Meeting Strangers Online: Feature Models for Trustworthiness Assessment

Angela Borchert[1](✉), Nicolás E. Díaz Ferreyra[2], and Maritta Heisel[1]

[1] University of Duisburg-Essen, Forsthausweg 2, 47057 Duisburg, Germany
{angela.borchert,maritta.heisel}@uni-due.de

[2] Hamburg University of Technology, Am Schwarzenberg-Campus 1, 21073 Hamburg, Germany
nicolas.diaz-ferreyra@tuhh.de

Abstract. Getting to know new people online to later meet them offline for neighbourhood help, carpooling, or online dating has never been as easy as nowadays by social media performing computer-mediated introductions (CMIs). Unfortunately, interacting with strangers poses high risks such as unfulfilled expectations, fraud, or assaults. People most often tolerate risks if they believe others are trustworthy. However, conducting an online trustworthiness assessment usually is a challenge. Online cues differ from offline ones and people are either lacking awareness for the assessment's relevance or find it too complicated. On these grounds, this work aims to aid software engineers to develop CMI that supports users in their online trustworthiness assessment. We focus on trust-related software features and nudges to i) increase user awareness, ii) trigger the trustworthiness assessment and iii) enable the assessment online. For that reason, we extend feature models to provide software engineers the possibility to create and document software features or nudges for trustworthiness assessment. The extended feature models for trustworthiness assessments can serve as reusable catalogues for validating features in terms of their impact on the trustworthiness assessment and for configuring CMI software product lines. Moreover, this work provides an example of how the extended feature models can be applied to catfishing protection in online dating.

Keywords: Feature models · Trustworthiness · Nudging · Social media · User-centred design

1 Introduction

The use of online services as a common practice is characteristic of today's digital age. Many activities that were carried out offline in the early 2000s have partly shifted to the online sphere. These include activities such as offering or seeking carpooling, neighbourhood help, a place to sleep, or even friendship or romance. For individuals, it is often necessary to go beyond their own peer group to succeed in these endeavours.

Social media platforms that offer so-called *computer-mediated introductions*, short CMI, provide users with the service of introducing them to strangers who potentially match their needs [1]. Examples of CMI are the Sharing Economy and online dating. While Sharing Economy connects unknown private individuals for a (monetary) exchange of goods or services, online dating tries to bring together people with compatible interests in terms of social relationships [1]. Apart from realising one's purposes, the benefits of CMI lie in the large number of options being proposed to the user independent of time and space. CMI eases interaction with strangers, which would not necessarily be possible offline because of missing social contact points. At the same time, strangers pose a high risk. Due to a lack of prior knowledge, it is difficult for individuals to assess the intentions of strangers and predict their behaviour. Therefore, CMI usage involves risks related to both online and offline interaction. Regarding Sharing Economy, risks are for example poor service or products, being underpaid by other users, or robbery [2,3]. In terms of online dating, risks involve among others damaged self-esteem, romance scam, sexually transmitted diseases, or sexual assault [4]. Furthermore, risks not only originate from users but also from service providers and the used software application. Companies may violate their users' privacy by misusing personal data [5]. Software applications pose security risks [6].

However, risks are tolerated if individuals perceive the party with whom they interact as trustworthy [7]. They then believe that the other party is willing and able to behave according to their positive expectations [8]. Perceived trustworthiness further impacts the decision-making process whether to start or continue an interaction [9]. Hence, not only private commerce and dating have shifted to the online sphere but also related psychological behaviours like assessing the trustworthiness of others.

Yet, it is a challenge for individuals to perform a trustworthiness assessment. Most often, it is an unconscious process in which individuals follow their gut feeling [10]. Furthermore, cues for evaluating properties of trustworthiness are different online than offline. They are hard to grasp and susceptible to manipulation [11]. Therefore, many CMI users perceive it as difficult to assess trustworthiness online. Instead, they wait for a real-world encounter to check on the other person [4]. Despite the relevance of trustworthiness assessments for safe CMI usage, CMI users perform them insufficiently, either because they are not sufficiently aware of them or because they consider them as complex.

Against this background, we ask ourselves, firstly, how CMI users can be prompted to perform a trustworthiness assessment and, secondly, how they can assess the trustworthiness of other parties online in the best possible way. As the CMI application is the main user interface, it mediates the trustworthiness assessment of other users, the respective service provider and itself as a technology [12]. Therefore, software engineers need to develop CMI applications that support their users in the assessment process. This can be achieved by integrating properties of trustworthiness into software development and the design of the graphical user interface [13].

