2003 춘계학술발표회 논문집
한국원자력학회

KNICS 원자로보호계통 소프트웨어 검증을 위한 체크리스트 개발

Development of Checklists for Verification of KNICS Reactor Protection System Software

손한성, 이장수, 김장열, 차경호, 천세우, 권기춘
한국원자력연구소
대전광역시 유성구 덕진동 150

유영제, 김두환, 김영진
BNF ㈜
대전광역시 유성구 덕진동 150

요 약

원자로보호계통과 같은 안전에 매우 중요한 시스템은 소프트웨어에 대한 철저한 검증을 요구한다. 원전계측제어시스템개발사업단에서 개발하고 있는 안전에 중요한 소프트웨어에 대한 검증은 크게 검토(Review), 인스펙션(Inspection), 정형 검증(Formal Verification)의 세 가지 업무로 이루어 지는데, 검토보다는 인스펙션이, 인스펙션보다는 정형 검증이 보다 심층적인 검증이라 볼 수 있다. 본 논문은 인스펙션을 수행하는데 필요한 체크리스트에 대하여 소개한다. 본 연구에서 개발된 체크리스트는 검증 업무의 주안점이 되는 완전성(Completeness), 일관성(Consistency), 정확성(Correctness)에 초점이 맞추어져 있다.

Abstract

Safety critical systems such as reactor protection systems in nuclear power plants require strict and thorough software verification. In KNICS project, verification of safety critical software is being performed by using review, inspection, and formal verification, which are all prevailing verification techniques. The depth of verification goes deeper as it moves from review to inspection, to formal verification. This article introduces the checklists that are needed for inspection. The checklists developed in this work are focused on completeness, consistency, and correctness that are main viewpoints of software verification generally.

1. 서론

원전계측제어시스템의 경우, 시스템에 내장되는 소프트웨어에 대하여 다양한 측면으로, 또한 다양한 기법을 이용하여 확인 및 검증(Verification and Validation)을 수행할 필요가 있다. 특히 원자로보호계통과 같은 안전에 매우 중요한 시스템은 소프트웨어에 대한 철저한

검증을 요구한다. 규제기준에서 채택하고 있는 IEEE Std. 7-4.3.2[1]이나 IEEE Std 1012[2]나 IEEE Std 1074[3] 등과 같은 산업표준에서는 소프트웨어 개발 주기의 각 단계별로 확인 및 검증을 수행할 것과 또한 안전에 중요한 정도에 따른 철저한 확인 및 검증 수행을 권장하고 있다.

소프트웨어에 대하여 철저한 검증을 수행하기 위해서는 요구사항분석, 설계, 구현 등의 활동을 체계적으로 수행하는 것이 전제되어야 한다. 이는 검증 가능한 소프트웨어를 생산할 때 철저한 검증이 이루어질 수 있기 때문이다. 본 연구진은 규제기준을 만족함과 동시에 검토, 인스펙션, 그리고 수학적 명세 언어로 설계하여 수학적인 증명 방법으로 검증을 수행하는 정형기법 등을 이용하여 소프트웨어에 대한 철저한 검증을 수행하고 이를 통하여 소프트웨어의 안전성 및 신뢰성에 대한 확신을 높이기 위한 노력을 하고 있다. 이러한 노력의 일환으로 원전계측제어시스템사업단에서 개발하는 안전에 중요한 소프트웨어에 대한 소프트웨어 검증 절차서를 개발하였다[4]. 본 논문은 이러한 검증 기법들 가운데 인스펙션을 수행하는데 필요한 체크리스트에 대하여 소개한다. 본 연구에서 개발된 체크리스트는 검증 업무의 주안점이 되는 완전성(Completeness), 일관성(Consistency), 정확성(Correctness)에 초점이 맞추어져 있다.

