To appear in the Proceedings of the 13th International Software Product Line Conference, San Francisco, CA, 2009

# Modeling PLA Variation of Privacy-Enhancing Personalized Systems

Scott A. Hendrickson*, Yang Wang*, André van der Hoek, Richard N. Taylor, Alfred Kobsa
Institute of Software Research
University of California, Irvine
Irvine, CA 92697 USA
{shendric, yangwang, andre, taylor, kobsa}@uci.edu

## Abstract

*Privacy-enhancing personalized (PEP) systems address individual users' privacy preferences as well as privacy laws and regulations. Building such systems entails modeling two different domains: (a) privacy constraints as mandated by law, voluntary self-regulation, or users' individual privacy preferences, and modeled by legal professionals, and (b) software architectures as dictated by available software components and modeled by software architects. Both can evolve independently, e.g., as new laws go into effect or new components become available. In prior work, we proposed modeling PEP systems using a product line architecture (PLA). However, with an extensional PLA, these domain models became strongly entangled making it difficult to modify one without inadvertently affecting the other. This paper evaluates an approach towards modeling both domains within an intensional PLA. We find evidence that this results in a clearer separation between the two domain models, making each easier to evolve and maintain.*

## 1 Introduction

To provide personalized services such as customized recommendations, a personalized website collects and uses users' personal data, which raises various privacy concerns [17]. We use the term *privacy constraints* to denote users' privacy preferences as well as privacy laws and regulations that are in effect. We call personalized systems that address these privacy constraints *privacy-enhancing personalized* (PEP) systems. Modeling such systems concerns expressing two different domain models: privacy constraints and their interdependencies, as managed by legal professionals; and the structural features of a software architecture, as managed by software architects. We refer to these models as the *privacy model* and *software model*, respectively. Both can evolve independently: The privacy model can evolve as new laws or self-regulations are put into effect or new user privacy preferences arise. The software model can evolve when new components become available.

In prior work [28], we found that privacy constraints may affect the admissibility of certain components or features in a PEP system. Therefore, different sets of privacy constraints may lead to different architectures of a PEP system. Based on this finding, we proposed modeling a PEP system as a product line architecture (PLA) [29]. Doing so allows a PEP system to dynamically select a product architecture for each user based on their current privacy constraints, which can change over time.

Currently, PLA modeling approaches are predominantly *extensional*, i.e., they model a single, monolithic architecture that simultaneously represents all possible products using variation points and guards of some form, e.g., [27][13][8][25]. These approaches could be viewed as "configurable architectures", where an architect obtains an individual product architecture by resolving each variation point based upon a selection of desired attributes. While these approaches adequately model PLA variation, they suffer from a sizable mismatch between conceptual variability (i.e., the features through which architects logically view and interpret product differences) and actual variability (i.e., the modeling constructs through which the logical differences must be expressed). As a result, the actual model exhibits a high degree of redundancy, scattering and tangling of the conceptual model it represents making it difficult to interpret and modify [14].

Alternatively, *intensional* [5] approaches are gaining ground, e.g., [14][2][1][7]. With intensional approaches, product architectures are *composed* from different modeling constructs that represent features at some level. Our previous work presented an intensional approach where an architect composes product architectures from a collection of change sets and is guided by constraints expressed as relationships [14]. Together, change sets and relationships form the basis for modeling features and feature models.

[*] First authors listed in alphabetical order.

Our earlier work found that the structural features of a feature model are better expressed using change sets and relationships than the modeling constructs of an extensional approach [14]. This paper extends our earlier work in two ways. First, we apply the intensional approach to a domain where a conceptual model exists, that of the privacy constraints, and where product configurations cannot practically be predefined and must instead be dynamically created. Second, we show evidence that a PLA-based PEP system is easier to maintain when represented using an intensional approach than when represented using an extensional approach.

In the remainder of this paper, we briefly explain several background concepts in Section 2. We then describe the motivating example PEP system used for our comparative analysis in Section 3. Thereafter, we present how to model the system extensionally and intensionally in Section 4. Subsequently, in Section 5, we introduce three common evolution scenarios of PEP systems and show how both modeling approaches would adapt to these changes. We then discuss insights gained from our evaluation in Section 6. Finally, we discuss related work in Section 7 and conclude in Section 8.

# 2 Background

The work presented in this paper relies on concepts from personalization and privacy, software architecture and configuration management. We introduce these concepts here.

## 2.1 Personalization and Privacy

Advantages of web personalization have been demonstrated for both online customers (e.g., getting personalized content) and vendors (better customer retention) [6]. However, numerous opinion polls and empirical studies have revealed that Internet users have considerable concerns regarding the disclosure of their personal data to websites, and the monitoring of their Internet activities (see [17] for an overview). Most privacy laws that have emerged in response are applicable only within the boundaries of the corresponding country, e.g., Germany. However, some privacy laws may be applicable beyond a country's border so long as the services are provided to its citizens and permanent residents, e.g., the Australian privacy law. Laws, along with industry self-regulation and user privacy preferences, place requirements on the collection, storage and processing of personal data. These may involve proper data acquisition, mandatory purpose of use notifications, restrictions on permissible data transfers (e.g., to third parties and/or across national borders) and requirements for certain data processing (e.g., organization, modification and destruction). Other provisions specify user opt-ins (e.g., asking for their consent before collecting their data), opt-outs, informing users about their rights (e.g., regarding the disclosure of the processed data), adequate security mechanisms (e.g., access control), and the supervision and audit of personal data processing [28]. Finally, interdependencies between privacy constraints may also exist, e.g., privacy requirements from European Union (EU) directives must be implemented by all of its member states in their own privacy laws.

