시스템 오브 시스템즈의 런타임 검증 속성 명세 유형 분석 및 적용*

조은호○, 박수민, Lingjun Liu, Chua Kiat Kian Anthony, 배두환
한국과학기술원
{ehcho, smpark, riensha, anthonyc, bae}@se.kaist.ac.kr

Analysis and Application of Runtime Verification Property Specification Patterns of System of Systems

Eunho Cho, Sumin Park, Lingjun Liu, Chua Kiat Kian Anthony, Doo-Hwan Bae
School of Computing, Korea Advanced Institute of Science and Technology (KAIST)

요 약

시스템 오브 시스템즈(System-of-Systems, SoS)는 독립성을 가진 구성 시스템(Constituent System, CS)으로 이루어진 복합적인 시스템으로, 복잡성이 높아 검증을 진행하는 것이 매우 어렵다. 특히, 정형 검증 기법, 정리 증명, 모델 체킹과 같은 검증 방식은 상태 폭발 문제가 발생하여, 검증 시간이 오래 걸리고, 효율성이 떨어진다. 반면, 런타임 검증은 모델을 고려하지 않고 시스템의 실행만 고려하므로, 기존 검증 기법보다 빠르게 검증이 가능하다. 시스템 오브 시스템의 런타임 검증을 체계적으로 진행하기 위해 필요한 검증 속성 명세 유형과 그 적용에 관한 연구는 활발히 이루어지지 않은 상황이다. 따라서, 본 연구에서는 명세 유형을 분석하여 SoS에 사용하기 적합한 런타임 검증 속성 명세 유형을 제시한다. 또한, 정립된 명세 유형을 SoS의 시뮬레이션 도구에 적용하고, 오류 모델을 통한 런타임 검증 결과의 분석과 검증 오버헤드의 분석를 제시함으로써, SoS의 런타임 검증 속성 명세 유형을 통한 런타임 검증이 효과적임을 보인다.

1. 서론

시스템 오브 시스템즈(System-of-Systems, SoS)는 관리적/운영적 독립성을 가진 구성 시스템(Constituent System, CS)으로 이루어진 복합적이고 진화하는 시스템이다. 이러한 복잡성이 높은 시스템의 경우, 시스템이 실행되는 동안 창발적 특성이 발생하여 검증을 진행하는 것이 매우 어렵다. [1] 특히, 정형 검증 기법(Formal Verification Method), 정리 증명(Theorem Proving), 그리고 모델 체킹(Model Checking)과 같은 방식의 경우, 상태 폭발 문제(State Explosion Problem)가 발생하여 검증 시간이 오래 걸리고 효율성이 떨어진다. [2]

반면, 런타임 검증(Runtime Verification)은 정형 검증 기법, 정리 증명, 그리고 모델 체킹과 같은 기존의 검증 기법보다 가볍게 사용할 수 있다. 런타임 검증은 프로그램이 실행될 때, 프로그램을 분석하고 실행 결과를 통하여 검증을 진행하는 동적 소프트웨어 분석 및 접근 방식이다. 즉, 런타임 검증은 코드 또는 모델을 고려하여 검증을 진행하는 것이 아니라, 시스템의 실행만 고려하기 때문에 [3] 시스템의 복잡성이 증가하더라도 검증을 효율적으로 진행할 수 있다. 하지만, SoS 의 런타임 검증을 체계적으로 진행하기 위해 런타임 검증 속성 명세 유형(Property Specification Pattern)은 그 필요성에도 불구하고, 아직 정립되지 않은 상황이다. [4]

따라서, 본 논문에서는 검증 속성 명세 유형을 분석하여, SoS 의 런타임 검증의 검증 속성 유형을 정립하고자 한다. 또한, 정립된 런타임 검증 속성 유형을 SoS 런타임 검증 도구에 적용하여, 오류 모델을 사용한 런타임 검증 결과의 분석과 런타임 검증의 오버헤드의 분석을 통해 SoS 의 런타임 검증의 적합성과 효율성을 보이고자 한다.

