내장형 소프트웨어 개발 프로세스와 기술자료 관리

정창민* · 변재정*

Embedded Software Development Process and Systematic Technical Information Management

Chang-Min Jeong* · Jai-Jeong Pyun*

Abstract

As the functional and nonfunctional requirements of defence systems become more complex and precise, concerning about the systematic development of software embedded into the defence systems has become surged up. In order to develop more reliable and correct software, and to extend the life cycle of the developing software, adoption of some engineering technologies such as development process, methodology, CASE tools and documentation is essentially required. This paper introduce an approach to technical information management in embedded software development process, with an instance, airborne ECM system development project. Particularly, we suggest and explain how to guide the software development according to process and methodology, and how to generate technical documents using CASE tool.

Keyword : Defense Systems, Embedded Software, Documentation, Engineering Technology, CASE Tools, Methodology

1. 서 론

최근 국방 무기체계의 기능과 성능에 대한 요구사항이 복잡해지고, 무기체계의 수명을 연장하기 위한 노력이 증대하면서, 무기체계의 기능을 체계적으로 제어하고 성능 향상을 용이하게 수행할 수 있는 내장형 소프트웨어의 효과적인 개발에 더욱 더 관심을 갖게 되었다. 신뢰성 있는 내장형 소프

* 국방과학연구소 정보기술연구부

트웨어를 개발하고, 이의 수명을 연장하기 위해서는 소프트웨어 개발 과정에서 공학적인 개발 방법론의 적용과 체계적인 기술자료 관리가 필수적으로 요구된다.

과거 국방 무기체계 개발사업에 있어서 내장형 소프트웨어가 차지하는 비중은 하드웨어 부분에 비해 매우 미약했다. 한때, 전자 회로의 집적화 및 고성능 소형화 기술 개발로 인하여 무기체계 기능을 소프트웨어 대신 하드웨어로 구현하는 추세에 있었으나, 최근에는 기능의 제어 용이성 및 임무 적응성을 위해 하드웨어 대신 소프트웨어 중심의 기능 개발 추세로 변화해 가고 있다.

이러한 추세에 따라 국방부 및 국방 품질관리소에서는 최근 1~2년에 걸쳐 국방 무기체계 소프트웨어 개발관리를 위한 방안을 정립하고, 이를 훈령화(국방부 훈령 699호)하여 국방 무기체계의 내장형 소프트웨어 개발에 적용하도록 유도하고 있다[7]. 이는 그 동안 개발된 내장형 소프트웨어는 규격화 및 표준화 과정에서 누락되거나 별도 관리되지 않아서, 개발 과정에서조차 소프트웨어의 기술 관리가 이루어지지 않았던 점을 보완하기 위함이다.

본 논문에서는 국방 분야에서의 내장형 소프트웨어를 개발하면서 적용한 개발 방법과 기술 관리 방안을 소개한다. 객체지향 개념을 통한 내장형 소프트웨어를 개발할 때, 복잡도, 규모, 도메인 특성에 따라 방법론을 적절하게 재구성하고, 이를 토대로 문서화를 통한 기술 관리 방안을 연구 적용하였다.

2. 공학적 기술 관리 요소

국방분야의 내장형 소프트웨어 개발에 있어서 고려되어야 하는 공학적 기술 요소들은 다음과 같이 요약할 수 있다. 이들에 대한 체계적인 관리는 개발 소프트웨어에 대한 품질을 향상시키고, 개발 비용을 절감하는 최선의 방법이라고 판단된다.

2.1 개발 프로세스 관리

소프트웨어 개발 프로세스는 고품질의 소프트웨어를 개발하기 위한 3가지 주체, 즉 사람(actor), 작업(activity), 자료 정보(asset)를 체계적으로 정의하고, 개발 절차상에서 이들간의 합당한 연관 관계를 부여하는 것이다. 즉, 정해진 작업을 누가 언제, 어떻게 수행하여 어떤 자료를 생성해야 하는가에 대한 정의가 프로세스 정립이라 할 수 있다. 국방 분야에서의 소프트웨어 개발 프로세스는 DoD-STD-2167A, DoD-STD-7935A, MIL-STD-498, ISO 12207 등으로 변천되어 왔으며, 특히 ISO 12207과 같은 경우는 소프트웨어 자체에 대한 개발 절차뿐만 아니라 문서화, 형상관리, 품질 보증, 감리, 교육 훈련 등과 같은 종합적인 절차를 다루고 있다.