## 2.2 Software Architectures

*Software architectures* provide high-level abstractions for representing a system's structure, behavior, and key properties. These are generally expressed using an *architecture description language* (ADL) [19], which captures concepts such as the elements from which systems are built, interactions among those elements, patterns that guide their composition, and constraints on these patterns [23].

Whereas "normal" software architectures define the architectural structure of a single software system, a *product line architecture* (PLA) simultaneously defines the architectural structure for a set of closely-related systems or products [4]. As such, it must provide a basis by which an architect may understand and manipulate the commonalities and variabilities existing among each product constituting the PLA and also must support the creation of each individual product architecture from the PLA, for instance to deploy an individual product to a client.

A number of ADLs support the specification of product line architectures, typically distinguishing *core elements* from *variation points*. Variation points, architectural elements themselves, specify places in the PLA where differences exist among specific product architectures. For instance, Koala uses switches [27], xADL 2.0 [8] and Ménage [13] allow optional, variant, and optional variant elements, and COVAMOF utilizes optionals, alternatives, optional variants, variants, and values [25]. Most express these differences in some form of configuration or constraint language and promote commonly-used rules to first-class language constructs. For instance, xADL 2.0 uses Boolean expressions and Koala has a language construct for switches that route connections to one of several alternative interfaces.

## 2.3 Configuration Management

The discipline of *configuration management* (CM) has been primarily concerned with capturing the evolution of a software system at the source code level [10]. For this, it has extensive and detailed mechanisms and procedures for storing multiple versions of code and allowing multiple developers parallel access to that code [5]. Automated conflict detection and merge routines help in reconciling overlapping changes that may arise as a result of parallel development [20].

Of interest to this paper are the concepts of extensional and intensional versioning [5]. In *extensional versioning*, the configuration management system focuses on managing versions of artifacts that result after making changes. Typically, a version graph is used to relate different versions of an artifact; developers retrieve a particular version, modify it, and then add the new version to the graph when complete.

In contrast, *intensional versioning* makes changes a first class entity, inverting the relationship between versions and changes [5]. Instead of ensuring that each version is uniquely stored and accessible, intensional versioning stores each change as a change set (a "delta") independently from the other changes. So, instead of requesting a version of an artifact, developers retrieve an artifact by requesting a series of change sets from which a "version" is constructed. Similarly, after modification of this "version", the delta between this new and the original version is stored as an individually-identifiable change set. This has the advantage that new incarnations of an artifact can be composed by mixing and matching different change sets.

## 3 Motivating Example

As a motivating example, consider an online movie recommender system[1]. To make personalized recommendations, our system can utilize three features: (1) *cross-site tracking*, to observe what other websites a user visits, (2) *single session logs*, which provide information about the pages a user has visited on our website during the user's current session, and (3) *multiple session logs*, which additionally provide information about the pages a user has visited on our website in the past, during previous sessions. The first feature is optional while the singe and multiple session log features are alternatives, meaning that only one may be selected. In this paper, we refer to these features and the feature model that binds them, collectively as the *software model*.

Being a PEP system, our example also has a *privacy model*, which defines the various privacy constraints placed on the system according to the users stated preferences and location. For our example, we chose to deploy the system in Germany, the United Kingdom, and the United States. The privacy model must ensure that the product obtained from the software model adheres to the individual laws of each country.

Initially, our system's privacy model recognizes two conditions for German citizens[2]:

- Cross-cite tracking is prohibited, without consent.
- Multiple session logs are prohibited, without consent.

Upon navigating to our website, a user can explicitly specify their privacy preferences (possibly indicating consent) for a set of pre-defined options, or leave these preferences unspecified.

Our system dynamically generates a product tailored to each individual user's personal privacy constraint preferences. It also ensures that the resulting product adheres to the privacy laws and regulations in effect for the user, as indicated by the user's location. If preferences are not explicitly stated, the system chooses features that provide the most information to the system, which are legally allowed by the user's location. Thus, German citizens without explicit preferences are provided a system with the single session logs feature while citizens from the United Kingdom and the United States without explicit preferences are provided systems with the cross-site tracking and multiple session logs features. Preferences, of course, if stated, override these defaults when the law allows.

## 4 Extensional and Intensional Modeling

In this section, we model our example PLA both extensionally and intensionally, noting intricacies of both approaches. We conclude this section with a reflection on the two modeling approaches.

### 4.1 Modeling Extensionally

Taking an extensional modeling approach, we use variation points to denote the places where variabilities occur in the PLA. The resulting *software model* for our system is shown in Figure 1. In this figure, core elements belonging to all product architectures are shown as solid boxes or lines, optional elements as dashed boxes or lines, and variant elements as large boxes containing the variants. Additionally, but not shown, each variation point is annotated with a Boolean guard that indicates when that element is to be included in or excluded from a particular product architecture.

