기존에 제안된 휘처 관계 타입 분석 및 비교 연구

이혜선*, 강교철**
*포항공과대학교 컴퓨터공학과
**포항공과대학교 IT 융합공학과
e-mail : compial@postech.ac.kr

Feature Relationships in Software Product Lines: Survey

Hyesun Lee*, Kyo Chul Kang**
*Dept. of Computer Science, Pohang University of Science and Technology (POSTECH)
**Division of ITCE, POSTECH

요 약

소프트웨어 제품라인의 핵심 자산을 개발할 때 제품라인에 포함된 휘처 사이의 관계를 고려하여 핵심 자산을 개발하지 않으면, 한 휘처의 변화가 핵심 자산의 많은 부분에 영향을 미칠 수가 있고, 미처 고려하지 못한 휘처 관계에 의해 제품이 동작중에 의도치 않은 문제가 발생할 수 있다. 휘처 사이에 존재할 수 휘처 관계 타입을 미리 알고 있다면, 실제 휘처 관계를 분석할 때 도움이 되고, 각 타입별로 어떻게 핵심 자산을 개발해야 하는지 가이드라인을 제시할 수 있다.

기존에 많은 연구자들이 휘처 관계 타입을 제시하였지만 아직까지 제안된 휘처 관계 타입들을 비교하고 분석하는 연구가 이루어지지 않았다. 이에 본 논문에서는 지금까지 제안된 휘처 관계 타입들을 살펴보고, 제안된 관계 타입들을 비교 및 논의하고자 한다. 또한 기존에 제안된 휘처 관계 타입 중 휘처 사이의 관계로 보기 어렵거나 중복이 되는 타입을 제외하여 휘처 관계 타입 리스트를 정리 및 제안하였다.

1. 서론

소프트웨어 제품라인 공학([1])은 소프트웨어 재사용 방법 중 하나로 최근 많은 학계 및 산업계에서 주목받는 방법이다. 제품을 개별적으로 개발하고 유지보수하는 기존 소프트웨어 공학 방법과 달리, 소프트웨어 제품라인 공학은 어떤 도메인에서 공통된 기능을 공유하는 유사한 제품군을 고려하여 핵심 자산을 개발하고, 핵심 자산으로부터 개별 제품을 개발하는 개념을 제시한다. 핵심 자산으로부터 계획된 제품들을 개발할 수 있기 때문에 제품의 생산성을 높일 수 있고, 제품마다 자산을 따로 개발하고 관리하는 것이 아니라 핵심 자산을 개발하여 이를 관리하면 되기 때문에 소프트웨어의 품질을 높일 수 있다.

핵심 자산에는 제품들 사이의 차이가 나는 부분이 가변점(Variation Point) 및 가변부(Variant)로 구현이 되어 있다. 가변부에 해당하는 부분은 자산으로부터 개발할 제품이 무엇이냐에 따라서 가변점에 바인딩 되거나 되지 않을 수 있다. 구현된 가변점 사이에는 의존성이 있을 수 있는데, 자산의 유지보수를 쉽게 하기 위해서는 가변점 사이의 의존성을 구현으로부터 분리하여 모델하고 관리하여야 한다. 그렇지 않을 경우 구현된 코드에서 가변점 사이의 의존성을 파악하기가 어려워서 자산을 유지보수하기가 어려워진다.

휘처 모델([2])은 가변점 사이의 의존성을 표현하는 모델 중 가장 널리 사용되는 모델이다. 휘처 모델은 제품라인에 포함된 제품들의 공통점 및 차이점을 휘처 단위로 파악하며, 휘처 사이의 계층적인 구조 관계를 부모-자식 관계로 이해하기 쉽게 표현한다. 또한, 제품에 따라 포함되거나 포함되지 않을 수 있는 선택(Optional) 및 택일(Alternative) 휘처 사이에 설정 의존 관계(Configuration Dependency)가 있을 수 있는데 이를 조합 규칙(Composition Rule)으로 표현한다. 예를 들어, 어떤 휘처가 선택될 때 다른 휘처가 반드시 선택되어야 하는 관계를 필요(Require) 관계로 정의할 수 있고, 어떤 휘처가 선택될 경우 다른 휘처는 선택될 수 없는 관계를 배제(Exclude) 관계로 정의할 수 있다.