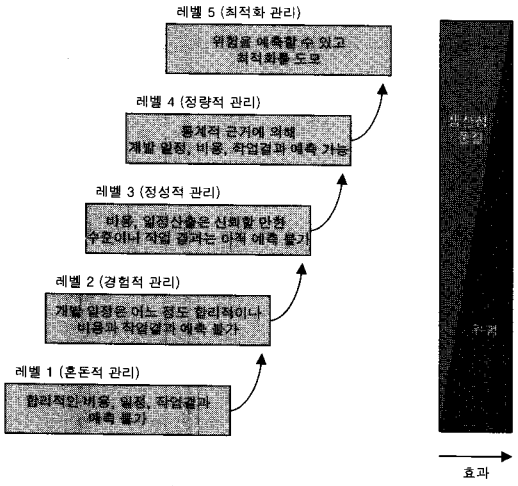
개발 프로세스는 소프트웨어 개발 과정에 대한 마일스톤을 정하고, 이를 기준으로 개발되는 소프트웨어의 품질을 관리하기 위한 수단이다. 따라서 소프트웨어의 특성 및 규모, 개발 기간 및 비용에 따라 합당한 프로세스를 정립하여 적용하는 것은 중요하다.

2.2 Software-Capability Maturity Model (SW-CMM)

SW-CMM은 1987년 카네기멜론 대학의 SEI (Software Engineering Institute)에서 Philip Crosby의 품질 성숙도 구조의 개념을 소프트웨어 프로세스에 적용하여 개발한 모델이다[5]. 처음에는 미국방성이 발주하는 소프트웨어 개발사업의 위험성을 줄이기 위해 방산업체의 소프트웨어 개발능력을 평가할 수 있도록 개발되었으나, 이후 SW-CMM이 갖고 있는 소프트웨어 프로세스 성숙도에 대한 개념이 국방 관련 사업만이 아닌 민간 소프트웨어 개발업체에서도 인정을 받으면서 현재 미국, 일본 등 여러 나라에서 소프트웨어 개발조직의 소프트웨어 프로세스 성숙도를 진단하고 개선방안을 도출하는 데 사용되고 있다.

SW-CMM은 [그림 1]과 같이 소프트웨어 개발 조직의 성숙도를 5단계로 구분하고 있는데, 그림에

서 알 수 있듯이 레벨이 높아짐에 따라 생산성과 품질이 높아지는 반면에 개발 위험은 감소한다.



[그림 1] SW-CMM의 소프트웨어 개발조직의 성숙도 모형[5]

국방 분야에서의 소프트웨어 개발 프로세스인 MIL-STD-498과 SW-CMM과의 관련성을 살펴보면 <표 1>과 같다.

〈표 1〉 MIL-STD-498과 SW-CMM과의 관련성

	범 위	초 점	비 고
SW-CMM	조 직	프로세스	
MIL-STD-498	사 업 (프로젝트)	소프트웨어	SW-CMM 기준 Level 3 권고/요구

MIL-STD-498은 소프트웨어 개발조직이 준수하여야 하는 소프트웨어 개발 및 문서화에 대한 요구사항을 명시한 표준지침(What to do)으로 적용되고 있으며, SW-CMM은 소프트웨어 개발 프로세스의 올바른 적용과 산출물에 대한 품질향상을 목표로 하고 있다. 국방 분야의 내장형 소프트웨어 개발 표준지침은 MIL-STD-498을 근간으로 하고 있으므로 표준지침을 이행하기 위해서는 소프트웨어 개발조직이 이에 부합하는 소프트웨어 프로세스를 보유하고 있어야 한다. Reed Sorensen은 MIL-STD-498과 SW-CMM과의 관련성을 언급하면서 소프트웨어 개발 조직이 MIL-STD-498의 개발표준에 부합하게 소프트웨어를 개발·관리하기 위해서는 SW-CMM 기준으로 볼 때 성숙도 수준이 Level 3은 되어야 한다고 주장한다[6].