2. KNICS 원자로보호계통 소프트웨어 검증 절차

원전계측제어시스템사업단은 KNICS(Korea Nuclear Instrumentation and Control System)의 일환으로 원자로보호계통을 개발하고 있다. KNICS 원자로보호계통 소프트웨어는 철저한 검증을 거쳐 개발되며, 이를 위해서 다음 그림 1과 같은 검증 절차를 따르고 있다.

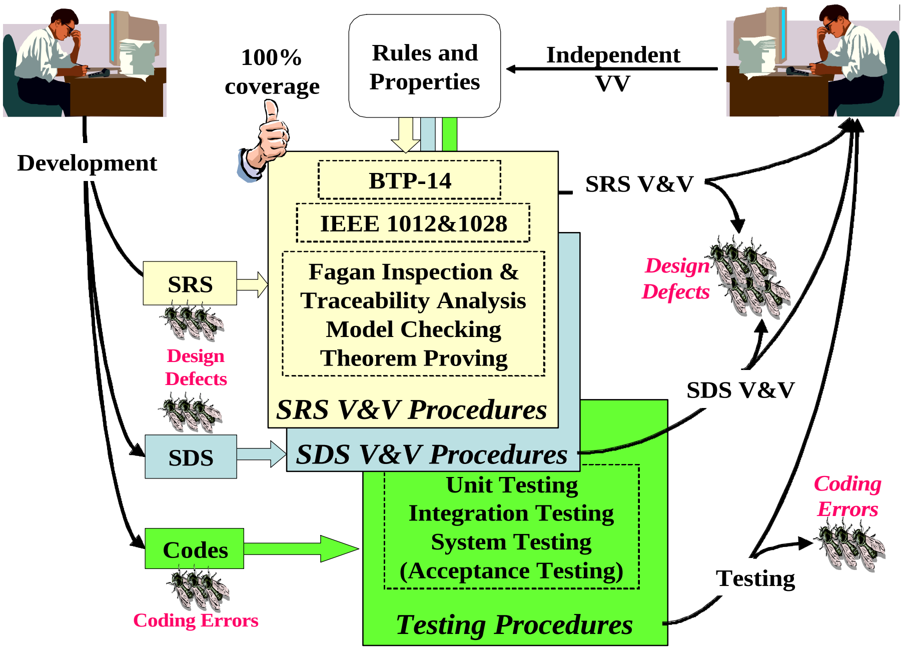

그림 1. KNICS 원자로보호계통 소프트웨어 검증 절차

그림 1은 본 연구진이 개발한 검증 절차가 크게 규제요건을 만족하는가에 대한 검토, 인스펙션 및 추적성 분석, 모델체킹 및 정리증명과 같은 정형적 검증으로 구성되어 있음을 보여준다. 검증 절차에 대한 보다 상세한 내용은 본 연구진의 또 다른 논문 [5]를 참조하기 바란다.

3. 인스펙션(Inspection)

그림 1에서 알 수 있듯이 KNICS 원자로보호계통 소프트웨어 검증을 위해 채택한 인스펙션 방법은 Fagan 인스펙션 방법이다. Fagan 인스펙션은 소프트웨어 품질을 검증하기 위한 소프트웨어 검토(review) 방법 중의 하나로서 1970년대 중반 IBM에 있던 Fagan에 의해 제안되었다[6]. Fagan 인스펙션에서는 인스펙션의 각 단계를 체계화하여 일정한 작업 순서를 따르게 하며, 참가자의 역할을 명확히 정의하고 소프트웨어 개발에 참여하지 않은 사람들을 인스펙션에 참가하게 함으로써 다양한 에러 검출의 시각을 구할 수 있게 하였다. 이러한 장점으로 인해 Fagan 인스펙션은 학계에서뿐만 아니라 실제 산업계에서도 많이 적용되어 효과를 나타내고 있다.

Fagan 인스펙션이 처음 제안되었을 때에는 주로 소프트웨어의 설계나 코드에 적용되던 것이 지금은 요구사항 명세, 설계, 코드, 테스트 플랜 등 소프트웨어 개발과정의 전 단계로 확장 적용되어 소프트웨어의 품질 향상 기법으로 더욱 그 사용범위가 넓어지고 있다.