휘처 사이에는 앞서 설명한 구조적 관계 및 설정 의존 관계 외에도 다양한 타입의 관계가 존재할 수 있다. 예를 들어 엘리베이터 제어 시스템 제품라인에서 자동 문닫힘 휘처는 승객 서비스 휘처가 활성화(Active) 됐을 때만 실행이 될 수 있는 휘처로서, VIP 서비스 또는 유지보수 서비스가 활성화 됐을 경우에는 실행이 될 수 없다. 자동 문닫힘과 다양한 서비스(승객 서비스, VIP 서비스, 유지보수 서비스 등) 휘처 사이에 존재하는 이 관계는 구조적 관계 및 설정 의존 관계와는 다른 타입의 관계이다[1].

핵심 자산을 개발할 때 이와 같은 휘처 사이의 관계를 잘 분석 및 고려하여 핵심 자산을 개발하지 않으면, 한 휘처의 변화가 핵심 자산의 많은 부분에 영향을 미칠 수가 있고, 미처 고려하지 못한 휘처 관계

[1] 이 관계를 Lee et al.([3])은 동시 활성화 관계(Concurrent-Activation Dependency)로 정의하였다.

에 의해 제품이 동작할 때 의도치 않은 문제가 발생할 수 있다. 만약 휘처 사이에 어떤 관계들이 존재할 수 있는지 휘처 관계 타입을 미리 알고 있다면, 실제 휘처 관계를 분석할 때 도움이 될 것이고, 각 타입별로 어떻게 핵심 자산을 개발해야 하는지 가이드라인을 제시할 수 있을 것이다.

기존에 많은 연구자들([3-10])이 여러휘처 관계 타입을 제시하였지만 아직까지 제안된 휘처 관계 타입들을 비교하고 분석하는 연구가 이루어지지 않았다. 이에 본 논문에서는 지금까지 제안된 휘처 관계 타입들을 살펴보고, 제안된 관계 타입들을 비교 및 논의하고자 한다.

본 논문에서 대상으로 하고 있는 휘처 관계 타입은 휘처 모델이 포함하고 있는 구조 관계 및 설정 의존 관계 이외의 휘처 관계 타입이다. 또한, 소프트웨어 기능 휘처 사이의 관계 타입만을 대상으로 하며, 비기능 휘처 사이의 관계 타입 및 기능 휘처와 비기능 휘처 사이의 관계 타입은 연구 범위에서 벗어난다.

2. 기존에 제안된 휘처 관계 타입 비교 분석

기존 연구([3-10])에서 제안한 휘처 관계 타입을 정리하면 <표 1>과 같다. <표 1>에서 각 열은 각 기존 연구에서 제안한 휘처 관계 타입을 의미하고, 같은 행에 있는 휘처 관계 타입은 비슷한 의미를 나타내는 휘처 관계 타입을 의미한다. <표 1>의 첫 행부터 시작하여 행 별로 휘처 관계 타입을 비교하면서 제안된 휘처 관계 타입의 타당성을 논의하겠다.

1 행: 의도 관계 – 휘처들은 제공하는 기능의 많은 부분이 다른 휘처와의 상호작용에 바탕을 두고 있는데, Ferber et al.은 이러한 상호작용 관계를 의도 관계라고 정의하였다. 하지만 의도 관계는 의미가 모호하여 다른 연구에서 제안된 휘처 관계의 상당 수를 포함하고 있기 때문에 의도 관계와 비슷한 휘처 관계를 찾기가 어려웠다. 대부분의 휘처 관계가 상호작용에 바탕을 두고 있기 때문에 모든 휘처 관계 타입이 의도 관계에 속한다고 볼 수 있었으며, 때문에 휘처 관계 타입으로 보기에는 적합하지 않다고 판단 되었다.