본 논문의 구성은 다음과 같다. 2 장에서는 검증 속성 명세 유형과 관련한 선행 연구를 분석하여 SoS 의 런타임 검증 속성 명세 유형을 정리한다. 3 장에서는 분석된 명세 유형을 기반으로 런타임 검증을 SoS 에 적용한다. 4 장에서 결론 및 향후 발전 방향을 제시한다.

2. 시스템 오브 시스템즈의 런타임 검증 속성 명세 유형 분석

2.1 검증 속성 명세 유형 관련 선행 연구 분석

검증 속성 명세 유형은 검증 속성의 재사용성을 고려하여, 검증 속성 명세의 공통적인 부분을 추상화 및 유형화시켜 정리하였다. 검증 속성 명세 유형을 이용하면 시스템의 특정 사건과 검증이 이루어지는 범위를 나타내는 범위 유형과 필요한 변수를 추가하여 쉽게 검증 속성을 명세할 수 있다. 검증 속성 명세 유형이 정립되면, 도구 지원, 자동화, 그리고 확장에 유리하다.

Dwyer 외[5]는 처음으로 검증 속성 명세 유형을 제시하였다. 해당 연구에서는 검증 속성 명세 유형을 처음으로 정의하고, 각각의 유형을 형식 검증을 위하여 계산 트리 논리(CTL), 선형 시제 논리(LTL), 그리고 정규식으로 표현하였다.

Konrad 외[6]의 연구에서는 Dwyer 외[5]의 연구를 확장하여, 특정 시간 내에 반복되는 사건을 나타내는 Recurrence 유형, 특정 사건이나 상태의 지속 시간을 나타내는 'Duration' 유형을 새로 제시했다.

Autili 외[7]의 연구에서는 Dwyer 외[5]의 연구, Konrad 외[6]의 연구 등을 비롯해 검증 속성 명세 유형에 관한 연구를 종합하여 10 개의 기본 검증 속성 유형을 시간 제약이 포함된 유형인 Time-constrained, 확률적 요소를 표현한 Probabilistic 유형으로 확장할 수 있도록 정리하였다.

Blein 외[8]의 연구에서는 실행 기록에 대한 구체적인 검증을 목적으로 하여, 'Prevention' 유형을 추가하였으며, Bellini 외[9]의 연구에서는 특정 사건이나 상태가 지속되는 동안을 나타내는 'Presence', 'Absence' 범위 유형을 제시하였다.

표 1은 소개한 선행 연구들에서 제시된 검증 속성 명세 유형들과 검증 범위 유형들을 비교한 표이다. Dwyer 외[5]의 연구

* 이 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(No. 2015-0-00250, (SW 스타랩) 모델 기반의 초대형 복잡 시스템 분석 및 검증 SW 개발)과 과학기술정보통신부 및 정보통신기획평가원의 대학ICT 연구센터지원사업의 연구결과로 수행되었음 (IITP-2020-2020-0-01795)

표 1. 검증 속성 명세 유형 선행 연구 비교

평가 기준 / 연구	Dwyer 외 [6]	Konrad 외 [7]	Autili 외 [8]	Blein 외 [9]	Bellini 외 [10]
기본 검증 속성 명세 유형	Absence, Universality, Existence, Bounded Existence, Precedence, Response				
추가된 검증 속성 명세 유형	-	Recurrence, Duration	Until	Prevention	Recurrence, Duration
Time-constrained 유형	O	O	O	O	O
Probabilistic 유형	X	X	O	X	X
검증 범위 유형	Globally, Before, After, Between and, After until				Presence, Absence
검증 방식	정형 검증	정형 검증	정형 검증	정형 검증	정형 검증

에서 제시된 기본 검증 속성 명세 유형들과 검증 범위 유형 들은 동일하게 사용되었으며, 추가로 Recurrence, Duration, Until, Prevention 기본 유형이 제시되었다. 검증 범위 유형은 Bellini 외[9]의 연구에서 추가로 제시되었다. 제시된 선행 연구들에서는 해당 검증 속성 명세 유형들과 범위 유형들에 대한 정형 검증에 대한 논의만 이루어졌으며, 이를 런타임 검증이나 다른 검증 방식에 활용하는 방안에 대해서는 구체적으로 언급된 바가 없다.

