UML Components 방법론의 테일러링을 통한 GNSS 컴포넌트 추출

[0]진달래* 노혜민* 유철중* 장옥배* 이종훈**
*전북대학교 컴퓨터과학과
**한국전자통신연구원 컴퓨터소프트웨어기술연구소 공간정보기술센터
{dlajin, hmino}@cs.chonbuk.ac.kr {cjyoo, okjang}@moak.chonbuk.ac.kr jong@etri.re.kr

GNSS Component Extraction by UML Components Tailoring

[0]Dal-lae Jin* Hye-Min No* Cheol-Jung Yoo* Ok-Bae Chang* Jong-Hun Lee**
*Dept. of Computer Science, Chonbuk National University
**Electronics and Telecommunications Research Institute

요 약

GNSS 시스템을 구축하는데 있어서 기존의 절차지향이나 객체지향 방법에서 벗어나 컴포넌트 개발(CBD) 방법론을 이용하는 것이 재사용성, 유지 보수성, 비용 절감 및 효율성 측면에서 적합하다는 타당성이 제기되고 있다. 따라서 본 논문에서는 기존의 여러 CBD 방법론 중 GNSS 문제 도메인에 가장 적합한 UML Components 방법론을 테일러링하여 GNSS 컴포넌트 추출을 위한 프로세스를 정의한 후에, 그 프로세스에 따라 GNSS 컴포넌트를 추출한다.

1. 서 론

지금까지 GNSS를 기반으로 하는 여러 애플리케이션들은 절차지향 또는 객체지향 방법으로 구현되어 왔다. 그러나 이러한 기존의 방법을 이용해서 구현된 GNSS 애플리케이션은 차후에 시스템을 변경하거나 추가해야 할 경우, 그 기능들을 다시 구현해야 하므로 시간적·경제적으로 낭비를 가져온다는 단점이 있다. 이러한 점을 보완하기 위하여 현재 새로운 소프트웨어 패러다임으로서 위치를 확고히 하고 있는 컴포넌트 기반 개발(CBD) 방법론을 GNSS 도메인에 적용시킬 필요성이 제기되었다[1, 2].

소프트웨어의 적시성을 제공하며, 소프트웨어 개발 생산성에 기여하여 품질을 향상시킨다는 이점을 안고 있는 컴포넌트 기반 개발 방법론에는 RUP, CBD96, CBS/e, UNIFACE, UML Components 등이 있으며, 각 방법론들은 컴포넌트 추출기법을 포함하고 있다[3]. 이들 각자가 나름대로의 장단점을 갖고 있지만, 그 중에서도 UML Components 방법론은 시스템 컴포넌트의 정의가 분명하고, 비즈니스 컴포넌트와 시스템 컴포넌트가 명확히 분리되어 정의되어 있다는 점에서 본 연구의 GNSS 도메인에 가장 효율적인 것으로 판단하였다. 따라서 본 논문에서는 UML Components 방법론의 컴포넌트 추출 기법을 GNSS 시스템 도메인에 가장 적절하도록 테일러링한 후, 이 프로세스를 따라 GNSS 컴포넌트를 추출해 볼 것이다.

본 연구의 범위는 GNSS 컴포넌트 추출 단계까지로서 테일러링하고자 하는 프로세스를 컴포넌트 추출 단계까지로 제한하였다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 관련연구로서 여러 CBD 방법론의 컴포넌트 추출 기법들을 비교하고, 3장에서는 UML Components를 테일러링하여 컴포넌트 추출 프로세스를 정의한다. 4장에서는 3장에서 정의한 프로세스를 따라 GNSS 컴포넌트를 추출하고, 5장에서는 결론과 향후 연구과제를 제시한다.