3.1 구성

Fagan 인스펙션은 6개의 단계로 구성되어 있고 참가자들은 5개로 역할로 나뉘어져 정형화된 진행과정에 따라 소프트웨어의 결점을 찾아 낸다. Fagan 인스펙션에서는 크게 참가자를 조정자(moderator), 기록자(recorder), 낭독자(reader), 저자(author), 인스펙터(inspector)로 나누며, 참가자의 역할은 다음과 같다.

- 조정자

조정자는 인스펙션의 진행과정에 대한 책임을 지고 팀 구성원들이 그들의 능력을 최대한 발휘하도록 해야 한다. 우선 인스펙션을 하려는 소프트웨어가 시작조건에 적합한가 검증함으로써 인스펙션을 준비하고, 인스펙션 팀을 구성하고 역할을 나누며, 인스펙션 진행이 계획대로 잘 되도록 이끌어간다. 마지막으로 인스펙션 결과물이 종료조건에 적합한지 인증하는 것까지 책임져야 한다.

- 기록자

인스펙션에서 발견된 결점과 그에 관한 모든 정보를 체계적으로 기록한다. 이 기록은 인스펙션이 끝난 후 남는 구체적인 결과물로 재작업에 있어서 중요하게 사용되기 때문에 일정한 형식에 따라 명확하게 결점을 정리해야 한다.

- 낭독자

인스펙션 대상인 소프트웨어를 읽는 역할을 맡아서 그 소프트웨어를 체계적으로 분석한다. 한번에 한 단락씩 읽고 요약하여 그 뜻을 전달함으로써 나머지 팀 구성원들은 세부적인 사항에 집중할 수 있도록 한다. 그리고 일반적으로 저자는 낭독자의 역할을 같이 맡아서는 안 된다.

- 저자

저자는 소프트웨어를 개발한 사람이거나 그 소프트웨어를 변화시키는데 대해 책임을 가지는 사람이다. 낭독자가 답변할 수 없는 특정 문제에 대해 설명하고 그 자신의 그 소프트웨어에 대한 특별한 지식을 바탕으로 결점을 찾아낸다.

- 인스펙터

인스펙터는 인스펙션을 하려는 소프트웨어에 대해 사전 지식이 없는 사람일 수도 있다. 주어진 소프트웨어에 대해 각각 다른 관점과 관심을 가진 개개인이 인스펙션에 참여하는 것은 사고의 방향을 다양하게 함으로써 자료에 대한 종합적인 이해의 증진과 결점 검출 효과의 향상에 도움이 될 때가 많다.

3.2 절차

Fagan 인스펙션의 각 단계는 크게 계획(planning), 개관(overview), 준비(preparation), 인스펙션 회의(inspection meeting), 재작업(rework), 사후 점검(follow-up)으로 나누어진다. Fagan 인스펙션의 절차는 다음과 같다.

- 계획

인스펙션 전반에 관한 계획을 수립하는 단계이다. 저자가 자신의 자료가 인스펙션을 할 준비가 되었다고 생각하고 조정자에게 제출하면, 조정자는 그것이 시작 조건에 부합되는지 판단한다. 만약 그것이 인스펙션에 적당하면 저자와 조정자는 적당한 인스펙션 팀을 구성하고 전체적인 계획을 수립한다.

- 개관

개관 모임은 저자가 인스펙션할 소프트웨어에 대한 대략적인 설명을 하는 단계이다. 조정자는 개관이 팀원들의 소프트웨어에 대한 이해를 높이는데 기여를 할 것인가를 판단하고, 개관 단계를 할 것인가에 대해 결정한다. 보통 소프트웨어가 매우 중요하거나 매우 크고 복잡할 때 그리고 팀원들에게 익숙하지 않을 때 개관을 많이 한다.