To model the privacy model of our system, we define the following variables and their possible values:

```
Country = Germany | UK | USA
SessionLogPref
  = single | multiple | unspecified
CrossSiteTrackingPref
  = allow | disallow | unspecified
```

These variables are used in the Boolean guard expressions of each variation point. For example, the Boolean guard

[1] Inspired by the MovieLens system (www.movielens.org)

[2] Based on the German Telemedia law [9]

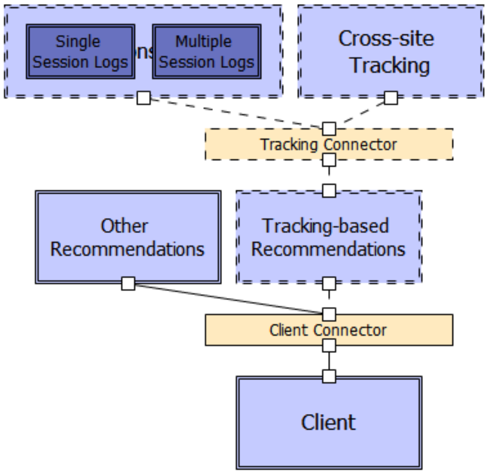


**Figure 1. Extensional Model**

for the "Cross-site Tracking" component (using a Java-like notation) is:

```
CrossSiteTrackingPref == "allow"
||(CrossSiteTrackingPref=="unspecified"
    && Country != "Germany")
```

The above guard states that the architectural element is to be included when the user has explicitly allowed it, or when the user has not stated a preference and the user is in a country other than Germany. A product is selected by evaluating each variation point's Boolean guard against values assigned to each variable, indicating whether the variation point is included in, or excluded from, the desired product.

As we can see, the software model and privacy model immediately become entangled as the privacy model is expressed in terms of the impact of a user's preferences and location on each and every variation point in the system. Furthermore, as features overlap in the extensional approach, guards become more complex. For instance, the "Tracking-based Recommendations" component should be included when any of the three features ("Single Session Logs", "Multiple Session Logs", and "Cross-Site Tracking") are enabled. The resulting guard is:

```
(SessionLogsPref == "single"
 ||(SessionLogsPref == "unspecified"
     && Country == "Germany"))
||(SessionLogsPref == "multiple"
 ||(SessionLogsPref == "unspecified"
     && Country != "Germany"))
||(CrossSiteTrackingPref == "allow"
 ||(CrossSiteTrackingPref=="unspecified"
     && Country != "Germany"))
```

From these example guards, three problems become apparent. First, information is *scattered*. The clause in the first example is the condition for selecting the cross-site tracking feature. However, the same clause is repeated at the end of the second example. For an architect to determine all elements affected by that feature, he must examine each and every guard throughout the architecture. Second, information is *tangled*. The second example contains three main clauses, each capturing the condition for selecting the "Single session logs", "Multiple session logs", and "Cross-site Tracking" features, respectively. To modify this guard, an architect must mentally extract these concepts, modify them, and then recombine them to update the guard's expression. Finally, information is *redundantly* expressed. The links and connectors surrounding the "Tracking-based Recommendations" component have the same guard, because they are included or excluded in unison with the component. These factors make interpreting and updating an extensional PLA a tedious and error-prone task.

A deployed system using our extensional PLA example would have a complete copy of the entire PLA as presented above. A new user visiting the website would be prompted for her privacy preferences. Once obtained, these values and the users country would be directly plugged into the variables used in the PLA guards. These guards would be resolved to obtain a specific system configuration matching the user's privacy constraints, which would then be instantiated for the user. If the user changes her privacy preferences, a potentially different system will then be instantiated for the user.

Modeling the entire system extensionally requires a total of 3 variables: 2 for the users preferences regarding each feature and 1 for the user's country. The model has 10 variation points (the "Session Logs" component and each of its variants have their own Boolean guard) and there are a total of 5 unique Boolean guards throughout the model.

## 4.2 Modeling Intensionally

Instead of using variation points and guards, the intensional modeling approach uses change sets and relationships to represent the PLA. We chose to use four categories of change sets in our intensional model: (1) feature change sets and relationships which are used to model the software model, (2) preference change sets and (3) country change sets which are used to model the privacy model, and (4) mapping change sets which are used to connect the two models.

The *feature change sets*, prefixed by "Feature", are used to model the structural features of the software system, as shown on the left-hand-side of Figure 2. For each element in a feature change set, an annotation with a "+" means that the element is added by the change set and an "x" means that the element is removed. The presence of a clear, dashed, "ghost" element indicates that the change set references, but does not actually modify, that particular element. For instance, the "Feature: Single Session Logs" change set adds the "Single Session Logs" component, its interface, and a link to an interface on the "Tracking Connector". However, it does not modify the "Tracking Connector" itself.

These feature change sets are merged together to compose different product architectures. The "Feature: Baseline" change set adds elements common to all products while the three feature change sets on the top row of the left-hand-side of Figure 2 add elements that are unique to their respective features. In the intensional model, feature overlap is modeled as a separate change set that adds the elements common to both features. The "Feature: Tracking-based Recommendations" change set is such a change set, adding, for instance, the "Tracking-based Recommendations" component needed by all three features. A relationship is used to ensure that this change set is included whenever any of the feature change sets are included.

The entire collection of change sets and relationships is shown on the right-hand-side of Figure 2 in a "variability spreadsheet". Each row represents a change set and each column to the right of the "Change Set" column represents a relationship (numbered from 1 to 15). Note that change sets and relationships are grouped according to the two domain models from which they originate: change sets and relationships for the software model are in the lower-right quadrant of the spreadsheet and managed solely by software architects, as indicated by the "Software" labels. Change sets and relationships for the privacy model as managed by legal professionals are in the upper-left quadrant of the variability spreadsheet, as indicated by the "Privacy" labels.