2.3 개발 방법론

소프트웨어 개발 방법론은 개발 단계에 따라 이루어지는 요구사항의 분석기법, 소프트웨어 설계기법 등에 대한 기술적인 요소를 다룬다. 내장형 소프트웨어를 개발하기 위한 방법으로는 Real-time SA/SD, COMET, OCTOPUS 등과 같은 방법론들이 제안되어 왔는데, 이들은 내장형 소프트웨어 갖는 실시간 제약 조건이나 스케쥴링 문제 그리고 하드웨어와 소프트웨어의 인터페이스 등을 방법론 차원에서 해결하기 위한 기법을 제공하고 있다.

개발 방법론은 자연어로 주어지는 소프트웨어 요구사항을 프로그램 코드로 가져가기 위한 정형화된 기법들을 자연스럽게 연결하며, 이를 통해 요구사항의 일관성이나 설계 정보의 명확성을 유지 관리할 수 있는 메커니즘을 제공한다.

2.4 모듈화, 계층화, 객체화

복잡한 기능을 갖는 소프트웨어 요구사항은 이해하기 어려우며, 분석 및 설계가 용이하지 못한 문제점이 있다. 이러한 문제는 일반적인 공학적 원리인 모듈화, 계층화, 객체화 등을 통해 해결할 수 있다. 즉 사용자 요구사항을 추상화 수준에 따라 계층적으로 분할하고, 이들을 객체화하여 이해하기 쉬운 모듈로 정의하는 것이다.

이러한 일반적인 공학적 기술은 개발 소프트웨어에 대한 이해를 높일 뿐만 아니라, 기능의 수정 변경 및 향후 유지 보수를 용이하게 하는 핵심 기술중의 하나이다.

2.5 재사용 및 정보 저장소

내장형 소프트웨어 개발에 있어서 범용으로 사

용될 수 있는 모듈이나, 체계간 동일 또는 유사 기능에 대해서는 기 개발 소프트웨어를 재사용 할 수 있는 기회를 제공해야 한다. 재사용 가능한 소프트웨어 모듈은 변경없이 다른 소프트웨어 개발에 활용될 수 있어야 한다. 이를 위해서는 개발된 모듈이 재사용할 수 있도록 하는 기술 정보가 함께 필요하며, 또한 재사용 모듈을 찾는 개발자에게 도서관과 같은 관리 저장소를 제공해야 한다.

재사용 기술의 적용 및 관리는 개발 기간을 단축하거나, 소프트웨어의 신뢰성을 높이는 요소가 된다.