Therefore, our research objective is to provide a method for software engineers to design, build and assess interactive applications that i) increase users' awareness of the trustworthiness assessment, ii) trigger the trustworthiness assessment and iii) enable the assessment online. We aim for a model-based approach by using extended feature models in a no-code development paradigm. Established feature models contribute to reusable, scenario-specific catalogues on which basis software product lines can be configured.

2 Theoretical Background

In this work, we adapt extended feature models to the context of trustworthiness and introduce software engineers how to use them to build user-centered, interactive CMI applications. This section provides a brief theoretical background of relevant research in trustworthiness assessment and how it can be considered in software development, software features and nudges, and feature models.

2.1 Trustworthiness Assessment

A trustworthiness assessment is a procedure on which basis an individual evaluates whether another party is trustworthy, and, thus, can be trusted [12]. In the past, various definitions for trust and trustworthiness arose.

Trust involves positive expectations regarding the actions of others [8]. Based on positive expectations, trust is the willingness to be vulnerable to another party and to depend on them [14]. Other researchers define trust as an individual's belief that another party possesses suitable properties necessary to perform as expected in a specific scenario [15]. This is in accordance with the definition of trustworthiness by Mayer et al. [16]. They identify ability, benevolence and integrity as the factors that capture the concept of trustworthiness in the context of interpersonal trust. Since then, these three properties have been applied to various other trust contexts, as for example trust in organizations and technology [15,17]. A multitude of other properties have been additionally related to the trustworthiness of *individuals, organizations* and *technology* by former research, as for example honesty, predictability and reputation [18]. Borchert & Heisel provide an overview of these properties that are unified under the term *trustworthiness facets*. Trustworthiness facets cover desirable properties that can be attributed to at least one of the three before-mentioned parties and positively impact their trustworthiness [19]. If facets are perceived as available, perceived trustworthiness increases. In the context of social media like CMI, trustworthiness facets affect people's trust in other CMI users (*computer-mediated interpersonal trust*), the service provider (*brand trust*), and the technical platform (*system trust*) [12]. CMI users evaluate trustworthiness facets in their trustworthiness assessment via the CMI platform. Assessing trustworthiness facets impact their decision with what service provider, application and other users they want to interact [19].

2.2 Identifying Trustworthiness Facets for Software Development

On these grounds, considering trustworthiness facets during software development is necessary to support CMI users in their trustworthiness assessment as good as possible [13]. Borchert & Heisel have conducted a literature review to provide an overview of trustworthiness facets [19]. Furthermore, they introduced a guideline for software engineers how to select appropriate facets for software development. Trustworthiness facets are determined in dependence on a specific problem. They can be derived by either analysing problematic characteristics or desired characteristics of a problem. Problematic characteristics are inherent to the problem context leading to trust issues. Desired characteristics resolve the trust issues of the problem. After obtaining such knowledge of the problem space, trustworthiness facets can be targeted by software features to address the problem in the solution space later in the software development process. Due to space constraints, we refer to Borchert & Heisel for the detailed procedure of the guideline [19].

2.3 Software Features and Nudges

Software features can be defined as user-accessible, useful concepts within software that realize both functional and non-functional requirements [20]. While functional requirements "describe the behavioral aspects of a system" [21], non-functional requirements are not directly related to system functionality [22]. They are rather a quality of a system functionality, such as performance or usability [21,22]. Overviews of software features have been collected in catalogues, such as the User Interface Design Pattern Library [23] or Welie.com - Patterns in Interaction Design [24].

Complying with the definition of software features, digital nudges can be categorized as features that are user-accessible and persuasive [25]. Digital nudges are user-interface design elements, more specifically "information and interaction elements to guide user behaviours in digital environments, without restricting the individual's freedom of choice" [26]. They can be characterized as soft paternalistic interventions and persuasive technologies [26]. Soft paternalistic interventions use information and guidance to point out safer and better choices for users' decisions without constraining their actions [27]. They can be used to increase user awareness concerning a specific issue [27]. Persuasive technologies try to change attitudes, behaviour or both [28]. In doing so, these technologies do not use any coercion or deception. In the context of persuasive technologies, the Fogg Behavioural Model names three requirements, which need to be considered simultaneously within the system for facilitating behaviour change of a target audience [28]. These are i) to encourage users' motivation, ii) to consider their ability to perform the targeted behaviour, and iii) to provide an effective trigger to show the targeted behaviour. Additionally, nudges may explain behaviour patterns of users and provide solutions for unfavourable behaviour [29]. Nudges can also be realized by presenting certain forms of content or information [26]. In order to create nudges, software engineers rely on nudge catalogues like the model for the design of nudges (DINU model) [26] or follow nudging design principles as from Sunstein [30].