Change sets added by legal professionals that are used in the privacy model are *empty*, only serving as "place holders" or "switches" for the privacy-related concepts they wish to model. These privacy-related change sets include the *preference change sets* prefixed by "Pref" that represent user preferences and the *country change sets* prefixed by "Country" that model users' countries.

The *mapping change sets* prefixed by "Mapping" serve as the common terms of discourse through which legal professionals and their privacy model interact with software architects and their software model. Professionals from each domain must agree on the meaning of these mapping change sets. Legal professionals add relationships to ensure that a selection of a country and preferences imply the appropriate set of mapping change sets, shown in the upper-left quadrant of the "variability spreadsheet" in Figure 2. Software architects add relationships to ensure that a selection of mapping change sets imply the appropriate feature change sets, shown in the lower-right quadrant of the "variability spreadsheet" in Figure 2.

Several representative relationships in Figure 2 are read as follows:

Relationship 4 expresses an internal mapping within the privacy model. In this case, the legal professionals chose to have a country selection indicate a default set of preferences that adhere to each country's privacy concerns. These, of course may be overridden by the users' actual preferences. As an AND relationship type, this relationship reads: if the "Country: Germany" change set is selected, and the user has not selected the "Pref: Multiple Session" preference change set, then the "Pref: Single Session" preference change set should be selected.

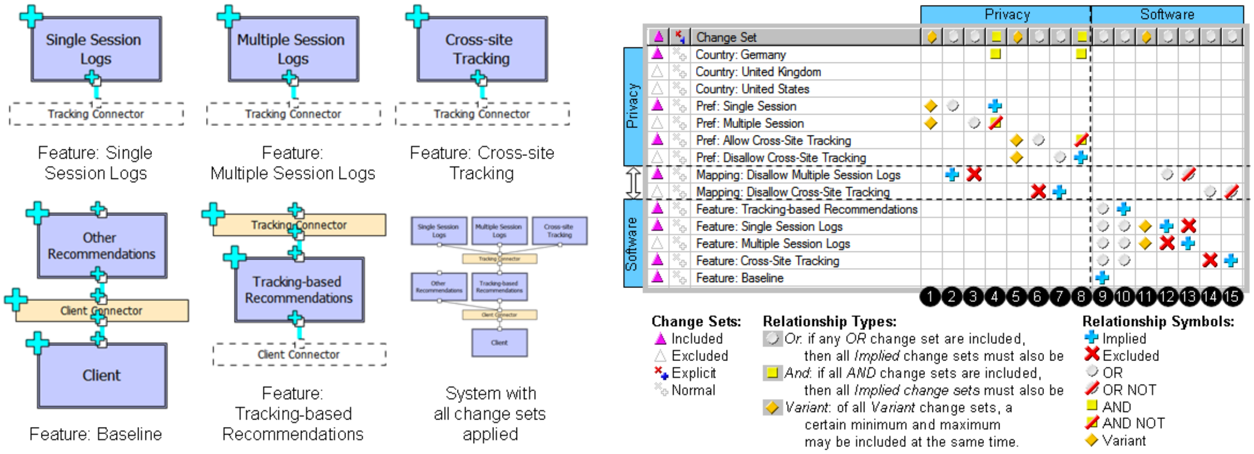
Relationship 2 expresses a mapping from the privacy model to the mapping change sets that are common to both models. As an OR relationship type, this relationship ensures that when the "Pref: Single Session" change set is selected, the "Mapping: Disallow Multiple Session Logs" change set is also selected.

Relationship 12 expresses a mapping from the mapping change sets to the software model. As an OR relationship type, this relationship ensures that when the "Mapping: Disallow Multiple Session Logs" change set is selected, the "Feature: Multiple Session Logs" change set is *not* selected and the "Feature: Single Session Logs" change set is selected.

Relationship 10 expresses an internal mapping within the software model. It captures the dependency that the tracking-based change sets on the top row of the left-hand-side of Figure 2 have on the "Feature: Tracking-based Recommendations" change set, which adds elements common to those three features. As an OR relationship type, this relationship ensures that when any of the three feature change sets are selected, the "Feature: Tracking-based Recommendations" change set is also selected.

Note that in Figure 2 the upper-right and lower-left quadrants of the variability spreadsheet are empty, indicating no cross-over occurs between these two models, other than through the mapping change sets. This is because the two domain models remain independent, interconnected only through the common mapping change sets. As such, professionals in each domain may freely modify the change sets and relationships used to capture the concepts relevant to their respective domains. Interaction is only needed when mapping change sets are added, removed, or their purposes are changed.

Our intensional PLA would be deployed in a similar fashion as the extensional PLA: the system would have a complete copy of the entire PLA as presented above and new users would be prompted for their privacy preferences. Once obtained, the user's preferences and country would be mapped to an initial selection of change sets. Thus, the