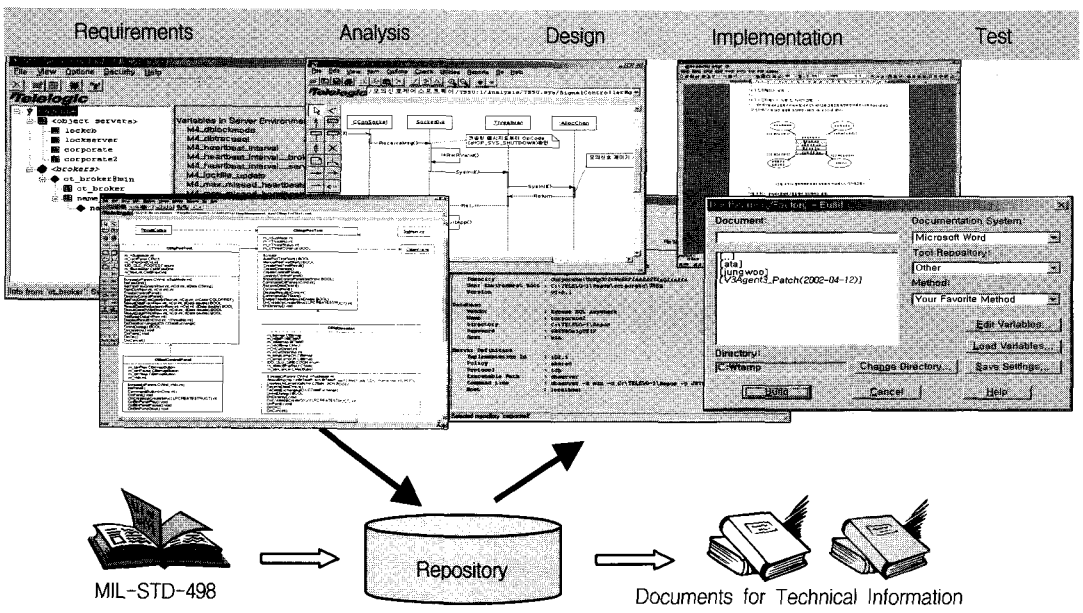
2.6 자동화 도구

사용자 요구사항으로부터 소프트웨어 코드 개발까지의 과정에서 많은 작업이 이루어지며, 또한 다양한 기술 정보들이 개발된다. 작업 수행의 용이성을 제공하고, 작업 산출물의 정확성 및 일관성을 보장받기 위해 CASE 도구와 같은 자동화된 도구의 필요성이 생긴다. [그림 2]에서 보는바와 같이 자동화 도구는 여러 개발자를 하나의 통합 개발 환경으로 묶어주기도 하고, 연계된 산출물의 부분적인 자동 생성과 일관성을 보장한다. 그러나 자동화 도구는 개발 방법론에 의존적이기 때문에 방법론에 대한 기본적인 이해를 통해 합당한 도구를 선택하는 것이 중요하다.

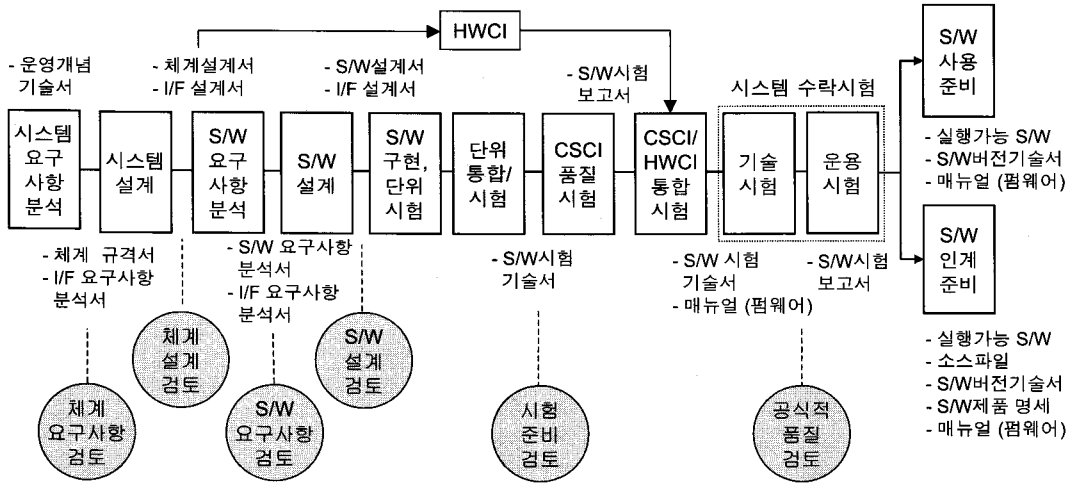
자동화 도구의 사용은 소프트웨어 개발자가 기술적인 측면에서의 전문성을 필요로 하기 때문에 인적 자원의 기술 관리가 필요하기도 하지만, 소프트웨어 자체에 대한 기술을 체계적으로 생성하고 필요한 시점에서 원하는 정보를 적절하게 제공받을 수 있다는 장점을 제공한다.

2.7 기술 자료 생성(문서화)

개발 프로세스에 따라 합당한 방법론을 선택하고, 분석 및 설계 등의 개발 활동을 지원하는 자동화 도구를 활용할 때, 소프트웨어 기술 관리 측면에서 수행하는 또 다른 작업은 기술 자료의 생성이다. 기술 자료는 개발 소프트웨어에 대한 정보를 담고 있는 문서이기 때문에 개발 과정에 따라 생명력 있는 관리가 필요하다.



[그림 2] 자동화도구 사용 예



[그림 3] 국방 소프트웨어 개발 프로세스(MIL-STD-498 기준)

소프트웨어 개발 과정에서 요구사항 변경이나 설계 변경 등을 반영하기 위해서는 기술 정보에 대한 수정이 요구된다. 즉, 일관된 정보를 유지 관리하여, 개발자가 구 버전의 기술 정보를 참조하는 일이 없도록 해야 한다.

기술 자료는 그 용도 및 작성자에 따라 매우 다양한 형태로 작성된다. 개발이 진행되면서, 기술 정보의 수준은 상세화되고, 개발자 정보와 사용자 정보 등으로 분리되기도 한다. 따라서 체계적인 기술 자료의 생성 및 관리가 소프트웨어 개발에 있어서 매우 중요하다할 수 있다.

