시스템 안전성을 위한 복잡도 메트릭스 고찰

이진호 ○,황대연, 심재환, 최진영

고려대학교 컴퓨터학과

{jhlee, dyhwang, jhsim, choi} @formal.korea.ac.kr

A study on the complexity metrics for system safety

Jeanho Lee○, Daeyon Hwang, Jeahwan Sim, Jinyoung Choi

Dept.Computer Science, Korea University

요 약

시스템의 복잡도를 정의하고 측정하는 일은 보다 신뢰성있고 효율적인 시스템을 만들기 위한 초석이다. 복잡도가 증가함에 따라 복잡도에 영향을 받는 안전필수 시스템의 안전성을 정의하고 측정할 메트릭스가 필요성도 증가하고 있다.

본 논문에서는 임베디드 시스템의 복잡성과 안전성을 고찰하고, 안전성의 성질을 서술하는 복잡도 메트릭을 제안한다.

1. 서 론

임베디드 시스템의 발달은 안전필수 시스템(safety-critical system)의 확산과 더불어 안전필수 시스템에서 소프트웨어와 하드웨어의 경계가 없어지는 결과를 가져오고 있다[8].

임베디드 시스템은 환경과 상호작용하는 하드웨어 장치들의 조합과 시스템의 행위를 정의하는 소프트웨어 사이의 상호의존성이 존재한다.

소프트웨어의 신뢰성 예측과 모델링을 위해 소프트웨어의 복잡도 메트릭들을 정의하고 적용하는 연구가 진행되었다[10]. 안전성 측정을 위해 시스템 혹은 소프트웨어의 복잡도 메트릭들을 이용하는 연구는 부족하다.

2장에서는 시스템의 안전성과 복잡도 메트릭의 특성을 살펴보고, 3장에서는 임베디드 시스템의 안전성 관련 복잡도 메트릭을 제안하고, 간단한 안전필수 임베디드 시스템인 인슐린 펌프 예제를 가지고 적용해본다. 4장에서 결론 및 향후 연구과제에 대해 기술한다.

2. 관련 연구

2.1 복잡도

IEEE 표준 용어사전에 따르면 복잡도는 “시스템이나 구성요소가 이해하고 검증하기 어려운 디자인이나 구현을 가지고 있는 정도”로 정의된다.[6].

복잡도에 관한 연구는 수학, 컴퓨터 과학, 시스템 과학, 경제학, 심리학, 사회과학 등의 여러 분야에 걸쳐 다양하게 이루어지고 있다. 복잡도의 측정 단위와 측정 기법도 적용 분야마다 다르게 정의되고 사용되고 있다. coskun&Grabowski 의 연구는 적용분야와 측정 수준을 기준으로 소프트웨어의 복잡도와 측정 모델을 분류하였다[1]. 임베디드 시스템을 소프트웨어의 한 종류로 가정하고 기존의 소프트웨어 복잡도 메트릭을 적용하여 분류하였다. 시스템 수준의 복잡도를 표현하기 위해서는 새로운 시스템 복잡도 측정 기법과 모델링이 요구된다.

Mcdermid 의 연구에 따르면, 복잡도의 원인은 크게 내부 요인과 외부 요인으로 나눌 수 있고, 3가지 특성으로 규모(scale), 다양성(diversity), 연결성(connectivity)을 갖는다[9]. Grassberger 는 시스템 복잡도가 시스템의 구조와 관계가 있다는 것을 발견했다. 완벽하게 질서가 정해진 구조나 완벽하게 무질서한 구조는 구조적 복잡도(structural complexity)가 낮고, 질서(order)와 임의성(randomness)이 혼합될수록 더욱 복잡해진다. 구조적 복잡도가 낮으면 선형성(linearity)이 높고, 복잡도가 높으면 비선형성(non-linearity)이 높다 [4].

