버그를 확인하여 품질비용을 낮출 수 있으며, 또한 개발 완료 후 검증과정과 유지보수 측면에서도 소프트웨어 테스팅 과정이 필수적으로 요구되고 있다. 열차제어시스템 소프트웨어는 하드웨어에 의존적이고 높은 안전성이 요구되는 바이탈한 소프트웨어여서 자동화된 도구를 통한 테스팅이 필요하지만, 아직 일반 산업용 임베디드 시스템의 소프트웨어를 대상으로 일부 항목에 대한 테스팅 도구만이 소개되고 있다.

IEC 61508과 IEC 62279 같은 국제표준들에서 철도시스템 소프트웨어에 관련된 안전관련 사항들이 요구되고 있지만, 일반 산업용 제어시스템을 위한 테스팅 도구들은 철도 관련 국제 표준의 요구사항을 일부 만족시키지 못하고 있다[3]. 따라서 본 논문의 열차제어시스템 소프트웨어 테스팅을 위한 자동화 도구를 개발하기 위해 <그림 2>처럼 철도시스템 소프트웨어의 안전관련 국제 표준의 분석을 통해 철도 소프트웨어 안전성 평가 가이드라인을 작성하였으며, 이 가이드라인에서의 테스팅 항목 중 반드시 자동화가 필요한 아이템들을 분석하여 추출하였다.

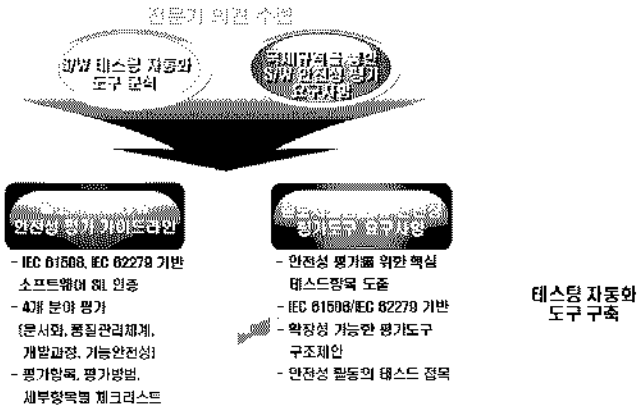

그림. 2 열차제어시스템의 테스팅 자동화 도구 구축 과정

국제표준에서 의무적인 검증항목으로 요구하는 것 중에 하나가 바로 바이탈한 철도시스템 소프트웨어의 변경영향 분석이다. 변경영향 분석은 개발하고자 하는 대상 소프트웨어에 변경이 이루어졌을 때, 실제 변경된 부분 및 이 변경으로 인해 영향 받는 부분을 확인하고 파악하기 위한 수단으로 활용되어진다. 이처럼 변경영향 분석은 소프트웨어의 변경이 미치는 파급효과의 범위를 분석하는 기술로서, 변경에 대한 요청이 있을 때 이 요청에 대한 처리 방안을 결정하면서 이루어진다. 즉, 변경 요청에 대한 설계 결정시에 변경에 대한 파급 효과를 점검함으로써 변경이 또 다른 오류를 유발하지 않도록 사전에 방지하는 효과가 있다.

3. 열차제어 소프트웨어 변경영향 분석 자동화 도구 개발

이처럼 철도관련 국제규격에서 소프트웨어 검증을 의무적으로 요구하고 있으며, 이에 따른 테스팅을 자동화하여 분석하는데 용이한 평가항목 중 하나가 바로 소프트웨어 변경영향 분석이므로 지원도구로서 본 장에서와 같이 철도 소프트웨어 변경영향 분석도구를 설계하였다.