- 준비

각 팀원들이 각자 미리 정해진 시간에 따라 개인적으로 관련된 자료를 읽어서 인스펙션을 준비하는 단계이다. 체크리스트를 살펴보거나 전형적인 결점을 유형에 따라 순위를 매겨 놓은 자료 등을 살펴보는 것을 포함한다. 준비 동안에 발견된 결점은 기록되어야 하지만, 개인적인 준비 동안 발견된 결점이 모두 최종적인 결점으로 확정되는 것은 아니다. 실제 최종적인 결점은 인스펙션 회의에서 결정된 것으로 국한한다.

- 인스펙션 회의

인스펙션 회의는 인스펙션 과정의 가장 중요한 단계이다. 인스펙션 회의는 문제의 해결책을 찾으려고 해서는 안되며 단지 에러 검출에만 목적을 두어야 한다. 조정자는 미리 정해진 형식에 의해 인스펙션을 진행시킨다. 먼저 모임의 진행 순서를 모두에게 알리고 각자의 역할을 주지시키며 참가자들을 서로에게 소개한다. 그리고 낭독자가 모임을 이끌게 되는데 우선 한 번에 한 문장을 읽고 나머지 팀원들은 방금 읽은 부분에 대한 에러를 찾아낸다. 제기된 에러는 토론을 거치게 되고 에러의 진위 여부와 종류를 결정한다. 모두가 에러임을 동의하면 기록자는 이것을 에러 리스트에 기록하고 이 리스트는 후에 저자의 재작업 자료로 이용된다. 저자는 단지 모임에서 특정 에러에 대한 이해를 돕는 설명을 하는 역할을 맡는다. 모임은 한번에 두 시간을 넘지 않도록 하며 하루에 한번 갖는 것을 원칙으로 하되 필요하다면 두 번까지 할 수 있다. 조정자는 최종적으로 보고서를 작성하고 주어진 자료가 인스펙션을 통과했는지 결정해야 한다. 또한 각각의 결점들은 위치, 심각도, 범주, 타입의 항목에 따라 분류되어 기록되어야 한다.

- 재작업

재작업 단계는 조정자의 인스펙션 리포트에 기록된 발견된 모든 결점에 대해 저자가 수정을 가하는 단계이다. 이렇게 저자에 의해 수정된 개정판(revision)이 완성되고 검증되어야 비로서 인스펙션이 성공적으로 마치는 것이다. 즉, 재작업 단계는 인스펙션 한 결과물이 종료 조건에 만족하도록 하는 단계라 할 수 있다.

- 사후 점검

사후 점검 단계에서는 저자의 개정판(revision)이 정식으로 검증되는 단계이다. 이는 조정자가 인스펙션 동안 발견된 결점이 재작업 단계에서 해소되었는지를 결정한다. 이러한 검증은 인스펙션을 처음부터 다시 함으로써 이루어지는데, 이때 처음의 인스펙션과는 달리

변화한 점과 그에 의존하여 변하는 점 등에 집중해서 인스펙션을 수행한다. 이러한 사후 점검을 통과해야 최종적으로 하나의 소프트웨어에 대한 인스펙션이 끝난다.

위와 같이 설명된 인스펙션 절차에서 알 수 있듯이, 체크리스트는 인스펙션에서 매우 중요한 위치를 차지한다. 이는 인스펙션을 준비하는 단계에서 면밀한 인스펙션을 위한 근거로서 체크리스트가 활용되며, 인스펙션 회의에서도 체크리스트가 인스펙션 결과를 도출하는 기준으로 활용되기 때문이다. 다음 4절에서 KNICS 원자로보호계통 소프트웨어에 대한 체크리스트 중 요구사항 분석 단계의 검증을 위한 체크리스트 개발에 대하여 간략하게 소개한다.