2.4 Feature Models

Feature Models can be traced back to the Feature-Oriented Domain Analysis by Kang et al. [31] and to Czarnecki and Eisenecker [32]. They can be allocated to Software Product Line Engineering, which encompasses methodological approaches to develop software based on the reusability and variability of software features [33]. By feature models, software requirements are expressed on the abstract level of software features. Requirements are predefined by the software engineer and not part of the model. Based on the represented software features, feature models can be used for the development, configuration, and validation of software systems [34]. An example of a feature model is presented in Fig. 4 on page 14.

Feature models structure software features in a hierarchical tree. Software features are represented by single terms as nodes. At the root is the so-called concept feature, which represents a whole class of solutions. With increasing tree layers, the abstraction level of software features becomes more and more concrete. Features can be refined and become parent features by adding descriptive, related, more detailed child features in the next layer. These are called sub-features. Leaf elements of a tree are the ones that are most concrete and are called properties.

Based on the relationship of parent and child features, software product lines can be configured. This means that the software engineer can decide which features to include in a product. Relationships are modelled by the links between the features. They are depicted in Fig. 1. Links can mark single features as mandatory (simple line or line with filled bullet connected to feature) or optional (line with empty bullet next to feature). OR- or XOR-links can express the optionality of a set of sub-features. They are modelled by a semi-circle covering emanating links from a parent feature. The OR-link is modelled by a filled semi-circle. It means that at least one sub-feature must be included in an instance of the software product line. The XOR-alternative-link is modelled by an empty semi-circle. Only one sub-feature shall be selected for the software product while the others are consequently excluded. In addition to relationships between parent and child feature, there are cross-tree constraints. A dashed arrow is a requires-link. It denotes that a software feature implies another feature to be included

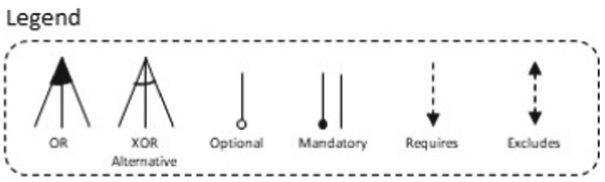


Fig. 1. Relationship links for feature models.

in the system. A double-sided dashed arrow expresses an excluded-link. Two features cannot be part in a software product line at the same time.

In the past, the basic feature model has been further extended. Extensions like the one of Benavides et al. have added additional information in form of so-called attributes to software features [35]. Attributes are any characteristic of a feature that are measurable. They are modeled next to the related feature in own boxes connected by a dotted link. Attributes may include a value that can either be continuous or discrete or mathematical relations between one or more attributes of a feature. These can be used for the validation of features [35].

3 Trust-Related Software Features

In this work, we aim to establish feature models whose features support the online trustworthiness assessment performed by CMI users. In order to contribute to the trustworthiness assessment, software features need to be trust-related. This means that they relate to at least one trustworthiness facet. Trustworthiness facets can be regarded as non-functional requirements of software features.

As we have mentioned in Sect. 1, trustworthiness assessments are a challenge for users. They are either not aware of the assessment itself or its relevance, or they perceive it as too difficult to execute online. On these grounds, we identify three categories of trust-related software features based on their purpose to support the trustworthiness assessment. **Awareness features** increase the user awareness of the trustworthiness assessment. They tend to make users realize the relevance of the assessment. **Trigger features** nudge users to perform the trustworthiness assessment. **Empowerment features** enable the user to perform the trustworthiness assessment. Most often, they provide information or interaction elements necessary for performing the trustworthiness assessment.

All three categories can additionally meet the definition of digital nudges, if they are designed to convince users to perform a trustworthiness assessment or if they shall raise user awareness [27,29]. Only empowerment features do not necessarily need to be realized in form of nudges. Instead, they may be "regular" software features.

4 Feature Models for Online Trustworthiness Assessments

Feature models are a useful approach to design software that supports its users in their online trustworthiness assessment. In the following, we introduce how we adapted the notation of the models for that purpose. Moreover, we explain how feature models for online trustworthiness assessments can be created.