|  |  |  |  | Privacy |  |  |  |  |  |  |  | Software |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  | ▲ | ✕✚ | Change Set | ◆ | ○ | ○ | ■ | ◆ | ○ | ○ | ■ | ○ | ○ | ◆ | ○ | ○ | ○ | ○ |
| Privacy | ▲ | ✕✚ | Country: Germany |  |  |  | ■ |  |  |  | ■ |  |  |  |  |  |  |  |
|  | △ | ✕✚ | Country: United Kingdom |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | △ | ✕✚ | Country: United States |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | ▲ | ✕✚ | Pref: Single Session | ◆ | ○ |  | ✚ |  |  |  |  |  |  |  |  |  |  |  |
|  | △ | ✕✚ | Pref: Multiple Session | ◆ |  | ○ | ⧄ |  |  |  |  |  |  |  |  |  |  |  |
|  | ▲ | ✕✚ | Pref: Allow Cross-Site Tracking |  |  |  |  | ◆ | ○ |  | ⧄ |  |  |  |  |  |  |  |
|  | △ | ✕✚ | Pref: Disallow Cross-Site Tracking |  |  |  |  | ◆ |  | ○ | ✚ |  |  |  |  |  |  |  |
| ⇕ | ▲ | ✕✚ | Mapping: Disallow Multiple Session Logs |  | ✚ | ✖ |  |  |  |  |  |  |  |  | ○ | ⊘ |  |  |
|  | △ | ✕✚ | Mapping: Disallow Cross-Site Tracking |  |  |  |  |  | ✖ | ✚ |  |  |  |  |  |  | ○ | ⊘ |
| Software | ▲ | ✕✚ | Feature: Tracking-based Recommendations |  |  |  |  |  |  |  |  | ○ | ✚ |  |  |  |  |  |
|  | ▲ | ✕✚ | Feature: Single Session Logs |  |  |  |  |  |  |  |  | ○ | ○ | ◆ | ✚ | ✖ |  |  |
|  | △ | ✕✚ | Feature: Multiple Session Logs |  |  |  |  |  |  |  |  | ○ | ○ | ◆ | ✖ | ✚ |  |  |
|  | ▲ | ✕✚ | Feature: Cross-Site Tracking |  |  |  |  |  |  |  |  | ○ | ○ |  |  |  | ✖ | ✚ |
|  | ▲ | ✕✚ | Feature: Baseline |  |  |  |  |  |  |  |  | ✚ |  |  |  |  |  |  |
|  |  |  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

**Change Sets:**
▲ Included
△ Excluded
✕✚ Explicit
✕✚ Normal

**Relationship Types:**
○ *Or*: if any *OR* change set are included, then all *Implied* change sets must also be
■ *And*: if all *AND* change sets are included, then all *Implied change sets* must also be
◆ *Variant*: of all *Variant* change sets, a certain minimum and maximum may be included at the same time.

**Relationship Symbols:**
✚ Implied
✖ Excluded
○ OR
⊘ OR NOT
■ AND
⧄ AND NOT
◆ Variant

**Figure 2. Change Sets and Relationships**

use of change sets as opposed to variation points would be transparent to the end user – only the legal and software professionals need know its details.

From the initial selection of change sets, the system would automate the process of selecting mapping change sets and feature change sets as dictated by the system's relationships. To do so, the system could continually scan through the relationships, selecting implied change sets and unselecting excluded change sets for any relationship that is violated until either one of two conditions occur. First, if a configuration does not violate any relationships, then a complete system matching the user's privacy constraints has been configured, which would then be instantiated for the user. Otherwise, the system will eventually hit a violated variant relationship or revisit a previous selection of change sets, indicating that a system cannot be composed, e.g., because of a contradiction in the relationships.

The entire system, modeled intensionally, has a total of 14 change sets: 4 for user preferences, 3 for a user's country, 2 for mappings between the two domains, and 5 for system features. The model includes 15 relationships: 4 for interdependencies between privacy constraints (Relationship 1, 4, 5, 8), 3 for interdependencies between architectural features (Relationship 9, 10, 11), and the remaining 8 for mappings from the privacy domain to the software domain. We note that the initial work of building the intensional model is more involved as it models more domain concepts explicitly, but it enables a greater degree of separation between both domain models and their interactions, making them easier to interpret, evolve and maintain.

## 5 Evaluation

In this section, we evaluate both models in terms of three scenarios that are likely to occur in the evolution of a privacy-enhancing personalized system: First, a new law is introduced that adds a new requirement not previously modeled in either domain. Second, a software architect chooses to modify their model by refactoring a feature. Third, a law is modified so as to include additional countries, but does not require or prohibit any new system features. Each scenario modifies the systems that result from the previous scenario, building on the previous scenarios' modifications.

### 5.1 Scenario 1: Modifying Both Models

This scenario is based on a European Union (EU) Directive on Privacy and Electronic Communications (2002) [12] mandating that tracking-based services require anonymization or user's consent. Anonymization of tracking affects both domain models because neither currently captures such a concept. We discuss updating the software model and the privacy model for each approach.

#### 5.1.1 Scenario 1: Extensional Model

To update the extensional PLA, we first focused on modifying the software model. To do so, we changed the "Tracking-based Recommendations" component into an optional-variant that contains an "Anonymous Tracking-based Recommendations" variant that anonymizes the data and an "Identifiable Tracking-based Recommendations" variant that does not anonymize the data. The new component is shown in Figure 3.

While the effort involved in the introduction of the new variants was minimal, we ran into two limitations when trying to assign them guards. First, the extensional model disallows the calculation of intermediate values, such as whether or not the user's country is within the European Union. As such, we had to treat the new anonymization constraint as if it were placed on each country within the Euro-

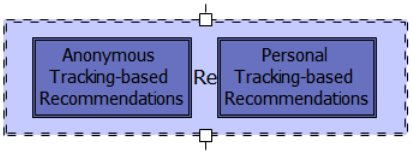

Anonymous Tracking-based Recommendations
Re
Personal Tracking-based Recommendations

Figure 3. New Optional-Variant

pean Union individually, namely Germany and the UK. Second, we found that we could not assign guards to the new variants without considering the privacy constraints, which are part of the privacy model, since these are directly represented in the guards. Instead, to assign Boolean guards the architect and legal professionals would have to work together to ensure that both the privacy and software models were correctly reflected in each guard.