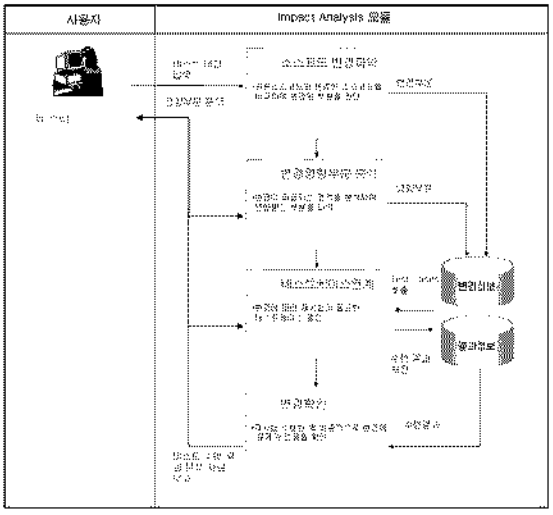

그림. 3 철도 소프트웨어 변경영향 분석 모듈의 기능 구성도

기본적으로 본 도구에서 철도 소프트웨어의 변경된 부분은 함수나 메소드의 시그너처 및 제어 구조를 기준으로 파악하며, 영향 받는 부분은 call graph와 control flow graph에서 순방향 분석(forward analysis) 및 역방향 분석(backward analysis)을 통해서 파악하도록 하였다. 변경의 파급 범위가 결정되

면 이 범위를 재시험하기 위한 테스트 케이스를 선택하는데, 변경영향 부분에 대한 재사용 테스트 케이스 선정은 테스트 케이스별 트레이스 정보를 통해서 분석하였다. 선택된 테스트 케이스를 재시험하여 변경의 문제 발생 여부를 확인하는 것이 본 도구 활용에 있어서의 최종 단계이다. 설계 및 구현한 소프트웨어 영향분석 도구의 기능 구성도는 <그림 3>과 같으며, 그림에서와 같이 소프트웨어 변경영향 분석 도구에서 가능한 기능들은 변경부분 파악, 변경영향 분석, 테스트 케이스 연계, 히스토리 관리 및 변경확인 등으로 선정하여 설계 및 구현하였다. 또한 본 자동화 도구가 소프트웨어 검증 단계 및 개발 단계에서도 활용이 가능하도록 하기 위해 변경영향 분석 결과를 변경정보로 저장하여 테스트 케이스를 호출할 수 있도록 구현하였으며, 그 수행결과를 저장하여 최종 변경 확인을 할 수 있도록 하였다.

본 논문에서 개발한 도구의 기능설계를 바탕으로 각각의 기능에 맞는 변경영향 분석 도구에 대한 화면 구현은 다음과 같은 절차를 따랐다. 먼저 철도 소프트웨어의 변경영향 분석을 효과적으로 지원하기 위해서 최근의 변경 내역뿐만이 아니라, 특정 소스 코드의 변경을 지속적으로 관리할 수 있도록 하였다. 이를 위해 변경 내역관리 기능은 사용자의 분석 순서에 따라서 변경 내역을 구분, 관리하여 사용자가 필요로 하는 경우에 각 회차를 기준으로 이전 회차와의 차이점을 보여줄 수가 있다. 다음 <그림 4>는 이러한 변경 내역 관리에 대한 화면과 소스코드 목록화면을 보여주는 그림이다. 그림에서와 같이 현재 확인하고 있는 변경 내역 분석에 포함되어 있는 소스코드 및 함수 목록을 확인할 수 있도록 지원하여 사용자가 원하는 변경 내역만을 선별적으로 추적 가능하도록 하였다.