4.1 Adaption of Feature Models

In order to support users in their trustworthiness assessment, we aim to cover all three software features types introduced in Sect. 3 within a model. For that reason, software features shall be labelled according to their type. Depending on the feature type, the labels are <<*awareness*>>, <<*trigger*>> or <<*empower*>>. Labelling shall be performed on the first layer underneath the concept feature.

In addition, we follow the approach of extended feature models like Benavides et al. [35] and include trustworthiness facets as attributes to software features. Trustworthiness facets that are related to software features shall be considered in their design to reflect the trustworthiness of at least on of the three parties end-user, service provider or application. Thereby, software engineers can model whether computer-mediated interpersonal trust, system trust, or brand trust is impacted by a software feature. In order to distinguish the three different kinds of trustworthiness facets in the notation, they shall be framed in different colours. Facets for computer-mediated interpersonal trust that represent the trustworthiness of other users shall be framed green. Facets for system trust reflecting the trustworthiness of the software application shall be framed orange. Facets for brand trust depicting the trustworthiness of the service provider shall be framed purple.

4.2 Feature Model Creation

Feature models are created for a specific problem to provide solution approaches in form of software features. As an application is usually confronted with multiple problems, sets of feature models need to be created for the software development of the application. Feature model creation consists of two phases: Building the model on a feature level and the facet attribution process. Latter can be subdivided into allocation phase and propagation phase.

Building the Model and Refining Features. As a first step, the problem to be addressed needs to be determined. Based on the problem, the concept feature, which represents an abstract solution approach to the problem, can be derived. Building the feature model follows the same procedure described in Sect. 2.4. Software features shall be refined in the following layers of the model in more concrete information, interaction or design elements. The elements shall correspond to the purpose of the three software feature types of Sect. 3. For modelling, engineers may rely on their creativity and expertise. External catalogues for software features and nudges may provide additional input. In doing so, engineers may consider the brand image by design. Furthermore, feature models can be created to match the business strategies of the service provider and to convey the brand message.

Facet Attribution Process. After identifying possible software features as solution approaches, trustworthiness facets are attributed to them in the facet attribution process. It is subdivded in the allocation phase and propagation phase. In the *allocation phase*, trustworthiness facets are related to each software feature of a feature model and added as their attributes. For facet identification, the guideline of Borchert & Heisel [19] can be used.

The facet allocation phase starts at the layer beneath the concept feature and continues feature-wise layer by layer. Working down the tree structure of the model, it might happen that differences emerge between the trustworthiness facets of parent and child features. Such differences can be explained by the principle that the whole is greater than the sum of its parts [36]. A parent feature might evoke other trustworthiness facets by the combination of its sub-features than a sub-feature alone. An example is presented in Sect. 6, Fig. 4. The parent feature "identity verification" as a whole shows the legal compliance of the service provider, whereas its parts in form of sub-features are not associated with this facet.

Furthermore, a feature involves facets based on the configuration of its sub-features. Depending on the configuration, different facets can be involved. For example, a warning message formulated in natural language and displayed in certain colours may reflect different facets depending on the actual wording of the message and the colour chosen to convey the message. It may be perceived as either caring or patronising. At that point, the software engineer points out a design direction how software features shall be perceived by determining trustworthiness facets. This means that by further processing with the facet allocation for features at a lower layer, new facets may be derived impacting parent features.

On these grounds, the *propagation phase* is important to realise the inheritance principle so that parent features are associated with the trustworthiness facets of their child features. Starting from the leaves of a model, the software engineer needs to check whether there are facets of features that are not yet allocated to their parent feature. If this is the case, respective facets shall be added to the parent feature above. This process is repeated until the concept feature of a model is reached. It is usual that the number of trustworthiness facets associated with a concept feature is so large that it undermines the clarity of a feature model. Therefore, all facets of a model shall be documented in a list instead of linking them as attributes to the concept feature. The list of trustworthiness facets shall be created in the end of the propagation phase. When creating the list of trustworthiness facets, descriptive information about the frequency of occurrence for each facet can be added. This information supports software engineers in evaluating the impact of a concept feature on the trustworthiness assessment. It is a first step to feature model validation.

5 Method for Using Feature Models for Supporting Online Trustworthiness Assessments

In the following, we introduce a method on how feature models for online trustworthiness assessment can be efficiently used for software development. We explain necessary input for and useful output of the feature models. These are valuable for the three steps of the method, which are feature model creation (Step 1), feature validation (Step 2) and software product line engineering (Step 3). The method is depicted in Fig. 2.