The resulting extensional model contained one additional variable to express a user's preference regarding anonymization and two new, unique Boolean guards, one for each new variant. The resulting model has 4 variables, 12 variation points, and 7 unique Boolean guards.

### 5.1.2 Scenario 1: Intensional Model

To update the intensional PLA, we again focused on updating the software model first. We added a new change set which replaces the "Tracking-based Recommendations" component with a "Anonymous Tracking-based Recommendations" component. This change set is shown in Figure 4. We then updated two relationships in the software model to correctly integrate the new feature change set with the other feature change sets.

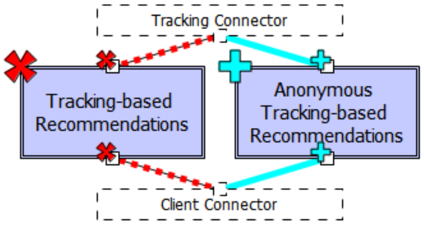

Tracking Connector
Tracking-based Recommendations
Anonymous Tracking-based Recommendations
Client Connector

Figure 4. New Change Set

To update the privacy model, we added 3 change sets and 5 relationships. Of significance was the fact that we could explicitly model the concept of being "In the European Union". To do so, we created a "Country: Member of the EU" change set and added a relationship implying that this change set will be selected when users indicate that they are in Germany or the UK.

For the intensional model, we added a total of 5 new change sets and 9 relationships, we also modified 2 relationships within the software model.

### 5.1.3 Insights from Scenario 1

In this scenario, we found that we could express new concepts in the intensional model (i.e., whether a country is part of the EU) that we could not directly express in the extensional model. We also note that updating the intensional model involved more actual changes, but allowed us to focus on smaller parts of the model while making those changes, such as focusing on one domain model at a time.

## 5.2 Scenario 2: Modifying the Software Model

In this scenario, the architect decided to refactor the anonymization feature introduced in the previous scenario. Instead of having two variants of the "Tracking-based Recommendations" component (one with anonymization and one without), the architect chose to utilize a single "Tracking-based Recommendations" component and an optional "Anonymization" component. This change was conceptually confined to only the software model of the system.

### 5.2.1 Scenario 2: Extensional Model

To implement this architectural change, the architect reverted the optional-variant component back to an optional component and added a new optional component, as shown in Figure 5.

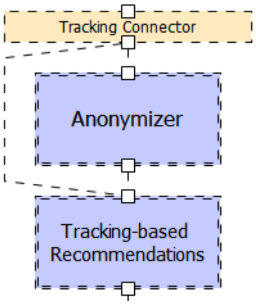


Figure 5. New Optional Component

Updating the model required making the structural changes and then updating the guards. Again, we found that we cannot assign guards without considering both domain models because they are intertwined. Additionally, the guard for the "Anonymizer" component turned out to be more complex than that of the "Tracking-based Recommendations" component (shown in Section 3) because it had to account for yet another preference variable. The guard for the new link in Figure 5 was equally complex.

The resulting extensional model contained two new, unique Boolean guards, however the two guards introduced in the previous scenario were removed. The resulting model has 4 variables, 13 variation points, and 7 unique Boolean guards.

### 5.2.2 Scenario 2: Intensional Model

In comparison, to update the intensional model, we modified the change set introduced in the previous scenario to that shown in Figure 6. These modifications were contained within the single change set and so we did not need to consider any other concepts modeled in the system, even the relationships of the software model. Instead, we already knew that the particular change set was included or excluded from the model when appropriate.

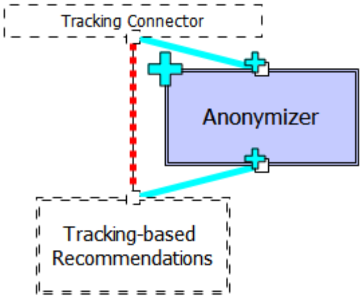


**Figure 6. Revised Change Set**

### 5.2.3 Insights from Scenario 2

We found that, for the extensional model, the amount of work involved was similar to that of the first scenario – each guard had to be interpreted with only a few being updated, and both domain models had to be considered when updating the guards. In contrast, the intensional model proved far superior in this scenario. The feature was already represented using a change set, which was introduced in the previous scenario. Thus, we did not have to examine anything beyond the boundaries of that change set, instead we simply modified the change set's contents.

## 5.3. Scenario 3: Modifying the Privacy Model

For the final scenario, we consider a change that is confined to the privacy model. Here, we imagine that the German privacy law that prohibits cross-site tracking has been adopted by the European Union, making it applicable to all of its constituent countries.

### 5.3.1 Scenario 3: Extensional Model

No structural modifications were necessary for the extensional model, however a *significant* number of Boolean guards had to be modified. In fact all Boolean guards needed to be updated except for four. This was because so many other optional parts of the extensional model need to be present to utilize the "Cross-site Tracking" component, such as the "Tracking-based Recommendations" and "Anonymizer" components as well as the links and connectors between them. Specifically, the guards of 9 different variation points, consisting of 4 unique guards needed to be updated. Though many guards were updated, the overall number of variables, variation points and unique Boolean guards of the resulting model remained the same.