2. 관련연구

2.1 CBD96 방법론의 컴포넌트 추출 기법

CBD96은 현재 Cool Software 사에서 자사의 COOL 시리즈 케이스 툴과 연계하기 위해 제안한 방법으로 프로세스 구현에 특별한 방법론을 제시하지는 않고 유스케이스 모델링, 비즈니스 타입 모델링, 상황 모델링, 인터페이스 모델링, 컴포넌트 명세 그리고 구조 모델링과 같은 모델링을 기준으로 작업의 단계를 구분한다[4, 5]. CBD96은 컴포넌트를 추출하기 위한 절차와 지침이 다른 방법들에 비해서 좀 더 체계적이고 구체적으로 제시되고 있는 장점이 있지만 몇 가지의 문제점이 존재한다. 첫째, 요구사항을 파악하기 위하여 시스템적 개념이 없는 현업 담당자가 타입 다이어그램을 그리게 되면, 어떠한 단어가 의미가 있는지 없는지 구별하기가 쉽지 않고 그들 사이에 관계성을 도출하기가 쉽지 않다. 둘째, 비즈니스 타입 모델은 시스템 분석가가 현업 담당자가 그린 타입 모델을 보고 시스템 측면에서 불필요한 것, 중복되는 것, 모호한 것, 의미가 없는 것을 제거하고 다시 의미가 있는 객체를 추가하여 비즈니스 타입 모델을 만든다. 그러나 비즈니스 타입 모델의 객체를 도출할 때 객체 추출의 구체적인 지침 없이 도출한다는 것은 경험이 많은 시스템 분석가의 직관에 의해서 객체를 추출해야 하는 모순이 있다. 셋째, 시스템 서비스 측면의 기능을 재사용 할 수 있는 시스템 컴포넌트를 추출하기 위한 지침과 절차가 없고 비즈니스 컴포넌트 식별 방법만을 중심으로 컴포넌트 추출이 이루어진다[6].

2.2 RUP 방법론의 컴포넌트 추출 기법

RUP는 Rational Software 사에서 Booch, Rumbough, Jacobson 방법론의 모델링 기법을 통합한 방법론이다. RUP는 요구사항 분석 단계의 산출물로 유스케이스 모델을 산출하고 분석 단계에서 각 유스케이스의 이벤트 플로우를 보고 인터페이스 객체, 컨트롤 객체, 엔티티 객체 추출을 통하여 CDB96 방법론보다는 좀더 체계적이고 효율적으로 객체를 도출해나간다. 클래스 다이어그램을 도출할 경우에도 인터랙션 다이어그램을 통해서 효율적으로 도출한다[4, 5]. 그러나 이러한 장점을 보유하고 있음에도 컴포넌트를 추출하기 위하여 널리 사용되지 못하고 있는 문제점은 컴포넌트를 추출하기 위한 방법만 제시하고 있지 제시한 방법을 사용해서 컴포넌트를 추출하기 위한 구체적인 지침과 절차를 제시하고 있지 않으므로 시스템 분석가와 직관과 경험에 의하여 컴포넌트를 추출할 수밖에 없다는 데에 있다[6].

2.3 UML Components 방법론의 컴포넌트 추출 기법

UML Components 방법론은 CBD96에서의 컴포넌트 개발 방법론을 확장하여 Cheesman과 Daniel이 제안하였다[7]. UML Components 방법론은 CBD96의 첫째, 둘째의 문제점을 가지고 있지만 시스템 컴포넌트의 정의가 분명하고, 비즈니스 컴포넌트와 시스템 컴포넌트가 명확히 분리되어 추출된다는 장점이 있다[8]. UML Components 방법론에서의 컴포넌트 추출은 시스템의 전체 도메인을 중심으로 개념 모델과 비즈니스 타입 모델을 생성하고 핵심 타입(core type)을 추출하여 그것들을 중심으로 관련된 타입을 그룹화하여 비즈니스 컴포넌트를 추출한다. 동시에 요구사항 분석을 통하여 추출된 유스케이스에 의해 시스템 컴포넌트를 추출한다. 그 이후에 인터랙션 다이어그램에 의해 추출된 비즈니스 컴포넌트를 정제하고 컴포넌트 아키텍처를 정

의하기 위하여, 추출된 시스템 컴포넌트에 포함될 비즈니스 컴포넌트를 찾아서 조립한다[7].

3. UML Components의 테일러링

본 논문에서는 위에서 제시한 기존 컴포넌트 추출 방법 중 시스템 컴포넌트와 비즈니스 컴포넌트의 정의가 명확히 분리되어 있어 시스템 서비스 측면의 기능적인 재사용이 가능하기 때문에 UML Components 방법론이 GNSS 문제 도메인에 가장 효율적이라고 판단하였다. 따라서 이 장에서는 UML Components 방법론이 포함하고 있는 컴포넌트 추출 기법을 좀더 상세하면서도 상황에 맞게 변화가 가능하고, 기본적으로 어느 정도의 구조를 갖추도록 테일러링하여 GNSS 개발 프로세스로 정의하였다.

