논문 2024-2-2 http://dx.doi.org/10.29056/jsav.2024.06.02

OSMI 방식으로 개발된 소프트웨어의 형상 관리를 활용한 완성도 감정

김명주*†

Appraisal of the Completeness of OSMI-based Software by using Software Configuration Management

Myuhng-Joo Kim*†

요 약

동일 업종의 자회사 다수를 보유한 그룹사의 경우, 자회사 간의 업무 일관성과 변화 통일성을 유지하면서도 자회사별 업무 독립성을 보장하기 위하여 단일소스 다중인스턴스(OSMI) 방식을 사용하여 업무 관리 소프트웨어를 공동 개발할 수 있다. 이처럼 다수의 자회사가 동일 개발사와 일괄 계약하여 OSMI 기반의 소프트웨어 개발을 착수했음에도 불구하고 일부 자회사의 인스턴스 개발이 실패하여 법적 소송까지 진행될 경우, 소송 관련 소프트웨어에 한정하여 개발 완성도를 산정해야 하는 감정 과정이 발생한다.

이미 개발이 완료된 자회사는 업무 과정에서 소스 코드를 계속 변경하며 사용하게 되므로 OSMI의 단일소스 특성 때문에 계약 만료일 시점에서의 감정 대상 소스 코드를 특정하기 어렵다. 만일 개발사가 소프트웨어 형상 관리(SCM)를 사용하여 OSMI의 소스 코드 변경 로그를 꾸준히 유지해 왔다면, 이 이벤트 데이터를 활용하여 소프트웨어 완성도를 산정할 수 있다. 2개의 자회사가 OSMI 기반의 제조실행시스템(MES)을 함께 개발한 후 발생한 소송에서 SVM이라는 형상 관리를 활용하여 소프트웨어의 완성도를 감정한 사례를 제시한다.

Abstract

A group company with multiple subsidiaries in the same industry may develop business management software using the One Source Multiple Instances (OSMI) methodology. This approach ensures consistency and uniformity of changes across subsidiaries while maintaining each subsidiary's operational independence. Despite several subsidiaries contracting with the same developer to create OSMI-based software, some may face development failures, leading to legal disputes. In such cases, it is necessary to evaluate the completeness of the software specifically related to the lawsuit.

Completed subsidiaries continually modify and use the source code, making it challenging to pinpoint the specific code for evaluation at the contract's end due to OSMI's single-source nature. However, if the developer has maintained OSMI source code event logs using Software Configuration Management (SCM), this data can be used to assess the software's completeness. An example is provided where SCM, specifically SVM, was used to evaluate software completeness in a lawsuit following the joint development of an OSMI-based Manufacturing Execution System (MES) by two subsidiaries.

한글키워드 : 단일소스 다중인스턴스, 완성도 감정, 소프트웨어 형상 관리, 제조 실행 시스템

keywords : one source multiple instance, appraisal of completeness, SCM, MES

* 서울여자대학교 정보보호학부 교수
† 교신저자: 김명주(email: mjkim@swu.ac.kr)

접수일자: 2024.06.01. 심사완료: 2024.06.04.
게재확정: 2024.06.20.

1. 서 론

같은 업종의 기업 다수를 자회사로 두는 경우 모기업인 그룹사는 자회사 경영의 독립성을 보장하면서도 그룹 차원에서의 업무 일관성을 유지하려고 노력한다. 기업의 주요 업무를 지원하는 ERP(전사적 자원관리), MES(생산 관리 시스템), MRP(자재 소요량 계획) 등 다양한 기업 경영 시스템은 이러한 노력의 중요한 대상물이다. 동종의 다수 자회사를 둔 그룹사는 기업 경영 시스템을 도입할 때 그룹사의 이러한 특징을 반영하여 단일소스 다중인스턴스(One Source Multiple Instance, OSMI)라는 개발 방식을 도입하기도 한다.