3. 개발 프로세스와 기술자료 관리

국방 분야에서의 소프트웨어 개발 프로세스는 MIL-STD-498을 중심으로 ISO 12207로 발전하였다. 이러한 개발 프로세스들은 소프트웨어 개발 과정을 5 또는 6개의 단계로 분리하고, 각 단계별로 수행해야 하는 작업과 산출 문서를 정의하고 있다.

[그림 2]는 국방소프트웨어 개발 수명주기 단계의 근간으로 적용되고 있는 MIL-STD-498 표준의 개발 프로세스를 나타낸 것이다.

[그림 3]의 각 사각형은 개발 프로세스에서의 작업 활동을 의미하고 있으며, 이들 과정을 통해 산

〈표 2〉 S/W 규모별 산출 기술자료 구성

단 계	산출 문서명	단순	보통	복잡
체계분석 및 설계	체계 규격서	체계개발계획서 (S/W 미작성)		
	체계 설계명세서			
	운영개념기술서		◐	○
S/W 분석 및 설계	S/W 개발계획서		○	○
	S/W 요구사항명세서		○	○
	I/F 요구사항명세서		◐	○
	S/W 설계기술서	○	○	○
	I/F 설계기술서		◐	○
	DB 설계 명세서			
S/W 시험 및 평가	S/W 시험계획서	○	○	○
	S/W 시험명세서	◐	◐	○
	S/W 시험보고서	◐	○	○
	S/W 버전명세서			○
	S/W 제품명세서	○	○	○
설치 및 인계	펌웨어 설치절차서	○	○	○
	S/W 설치계획서			
	S/W 인계계획서			○
	S/W 사용자 지침서	기술교범 수록 내용		
	S/W 입출력 지침서			
	S/W센터 운용자지침서			
	컴퓨터 운용지침서			
	컴퓨터프로그래밍절차서			

주 : ○ - 완전적용, ◐ - 부분 적용.

출되는 기술 자료가 매 작업에 대하여 정의되고 있다. 그런데 앞서 언급했던 것처럼 개발 소프트웨어의 규모 및 특성에 따라 서로 다른 개발 프로세스가 적용될 수 있기 때문에 개발 문서에 대한 테일러링도 필요하게 된다.

<표 2>는 개발하고자 하는 내장형 소프트웨어의 규모에 따라 작성해야 하는 문서를 재구성하여 제시한 것이다. 정의된 15종의 문서는 현재 국방부 훈령 699호에서 정의하고 있는 소프트웨어 기술문서들이다. 소프트웨어 규모의 구분은 일반적으로 소규모(단순)의 경우 2KDSI~8KDSI, 중규모(보통)의 경우는 32KDSI, 그리고 대규모(복잡)의 경우는 128KDSI 이상으로 구분한다. 이러한 구분에 따라 각각 4종, 7종, 그리고 15종의 기술 문서를 작성, 관리하게 되는데, 그 외에 필요한 설계 정보들은 다른 문서와 통합되어 작성될 수 있다[8].