선형적 시스템의 경우 구조적으로 단순하고 실행 경로가 선형적이다. 실행 경로는 순차적인 순서를 따라 진행된다. 시스템 구성요소(component)와 하위 시스템(subsystem)이 상호 작용하는 분기 경로(branching path)가 단순하다. 안전성과 관련한 위험원 사건(hazardous event)이 발생하면 사건들의 선형 경로를 따라 순차적으로 발생하게 된다.
비선형적 시스템의 경우 구조적으로 복잡하고 실행 경로가 선형적이지 않다.

2.2 시스템 안전성

안전성의 정의는 “사망, 부상, 직업병, 혹은 장비나 재산에 대한 손상이나 손실, 그리고 환경 손상”으로 정의된다[3]. 안전성의 평가는 구성요소나 부분이 아닌

시스템 전체를 대상으로 이루어진다. 시스템과 관련된 품질 항목으로 안전성과 신뢰성이 사용된다. 신뢰성과 가용성(availability)은 안전성의 필수조건이지만 충분조건은 아니다. 신뢰성은 시스템이 주어진 요구사항에 순응하는지에 초점을 맞추고, 안전성은 시스템이 요구사항에 순응하는지와 무관하게 오류없이 동작하도록 보장하는 것에 초점을 두고있다[12].

전통적인 시스템 안전성에 관한 메트릭은 확률에 기반한 정의를 사용하고 있다: MTTF(mean time to failure), MTTUF(mean time to unsafe failure), TMR(triple module redundancy) [7]. 소프트웨어가 시스템의 안전성에 영향을 끼치는 요소나 메트릭스에 대해 정의되어 있지 않고 연구가 미흡하다.

신뢰성에 대한 대표적인 정의는 두 가지로 정의된다: "시스템이나 시스템의 구성요소가 기술된 환경아래에서 특정 시간동안 필수 기능을 수행할수 있는 능력", 혹은, "어떤 장치가 정해진 시간동안 또는 사용량만큼 고장없이 수행할 확률" [6].

시스템 안전성에 영향을 미치는 시스템 복잡도는 3가지 요소를 가진다[mcdermid'00]: 규모(scale), 다양성(diversity), 연결성(connectivity). 규모는 시스템 안에 존재하는 요소들의 개수를 말하고, 다양성은 시스템이 얼마나 다양한 종류의 구성요소들로 이루어져 있는지를 의미한다. 연결성은 시스템 안의 구성요소들 사이의 상호관계를 말한다.

3. 안전성을 위한 임베디드 시스템 복잡도 메트릭스

본 연구에서는 임베디드 시스템의 안전성과 관련있는 복잡도를 측정하기 위해 mcdermid 의 시스템 안전성의 복잡도 요인 개념과 특성을 반영하고 grassberger 의 복잡도와 구조(structure)의 관계를 그래프 이론에 적용시킨 메트릭을 제안한다.

먼저, 안전성의 요인이 되는 시스템 복잡도의 특성인 규모, 다양성, 연결성에 대한 메트릭을 정의한다.

시스템의 규모를 확장시키는 것은 시스템 안에서 가장 복잡한 구성요소 혹은 개체에 의해 제약을 받는다. 왜냐하면 stress point 가 되기 쉽기 때문이다 [5]. 시스템의 규모 복잡도는 시스템의 구성요소나 개체 사이의 결합도(coupling)로 표시할 수 있고, 서로 다른 개체나 구성요소들 사이의 상호작용을 분석하여 결합도를 측정할 수 있다[11]. 선형적 구조인 경우에 적합하지만, 비선형적 구조에는 적합하지 않는 단점이 있다.

비선형적 구조에도 적용시키기 위해, Phukan 이 제안한 규모 복잡도 공식은 그대로 유지하지만 인자의 값을 다르게 정의한다. 내부 복잡도 (intra-complexity)의 경우 제어 명령문과 제어 조건의 변수의 개수만을 측정한 것과는 달리, 각각의 제어 명령문의 모든 가능한 분기의 개수와 제어 조건의 변수의 개수를 측정한다. 상호작용의 회수의 경우 현재 구성요소나 모듈에서 다른 구성요소나 모듈의 직접 호출 회수만을 측정한 것과는 달리, 간접 호출의 회수까지 측정한다.