본 논문에서 UML Componets 방법론을 테일러링한 내용은 다음과 같다. UML Componets 방법론에서는 요구사항 단계에서 비즈니스 개념 모델을 생성한 후, 컴포넌트 식별 단계에서 이것을 복사하여 비즈니스 타입 모델로 전환한 후 이를 정련하는데[7], 본 논문에서는 요구사항 분석을 통해 처음부터 비즈니스 클래스 모델을 추출하고 이를 비즈니스 범위가 맞을때까지 구성요소들을 추가 및 삭제하면서 정련하도록 테일러링하였다. 그리고 UML Componets 방법론에서는 비즈니스 타입 모델에서 핵심 타입을 추출하여 핵심 타입마다 인터페이스를 추출하지만, 본 논문에서 사용한 방법에서는 비즈니스 클래스 모델안의 각 핵심 비즈니스 클래스에 대해서 한 개의 비즈니스 인터페이스를 생성하도록 하였다. [그림 1]은 UML Components 방법론을 테일러링한 프로세스를 기술한다.

[그림 1] UML Components 방법론을 테일러링한 프로세스

Process 1 비즈니스 요구사항을 분석하고, 비즈니스 클래스 모델을 추출한다.
- **Process 1.1** 도메인 분석을 통하여 요구사항을 분석하고 요구사항 명세서를 작성한다.
- **Process 1.2** 요구사항 문서를 분석하여 액터와 유스케이스를 식별한다.
- **Process 1.3** 요구사항 문서를 분석하여 시나리오를 작성한다.
- **Process 1.4** 식별된 액터와 유스케이스를 이용하여 유스케이스 다이어그램을 생성한다.
- **Process 1.5** 추출된 유스케이스마다 각 유스케이스를 명세한다.
- **Process 1.6** 유스케이스를 계층적으로 명세한다.
- **Process 1.7** 요구사항 문서로부터 명사구 접근 방식으로 후보 클래스를 추출한다.
- **Process 1.8** 추출된 후보클래스들을 토대로 하여 비즈니스 클래스 모델링을 한다.

Process 2 Process 1에서 추출된 유스케이스를 토대로 메인 성공 유스케이스(main success usecase)를 추출한다.
- **Process 2.1** 요구사항 성공(success) 시나리오를 만족하는 유스케이스를 메인 성공 유스케이스로 결정한다.
- **Process 2.2** 요구사항 성공 시나리오를 만족하더라도 시스템의 주요 흐름상 상관성이 없는 유스케이스는 성공 유스케이스에서 제외한다.

Process 3 Process 2에서 추출된 성공 유스케이스마다 한 개의 시스템 인터페이스를 정의한다.

Process 4 Process 1에서 추출된 비즈니스 클래스 모델을 토대로 비즈니스 인터페이스를 식별한다.
- **Process 4.1** 후보 클래스들을 비즈니스의 범위에 맞을 때까지 추가 및 삭제하여 비즈니스 클래스 모델을 정제한다.
- **Process 4.2** 비즈니스 클래스 모델에서 생략된 세부사항을 보완함으로써 모델을 정제해 나간다. 즉, 각 타입별로 속성(attribute)을 정의하며, 모델에서 사용될 데이터 타입(type)을 정의하고, 연관관계의 다중성(multiplicity)을 표현함으로써 모델을 정제한다.
- **Process 4.3** 비즈니스 클래스 모델에서 핵심이 되는 클래스가 무엇인지 고려해서 핵심 클래스를 식별한다. 즉, 정보간의 의존성을 파악하여 어떠한 정보가 독립적으로 존재할 수 있는지를 고려하여 핵심 클래스를 추출한다.
- **Process 4.4** 비즈니스 클래스 모델 안의 각 핵심 클래스에 대해서 1개의 비즈니스 인터페이스를 생성한다. 비즈니스 인터페이스는 핵심클래스와 그 핵심클래스를 설명하고 있는 상세 클래스로 표현되는 정보들을 관리한다.