2 행: 사용 관계 (1) – Lee 와 Kang 및 Daizhong 과 Shanhui 가 제시한 사용 관계는 Parnas([11])가 제시했던 사용 관계(Uses Relationship)과 동일한 의미를 갖는 관계로, 기능의 정확성 관계에 초점을 맞춘 관계이다. 하지만 사용 관계라는 이름 때문에 의미에 혼란을 가

<표 1> 기존에 제안된 휘처 관계 타입 비교 표

	Ferber et al. ([4]) 2002	Lee와 Kang ([3]) 2004	Ye와 Liu ([5]) 2005	Lee et al. ([6]) 2006	Zhang et al. ([7]) 2006	Daizhong과 Shanhui ([8]) 2009	Liu et al. ([9],[10]) 2005, 2011
1	의도 관계 (Intentional Dependency)						
2		사용 관계 (Usage Dependency)				사용 관계 (Usage Dependency)	
3	자원 사용 관계 (Resource Usage Dependency)				흐름 관계(Flow Interaction)		
4	사용 관계 (Usage Dependency)						
5					정보 알림 관계 (Inform Interaction)		
6		변경 관계 (Modification Dependency)		변화 관계 (Change Dependency) – 상태 변화 관계 (State Change Dependency)		변경 관계 (Modification Dependency)	
7				변화 관계 (Change Dependency) – 행위 변화 관계 (Behavior Change Dependency)			
8				변화 관계 (Change Dependency) – 데이터 변화 관계 (Data Change Dependency)			
9				변화 관계 (Change Dependency) – 코드 변화 관계 (Code Change Dependency)			코드 변화 관계 (Code Change Dependency)
10		활성화 관계(Activation Dependency) – 배제 활성화 관계 (Exclusive-Activation Dependency)				활성화 관계(Activation Dependency)	
11		활성화 관계(Activation Dependency) – 종속 활성화 관계 (Subordinate-Activation Dependency)					
12		활성화 관계(Activation Dependency) – 동시 활성화 관계 (Concurrent-Activation Dependency)		담보 관계 (Collateral Dependency)			
13		활성화 관계(Activation Dependency) – 순차 활성화 관계 (Sequential-Activation Dependency)		순차 관계 (Serial Dependency)			
14		활성화 관계(Activation Dependency) – -		시너지 관계 (Synergetic Dependency)			
15			영향 관계 (Impact Dependency)		영향 관계(Influence Dependency)	영향 관계 (Impact Dependency) – 책임 영향 관계 (Responsibility Impact Dependency)	
16	환경 감응 관계 (Environment Induced Dependency)				자원 설정 관계 (Resource-Configure Interaction)	영향 관계 (Impact Dependency) – 자원 영향 관계 (Resource Impact Dependency)	
17					메타 레벨 설정 관계 (Meta-Level-Configure Interaction)		

져올 수 있다. 예를 들어, 두 논문 모두 사용 관계의 예로 다른 휘처로부터 받는 데이터가 정확해야 한다는 데이터 정확성 관계를 예로 들었는데, 데이터를 사용하는 관계와, 데이터가 정확해야 한다는 관계는 서로 다른 의미를 갖는다. 때문에 사용 관계라는 이름 대신 정확성 의존 관계라는 이름으로 변경하는 것이 의미를 파악하기 더 쉬울 것이라고 판단된다.

3-4 행: 사용 관계 (2) – Ferber et al.가 정의한 사용 관계는 앞서 Lee 와 Kang 과, Daizhong 과 Shanhui 가 소개한 기능의 정확성을 나타내는 사용 관계와는 다른 의미를 갖는다. Ferber et al.이 정의한 "사용한다" 의미는 모호하여 다른 휘처의 데이터를 사용하는 관계와 다른 휘처의 제어를 사용하는 관계(예를 들어 모터 제어 휘처를 사용하여 모터를 움직임)를 모두 포함할 수 있다. 하지만 Ferber et al.의 논문에서 사용 관계를 설명하는 예는 데이터 사용 관계만을 나타내었다. 데이터 사용 관계는 Zhang et al.이 정의한 흐름 관계와 의미가 같다. Ferber et al.이 정의한 자원 사용 관계 또한 자원의 데이터를 사용하는 관계인지 자원의 제어를 사용하는 관계인지 명확하지 않지만, 자원의 데이터를 사용하는 관계일 경우에는 데이터 사용 관계와 중복이 된다. 이처럼 사용 관계는 데이터 사용 관계와 제어 사용 관계를 포함할 수 있는데, 각 관계가 핵심 자산 설계 시 다른 의미를 가질 수 있기 때문에, 사용 관계를 데이터 사용 관계와 제어 사용 관계로 세분화하여 명확히 하는 것이 좋을 것이다. (제어 사용 관계는 15 행을 설명 시 다시 언급된다.)