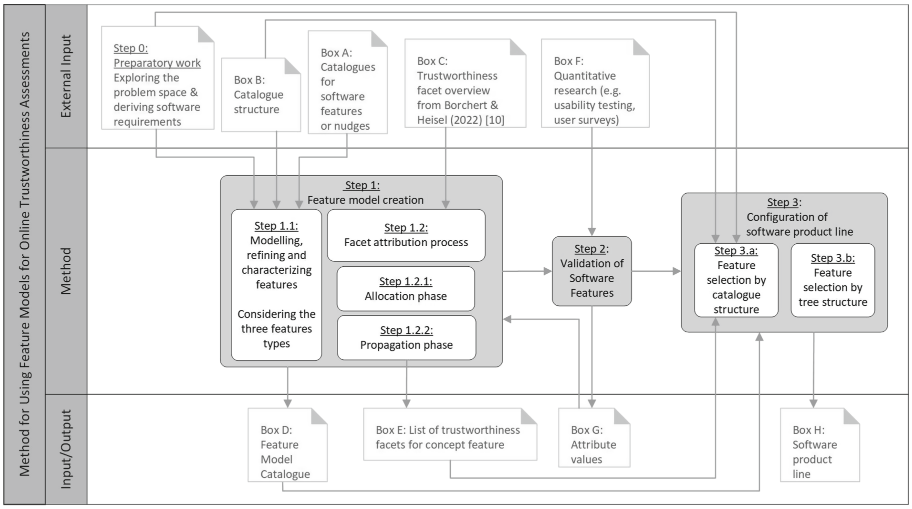

Method for Using Feature Models for Online Trustworthiness Assessments
External Input
Method
Input/Output
Step 0: Preparatory work Exploring the problem space & deriving software requirements
Box B: Catalogue structure
Box A: Catalogues for software features or nudges
Box C: Trustworthiness facet overview from Borchert & Heisel (2022) [10]
Box F: Quantitative research (e.g. usability testing, user surveys)
Step 1: Feature model creation
Step 1.1: Modelling, refining and characterizing features
Considering the three features types
Step 1.2: Facet attribution process
Step 1.2.1: Allocation phase
Step 1.2.2: Propagation phase
Step 2: Validation of Software Features
Step 3: Configuration of software product line
Step 3.a: Feature selection by catalogue structure
Step 3.b: Feature selection by tree structure
Box D: Feature Model Catalogue
Box E: List of trustworthiness facets for concept feature
Box G: Attribute values
Box H: Software product line

Fig. 2. Method for using feature models for online trustworthiness assessments.

5.1 Step 1: Feature Model Creation

Feature model creation can be divided into modelling software features (Step 1.1) and the facet attribution process (Step 1.2). The facet attribution process can be in turn divided into the allocation phase (Step 1.2.1) and the propagation phase (1.2.2). For the facet attribution process, the overview of trustworthiness facets and the guideline for facet selection by Borchert & Heisel serves as important input (Box C). The exact procedure for feature model creation is explained in Sect. 4.

In order to create feature models, preparatory work (Step 0) needs to be done first. Knowledge about the problem space, like user needs or software goals, as well as determining software requirements serve as input for creating feature models. Thereby, software features can be modelled that propose solution approaches to the problem. External catalogues of software features or nudges (Box A) like the User Interface Design Pattern Library [23] or the DINU model [26] may support the software engineer in this process. By creating feature models for a multitude of problems that a CMI application faces, a set of feature models tailored to the application is established, which can be regarded as a feature model catalogue (Box D). To be of use late for the configuration of software product lines, all software features of a feature model shall provide consistent information. Thus, software features shall be characterized according to the catalogue structure (Box B). The catalogue is introduced in Sect. 5.4.

5.2 Step 2: Feature Model Validation

The validation of feature models describes the process of testing the impact software features have on the online trustworthiness assessment and trust building. Since features are related to the trustworthiness assessment due to their facets, the validation involves testing to what extent users really associate the allocated trustworthiness facets with the software features. The feature model validation takes place after model creation (Step 2, Fig. 2). In order to realize validation testing, we refer to Arnowitz et al. [37]. They propose prototyping as an approach to let people experience single features in usability tests. Afterwards, participants can rate related trustworthiness facets on appropriate scales. For some trustworthiness facets, scientific scales already exist, as for example for ability, benevolence, integrity, and predictability [17]. Future work needs to support usability testing of those facets for which no scientific scales exist yet. Based on the answers of user ratings, quantitative attribute values can be calculated by which the impact of the features through the various facets is comparable (Box G). The attribute values shall be added within the feature model.