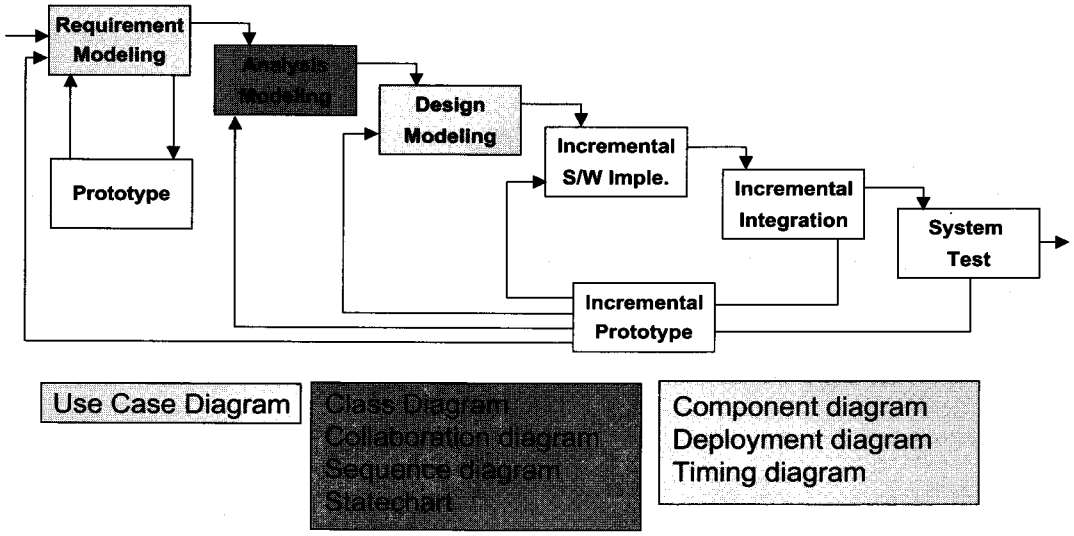
4. 개발 방법론과 기술자료 관리

내장형 소프트웨어를 개발하기 위한 개발 방법론은 매우 다양하다. 최근 객체지향 방법의 유용성이 여러 소프트웨어 개발 프로젝트에서 입증되면서, 내장형 소프트웨어의 개발에도 객체지향 개념을 도입한 개발 방법론이 등장하게 되었다. [그림 4]의 H. Gomaa가 제시한 COMET(concurrent object modeling and architecture design method) 방법론도 실시간 내장형 소프트웨어를 개발하기 위해 제안되었는데, 이 방법론은 점진적인 프로토타입 개발 방식에 의거하여 소프트웨어를 개발하도록 하고 있다.

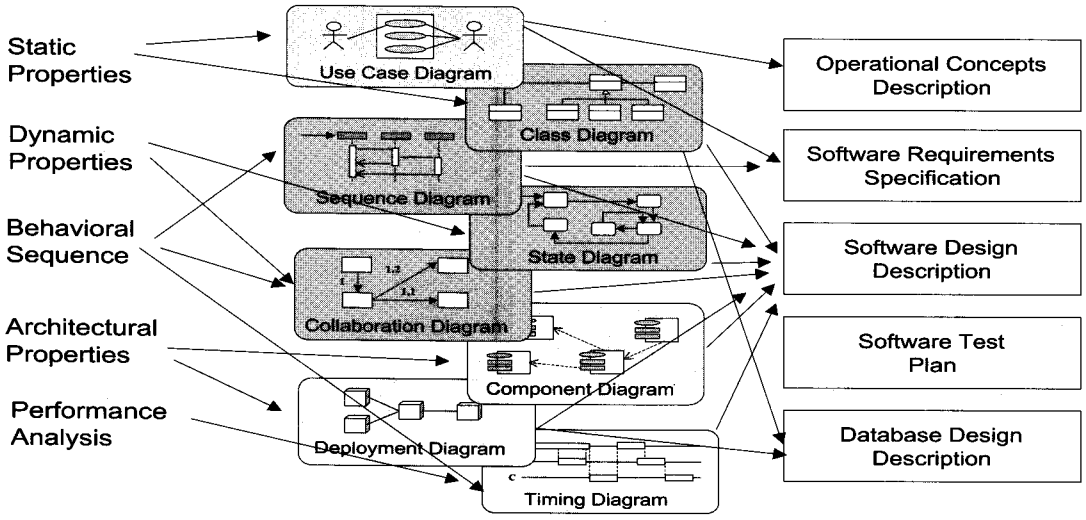
[그림 4]에서 보는 바와 같이 이 방법론에서 작성하는 다이어그램의 유형은 다음과 같다.

- Use Case diagram
- Class diagram
- Collaboration diagram
- Sequence diagram
- Statechart
- Component diagram
- Deployment diagram
- Timing diagram

이들 다이어그램들은 소프트웨어에 대한 요구사항과 설계 사항을 잘 표현할 수 있는 정형화된 기법들로써, 무형의 소프트웨어 모습을 시각적으로 표현하는 방법으로 제공한다. 방법론에서 제공하는 다양한 표현 방법들은 소프트웨어가 갖는 정적 특



[그림 4] COMET(Concurrent Object Modeling and architectural design mThod)