2.2 시스템 오브 시스템즈의 런타임 검증 속성 명세 유형

분석된 선행 연구를 기반으로, SoS 의 런타임 검증에 적합한 검증 속성들을 분석하여, 표 2 와 표 3 에 본 연구에서 사용하는 런타임 검증 속성 명세 유형과 범위 유형을 나타내었다.

표 2. 런타임 검증 속성 명세 유형

유형	설명
Absence	{P}가 절대로 만족되지 않아야 한다.
Universality	{P}가 항상 만족되어야 한다.
Existence	{P}가 언젠간 만족되어야 한다.
Bounded Existence	{P}가 최대 혹은 최소 x 번 만족되어야 한다.
Minimum Duration	{P}가 최소 x 시간 단위 동안 만족되어야 한다.
Maximum Duration	{P}가 최대 x 시간 단위 동안 만족되어야 한다.
Recurrence	{P}가 x 시간 단위 간격으로 반복해서 만족되어야 한다.
Precedence	{Q}가 만족된다면, 그 이전에 {P}가 만족되어야 한다.
Response	{P}가 만족된다면, 그 이후에 {Q}가 언젠간 만족되어야 한다.
Prevention	{P}가 만족된다면, 그 이후에 {Q}가 절대로 만족되지 않아야 한다.
Until	{Q}가 만족되기 전까지 지속해서 {P}가 만족되어야 한다.

표 2 는 정리된 11 개의 런타임 검증 속성 명세 유형을 나타낸다. {P}와 {Q}는 한 개 이상의 사건이나 상태를 뜻하며, 여러 사건이나 상태가 하나의 기호로 표시될 수 있다. 런타임 검증은 시스템이 실행되는 동안 진행되어야 하므로 기존에 정의되었던 Probabilistic 유형은 제외하였다.

표 3 은 런타임 검증에 사용될 수 있는 검증 범위 유형을 나타낸 것이다. {R}과 {S}는 한 개 이상의 사건이나 상태를 뜻하며, 여러 사건이나 상태가 하나의 기호로 표시될 수 있다. {T}는 시간 범위를 의미한다. Bellini 외[9]의 연구에서 제시된 Presence 와 Absence 범위 유형은 During 범위 유형으로 통합하였고, Time-constrained 명세 유형은 Interval 범위 유형으로 새롭게 제시했다.

표 3. 런타임 검증 속성 범위 유형

범위 유형	설명
Globally	언제나 달성되어야 한다.
Before {R}	{R}이 만족되기 전까지 달성되어야 한다.
After {R}	{R}이 만족된 이후에는 달성되어야 한다.
Between {R} and {S}	{R}이 만족된 이후부터 {S}가 만족된 이전까지 달성되어야 한다.
Interval {T}	{T}에 해당하는 시간에는 달성되어야 한다.
During {R}	{R}이 만족되는 동안 달성되어야 한다.

3. SoS 런타임 검증 속성 명세 유형 적용

SoS 런타임 검증 속성 명세 및 범위 유형을 이용하여 런타임 검증을 진행하기 위해, Park 외[10]에서 제시한 SoS 시뮬레이션 및 검증 도구에 런타임 검증을 적용하였다. 위에서 제시한 SoS 시뮬레이션 및 검증 도구는 CS 와 SoS 의 특징을 표현하기 위한 인터렉티브 시뮬레이션과, 시뮬레이션 결과를 이용하여 검증을 진행하는 통계적 검증기로 구분된다. 통계적 검증 방식은 정형 검증 기법과 모델 체킹 검증 기법보다 빠르게 검증을 할 수 있다는 장점이 있지만, 시스템의 크기가 거대해지고 복잡해짐에 따라 검증의 효율성이 떨어지는 공통점을 가지고 있다. 반면, 런타임 검증은 시스템의 복잡성에 제한되지 않는 검증 기법으로 기존의 검증 기법보다 더 빠르게 결과를 얻을 수 있다.