5 행: 정보 알림 관계 – Zhang et al.이 제안한 정보 알림 관계(한 휘처가 어떤 조건을 만족하였음을 알리기 위해 다른 휘처에게 정보를 전달하는 관계)에는 대응되는 관계를 찾지 못하였다. 정보 알림 관계는 휘처 사이에 존재하는 데이터 사용 관계를 어떻게 구현하느냐에 따라서 존재할 수도 있고 존재하지 않을 수도 있는 관계이다. 즉, 요구사항을 나타내는 휘처 사이의 관계가 아니라, 제품라인 핵심 자산의 구현에 따른 관련있는 관계이기 때문에 휘처 사이의 관계로는 적합하지 않다고 판단되었다.

6-7 행: 변경/변화 관계 – Lee 와 Kang 이 제시한 변경 관계와, Daizhong 과 Shanhui 이 제시한 변경 관계, Lee et al.이 제시한 상태 변화 관계 및 행위 변화 관계는 의미가 같다. 우선 Lee 와 Kang, 그리고 Daizhong 과 Shanhui 가 제시한 변경 관계의 의미는 행위 변경 관계지만 변경 관계라는 이름이 행위 변경 외에도 데이터 변경 등을 포함하는 것처럼 이해될 수 있다. 따라서 변경 관계라는 이름을 행위 변경 관계로 수정하는 것이 의미를 더 명확하게 할 것이다.

Lee et al.이 제시한 상태 변화 관계 및 행위 변화 관계를 먼저 살펴보면, 상태가 변한다는 것과 행위가 변한다는 것의 뜻이 모호한 것을 알 수 있다. 왜냐하면 휘처의 상태가 변함으로 인해 해당 휘처의 행위가 변할 수 있기 때문이다. 때문에 둘을 서로 다른 관계 타입으로 분류한다는 게 적합하지 않다고 판단된다.

앞서 설명한 대로 Lee 와 Kang, Daizhong 과 Shanhui 가 제시한 변경 관계와 Lee et al.이 제시한 상태 변화 관계 및 행위 변화 관계를 "행위 변경(Behavior Modification) 관계"로 통일하여 생각하였을 때, 휘처의 행위를 "변경"한다는 의미가 맞는 것인지 생각해 볼 필요가 있다. 어떤 휘처의 행위는 다른 휘처가 활성화 될 때 달라져야 할 행위까지를 포함하고 있을 것이기 때문에 "변경"이라는 의미가 옳지 않을 수 있다. 하지만 어떤 휘처가 활성화 되느냐 여부에 따라서 다른 휘처의 행위가 달라진다면 이 관계를 요구사항 분석 시에 명시해야 향후 제품라인 자산 개발 시에 이를 고려할 수 있다. 따라서 행위 변경 관계는 휘처 타입으로 고려되어야 한다고 판단된다.

8 행: 데이터 변화 관계 – Lee et al.이 제안한 데이터 변화 관계는 다른 연구에서는 제안되지 않은 관계이다. 휘처가 갖고 있는 데이터를 정의할 경우 휘처 사이에 존재할 수 있는 관계 타입이라고 판단된다.

9 행: 코드 변화 관계 – Lee et al.이 제안한 코드 변화 관계와 Liu et al.이 제안한 코드 변화 관계는 같은 의미를 갖는다. 코드 변화 관계는 요구사항을 나타내는 휘처 사이의 관계라기보다는 휘처를 구현한 자산 코드 사이의 관계이다. 때문에 휘처가 동일하더라도 핵심 자산을 어떻게 구현하느냐에 따라서 존재하거나 존재하지 않을 수 있는 관계이다. 따라서 휘처 사이의 관계로 파악하기에는 적합하지 않다고 판단된다.