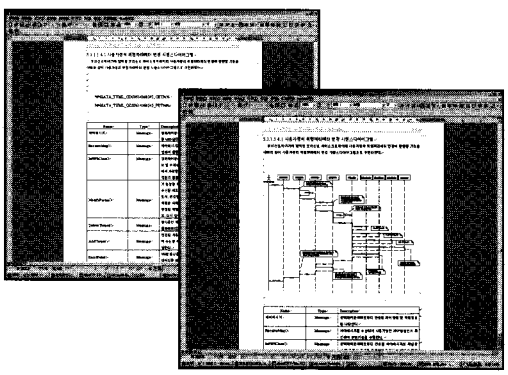
[그림 5] 기술문서와 엔지니어링 모델의 매핑 예

성(예를 들면, class diagram, component diagram), 동적 특성(예를 들면, sequence diagram, state-chart 등), 그리고 성능 특성들을 고려한 모델링을 가능하게 한다. 이들은 소프트웨어에 대한 복잡도를 단순화하고, 분할과 통합 개념 하에서 소프트웨어를 개발하기 위한 공학적 전략이라고 할 수 있다. [그림 5]에서와 같이 이러한 표현 방식은 소프트웨어 개발 과정에서 생성되는 기술문서와 연관되어 적절하게 문서화되어야 한다.

문서화는 MIL-STD-498에서 정의한 21종의 문서(또는 국방부 훈령 699호의 15종)에 대하여, 세부 내용을 어떻게 채워나갈 것인가 하는 것이다. 방법론에서 제공하는 소프트웨어 표현 기법을 적절하게 기술문서에 반영하기 위해서는 상호 매핑을 위한 템플릿(template)이 필요하다. 템플릿을 개발하기 위해서는 먼저, 문서화 지침에서 정의된 문서의 목차에 따라 어떠한 설계 정보를 어느 부분에 위치시키는가를 결정해야 한다. 이러한 결정을 위해서는 다음과 같은 요소들이 체계적으로 분석되어야 한다.

- 기술 문서에 포함시킬 다이어그램의 유형
- 선정된 기술문서별 다이어그램 할당
- S/W를 설명하기 위한 다이어그램의 연결 순서
- 복잡도 해결을 위한 계층적 공학 모델의 포함 수준

이러한 사항들은 개발자의 기술 수준이나 설계 문서의 작성 방안에 따라 융통성 있게 정의될 수 있다. 이러한 분석을 통해 문서 템플릿이 작성되는데, [그림 6]은 설계문서에 대한 템플릿의 예이다.



[그림 6] 설계문서에 대한 템플릿의 예

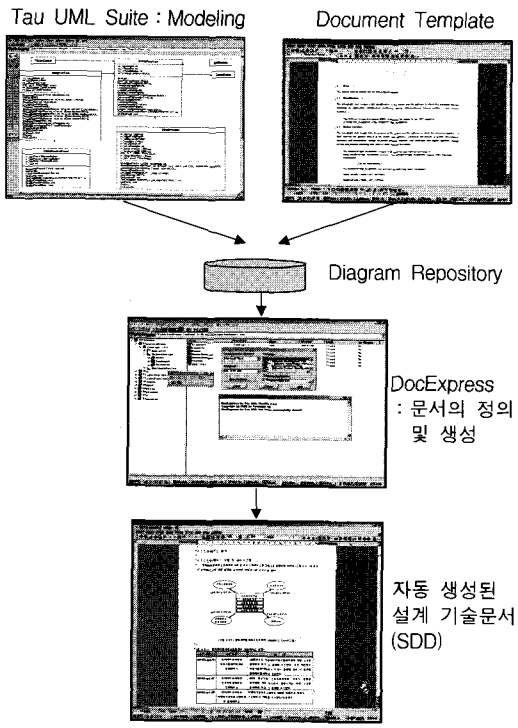
5. 자동화 도구를 통한 기술자료 관리 사례

항공용 전자전장비의 지원 소프트웨어의 개발은 국방부 훈령 699호의 소프트웨어 문서화 지침을 기

준으로 <표 2>의 중규모(보통) 수준에서 재구성하였으며, 객체지향 개발 방법론인 COMET를 적용하여 개발하였다. 내장형 소프트웨어 개발 과정에서 UML Suite를 이용하여 소프트웨어를 모델링하고, 공학적 모델들을 기술 문서로 작성하기 위해 DocExpress라는 문서화 지원 도구를 사용하였다. [그림 7]은 이러한 과정을 설명하는 한 예이다.