OSMI는 하나의 콘텐츠 자원에 대하여 다양한 사용처를 발굴해 내려는 마케팅 분야의 단일소스 다중사용(One Source Multi-Use)[1, 2] 개념을 소프트웨어 개발 분야[3]에 적용한 것이다. 자회사들의 공통된 업무용으로 개발한 소스 코드는 물론이고 자기 회사만의 고유 업무에 대해 추가로 개발한 소스 코드까지 포함하여 자회사 모두는 고유의 실행 소프트웨어 즉, 인스턴스를 생성하여 업무에 활용하게 된다. 이처럼 OSMI 방식으로 개발된 기업 경영 시스템의 경우, 대다수를 차지하는 공통 업무에 변경이 발생하게 되면 '단일소스'라는 특성을 통해 전체 자회사의 인스턴스 모두에게 동시에 반영되므로 그룹사 전체의 업무 일관성을 유지하는데 편리하며 비용 측면에서도 효율적이다.

그런데 이러한 OSMI 방식으로 다수 자회사에 기업 경영 시스템을 동시에 도입하려는 노력이 항상 성공하는 것은 아니다. 동일 그룹사 안에 존재하는 자회사라고 하더라도 자회사별로 기업 문화와 환경이 다르다 보면 어떤 자회사는 도입에 성공하지만 그렇지 못한 경우도 발생한다. 이러한 실패의 경우 계약기간 내 인스턴스 생성이 불가능하여 개발사와 발주사인 자회사의 갈등이 지속되며 결국 소송까지 이르게 된다. 해당 자회사는 개발사의 소프트웨어 완성도가 낮아서 사용할 수 없다고 주장하는 반면, 개발사는 OSMI 기반으로 개발하여 이미 도입에 성공한 다른 자회사를 근거로 이를 반박한다. 결국 법원은 계약서 상의 계약종료 시점에서 해당 자회사의 소프트웨어 완성도를 감정할 필요가 생긴다. 본 논문에서는 이처럼 OSMI 방식으로 개발된 기업 경영 시스템에 대하여 계약종료 시점에서의 완성도 산정 방식을 제시한다.

2. OSMI 방식으로 개발된 소프트웨어

2.1 OSMI 방식의 소프트웨어 개발 체계

완성도 산정의 대상을 식별하기 위하여 먼저 OSMI 방식의 소프트웨어 개발 체계를 그림 1처럼 간략하게 제시해 보았다. 고려할 사항을 최소화하기 위하여 단일소스를 공유하는 자회사는 A 회사와 B 회사로 모두 2개라고 가정했다.

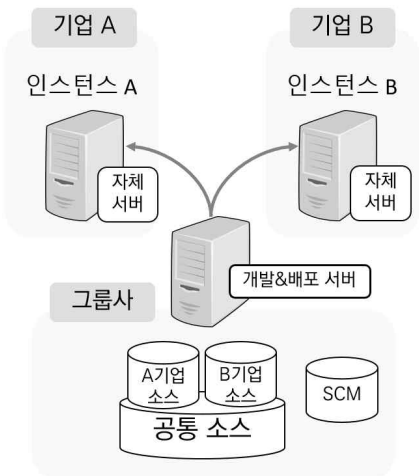

그림 1. OSMI 방식의 SW 개발 체계
Fig. 1. OSMI-based SW Development

2.2 형상 관리 도구와 완성도 산정

소프트웨어를 개발하는 수주기업은 개발 착수 때부터 소프트웨어 형상 관리(Software Configuration Management, SCM) 도구를 사용한다. 그런데 OSMI라는 개발 방식의 특성으로 인하여 수주기업은 개발이 완성된 후에도 이 SCM 도구를 발주기업에게 인계하여 단일소스의 변화를 지속하여 추적하면서 다중인스턴스를 운영하도록 한다. 이러한 과정에서 수주기업이 유지보수기업으로 역할을 바꾸어 참여하는 것이 현실이다.

다수가 참여하는 복잡한 소프트웨어 개발 프로젝트에서 소프트웨어의 버전 관리 또는 각자 만든 소스의 통합과 같은 문제를 해결하기 위해 형상 관리 즉, SCM을 사용한다. 보통은 중앙 집중 관리식 저장소를 만들어 그곳에 소스를 저장해 소스 중복이나 여러 문제를 해결한다. 이를 활용하여 과거 작업 이력을 관리할 수 있고 문제점을 파악할 수 있다.

소프트웨어 개발 현장에서 많이 사용하는 형상 관리 시스템은 주로 버전 관리에 초점을 맞춘다. 이러한 대표적인 형상 관리 도구 사례로는 로컬용 RCS[4], 클라이언트/서버용 SVN[5], 분산 관리용 GIT[6] 등을 꼽는다.