4. 인스펙션 체크리스트 개발

위에서 언급한대로 본 연구진은 인스펙션 체크리스트를 완전성, 일관성, 정확성에 대한 검증을 중심으로 개발하였다. 이러한 특성들은 검증의 목적이라고 할 수 있다. 한편, 인스펙션의 대상이 되는 소프트웨어 제품은 소프트웨어 생명주기 생성물 전반에 해당된다. 소프트웨어 요구사항 명세서(software requirements specifications), 설계 문서(design documents), 코드 문서(code listings), 시험 계획서(test plans), 시험 자료 명세서(test case specifications) 등이 모두 인스펙션의 대상이 된다. 또한 하드웨어 명세서나 하드웨어 설계도 인스펙션의 대상이다. 이와 같은 검증 목적과 검증 대상의 조합에 따라 다양한 인스펙션 형태가 존재하게 되며, 각 인스펙션 형태 별로 발견해야 할 결점의 형태 또한 다양하다. 이러한 결점의 형태를 정의하는 것이 인스펙션 준비 및 수행에서 가장 중요한 일이 된다.

본 연구에서의 체크리스트 개발은 먼저 일반적인 소프트웨어 공학적 인스펙션에서 활용된 경험이 있는 체크리스트를 참조하여 시작하였다. 그림 2가 요구사항 인스펙션 체크리스트를 위한 결점 형태의 예를 보여준다[7].

Completeness

1. Are all sources of input identified?
2. What is the total input space?
3. Are there any “don’t care” sets in the input space?
4. Is there a need for robustness across the entire input space?
5. Are there any timing constraints on the inputs?

…

Ambiguity

1. Are all special terms clearly defined?
2. Does each sentence have a single interpretation in the problem domain?
3. Is the input-to-output mapping clearly defined for each type of run?

Consistency

1. Do any of the designated requirements conflict with the descriptive material?
2. Are there any input states that are mapped to more than one output state?
3. Is a consistent set of quantitative units used? Are all numeric quantities consistent?

그림 2. 요구사항 인스펙션 체크리스트를 위한 결점 형태의 예

그림 2와 같은 자료를 기반으로 원전 계측제어계통 소프트웨어에 대한 검증에 지침을 제공하는 IEEE Std. 1012[2], NUREG/CR 6082[8] 등을 심층 분석하여 일반적인 체크리스트를 원전용 소프트웨어 검증을 위한 체크리스트로 수정 및 보완하는 연구를 수행하였다. IEEE Std. 1012는 소프트웨어 각 개발공정 별로 완전성, 일관성, 정확성 등에 대한 검증 직무 기준(task criteria)를 제시하고 있다. 각각의 직무 기준은 직접적으로 완전성, 일관성, 정확성에 대한 체크리스트가 될 수 있을 정도로 상세하다. 다만, 일부 직무

기준은 상당히 개념적이거나 포괄적이어서 좀 더 구체화될 필요가 있으며, 다른 직무 기준들과 심도(level of depth)를 조절할 필요가 있는 것으로 사료된다. 또한 기술적 검토로써 충분히 검증을 종결할 수 있는 직무 기준이 있는 반면, 좀 더 심층적인 검증을 수행해야 하는 직무 기준들이 있다. 본 연구에서는 이러한 조정 작업 및 선별 작업이 수행되었다. NUREG/CR 6082는 통신 시스템을 위한 지침을 제공하지만, 결국 통신 시스템도 원전계측제어시스템의 일부이므로 체크리스트 개발을 위한 많은 도움을 제공한다. 시스템의 거동이 결정론적인가, 데드록(dead lock) 및 라이브록(live lock)이 발생하지는 않는가 등이 대표적인 예이다.

다음 단계로서, 원전용 소프트웨어 검증을 위한 체크리스트를 KNICS 원자로보호계통 소프트웨어에 맞게 수정 및 보완하는 연구가 수행되었으며, 이 과정에서는 KNICS 원자로보호계통 소프트웨어를 명세한 정형 언어인 NuSCR[9]을 고려하여 체크리스트가 개발되었다. 주목해야 할 것은 KNICS 원자로보호계통의 소프트웨어 요구사항 명세가 어떠한 표기법들을 사용하여 어떠한 절차에 의하여 개발되는가에 대한 충분한 이해가 체크리스트에 반영되었다는 것이다. 이는 필수적인 요소로서, 특히 요구사항 명세서의 내부적인 일관성 등에 대한 체크리스트 개발에 중요한 근거가 된다.