10-14 행: 활성화 관계 – Daizhong 과 Shanhui 는 활성화 관계를 제시하였지만 활성화 관계의 세부 관계 타입을 제시하지는 않았다. 반면, Lee 와 Kang 과 Lee et al.은 다양한 타입의 활성화 관계를 제시하였는데, 이들 사이에는 일부 중복되는 부분이 있다. Lee 와 Kang 이 제시한 동시 활성화 관계는 Lee et al.이 제시한 담보 관계와 의미가 같다. Lee 와 Kang 이 제시한 순차 활성화 관계는 Lee et al.이 제시한 순차 관계와 의미가 비슷하지만 동일하지는 않다. Lee 와 Kang 이 제시한 순차 활성화 관계는 한 휘처의 동작이 끝난 후 다른 휘처가 동작해야 한다고만 정의했고, 바로 동작해야 하는지, 일정 시간 있다가 동작해도 되는지, 중간에 다른 휘처가 동작한 후에 동작해도 되는지 등은 명시하지는 않았다. 반면, Lee et al.이 제시한 순차 관계는 한 휘처의 동작이 끝난 후 다른 휘처가 즉시 동작해야 한다고 명시하고 있다. 한 휘처의 동작이 끝난 후 다른 휘처가 동작할 때, 언제 동작해야 하는지는 경우에 따라서 달라질 수 있을 것이다. 따라서 Lee 와 Kang 이 제시한 순차 활성화 관계와 Lee et al.이 제시한 순차 관계를 합쳐서 순차 활성화 관계로 정의하고, 이 관계를 분석할 때는 다음 휘처가 동작하는 시간도 함께 분석하도록 하면 될 것이다.

Lee 와 Kang 이 제시한 동시 활성화 관계와 Lee et al.이 제시한 담보 관계는 두 휘처가 동시에 동작해야 한다고만 정의하고 있지만, Lee et al.이 제시한 시너지 관계는 두 휘처가 동시에 동작하는데 동기화 되어야 한다고 정의하고 있다. 하지만 시너지 관계는 동시 활성화 관계 또는 담보 관계를 어떻게 구현하는 지에 따라서 달라질 수 있다. 때문에 시너지 관계는 휘처 관계로 파악하기에는 적합하지 않다고 판단이 되었다.

지금까지 논의한 내용을 정리하여, Lee 와 Kang 이

제시한 활성화 관계와 Lee et al.이 제시한 활성화 관계의 중복된 관계 및 휘처 관계 타입으로 적합하지 않은 관계를 제외하면 다음의 네 가지 단계를 도출할 수 있다: 배제 활성화 관계, 종속 활성화 관계, 동시 활성화 관계, 순차 활성화 관계.

15 행: 영향 관계 – Ye 와 Liu 가 정의한 영향 관계는 Zhang et al.이 정의한 영향 관계 및 Daizhong 과 Shanhui 가 정의한 책임 영향 관계와 같은 행으로 분류하였지만, Ye 와 Liu 의 정의가 모호하기 때문에 이외의 다른 관계 타입과 겹쳐질 수 있다. Ye 와 Liu 의 영향 관계는 의미가 모호하기 때문에 휘처 관계 타입으로 파악하기에는 적합하지 않다고 판단하였다.

Zhang et al.이 정의한 영향 관계는 이름만 봤을 때는 의미가 모호하지만 정의된 내용은 Daizhong 과 Shanhui 가 정의한 책임 영향 관계와 의미가 같다. 하지만 일반적으로 자산 디자인/개발 시에 컴포넌트/객체가 책임이 있다고는 이야기하지만, 사용자에게 제공되는 휘처가 책임이 있다고는 이야기 하지 않는다. 휘처는 제공해야 할 기능/서비스를 갖고 있고, 다른 휘처는 해당 휘처의 기능/서비스를 사용한다고 보는 것이 더 정확할 것이다. 하지만 기능/서비스 사용 관계라고 할 경우, 앞서 설명한 데이터 사용 관계와 구분이 명확하지 않다. 따라서, 기능/서비스 사용 관계를 데이터 사용 관계와 제어 사용 관계로 세분화하여 구분하는 것이 좋을 것으로 판단된다.

16 행: 자원을 통한 간접적인 관계 – Ferber et al.이 제안한 환경 감응 관계, Zhang et al.이 제안한 자워 설정 관계, Daizhong 과 Shanhui 가 제안한 자원 영향 관계는 모두 어떤 휘처가 자원을 변경함으로 인해 같은 자원을 사용하는 다른 휘처가 간접적으로 영향을 받는 관계를 의미한다. 하지만 이처럼 간접적인 관계를 모두 고려할 경우 고려해야 할 관계의 수가 많아지고 휘처 관계 분석이 복잡해진다. 따라서 자원을 통한 간접적인 관계를 고려하는 것보다는 자원을 나타내거나 관리하는 휘처와 그것을 이용하는 휘처 사이의 직접적인 관계를 고려하고, 간접적인 관계를 알아야 할 경우 이 직접적인 관계를 분석하여 도출하는 것이 더 적합할 것이다.