이러한 SCM은 관리 대상인 소프트웨어가 OSMI 기반으로 개발된 경우, 완성도를 산정하는 과정에서 중요한 근거자료를 제공한다. 같은 그룹사에 속해있지만, A 기업은 개발에 성공하여 현재 유지 보수 단계에 들어가 있는 반면 B 기업은 소송 과정에 들어있다고 가정해 보자. 발주기업인 B 기업과 수주기업인 개발사 C 사이에 체결한 계약을 근거로 계약종료 시점에서 B 기업용 소스 코드를 기반으로 완성도를 산정할 경우, 이미 현업에서 사용 중인 A 기업으로 인하여 계약종료 시점에서의 소스 코드를 특정하기 힘들어진다. 이 경우, 개발 초기부터 사용하여 현재 A 기업이 사용하고 있는 형상 관리 시스템의 이벤트 로그 내역을 시간대별로 분석하여 B 기업의 소스 코드를 간접 추정할 수 있다. 그리고 완성도 역시 SCM의 해당 이벤트 로그 내역을 분석하면 추정 산정이 가능해진다. 주로 공통 소스 코드에 대한 완성도 산정이 핵심이지만, 기업 B에 특화된 소스 코드에 대한 완성도 산정까지 추가로 고려하면, 기업 B의 전체 소스 코드에 대한 계약종료 시점에서의 완성도 산정이 가능하다.

3. OSMI 방식 소프트웨어의 완성도 산정

3.1 완성도 산정의 근거 및 과정

기업 B가 사용하는 소프트웨어 소스 코드에 대한 완성도 산정이지만 기업 A로부터 산정 근거를 인용할 수 있는 이유는 OSMI 개발 방식으로 두 기업의 소프트웨어가 함께 개발되었다는 점에 근거한다. A 기업과 B 기업별 특화된 소프트웨어가 비록 존재하기는 해도 궁극적으로는 공통 소프트웨어 중심의 단일소스라는 점이 핵심이다. B 기업과 달리 A 기업은 개발에 성공하여 공식 유지 보수 과정까지 통과하였으므로 A 기업의 소프트웨어 완성도는 이미 100%에 도달했다고 인정할 수 있다.

여기에 A 기업의 공식 유지 보수 과정에서 생성된 형상 관리 시스템의 이벤트 기록을 활용할 수 있으므로 이를 분석하여 각 시점에서의 유지 보수 관련 하자를 발굴하고 그 하자 수준을 판단하는 작업을 형상 관리 기록시간의 역순으로 반복함으로써 계약종료 시점에서의 B 기업의 감정 대상 완성도를 역산해 낸다. 만일 A 기업에 특화 프로그램에 대한 형상 관리 기록이 포함된 경우, 이는 감정 과정에서 제외하면 된다.

3.2 하자의 구분과 기준

이처럼 완성도를 산정할 때 중요하게 고려되는 요소는 '하자'이다. 발주자인 B 기업의 시각에서 기대했던 수준에 이르지 못한 구현 결과를 '하자'라고 정의한다. 하자는 그 수준의 심각도에 따라서 크게 '상'과 '하'로 구분할 수 있다. 수준이 '상'인 하자는 해당 프로그램의 '완성도'에 영향을 주는 반면, 수준 '하'인 하자는 해당 프로그램의 완성도에 영향을 주지 않는다.

수준 '상'인 하자는 프로그램의 완성도에 영향을 주는 하자로서, 비록 구현은 되었지만, 일부 기능이 미구현되었거나 잘못 구현되어 정상적인 시스템 오픈과 운영을 어렵게 만드는 하자이다. 수준 '상'인 하자는 심각도에 따라 등급을 다시 세분화할 수 있는데(예를 들어, 하자 등급 S, A, B) 세부 등급에 따라 완성도에 주는 영향력도 다르다. 형상 관리 이벤트 기록을 분석한 결과, 하자 등급 S로 판단된 경우, 프로그램의 완성도를 50%로 감소, 등급 A는 20%, 등급 B는 10% 감소시키는 것으로 판정했다. 이러한 감소율에 대하여 감정 표준으로 정한 기준이 아직 없어서 형상 관리 분석을 통해 통상적인 상황을 감안하여 감정인이 경험적으로 해당 비율을 정하였다.