Process 5 Process 3에서 추출된 시스템 인터페이스에 관한 시스템 컴포넌트 명세를 생성한다.
- **Process 5.1** 시스템 인터페이스 각각에 대해 별도의 컴포넌트 명세를 생성한다.
- **Process 5.2** 시스템 인터페이스가 중첩적이고 동일한 생명주기를 가지는 비즈니스 상의 사상들을 관리한다면, 여러 개의 인터페이스를 지원하는 하나의 컴포넌트를 명세 한다.

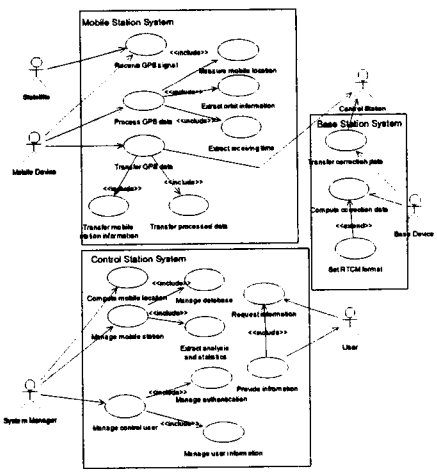
Process 6 Process 4에서 추출된 비즈니스 인터페이스 당 한 개의 비즈니스 컴포넌트 명세를 생성한다.

Process 7 Process 5와 6에서 생성된 시스템 컴포넌트와 비즈니스 컴포넌트의 명세 사이의 의존성을 대응되는 컴포넌트 명세의 인터페이스에 바인딩함으로써 최종 컴포넌트를 추출한다.

4. UML Components 테일러링 프로세스에 따른 GNSS 컴포넌트 추출

이 장에서는 3장에서 정의한 컴포넌트 추출 프로세스에 따라서 요구사항 및 비즈니스 클래스 그리고 컴포넌트를 추출한 후 컴포넌트를 명세하였다.

[그림 2]는 Process 1에서 비즈니스 요구사항을 분석하여 생성된 유스케이스 다이어그램이다.



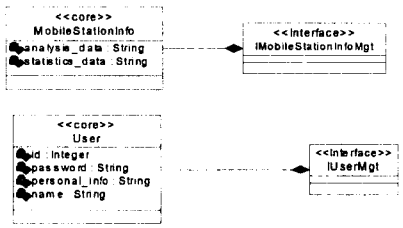
[그림 2] 유스케이스 다이어그램

Process 2와 Process 3 에서는 Process 1에서 추출된 유스케이스 토대로 메인 성공 유스케이스를 추출하고, 성공 유스케이스마다 한 개의 인터페이스를 정의한다. [표 1]은 유스케이스마다 추출된 시스템 인터페이스의 일부를 나타낸다.

Process 4에서는 Process 1에서 추출된 비즈니스 클래스 모델을 대로 비즈니스 인터페이스를 식별한 후, 비즈니스 클래스 모델 안의 각 핵심 클래스에 대해서 1개의 비즈니스 인터페이스를 생성한다. [그림 3]은 추출된 핵심 클래스와 비즈니스 인터페이스간의 관계를 나타낸다.

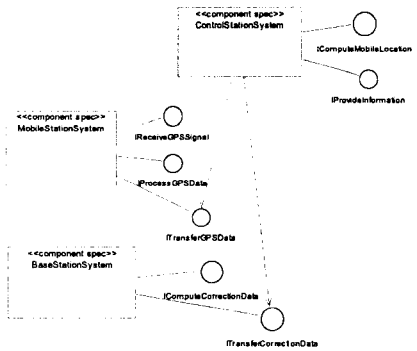
[표 1] 유스케이스와 시스템 인터페이스의 매핑

유스케이스	시스템 인터페이스
Receive GSP signal	<<Interface>> IReceiveGPSSignal
[UC 1] Receive GPS signal	Interface : IReceiveGPSSignal
Process GSP data	<<Interface>> IProcessGPSData
[UC 2] Process GPS data	Interface : IProcessGPSData
Transfer GSP data	<<Interface>> ITransferGPSData
[UC 3] Transfer GPS data	Interface : ITransferGPSData



[그림 3] 추출된 핵심 클래스와 비즈니스 인터페이스간의 관계

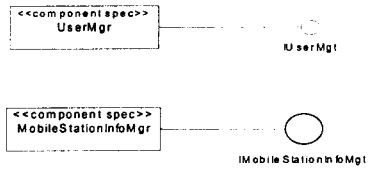
Process 5에서는 Process 3에서 추출된 시스템 인터페이스에 관 시스템 컴포넌트 명세를 생성한다. [그림 4]는 시스템 컴포넌트 명세를 나타낸다.