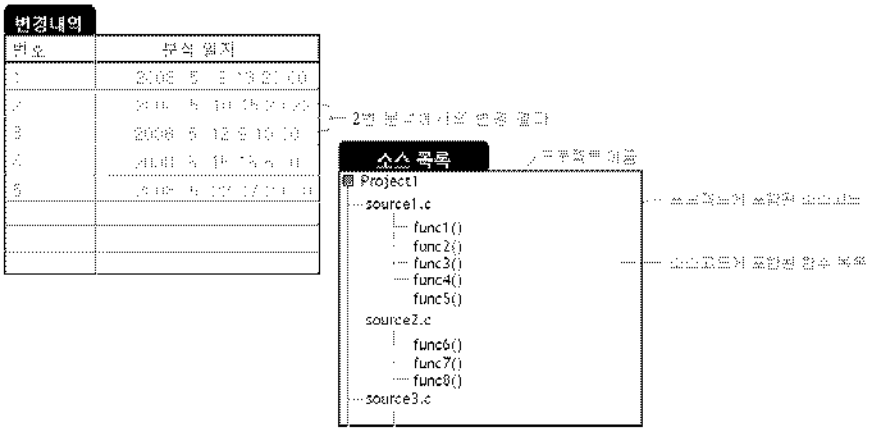

그림. 4 변경 내역 관리화면 및 소스코드 목록화면

또한 사용자 입장에서 변경영향 분석 결과를 쉽게 확인할 수 있도록 제어흐름 및 함수호출 그래프를 다음과 같이 구현하였다. 일반적으로 변경 내역 분석의 가장 기반이 되는 정보는 소스코드이지만, 본 개발 도구에서는 사용자가 이해하기에 어려움이 많은 것을 고려하여 <그림 5>처럼 소스코드로부터 추출된 제어흐름 그래프 모델을 사용하여 변경이 일어난 함수의 블록간의 변경영향 분석 내역을 파악할 수 있도록 하였다. 이 외에도 <그림 6>에서와 같이 함수 간의 호출관계를 보여주는 함수 호출그래프 수준에서 소스코드의 변경으로 인한 함수들의 콜 관계에 따라 각각의 변경이 영향을 끼치는 범위 또한 파악할 수 있도록 개발하였다.

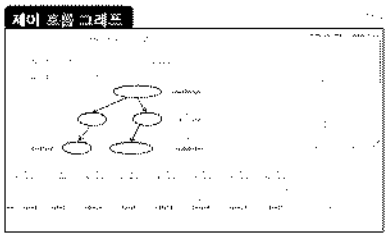

그림. 5 제어흐름 그래프 화면

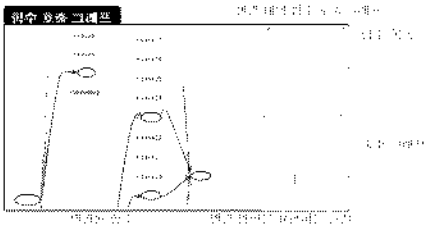

그림. 6 함수 호출 그래프 화면

마지막으로 소스코드의 변경영향 분석결과에 대한 내역 지원은 실제 <그림 7>에서와 같이 변경이 이루어진 소스를 변경 이전의 소스와 대비하여 제공함으로써 변경 내용을 명확하게 파악할 수 있도록 구현하였다. 특히 이와 같이 변경 이전과 현재 변경 내역에 대한 소스 코드를 확인할 수 있는 기능을 구현함으로써 본 개발 도구를 소프트웨어 검증을 위한 도구로서 뿐만 아니라 개발단계에서도 유용하게 활용될 수 있도록 한 것을 확인할 수 있다. 즉, 사전에 소스코드의 변경에 따른 소스코드들 간의 영향도를 분석함으로 소프트웨어 개발 및 유지보수에 있어서 효율성을 높일 수 있음을 의미한다.