수준 '하'인 하자는 프로그램의 완성도에 영향을 주지 않는 하자이다. 사용이 불편하거나 효율성이 떨어지는 경우 또는 수정이 필요하기는 하지만 시스템 오픈과 운영을 불가능하게 할 만큼의 수준은 아닌 등급 C인 하자이다. 즉, 오픈 직후, 유지 보수 과정이나 후속 인지 과정을 통해 발견되면 비교적 간단하게 해결할 수 있는 수준의 하자이다. 형상 관리의 이벤트 기록을 분석한 결과, 하자 등급 C로 판단된 경우에도 해당 프로그램의 완성도에는 영향을 주지 않는다.

3.3 단위 완성도

'단위 완성도'는 감정 대상인 각 단위 프로그램별 완성도를 의미한다. 단위 완성도는 0과 1 사이의 값을 갖는데 1이 만점이며 완성도가 떨어질수록 그 값이 작아진다. 형상 관리 이벤트 기록을 분석한 결과, 하자 등급 S인 프로그램의 단위 완성도는 50%를 감하여 단위 완성도를 0.5로 정한다. 하자 등급 A는 20%를 감하여 0.8, 등급 B는 10%를 감하여 0.9를 해당 프로그램의 단위 완성도로 정한다.

만일 어떤 프로그램이 형상 관리 기록에 아예 존재하지 않을 경우, 이는 '미구현' 혹은 '미완성'에 해당하고 이 프로그램은 패키지 형태의 공급이 아닌 시스템 통합(SI) 성격의 개발 프로그램에 해당한다. 이 경우 단위 완성도는 한국소프트웨어산업협회가 2022년 공표한 개발 단계별 기능점수 가중치[7]를 인용한다. 예를 들어, 어떤 프로그램이 감정 시점에서 미구현일 경우 단위 완성도는 0이지만, 분석 단계까지 마친 경우 0.19, 설계단계까지 마친 경우 0.43, 구현단계까지 마친 경우 0.75, 마지막 시험단계까지 모두 마친 경우는 단위 완성도가 1이 된다. 다음 표 1은 하자의 구분과 기준을 제시한 후 해당 단위 완성도를 어떻게 산정했는지 수치를 비교하여 보여준다.

표 1. 하자별 기준에 따른 단위 완성도 산정
Table 1. Unit completeness based on defect-specific criteria

수준	하자 등급	하자 기준 및 대표적 사례	단위 완성도
상	S	- 기능 다수는 구현되었으나 일부 기능이 구현되지 않음	0.5
	A	- 특정한 상황에서 정상적인 업무 진행에 장애를 줌 - 반복 수정에도 수준 B의 하자가 계속 발생하는 경우	0.8
	B	- 특정한 상황에서 원활한 업무 수행에 불편을 초래함	0.9
하	C	- 이용에 있어서 효율성이나 심미성을 떨어뜨림 - 이용자의 선호도에 따른 옵션 변경 수준	1.0

3.4 개발 가중치와 완성도 기여치

‘개발 가중치’는 각 단위 프로그램이 전체에서 차지하는 비중을 나타내는 값이다. 이 개발 가중치는 감정인이 수주기업인 개발사에게 감정 시작 전에 ‘감정착수 요청서’를 의뢰하여 회신받은 값을 기초로 감정인이 추가 조정하여 결정한다. 패키지 내 프로그램은 개발 가중치가 1이며, 시스템 통합(SI) 성격의 프로그램은 5 또는 10을 배정한다. 일부 미구현 프로그램의 경우, 프로그램의 하위 규모에 따라 감정인이 개발 가중치를 배정한다. 각 프로그램의 ‘개발 가중치’와 ‘단위 완성도’를 곱한 값을 ‘완성도 기여치’라고 부른다. 모든 단위 프로그램의 개발 가중치를 합하면, 감정 대상의 전체 개발 가중치가 된다.

3.5 전체 완성도 산정

감정 대상인 소프트웨어 전체에 대한 완성도를 산정하기 위하여 먼저 두 단계를 통해서 개별 완성도를 산정한 후, 이들을 통합하여 전체 완성도를 산정하는 방법을 사용한다.