Fan-in 과 fan-out 의 경우는 현재 구성요소나 모듈에서 종료되는 지역적 제어와 시작되는 지역적 제어의 개수를 측정한다.

시스템의 다양성은 시스템의 구성요소나 모듈의 종류의 개수로 표시할 수 있다. 구성요소나 모듈의 종류가 다양할수록 각 요소들을 분석하는데 더욱 많은 시간과 노력이 소요된다. Control flow graph 를 구성하고, 각 node 의 out degree 를 측정한다.

시스템의 연결성은 구성요소나 모듈의 상호관계를 나타낸다. 그래프 이론에서 비선형 시스템의 그래프는 선형적 독립 경로를 포함한다[2]. 구성요소 사이의 상호 관계를 정의하기 위해 cyclomatic 복잡도, control flow graph 의 degree , path 의 개수, Shepard 의 복잡도 4가지를 측정한다.

3.1 사례 연구

sommerville 의 인슐린 펌프 모델을 대상으로 적용하였다[12]. 개인용 인슐린 펌프는 임베디드 센서를 이용하는 안전필수 시스템으로서 주기적으로 혈압을 체크하여 혈당을 정상 수준으로 유지하도록 인슐린을 혈관에 투여하는 일을 한다. 본 연구에서 사용한 예제는 실제 구현 코드가 아닌 모델링 자동화 도구를 통해 자동 생성된 java 코드를 사용하였다.

인슐린 펌프 시스템의 구조는 그림 1과 같다.

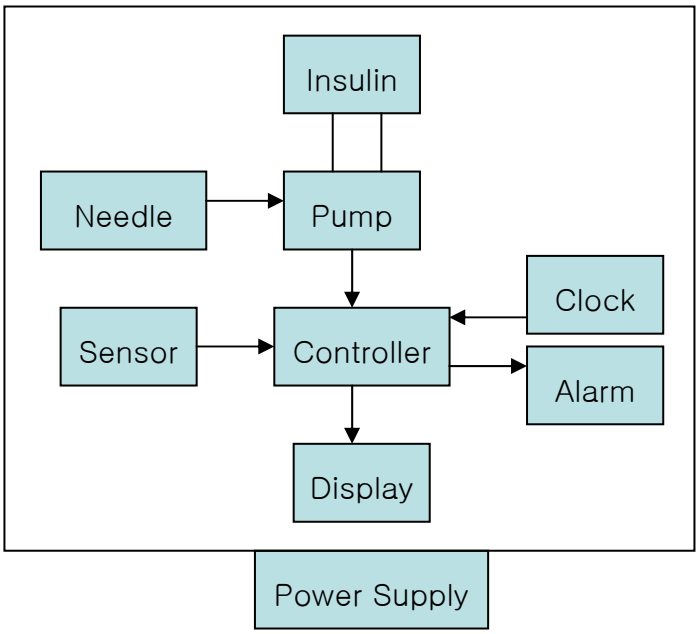

그림1 인슐린 펌프 시스템의 구조

환자의 신체에 삽입된 센서가 주기적으로 환자 혈액의 글루코스(glucose)의 수준을 감지하여 시스템에 전송하면, 제어기는 센서로부터 전달된 혈액 글루코스 수준값을 읽는다. 시스템은 연속으로 읽은 값들을

비교하여, 혈당 수준이 상승했다고 판단되면 인슐린을 주입시켜 혈당값을 떨어뜨린다. 최상의 상황은 혈당의 수준을 일관되게 어떤 안전한 영역대 안에서 지속시키는 것이다.

생성된 6개의 자바 클래스 파일을 분석하였고, 각 클래스 파일마다 포함되어 있는 대표적인 내부 함수들을 분석하였다. 표1은 외부 복잡도의 측정 결과로서 상호 복잡도와 fan-in, fan-out 값을 측정하였다.