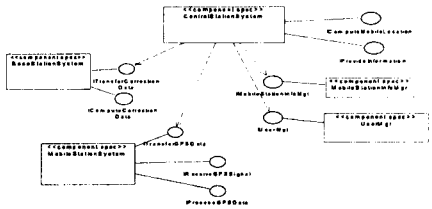
[그림 4] 시스템 컴포넌트 명세

Process 6에서는 Process 4에서 추출된 비즈니스 인터페이스 당 개의 비즈니스 컴포넌트 명세를 생성한다. [그림 5]는 비즈니스 컴포넌트 명세에 의해 제공되는 인터페이스를 나타낸다.

마지막으로 Process 7에서는 Process 5, 6에서 생성된 시스템 컴넌트와 비즈니스 컴포넌트의 명세 사이의 의존성을 대응되는 컴포넌트 명세의 인터페이스에 바인딩함으로써 최종 컴포넌트를 추출한다. [그림 6]은 시스템 컴포넌트 명세와 비즈니스 컴포넌트 명세 사이의 의존성을 고려하여 만든 컴포넌트 아키텍처를 나타낸다.



[그림 5] 비즈니스 컴포넌트 명세



[그림 6] 컴포넌트 아키텍처

5. 결론 및 향후연구

본 논문에서는 GNSS 시스템을 구축하기 위해, 컴포넌트 기반 개발(CBD) 방법을 적용한 UML Components 방법론을 테일러링하여 GNSS 컴포넌트 추출 프로세스를 정의하고 이를 바탕으로 GNSS 컴포넌트를 추출하였다.

본 논문이 갖는 의의는 비록 거대한 규모의 GNSS 기반 시스템이라 할지라도, 컴포넌트의 사용성에 따라 잘 추출된 컴포넌트는 향후 GNSS를 기반으로 하는 도메인들에서 재사용 될 수 있다는 점, 그리고 GNSS 컴포넌트를 추출하거나 개발할 때 시스템에 적합하도록 잘 정의된 프로세스를 이용한다면 GNSS 시스템 개발시간 및 비용절감에 많은 공헌을 한다는 데에 있다.

향후 GNSS 범주에 속하는 여러 도메인에 활용될 수 있는 컴포넌트를 본 논문에서 테일러링한 프로세스에 따라 추출하고, 추출된 컴포넌트와 도메인과의 상관성 및 공통성을 분석하여 그에 맞게 컴포넌트를 분류하는 것과, 본 논문에서 현재까지 추출한 컴포넌트 중 일부를 구현해봄으로써 구체적인 유용성을 검증해 보는 것이 필요하다.

참고문헌

[1] 권호열, "소프트웨어 개발 프로세스의 연구동향", Vol. 20, No. 3, 정보과학회지, pp.6-14, 2003.

[2] Paul Allen, 김경주, "CBD 프로세스가 갖춰야 할 기본 요소들", 제19권 제2호, 정보과학회지, pp.40-49, 2001.

[3] A. W. Brown and K.C. Wallnau, "The Current State of CBSE IEEE Software, Vol.16, No.5, pp.37-46, Sept./Oct. 1998.

[4] Ivar Jacobson, Grady Booch, James Rumbaugh, *The Unifie Software Development Process*, Addison-Wesley, 1999.

[5] 오영배, 나회동, 박준성, 백두권, "컴포넌트 기반 소프트웨어 개발 프로세스", 제20권 제3호, 정보과학회지, pp.15-22, 2001.

[6] 조진희, 이우진, 김민정, 신규상, "CBD 방법론의 컴포넌트 식별 방법의 비교", 정보처리학회지, 제7권 제2호, pp.515-518, 2000.

[7] John Cheesman & John Daniels, *UML Components : A sim Process for specifying component-based software*, Addison-W 2000.

[8] Yu, Young Ran, Kim, Dong Kwan, Kim, Soo Dong, "Connector Modeling Method for Component Extraction", Asa-Pacific Software Engineering Conference(APSEC99), pp.46-53, Dec. 1999