첫 번째 단계에서는 개발사에 의하여 완성된 전체 소프트웨어에 대하여 형상 관리 이벤트를 참조하여 계약종료 시점에서의 하자 수준을 결정함으로써 프로그램별 단위 완성도를 산정한다.

두 번째 단계에서는 “미구현”과 “미완성”에 대한 다툼이 있는 프로그램에 대하여 제반 제출 서류를 분석하여 소프트웨어 개발단계에 따른 기능점수 가중치[7]를 토대로 프로그램별 단위 완성도를 산정한다. 만일 다툼이 있는 프로그램이 첫 번째 단계에 속한다면 이를 첫 번째 단계로 위임하여 중복 산정을 피한다.

이같이 두 단계를 거쳐 산정한 각 단위 프로그램의 ‘개발 가중치’에 ‘단위 완성도’를 곱한 ‘완성도 기여치’의 합(B)을 ‘개발 가중치’의 합(A)으로 나눈 백분율(%)이 전체 완성도가 된다. 그림 2는 OSMI 방식으로 개발된 소프트웨어에 대하여 형상 관리 이벤트를 활용하여 완성도를 산정하는 방법과 절차를 통합적으로 보여주고 있다.

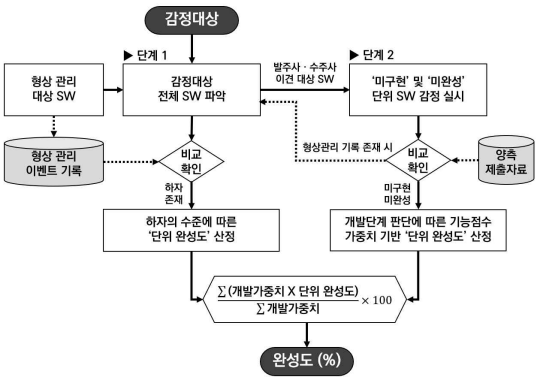

그림 2. 완성도 산정을 위한 감정 방법 및 절차
Fig. 2. Appraisal methods and procedures for calculating completeness

4. 완성도 산정 사례

4.1 감정이 이루어진 소송 사례 개요

본 완성도 산정의 감정이 이루어진 소송 사례를 먼저 소개한다. 감정 대상은 같은 제품군을 생산·제조하는 A, B 두 개의 기업이 동시 발주한 제조 실행 시스템(Manufacturing Execution System, MES)[8-10]을 OSMI 방식으로 신규 개발하는 프로젝트에서 발생했다. MES는 실시간으로 공장의 제조 공정을 모니터링하고 제어함으로써 제조 생산 공정의 효율성을 극대화하고 생산력을 증가시키는 기업 정보 시스템이다. MES의 주요 기능으로는 생산 추적 및 관리, 품질관리, 투입 자원 최적화, 주문 처리와 스케줄링 등을 꼽을 수 있다. MES는 ERP와 상호 작용하여, 전사적인 공정 및 자원관리를 위한 중요한 정보를 제공한다. 본 프로젝트를 수주한 개발사는 MES 솔루션으로 자사의 패키지 솔루션을 도입하여 이를 기초로 하여 피고의 요구 분석에 따른

커스터마이징 및 추가개발을 통한 시스템 통합(SI) 작업을 진행했다. 개발사는 형상 관리 시스템으로는 SVN을 사용하였다. 계약종료 시점에서 기업 A는 프로젝트에 성공하여 오픈한 후 공식 유지 보수 과정을 통하여 현업에서 사용하고 있는 반면, 기업 B는 프로젝트가 실패하여 법정 소송으로 진행되었다. 담당 법원은 기업 B와 관련된 소프트웨어 소스 코드에 대하여 계약종료 시점에서의 완성도 감정을 의뢰하였다.

4.2 감정 방법

본 감정 사례에서는 두 단계를 통해서 완성도를 산정한 후, 이들을 통합하여 전체 완성도를 산정하는 방법을 사용하였다. 첫 번째 감정 단계에서는 기업 B의 개발요청을 대상으로 계약종료 시점에서 총 455개 단위 프로그램의 완성도를 산정한다. 그리고 두 번째 감정 단계는, '미구현'과 '미완성' 이슈에 관하여 기업 B와 개발사 이견이 존재하는 13개 단위 프로그램에 대한 완성도를 산정한다.