### 5.3.2 Scenario 3: Intensional Model

To make this change in the intensional model, Relationship 4 from Figure 2 was duplicated and modified to reference the "Country: Member of the EU" change set added in scenario 1 rather than the "Country: Germany" change set. While it would have been possible to simply modify Relationship 4 itself, we chose to duplicate it so that the privacy model accurately reflects the fact that both Germany *and* and the EU prohibit cross-site tracking without consent, in case the laws change in the future.

### 5.3.3 Scenario 3: Insights

We found in this scenario that the extensional model had to be significantly changed in order to reflect changes in the privacy model. This is because the privacy model is implicitly embedded through the extensional model at each point of variation. In the intensional model, we found that the graphical representation of relationships aided in identifying the specific relationship that we wanted to copy, we simply looked at the relationships referencing the cross-site tracking preference change sets.

## 6 Discussion

We observed a number of tradeoffs between the two approaches from our initial modeling of the systems in Section 4 and the evolution scenarios in Section 5.

We found during the initial modeling and subsequent modification of the intensional and extensional models that introducing *new* concepts into the intensional model was generally more tedious, but easier than introducing these concepts in the extensional model. It was more *tedious* because, when compared to the Boolean guards in the extensional model, relationships in the intensional model tended to express small, individual dependencies. At the same time, the intensional model captured a more complete specification (such as expressing whether a country is part of the EU, in Scenario 1) of both domain models. As a result, more change sets and relationships were necessary to model the system. However, it was *easier* to model the concepts in the intensional model for these same reasons: smaller, isolated relationships were easier to express and the model was capable of modeling the additional concepts needed. This is as compared to the Boolean guards that were modeled in the extensional approach. For the extensional model, it was initially very difficult to determine how the relevant variables could be combined into single expressions and concepts such as the existence of the EU could not be directly modeled.

Even though the number of modeling constructs used in the intensional model was greater, interpreting and verifying the intensional model was significantly easier because we were able to focus on smaller, more isolated concepts. For instance, to verify that relationships captured the desired concepts correctly, we were generally able to talk through a logical argument. We did make a truth table for Relationship 4 in Figure 2, which is perhaps the most unintuitive relationship, but it was small. In contrast, to verify that the Boolean expressions of the extensional model captured the desired concepts correctly, we had to create much larger truth tables.

Since the intensional approach models domain concepts explicitly and independently, it enables a better separation of the domain models. The privacy model explicitly represents various privacy constraints and their interdependencies. The software model explicitly represents architectural features and their interdependencies. For example, in order to implement the change introduced in Scenario 2, only the change sets and relationships of the software model needed to be examined and in turn only one change set needed to be updated. On the other hand, implementing the change in the extensional model involved both domain models as they were entangled in the Boolean guard expressions.

The clearer separation of domains may also lead to a better division of work. Law professionals can primarily focus on the privacy model, while software architects can largely focus on the system model. Only the mappings from privacy constraints to architectural features demand knowledge of both domains. For instance, in order to realize the change introduced in Scenario 3 in the intensional model, law professionals only needed to update a single relationship in the privacy model without any involvement of the software architects. In contrast, implementing the same change in the extensional model required knowledge from both domains. In future work, it will be useful to add support for grouping and categorizing change sets and relationships according to their respective domain model.

## 7 Related Work

Our work is related to several different efforts. The field of feature engineering has worked on modeling features and their relationships for a number of years [15][3]. Recent work has focused on modeling quality attributes in addition to functional features [11]. These usually stay at the conceptual level, although some exceptions exist [26]. For instance, Lago and Van Vliet built an extensive mapping from conceptual features to architectural components [18] and Niemelä et. al. present an approach for mapping quality attributes to architectural components [21]. We believe that our work complements theirs by presenting a PLA approach that reduces the complexity of such mappings.

Our work is also related to other intensional versioning approaches. AHEAD, an example of feature-oriented programming [24], uses a compositional expression language that can be generically mapped onto rules that govern the underlying composition process [1] and has been applied to PLAs. Similarly, our work is related to multi-dimensional separation of concerns [22]. Here, change sets are close to modules, and change set selection with our merge algorithm, is close to a hypermodule. However, most research in these areas has focused on the source code level.

Aspect-oriented programming [16] is also related to our work. By their very nature, change sets are close to aspects in their compositional capabilities; in fact, one could probably support the other and vice versa. However, compared to aspects, our work has taken the generic change set idea, modified it to support PLAs, and built enhanced support through detailed relationships.

In the field of privacy-enhanced technology, the Platform for Privacy Preferences (P3P) [30] enables websites to express their privacy policies in a standard machine-reable format that can be retrieved automatically and interpreted by user agents. Client-side agents can then inform users about the sites' privacy policies and warn them when those deviate from their privacy preferences. However, P3P does not enforce privacy policies nor does it support different policies for different users. By itself, it is therefore not an answer to the need for privacy tailored to different users' privacy constraints. In contrast, our PLA-based approaches can automatically generate a particular personalized system that caters to each individual user's privacy constraints.

## 8 Conclusions

This paper presents a detailed comparison between the extensional and the intensional approach in modeling the PLA variations for an example PEP system. We found that when modeled in an *extensional* manner, the two domain models of the PEP system become highly entangled and difficult to interpret or modify. This can become a significant problem as each model may evolve independently, e.g., as laws change (thus the privacy model changes) or new technologies became available for the system (thus the software model changes). In contrast, we found evidence that although the initial effort in creating the intensional model is more involved as it models more domain concepts explicitly, the intensional model enables a greater degree of separation between both domain models and their interactions, making each easier to interpret, evolve, and maintain.