이러한 일련의 연구를 통하여 개발된 체크리스트는 인스펙션의 원활한 수행을 위하여 그림 3과 같이 계층화되었다.

항목 분류	관련 근거	상세 검증 항목		검증 결과
계층 1		계층 2	계층 3	

그림 3. 개발된 체크리스트의 양식 및 계층화된 구조

예를 들어, 완전성에 대한 체크리스트를 기능 정의의 완전성, 입력 및 출력 정의의 완전성, 소프트웨어 행위 명세의 완전성, 인터페이스 완전성 등으로 분류하여 계층 1에 두었고, 여기에 개발된 체크리스트들을 할당하여 계층 2에 두었으며, 계층 2의 체크리스트가 더욱 세분화 될 필요가 있을 경우에는 세분화하여 계층 3에 두었다. 일관성 및 정확성에 대한 체크리스트도 동일한 방법으로 계층화하였다.

이러한 체크리스트를 근거로 인스펙션을 수행한 검증 결과는 소프트웨어 개발 팀의 재작업 과정에 반영되며, 검증 팀은 이를 토대로 최종적인 검증 의견을 개진하여 검증 보고서를 발간한다.

5. 결론

본 논문에서는 원전계측제어시스템 소프트웨어에 적합한 검증 기법이라 할 수 있는 인스펙션에 대하여 소개하고, 인스펙션을 수행하기 위한 체크리스트가 어떻게 개발되었는가를 소개하였다. 개발된 소프트웨어 인스펙션 체크리스트는 일반적인 체크리스트로부터 시작하여 단계적으로 구체화하였으므로, 시스템에 의존적인 부분만을 변경한다면, 원자로보호계통 소프트웨어 외에도 많은 원전계측제어시스템 소프트웨어에 적용할 수 있을 것이다. 또한, 더욱 효과적인 검증 수행을 위하여 계층화된 체크리스트를 제안하였다. 계층화된 체크리스트는 인스펙션을 수행하는 검증 팀이 관심을 집중할 수 있게 해줄 것이다.

알림

이 연구는 원자력 기반기술 분야 원전계측제어시스템개발 사업으로 수행되는 디지털 계측제어 인허가 확보기술 개발 과제에서 수행되었음.

참고문헌

[1] IEEE, “IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations, ” IEEE Std 7-4.3.2-1993, Sep., 1993.

[2] IEEE, “IEEE Standard for Software Verification and Validation Plans,” ANSI/IEEE Std. 1012-1986, Feb., 1987.

[3] IEEE, “ Standard for Developing Software Life Cycle Processes,” IEEE Std. 1074-1997.

[4] 차경호 외, “소프트웨어 설계검증 승인요건에 대한 일치성 분석,” 2002 추계학술발표회논문집, 한국원자력학회, 2002.

[5] 차경호 외, “The KNICS Approach to Verification and Validation of Safety-Critical Software for RPS Prototype,” 2003 춘계학술발표회논문집, 한국원자력학회, 2003. (발표 신청)

[6] M. E. Fagan, “Design and Code Inspections to Reduce Errors in Program Development, ” IBM Systems Journal, Vol. 15, No. 3, 1976.

[7] A. F. Ackerman, L. S. Buchwald, F. H. Lewski, “Software Inspections: An Effective Verification Process, ” IEEE Software, Vol. 6, No. 3, May 1989, pp. 31-36.

[8] G. G. Preckshot, “ Data Communications, ” NUREG/CR 6082, May 1993.

[9] T. Kim and S. Cha, “Automated Structural Analysis of SCR-style Software Requirements Specifications Using PVS. Journal of Software Testing, Verification, and Reliability, Vol. 11, No. 3, pp. 143-163, 2001.