첫 번째 감정 단계에서는, 455개 단위 프로그램에 대하여 기업 A가 이미 오픈하여 유지 보수하는 과정을 추적한 형상 관리 시스템 SVN의 이벤트 기록 978건을 토대로 계약종료 시점까지 시간을 거슬러 올라가면서 그동안 해결된 하자의 수준을 결정함으로써 해당 프로그램의 단위 완성도를 산정했다. 이는 감정 대상인 MES가 OSMI 방식으로 개발되었으며 단일 배포 서버를 통해 기업 A와 기업 B라는 두 개의 동종 기업에 별개의 인스턴스로 배포되는 형태를 취하고 있다는 특징에 감정 방법의 근거를 두었다.

두 번째 감정 단계에서는, '미구현'과 '미완성' 이슈가 있는 13개 프로그램에 대하여 기업 B와 개발사가 각자의 입장을 제시하여 제출한 서류 및 SVN 이벤트 기록을 근거로 하여 계약종료 시점에서의 프로그램 개발단계를 먼저 결정하였다. 그리고 개발 단계별 기능점수 가중치[7]를 토대로 해당 프로그램의 단위 완성도를 산정하였다.

이렇게 두 단계에 걸쳐 감정 대상 프로그램 각각에 대하여 단위 완성도를 산정하면, 각 프로그램의 개발 가중치와 단위 완성도를 곱한 결과를 산정할 수 있게 된다. 전체 완성도(%)는 모든 감정 대상 프로그램에 대하여 '개발 가중치와 단위 완성도를 곱한 결과의 합'을 '전체 개발 가중치의 합'으로 나눈 후 100을 곱하면 얻을 수 있다.

4.3 감정 결과

앞서 그림 2에서 제시한 감정 절차를 따라 완성도를 감정한 결과, 다음 표 2와 같이 87.24%의 완성도가 산정되었다.

표 2. 완성도 산정 결과
Table 2. results of completeness calculation

감정 단계	감정 대상	개발 가중치 (A)	완성도 기여치 (B)
1단계	MES 366개 프로그램	3,431.0	3,366.5
	EES 89개 프로그램	143.0	143.0
2단계	Global MES 미구축	403.8	0.0
	PDA프로그램 미완성	30.0	12.9
	수축율 화면 미구축	30.0	0.0
합계		4,037.8	3,522.4
전체 완성도 = $\frac{B}{A} \times 100\,(\%)$		87.24%	

1단계에서는 제조 실행 시스템(MES) 내 366개 단위 프로그램과 이를 지원하는 설비 공학 시스템(EES) 내 89개 단위 프로그램에 대하여 완성도를 산정했다. EES의 경우 완성도에 대한 다툼의 여지가 없었으나 MES는 그렇지 않아서 형상 관리 시스템 SVN(SubVersioN)의 이벤트 로

그를 참조하여 완성도를 산정하였다. 전체 455개 프로그램 중에서 하자 등급이 A인 프로그램이 26개, 등급 B가 14개, 등급 C가 122개로 판정되었다. 하자 등급 A의 단위 프로그램은 단위 완성도가 0.8점, B는 0.9점, C는 완성도에 영향을 주지 않는 1점을 단위 완성도 값으로 할당했다.받는다. 자신의 개발 가중치에 단위 완성도를 곱하면 자신의 “완성도 기여치”가 된다.

2단계에서는 완성도에 대한 다툼이 있는 13개 프로그램 중에서 8개는 앞선 1단계 감정에 포함하여 완성도를 산정하도록 이동하였고, 2개는 제출된 자료를 근거로 감정인이 판단하여 감정 대상에서 제외하였다. 나머지 3개 프로그램 중 “미구축”의 경우, 완성도 가중치로 0을 부여하였으며 “미완성”의 경우 개발 단계별 기능점수 가중치를 토대로 해당 프로그램의 단위 완성도를 산정하였다.