## 9 Acknowledgement

This work is partially supported by NSF grant numbers CCR-0093489, DUE-0536203, IIS-0205724, IIS-0308277 and a Google Research Award. We thank Eric Dashofy and the SPLC09 reviewers for their insightful comments.

## References

[1] D. Batory. Feature-oriented programming and the AHEAD tool suite. In *Proc. of the ICSE04*, pages 702–703.

[2] Bell Labs Lucent Technologies. Sablime v5.0 user's reference manual. Technical report, 1997.

[3] D. Beuche, H. Papajewski, and W. Schröder-Preikschat. Variability management with feature models. In *1st Workshop on Softw. Variability Mgmt.*, pages 72–83, 2003.

[4] J. Bosch. *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach.* Addison-Wesley, 2000.

[5] R. Conradi and B. Westfechtel. Version models for software configuration management. *ACM Computing Surveys*, 30(2):232–282, 1998.

[6] P. Consortium. Personalization & Privacy Survey, 2000.

[7] T. Cottenier, A. van den Berg, and T. Elrad. The motorola WEAVR: Model weaving in a large industrial context. In *Proc. of Intl. Conf. on Aspect-Oriented Software Development*, 2006.

[8] E. Dashofy, A. van der Hoek, and R. N. Taylor. A comprehensive approach for the development of XML-based software architecture description languages. *ACM Trans. on Softw. Eng. and Methodology*, 14(2):199–245, 2005.

[9] DE-TML. German telemedia law. Technical report, 2007.

[10] J. Estublier, D. B. Leblang, G. Clemm, R. Conradi, A. van der Hoek, W. Tichy, and D. Wiborg-Weber. Impact of the research community on the field of software configuration management. *ACM Trans. on Softw. Eng. and Methodology*, 14(4):383–430, 2005.

[11] L. Etxeberria, G. Sagardui, and L. Belategi. Modeling variation in quality attributes. In *Proc. of the 1st Intl. Workshop on Variability Modelling of Software-intensive Systems*, pages 51–59, 2007.

[12] EU. Directive 2002/58/ec on processing of personal data and the protection of privacy in the electronic communications sector. 2002.

[13] A. Garg, M. Critchlow, P. Chen, C. V. d. Westhuizen, and A. v. d. Hoek. An environment for managing evolving product line architectures. In *Proc. of the Intl. Conf. on Software Maintenance*, pages 358–367, 2003.

[14] S. A. Hendrickson and A. van der Hoek. Modeling product line architectures through change sets and relationships. In *Proc. of the ICSE07*, pages 189–198.

[15] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical CMU/SEI-90-TR-21, Software Engineering Institute, November 1990.

[16] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. In *Proc. of the 11th European Conf. on Object-Oriented Prog.*, pages 220–242, 1997.

[17] A. Kobsa. Privacy-enhanced personalization. *Communications of the ACM*, 50(8):24–33, 2007.

[18] P. Lago, E. Niemelä, and H. van Vliet. Tool support for traceable product evolution. In *Proc. of the 8th European Conf. on Software Maintenance and Reengineering*, pages 261–269, 2004.

[19] N. Medvidovic and R. N. Taylor. A classification and comparison framework for software architecture description languages. *IEEE Trans. on Softw. Eng.*, 26(1):70–93, 2000.

[20] T. Mens. A state-of-the-art survey on software merging. *IEEE Trans. on Softw. Eng.*, 28(5):449–462, 2002.

[21] E. Niemelä, A. Evesti, and P. Savolainen. Modeling quality attribute variability. In *Proc. of the 3rd Intl. Conf. on Evaluation of Novel Approaches to Softw. Eng.*, pages 169–176, 2008.

[22] W. H. Peri Tarr, Harold Ossher and S. M. S. Jr. N degrees of separation: Multi-dimensional separation of concerns. In *Proc. of the ICSE99*, pages 16–22.

[23] D. E. Perry and A. L. Wolf. Foundations for the study of software architecture. *ACM SIGSOFT Softw. Eng. Notes*, 17(4):40–52, 1992.

[24] C. Prehofer. Feature-oriented programming: A fresh look at objects. In *Proc. of the 11th European Conf. on Object-Oriented Prog.*, pages 419–443, 1997.

[25] M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch. COVAMOF: A framework for modeling variability in software product families. In *Proc. of the SPLC04*, pages 197–213.

[26] C. R. Turner, A. Fuggetta, L. Lavazza, and A. L. Wolf. A conceptual basis for feature engineering. *Journal of Systems and Software*, 49(1):3–15, 1999.

[27] R. van Ommering, F. van der Linden, J. Kramer, and J. Magee. The Koala component model for consumer electronics software. *Computer*, 33(3):78–85, 2000.

[28] Y. Wang and A. Kobsa. Impacts of privacy laws and regulations on personalized systems. In A. Kobsa, R. K. Chellappa, and S. Spiekermann, editors, *PEP06: Workshop on Privacy-Enhanced Personalization*, pages 44–46.

[29] Y. Wang, A. Kobsa, A. van der Hoek, and J. White. PLA-based runtime dynamism in support of privacy-enhanced web personalization. In *Proc. of the SPLC06*, pages 151–162.

[30] R. Wenning and M. Schunter, editors. *The Platform for Privacy Preferences 1.1 (P3P1.1) Specification.* W3C Working Group Note, 2006.