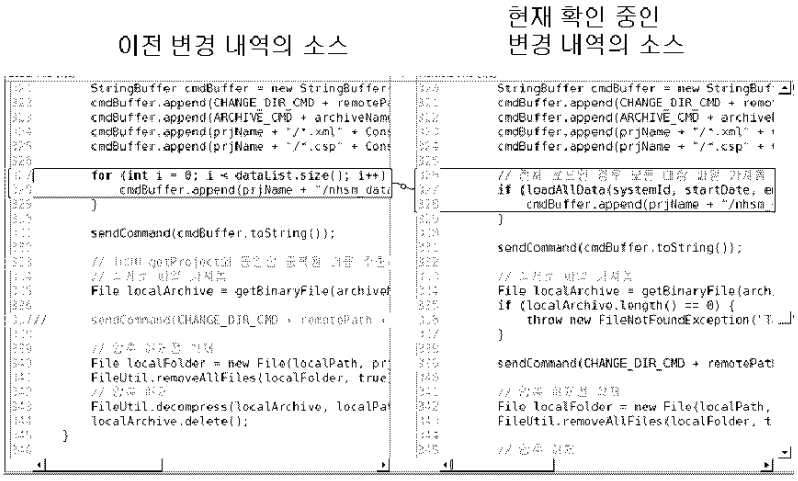

그림. 7 소스코드 변경 내역 윈도우

이와 같은 각각의 기능을 구현하여 개발한 철도 전용 소프트웨어 변경영향 분석 자동화 도구의 전체 실행 화면에 대한 결과는 <그림 8>과 같다. 그림에서와 같이 각종 기능을 제공하는 메뉴부분과 분석하고자 하는 소프트웨어의 소스파일 및 함수리스트를 보여주는 영역, 소스분석이 수행된 시점을 기록하는 변경기록 영역으로 본 도구의 GUI를 구성한다. 또한, 소스코드 및 그래프 영역을 구성하여 소스코드의 변경 내역 및 제어흐름과 함수호출 그래프의 변경영향 분석결과를 보여줄 수 있도록 하였다. 각각의 구성요소들은 서로 연관관계가 있으며, 그 연관관계는 관점에 따라서 틀릴 수가 있다.