5. 결 론

OSMI 방식으로 개발하여 운영하는 기업용 소프트웨어는 전체 소스 코드를 중앙 그룹사에서 단일형태로 유지하면서도 기업별 상황에 맞는 인스턴스를 사용할 수 있다는 측면에서 경영의 일관성 유지 및 비용 효율성 측면에서 유리하다. 그러나 공동 개발에 참여한 일부 기업에서 소송이 발생할 경우, OSMI 방식은 감정 대상을 특정하는데 어려움을 겪는다.

본 논문에서는 OSMI 방식에서 형상 관리 도구를 사용한다는 점을 착안하여, 이미 성공적으로 도입한 기업을 중심으로 형상 관리 이벤트를 역추적함으로써 계약종료 시점의 완성도를 산정하는 기법을 제시하고 이를 적용한 감정 사례를 소개하였다.

감정 자체가 감정인의 경험과 지식을 토대로 한 주관적 판단을 상당 부분 인정하고는 있지만 하자의 수준에 따른 단위 완성도 산정은 소송 당사자에게 있어서 여전히 정성적이고 주관적으로 다가와서 감정 결과에 대한 수긍에 어려움을 준다. 따라서 이 주제에 대한 심층적인 추가 연구를 통해 어느 정도 구체적인 감정 가이드라인이 제공되면 향후 유사한 감정이 발생할 경우, 지금다 높은 객관성을 보여줄 것이라고 기대한다.

이 논문은 서울여자대학교 학술연구비의 지원에 의한 것임(2024-0038).

참 고 문 헌

[1] Cho, Seong-Ryong, et al. “A Case Study of Broadcasting Contents Using One Source Multi Use Strategy”, Journal of Broadcast Engineering, vol. 12, no. 5, The Korean Institute of Broadcast and Media Engineers, pp. 423 - 434, 2007, https://doi.org/10.5909/jbe.2007.12.5.423

[2] Kim, Young-Jae. “A Conceptual Framework for One Source Multi Use Strategy of Culture Content”, Cartoon and Animation Studies, vol. 28, Korean Society of Cartoon Animation Studies, pp. 155 - 180, 2012, https://doi.org/10.7230/koscas.2012.28.155

[3] Y. -H. Chang, K. -B. Yoon and D. -W. Park, “A Study on the Development of One Source Multi Use Cross-Platform Based on Zero Coding”, 2013 International Conference on Information Science and Applications (ICISA), Pattaya, Thailand, pp. 1-3, 2013, https://doi.org/10.1109/ICISA.2013.6579501

[4] Walter F. Tichy, “Design, implementation, and evaluation of a Revision Control System”, In Proceedings of the 6th

International Conference on Software Engineering (Tokyo, Japan) (ICSE '82). IEEE Computer Society Press, Washington, DC, USA, pp. 58 - 67, 1982.
[5] C. Michael Pilato, Ben Collins-Sussman, Brian W. Fitzpatrick, "Version Control with Subversion", 2nd Edition, September 2008, O'Reilly Media, Inc., ISBN: 9781449379353
[6] Mariot Tsitoara, "Beginning Git and GitHub - Version Control, Project Management and Teamwork for the New Developer", Apress, 2024, ISBN: 9798868802157
[7] Korea Software Industry Association, "The Software Cost Estimation in Software Projects", Korea Software Industry Association, 2022, https://www.sw.or.kr/site/sw/ex/board/View .do?cbIdx=276&bcIdx=53628
[8] SAP, "Manufacturing execution system overview", June 2024, https://www.sap.com/products/scm/execution-mes/what-is-mes.html
[9] IBM, "What is a manufacturing execution system (MES)?", June 2024, https://www.ibm.com/topics/mes-system,
[10] Saenz de Ugarte, B., Artiba, A., & Pellerin, R., "Manufacturing execution system - a literature review.", Production Planning & Control, 20(6), pp. 525 - 539, 2009, https://doi.org/10.1080/09537280902938613

저 자 소 개

김명주(Myuhng-Joo Kim)

1986.2 서울대학교 컴퓨터공학과 졸업
1988.2 서울대학교 컴퓨터공학과 석사
1993.8 서울대학교 컴퓨터공학과 박사
1993.9-1995.8 서울대학교 컴퓨터신기술공동연구소 특별연구원
1995.9-현재 : 서울여자대학교 교수
<주관심 분야> 정보보호, 디지털윤리