본 장에서는 Park 외[10]에서 제시한 도구에 정립된 런타임 검증 속성 명세 유형 및 범위 유형을 적용하여 런타임 검증의 결과를 분석하고, 검증 과정에서 발생하는 오버헤드를 측정하였다. 대상 도구가 이산 시간 단위를 기반으로 하므로, 런타임 검증은 한 시간 단위마다 이루어진다. 한 시간 단위의 시뮬레이션 실행이 완료되면, 해당 시뮬레이션의 로그와 실행 기록에서 분석에 필요한 값을 추출하는 런타임 모니터링을 시행한다. 런타임 모니터링을 통해 얻은 값을 기반으로 11 가지의 검증 속성 명세 유형과 6 가지의 검증 범위 유형에 정의된 분석을 시행하여 런타임 검증을 매 시간 단위마다 수행한다. 이 연구에서는 2 장에서 제시된 11 가지의 검증 속성 명세 유형과 6 가지의 검증 범위 유형을 모두 사용하여 총 66 가지의 검증 속성에 대한 런타임 검증을 시행하였다.

런타임 검증의 결과 분석을 위해서, 프로그램이 강제로 종료되지 않는 수준에서 사용자가 발생시킬 수 있는 5 가지의 오류 모델을 이용하여 런타임 검증을 진행하였다. 오류 모델의 런타임 검증 결과와 기존 모델의 런타임 검증 결과의 차이가 각

오류마다 다른 양상을 보일 경우, 특정 모델의 런타임 검증 결과와 기존 모델의 런타임 검증 결과의 차이를 보고, 어떤 오류가 삽입되었는 지 예측할 수 있다.

표 4 는 오류 모델을 통한 런타임 검증 분석 결과를 보여준다. 분석 결과, 런타임 검증 결과의 변화를 통해 모델에 변화가 있음을 검출할 수 있었다. 또한, 오류 모델의 유형마다 변경된 검증 결과의 비율과 오류 모델의 종류마다 가장 많은 검증 결과의 변화를 준 검증 속성 명세 유형이 상이함을 확인하였다. 결론적으로, 변화가 발생한 검증 속성 명세 유형 및 범위를 분석하여 시스템의 오류 위치를 예측하는 것이 가능하였다.

표 4. 오류 모델을 통한 런타임 검증 결과 분석

오류 모델	변경된 검증 결과 (%)	변화가 발생한 검증 속성 명세 유형
CS 의 동작 오류 1	33.3%	Absence, Maximum Duration
CS 의 동작 오류 2	51.5%	Existence, Precedence
CS 의 위치 오류	60.6%	Absence, Existence, Bounded Existence
CS 의 동작 순서 오류	50.0%	Existence, Precedence
CS 의 이동 속도 오류	19.7%	Maximum Duration, Recurrence

런타임 검증의 오버헤드 분석을 위해 시뮬레이션에 런타임 모니터링 및 검증 기법을 적용한 모델과 적용하지 않은 모델 사이의 전체 실행 시간을 비교하였다. 런타임 검증의 오버헤드는 시뮬레이션의 로그 혹은 시행 기록을 추출하기 위한 런타임 모니터링과 모니터링한 값을 분석하는 데에서 기인한다. 따라서, 검증 속성이 많아, 모니터링이 필요한 값이 많아지고, 필요한 분석이 많아질수록, 더 많은 오버헤드가 발생하게 된다.

오버헤드 값은 시뮬레이션 실행마다 조금씩 다르게 측정되기 때문에 오차 범위를 줄이고, 실험의 신뢰성을 높이기 위해 같은 환경에서 총 100 회 반복 실행하여, 평균값을 통해 오버헤드 값으로 계산하였다. 표 5 는 런타임 검증의 오버헤드 결과를 보여주며, 런타임 검증의 여부에 따라 13.6954 % 정도의 오버헤드가 발생함을 보여준다.