표 1 모듈 외부 복잡도 결과

Module	intra	Fan-in	Fan-out
Sensor	16	2	2
Power-supply	12	1	4
Insulin-pump	50	2	4
Controller	166	8	14
Clock	72	2	2
Insulin-reservoir	5	2	2

표2는 내부 복잡도의 측정 결과로서 각 함수마다의 제어 경로를 구성하고 out degree 값과 path의 개수를 측정하였다.

표 2 모듈 내부 복잡도 결과

function	logical	data	total
sensor_setReading	2	7	9
sensor_getReading	4	7	10
Power-supply_batteryLevel	5	5	6
Power-supply_setBatteryLevel	4	4	5
Insulin-pump_setPumpWorking	4	14	16
Insulin-pump_deliverInsulin	1	5	6
controller_startClock	3	6	7
controller_changeState	26	18	41
controller_setDose	16	3	22
clock_run	35	8	40
Insulin-reservoir_replaceInsulin	6	3	11

4. 결론 및 향후 연구

본 연구에서는 소프트웨어의 복잡도 메트릭스들을 살펴보고 시스템 안전성의 특성을 고찰해보았다. 본 연구에서는 임베디드 시스템의 안전성과 관련한 새로운 복잡도 메트릭스들을 제안하고, 소규모 단위의 실험을 수행하였다. 실제 현장에서 사용되는 안전 필수 시스템 대신에 대중에게 알려진 안전필수 모델을 사용하고, 실제 구현된 안전 필수 시스템의 코드대신에 모델링을 통한 자동 생성된 코드를 사용하는 제한된 실험을 통해 단순한 샘플 데이터밖에 얻지 못해서 제안한 메트릭의 유효성 검증(validation)이 충분하게 이루어지지 못했다. 향후 다양한 실험이나 업계에서 사용되는 대량의 실제 데이터를 통해 확보한 충분한 실험 데이터를 가지고 제안한 메트릭의 유효성에 대한 검토(validation)가 필요하다.

본 연구에서는 구조적 분석을 통해 안전성 관련 정적인(static) 복잡도를 정의하였는데, 향후에는 상태 변화를 통한 행위적 분석으로 안전성의 원인성(causality) 관계나 핵심 경로(critical path)같은 동적인(dynamic) 복잡도를 정의해야 할 것이다.

향후 과제로는 복잡도 관련 메트릭을 임베디드 시스템의 안전성과 신뢰성을 측정하고 예측 지표로 활용하는 연구가 필요하다.

참고문헌

[1] E. Coskun & M. Grabowski. An Interdisciplinary model of complexity in embedded intelligent real-time systems. Information and Software Technology, 43(2001):527-537, 2001

[2] Reinhard Diestel. Graph Theory. Springer-Verlag, 2005.

[3] Report No. MIL-STD-882D. U.S. Department of Defense, 1997.

[4] P. Grassberger. Toward a quantitative theory of selfgenerated complexity.
Intl. J. Theo. Phys. 25, 9 (1986), 907-938, 1986.

[5] S. Henry & D. Kafura. Software structure metrics based on information flow. IEEE Transactions on Software Engineering, 7(5):510-518, 1981.

[6] IEEE Computer Society: IEEE-STD 610.12-1990: Standard Glossary of Software Engineering Terminology. IEEE Press, 1990.

[7] Johnson, B.W., Design and Analysis of Fault Tolerant Digital Systems, Addison-wesley Company, 1989.

[8] R. Lutz. Software Engineering for Safety: A Roadmap. In Future of Software Engineering, ACM Press, pp.3-24, 2000.

[9] J. McDermid. Complexity: Concept, Causes and Control. In IEEE International Conference on Engineering of Complex Computer(ICECCS 2000), 2000.

[10]J. Munson & T. Khoshgoftaar. The Use of Software Complexity Metrics in Software Reliability Modeling. Proceedings of the IEEE International

Symposium on Software Reliability Engineering, pp.2-11, May 1991.
[11] A. Phukan, M. Kalava & V.Prabhu. Complexity metrics for manufacturing control architectures based on software and information flow. Computers & Industrial Engineering 49(2005):1-20, 2005.
[12] I. Sommerville. Software Engineering. 8th Ed, Addison-wesley, 2004.