[그림 7]에서 보는 바와 같이, 먼저 객체지향 기반의 Use Case diagram, Class diagram, Sequence diagram 등을 작성하여 레포지토리에 저장하고, 문서 템플릿을 정의하여 저장한다. DocExpress를 이용하여 공학 모델과 문서 템플릿을 연결하여 자동으로 기술 문서를 생성하게 된다.

이러한 기술 문서의 생성 관리는 공학적 모델의 변경에 대한 기술 문서로의 반영을 자동적으로 수행할 수 있기 때문에 문서에 대한 일관성 제공은 물론, 개발 과정에서 소프트웨어 형상을 체계적으로 유지할 수 있다는 이점을 제공한다.



[그림 7] CASE 도구를 이용한 기술 자료 생성

6. 결 론

국방 분야의 내장형 소프트웨어를 개발할 때, 고려되어야 하는 기술적 관리 요소들은 개발 프로세스를 비롯하여 개발 방법론, 자동화 지원 도구, 문서화 방안 등으로써, 이들은 개발 대상 소프트웨어에 대한 품질을 향상시키고, 신뢰성을 증진시키는 매우 중요한 공학적 기술들이다. 특히, 국방 소프트웨어 개발에서 SW-CMM 기준의 레벨 3이 요구되는 시점에서는 앞서 제시된 공학적 기술의 적용 없이는 체계적인 기술관리가 불가능해진다고 볼 수 있다. 따라서, 보다 효과적이고 우수한 품질의 생명력 있는 소프트웨어 개발을 위해서는 소프트웨어 개발 프로세스와 기술자료 관리 측면에서 성숙도 향상이 요구된다.

본 논문에서는 항공용 전자전장비의 소프트웨어 개발시에 적용되었던 기술자료 관리 방안을 소개하였다. 자동화 도구를 이용한 기술 자료 생성 관리는 분석, 설계 자료의 재사용과 변경 추적의 용이성 등과 같은 이점을 제공해 준다.

참 고 문 헌

[1] David Simon, *An Embedded Software Primer*, Addison-Wesley, 1999.

[2] H. Gomaa, *Software Design Methods for Concurrent and Real-Time Systems*, Addison-Wesley, 1993.

[3] MIL-STD-498, *Software Development and Documentation*, 1994.

[4] TeleLogic, *Tau UML Suite and DocExpress User's Manual*, 2001.

[5] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber, "The Capability Maturity Model for Software," *Software Engineering Institute*, 1993.

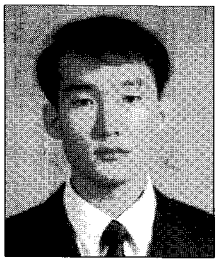
[6] Reed Sorensen, *MIL-STD-498 and the*

CMM : *How Do They Relate?*, http://www.stsc.hill.af.mil/CrossTalk/1995/nov/MILSTD.html, 1995.

[7] 국방부 훈령 699호, 「무기/비무기체계 내장형 소프트웨어 개발관리에 관한 세부지침」, 2001.

[8] 변재정, "무기체계 내장형 소프트웨어 개발관리 단계적 적용방안", 「2001 국방 소프트웨어 심포지엄」, 2001.

◈ 저 자 소 개 ◈



정 창 민 (min@add.re.kr)

경북대학교에서 컴퓨터공학 학사와 석사(1996) 학위를 취득하였다. 현재 국방과학연구소에서 선임연구원으로 근무 중이며, 항공용 전자전 장비 개발에 참여 하고 있다. 주요 관심분야로는 S/W 개발방법론, 무기체계 내장형 소프트웨어 개발 및 관리 등이다.

변 재 정 (jjpyun@add.re.kr)

충북대학교와 숭실대학교에서 통계학 이학사(1982)와 전산학 공학석사(1984) 학위를, 그리고 미국 일리노이공과대학교에서 전산학 공학박사(1996) 학위를 취득하였다. 1984년 한국국방연구원에 입사 후 각종 국방 소프트웨어 개발사업에 참여하였고, 1999년 1월부터 현재 국방과학연구소에 책임연구원으로 근무 중 이다. 주요 연구분야로는 사이버 3D 전투모의 게임엔진, 정보보호인증 방법, 무기체계 내장형 소프트웨어 개발 및 관리, 국방 CALS 체계 구축, CMM 기반 소프트웨어 프로세스 평가 및 개선 등의 분야이다.