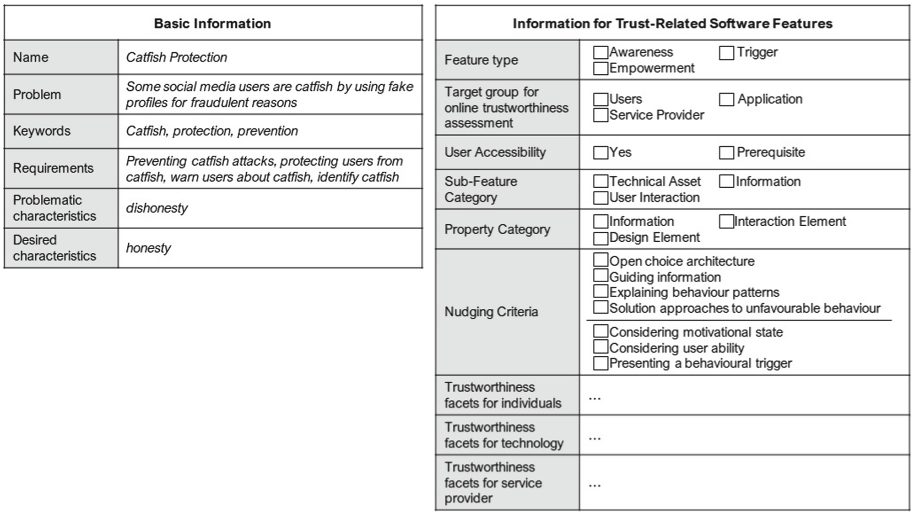
For validation reasons, additional attributes are useful for measuring the success rate of software features according to their feature type. For awareness features, *user awareness* is a suitable attribute to measure the feature's impact on how aware users are about the relevance of the trustworthiness assessment. Regarding trigger features, the *conversion rate* for trustworthiness assessments is interesting to know. It represents whether users really performed a trustworthiness assessment after interacting with the trigger feature. Thereby, it can be tested to what extent a trigger feature is convincing. For empowerment features, their *usefulness* to actually assess the trustworthiness of others is an indicator how well the system supports the online trustworthiness assessment.

5.3 Step 3: Configuration of Software Product Lines

The last step of the method is the configuration of a software product line by using the feature models for online trustworthiness assessment (Step 3). The configuration can be performed by either using the catalogue structure (Box B) for selecting software features (Step 3.a) or by considering the tree structure of the feature models (Step 3.b). For the catalogue, the preparatory work (Step 0) and the list of trustworthiness facets for the concept feature of a model (Box E) serve as input to consider trustworthiness facets during configuration. Concerning the tree structure, the notation of feature models provides the software engineer a decision basis what software features are mandatory or optional. The output of the configuration is a tailored software product line (Box H).

5.4 Feature Model Catalogue

As mentioned before, a feature model catalogue contains reusable, tailored solution approaches to specific trust problems of an application. As software features may be additionally modelled in terms of conveying a brand image via the application, a catalogue provides input for realising business strategies of the respective service provider. To have consistent information about the software feature available, software engineers shall follow the catalogue structure in Fig. 3 during



Basic Information	
Name	*Catfish Protection*
Problem	*Some social media users are catfish by using fake profiles for fraudulent reasons*
Keywords	*Catfish, protection, prevention*
Requirements	*Preventing catfish attacks, protecting users from catfish, warn users about catfish, identify catfish*
Problematic characteristics	*dishonesty*
Desired characteristics	*honesty*

Information for Trust-Related Software Features	
Feature type	☐Awareness ☐Trigger ☐Empowerment
Target group for online trustworthiness assessment	☐Users ☐Application ☐Service Provider
User Accessibility	☐Yes ☐Prerequisite
Sub-Feature Category	☐Technical Asset ☐Information ☐User Interaction
Property Category	☐Information ☐Interaction Element ☐Design Element
Nudging Criteria	☐Open choice architecture ☐Guiding information ☐Explaining behaviour patterns ☐Solution approaches to unfavourable behaviour
	☐Considering motivational state ☐Considering user ability ☐Presenting a behavioural trigger
Trustworthiness facets for individuals	...
Trustworthiness facets for technology	...
Trustworthiness facets for service provider	...