17 행: 메타 레벨 설정 관계 – Zhang et al.이 제안한 메타 레벨 설정 관계는 기존 연구에서 제안된 휘처 관계 중 이에 대응되는 관계를 찾지 못했다. 이 관계는 휘처 사이의 바인딩 시간을 고려하여야 분석될 수 있는 관계이다. 하지만 휘처의 바인딩 타임은 고정된 것이 아니라 마케팅 관점에 따라서 변할 수 있는 요소이다. 따라서 바인딩 타임을 휘처와 강하게 결합하여 휘처 관계를 찾는 것은 적합하지 않다고 판단된다.

3. 결론 및 향후 연구

본 논문에서는 지금까지 제안된 휘처 관계 타입을 소개하고, 제안된 관계 타입을 비교 분석한 뒤에 타당성을 논의하였다. 중복된 관계 타입 및 휘처 관계 타입으로 타당하지 않은 관계 타입을 제외한 휘처 관계 타입 리스트를 정리하면 다음과 같다.

- **데이터 사용(Data Usage):** 어떤 휘처가 다른 휘처로부터 데이터를 받아서 동작에 사용하는 관계.
- **제어 사용(Control Usage):** 어떤 휘처가 다른 휘처의 제어 기능/서비스를 사용하는 관계
- **데이터 변경(Data Modification):** 어떤 휘처의 동작이 다른 휘처의 데이터를 변경하는 관계.
- **행위 변경(Behavior Modification):** 어떤 휘처가 활성화 될 때 다른 휘처의 행위가 바뀌는 관계.
- **활성화(Activation):** 어떤 휘처의 활성화 상태가 다른 휘처의 활성화 상태에 영향을 미치는 관계. 배제 활성화, 종속 활성화, 동시 활성화 및 순차 활성화로 세분화될 수 있다.

향후에는 본 논문에서 정리한 휘처 관계 타입 리스트를 바탕으로 다양한 소프트웨어 제품라인의 휘처 관계를 분석하여, 휘처 관계 타입 리스트가 충분한지 검증하고 필요할 경우 리스트를 보완할 계획이다. 또한 비기능 휘처 사이의 관계 및 비기능 휘처와 기능 휘처 사이의 관계 분석에 대해서 연구할 계획이다.

Acknowledgement

이 논문은 정보통신산업진흥원의 SW 공학 요소기술 연구개발사업 및 한국연구재단을 통해 교육과학기술부의 세계수준의 연구중심대학육성사업(WCU, R31-10100)으로부터 지원받아 수행되었습니다.

참고문헌

[1] F Van der Linden et al., Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering, Springer-Verlag, 2007.

[2] K. C. Kang et al., “Feature-Oriented Domain Analysis (FODA) Feasibility Study,” CMU/SEI-90-TR-21, November 1990.

[3] K. Lee and K. C. Kang, “Feature Dependency Analysis for Product Line Component Design,” ICSR’04, LNCS 3107, pp. 69-85, 2004.

[4] S. Ferber et al., “Feature Interaction and Dependencies: Modeling Features for Reengineering a Legacy Product Line,” SPLC'02, LNCS 2379, pp. 235-256, 2002.

[5] H. Ye and H. Liu, “Approach to Modeling Feature Variability and Dependencies in Software Product Lines,” IEE Proc.-Softw., vol. 152, No. 3, 2005.

[6] Y. Lee et al., “An Approach to Managing Feature Dependencies for Product Releasing in Software Product Lines,” ICSR'06, LNCS 4039, pp. 127-141, 2006.

[7] W. Zhang et al., “Feature-Driven Requirement Dependency Analysis and High-Level Software Design,” Requirements Eng (2006) 11: 205-220.

[8] L. Daizhong and D. Shanhui, “Feature Dependency Modeling for Software Product Line,” ICCET'09, pp. 256-260, 2009.

[9] J. Liu et al., “Modeling Interactions in Feature Oriented Software Design,” ICFI'05, Leicester, United Kingdom, 2005.

[10] D. Batory, et al., “Feature interactions, products, and composition,” GPCE'11, 2011.

[11] D. L. Parnas, “Designing Software for Ease of Extension and Contraction,”IEEE Transaction on Software Engineering, vol. SE-5, no. 2, March 1979.