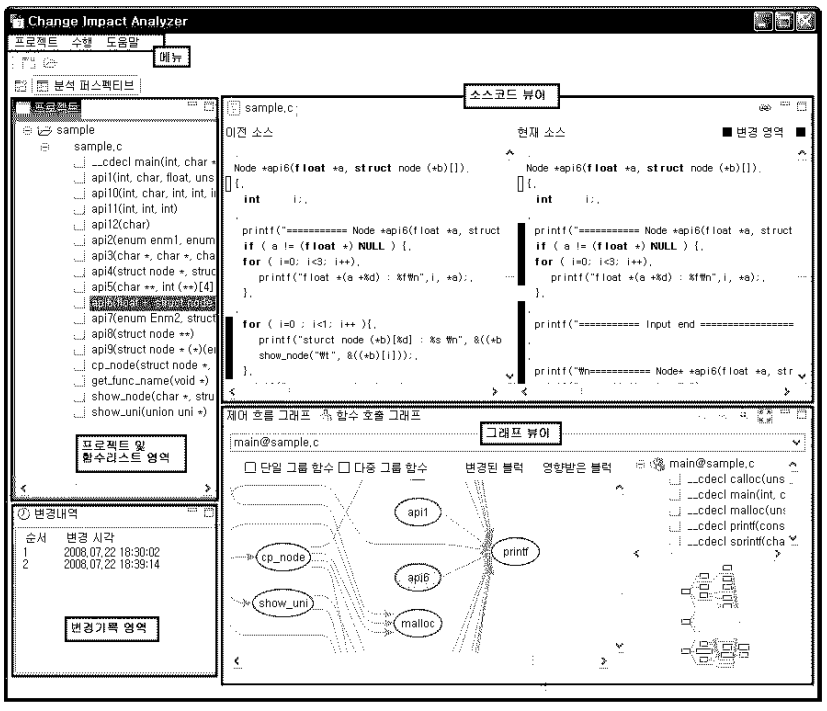

그림. 8 개발한 변경영향 분석 자동화 도구의 전체 윈도우

예를 들면, 변경기록 영역에 포커스를 맞춰서 따져보면 변경기록 중 알고자 하는 번호를 선택했을 경우, 해당 변경내역의 소스목록과 함수목록을 구성하여 함수호출 그래프에 이전 내역에서의 변경영향을 표시해주게 된다. 이와 달리 소스목록 관점에서 연관관계를 따져보면, 원하는 소스코드를 선택하여 해당 소스 코드의 이전변경 내역과의 차이점을 비교할 수 있고 함수를 선택한다면 해당 함수의 변경내역을 제어흐름 그래프에서 분석결과를 보여준다. 또한 함수호출 그래프에서 특정함수를 선택했을 경우는 해당 함수의 이전내역과의 차이점을 소스화면으로 보여주는 동시에 해당 함수의 제어흐름 그래프를 보여줌으로써 세부적인 변경사항을 확인하는 기능을 제공해준다.

4. 결 론

최근 들어 컴퓨터 기술의 발달에 따라 철도시스템들이 컴퓨터 소프트웨어에의 의존성이 급격하게 증가하고 있으며, 이러한 기술발전에 따라 바이탈한 철도 소프트웨어에 높은 신뢰성과 안전성이 요구되고 있다. 이에 따라 철도시스템 소프트웨어관련 국제표준에서 의무사항으로 요구하고 소프트웨어 검증 항목 중 하나인 변경영향에 대한 정확한 분석을 위해 본 논문에서 국내외 처음으로 개발한 자동화 도구를 제시하였다. 먼저 개발한 철도전용 소프트웨어 변경영향 분석 도구의 기능 설계에 대해 설명하였고, 그 구현 결과를 구체적으로 보여주었다.

이와 같은 변경영향 분석도구는 소스코드 변경으로 인한 함수 콜 관계 및 블록간의 영향에 대한 분석결과를 사용자 입장에서 파악하기 쉽도록 제어흐름 그래프와 함수호출 그래프로 표현하였다. 이러한 철도 소프트웨어 변경영향 분석 자동화도구는 기본적으로 소프트웨어 검증과 유지보수 단계에서 활용될 도구이며, 동시에 소프트웨어 개발과정에서도 충분히 활용도가 높을 수 있다고 본다. 구조면에서 점점 복잡해져가는 임베디드 소프트웨어를 한 사람이 아닌 여러 개발자에 의해 분산되어 개발하다보면, 특정 모듈의 소스코드에서 변경이 일어나도 변경이 일어난 소스코드와 연관이 있는 다른 부분들을 미처 파악하지 못하여 큰 오류를 유발할 소지가 있다. 이로 인해 발생되는 소스코드 변경에 뒤따른 추가적인 개발 작업의 비용 소모는 경우에 따라 막대할 수도 있기 때문에 본 도구를 개발단계에서 활용한다면, 사전에 소스코드 변경에 따른 영향을 분석함으로써 효율성을 최대화시킬 수 있을 것이다. 즉, 본 도구를 소프트웨어 검증 및 개발 단계에서 널리 이용한다면, 이를 통해 바이탈한 철도 소프트웨어의 오류를 미연에 방지하여 안전성과 신뢰성을 확보하는데 크게 기여할 수 있을 것이다.

참고문헌

1. IEC 61508, "Railway Applications – The specification and demonstration of RAMS", 1998.
2. IEC 62279, "Railway Applications – Software for railway control and protection systems", 2002.
3. 황종규, 조현정, 김형신, “열차제어시스템 소프트웨어 안전성 평가도구의 설계”, 한국철도학회 논문집, 제11권 제2호, pp.139-144, 2008. 4.
4. Zage, W.M and Zage, D.M., "Evaluating design metrics on large-scale software", IEEE Trans. on Software Eng., pp.75-81, Vol. 10, Issue 4, Jul 1993.
5. M. Fewstar, D. Graham, “Software Testing Automation: Effective use of test execution tools”, ACM Press, Addison Wesley, 1999.