표 5. 런타임 검증 오버헤드 분석

전체 실행 시간 (100 회 평균 값)		오버헤드
런타임 검증 (X)	16.80896 초 (A)	13.6954 %
런타임 검증 (O)	19.11102 초 (B)	(B-A) / A * 100 (%)

동일한 시나리오에 대하여 통계적 검증을 진행할 경우, 몇 시간 또는 하루 이상의 시간이 소요되는 결과와 비교하였을 때 매우 작은 수치라고 볼 수 있다. 하지만 통계적 검증은 통계적 추측의 하나로, 표본 정보를 이용하여 모집단의 모수에 대한 주장의 진위를 판정하는 방법이고, 런타임 검증은 시스템이 동작하는 동안 검증 속성을 통해 검증을 진행하는 방법으로 시스템 사이의 상호작용 등으로 발생할 수 있는 예상치 못한 시스템의 동작과 같이 시스템이 진행되는 동안에만 확인 가능한 행동들에 대하여 검증이 가능하다. 즉, 런타임 검증의 검증 시간이 빠르다는 이유로 런타임 검증이 통계적 검증 또는 모델 체킹과 같이 검증 시간이 오래 걸리는 검증 기법보다 우수하다고 말할 수 없다. 따라서, 런타임 검증 기법과 기존의 검증 기법을 함께 활용하여 검증을 진행할 경우 더 효율적으로 시스템을 검증할 수 있을 것으로 예상된다.

4. 결론

본 연구에서는 타 검증의 검증 속성 유형들을 분석하여 런타임 검증의 검증 속성 명세 유형을 정립하였다. 또한, 정립된 런타임 검증 속성 명세 유형들을 활용할 수 있도록, SoS 시뮬레이션 도구에 런타임 검증 지원을 추가하였다. 실험 결과 런타임 검증을 통해 오류를 검출하고, 해당 오류의 위치를 예상하는 것이 가능하였다. 또한, 검증 시간의 오버헤드가 크지 않음을 확인하였고, 복잡하고 거대한 시스템에 적용할 수 있음을 확인하였다.

향후 연구에서는 SoS 만의 특성을 보여주는 검증 속성 명세 유형을 제안하고자 한다. 또한, 런타임 검증을 진행하는 과정에서 발생하는 오버헤드 크기를 더 줄이는 것을 목표로 한다.

참고 문헌

[1] Nielsen, Claus Ballegaard, et al. "Systems of systems engineering: basic concepts, model-based techniques, and research directions." ACM Computing Surveys (CSUR) 48.2 (2015): 1-41.

[2] Clarke, Edmund M., et al. "Model checking and the state explosion problem." LASER Summer School on Software Engineering. Springer, Berlin, Heidelberg, 2011.

[3] Leucker, Martin, and Christian Schallhart. "A brief account of runtime verification." The Journal of Logic and Algebraic Programming 78.5 (2009): 293-303.

[4] Bianculli, Domenico, et al. "4.3 Property Specification Patterns for Runtime Verification." A Shared Challenge in Behavioural Specification: 75. 2017.

[5] Dwyer, Matthew B., George S. Avrunin, and James C. Corbett. "Property specification patterns for finite-state verification." *Proceedings of the second workshop on Formal methods in software practice*. 1998.

[6] Konrad, Sascha, and Betty HC Cheng. "Real-time specification patterns." *Proceedings of the 27th international conference on Software engineering*. 2005.

[7] Autili, Marco, et al. "Aligning qualitative, real-time, and probabilistic property specification patterns using a structured english grammar." *IEEE Transactions on Software Engineering* 41.7 (2015): 620-638.

[8] Blein, Yoann, et al. "Extending specification patterns for verification of parametric traces." *Proceedings of the 6th Conference on Formal Methods in Software Engineering*. 2018.

[9] Bellini, Pierfrancesco, Paolo Nesi, and Davide Rogai. "Expressing and organizing real-time specification patterns via temporal logics." *Journal of Systems and Software* 82.2 (2009): 183-196.

[10] Park, Sumin, et al. "SIMVA-SoS: Simulation-based Verification and Analysis for System-of-Systems." *2020 IEEE 15th International Conference of System of Systems Engineering (SoSE)*. IEEE, 2020.