Fig. 3. Catalogue structure for feature models for trustworthiness assessments for the example of catfishing.

modelling (Step 1.1). The structure is divided into basic information and information for trust-related software feature. Based on this information, software features can later be identified or searched within the catalogue. The catalogue structure may be extended by further characteristics than proposed here. The basic information in Fig. 3 is already applied to the example of catfishing from Sect. 6.

The *basic information* of a catalogue aims at the concept feature of a model. Basic information includes the *name* of the concept feature as well as a description of the *problem* that the concept feature addresses. *Keywords* provide an overview about the issue. Furthermore, the software *requirements* that were determined by preparatory work (Step 0) shall be added to document what the concept feature realises. In addition, basic information includes *problematic* and *desired characteristics*, which relate to the identification of trustworthiness facets described in Sect. 2.2.

The second part of the catalog structure contains *information for trust-related software features*, which are presented in the form of characteristics to be checked. Several characteristics of an information category can be applicable at the same time. The information category *feature type* refers to the three software features types awareness, trigger or empowerment from Sect. 3. Moreover, software engineers can select the *target group for the online trustworthiness assessment* for which a feature is intended. The target group can be users, the application or the service provider. Another information category is *user accessibility*. Features can be distinguished between being either user-accessible or being prerequisites for another feature to be user-accessible, such as underlying algorithms of an user interface element. Further information categories are *sub-*

feature and *property category*. Sub-features can be categorized as a technical asset (e.g., algorithm), information (e.g., user data), or user interaction (e.g. confirmation request). Properties can be related to information (e.g., telephone number), interaction element (e.g., button), or design element (e.g., graphical symbol). The information category *nudging criteria* refers to the definitions of nudges and persuasive technologies from Sect. 2.3. Features may comply to the nudging criteria of an open choice architecture, guiding information, explanations of user behaviour patterns or solution approaches to unfavourable behaviour. In terms of persuasive technologies, features may consider users' motivational state, consider users' ability for the targeted behaviour or present a trigger to act in accordance to the target behaviour. The last three information categories of the catalogue structure are the trustworthiness facets for individuals, technology and the service provider. For these, the list of trustworthiness facets of a concept feature (Box E) serve as input. Software engineers may choose facets in which they are interested.

6 Example: "Identity Verification" for Catfish Protection

For demonstrating feature models for online trustworthiness assessments, we chose the scenario of catfishing in online dating. Catfishing is a phenomenon, where online dating users, known as catfish, create user profiles with fake identities for fraudulent reasons [38]. It is a suitable example, because catfish are reason for trust issues among users. Online trustworthiness assessments help users to identify catfish and resolve their concerns. Based on this knowledge, we filled out the basic information of the catalogue structure from Sect. 5.4, Fig. 3. Catfishing is the problem that shall be tackled by catfish protection. Suitable keywords are catfish, protection and prevention. Software requirements for catfish protection can be to prevent catfish attacks, to protect users from catfish, to warn users about them or to identify catfish.

In the following, we demonstrate how the feature models can be used for catfish protection by the feature "identity verification". Identity verification helps to resolve the uncertainty whether another user has created a fake profile [39]. It is known to be an interactive tool for self-presentation, which increases users' reputation and allows them to rate the trustworthiness of other users. Furthermore, it is combined with persistent labelling in a user profile on which basis users can derive whether the identity is verified. Therefore, we classify identity verification as an empowerment feature to perform a trustworthiness assessment. Due to space limitations, the model can be extended by further features. We only explain parts of it.

6.1 Example: Feature Model Creation

Identity verification is introduced as an empowerment feature after the concept feature on the second layer (see labelling <<empower>>, Fig. 4). In the third layer of the model, we refine identity verification in three mandatory features -

a verification algorithm, user profile and the notification about the verification status. Verification algorithms most often try to link a user profile to additional identifying information. Therefore, we connect the verification algorithm with the user profile by a require-link. The verification algorithm has three properties: “photo of ID card”, “phone number”, and “Facebook account”. They represent the additional information that may be used for the verification of the user profile. By the OR-link, the model depicts that the algorithm needs to consider at least one of the options.

If a system has the software requirement to inform its users about the verification status of other users, a notification about the status is a suitable feature to address this requirement. In order to realize notifications, knowledge about the verification status is required. Therefore, we use the require-link to connect the feature notification with the feature verification algorithm. As a next step, we identify three different verification statuses, that are “verified”, “not completed” and “fake” (see right sub-tree, layer four of model). For determining features how to express the statuses, we follow the principle of familiarity. It says that the frequency with which users have already encountered design elements like symbols lead to an increased usability and understanding [40]. Therefore, we choose features that are frequently used by other applications and are thus already well-known to users in their appearance and meaning. For the verified status, a graphical symbol in form of a green check mark next to the name of a user profile can be used. Online dating applications such as Tinder[1] already use this symbol for verified profiles. In case of uncompleted identity verification processes, we want to emphasize this by an orange question mark next to the name of a profile. This symbol is inspired by the green check mark described before. If a fake identity has been identified by the verification algorithm, the catfish should no longer be available for matching. Furthermore, users should no longer be able to interact with catfish when they already had a match. To visualise the inactivity of a catfish profile, the profile and the corresponding chat are presented greyed out for the other users.

6.2 Example: Catalogue Information

As part of feature modeling and refinement, the software engineer has to assign trust-related software feature characteristics from the catalogue structure to each feature. For this example, we demonstrate this for the green check mark feature. Belonging to the identity verification feature, the green check mark is an empowerment feature. It is illustrated on the graphical user interface so that users can assess the trustworthiness of other users. Therefore, it is user-accessible. Being a property, the green check mark can be categorized as a design element and information, because it conveys the message that another user has passed the identity verification. Concerning the nudging criteria, the green check mark provides information that may guide user behaviour. For trustworthiness facets that are associated with the green check mark, we refer to Sect. 6.3.

[1] www.tinder.com.

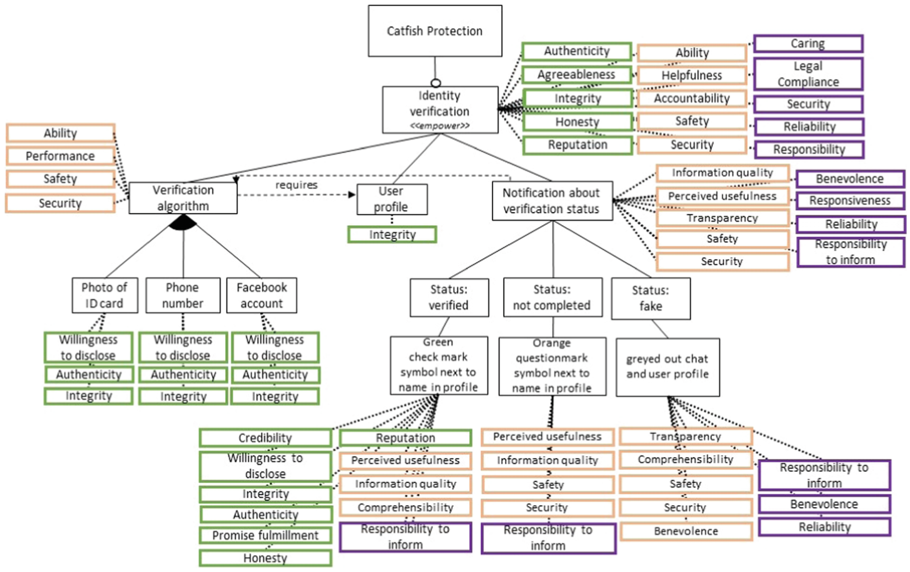


Fig. 4. Feature model for the empowerment feature “identity verification” for catfish protection after the allocation phase. (Color figure online)

6.3 Example: Trustworthiness Facet Attribution Process

After the model has been established, we proceed with the trustworthiness facet attribution. For the allocation phase, we start with the empowerment feature “identity verification”. Following the guideline for selecting appropriate trustworthiness facets [19], we first acquire an understanding of the actual problem to which identity verification serves as a solution approach. As mentioned before, identity verification shall resolve users’ concerns about fake profiles by proving that an identity is true. Thinking of problematic characteristics of a catfish, catfish are dishonest. We add this information to the catalogue structure for catfish protection (see Fig. 3). Catfish are likely to not perform an identity verification to hide the fraud. Therefore, they would not comply to the application’s norms in absolving a verification. Based on this problematic characteristic, we check the overview of trustworthiness facets [19] on semantically opposite trustworthiness facets by definition (Box C, Fig. 2). As a result, we assume that users, who perform an identity verification are associated with authenticity, agreeableness, integrity with the norms of the application, honesty and good reputation (green frames, second layer, Fig. 4).

As a next step, we consider how including identity verification in the system might impact users’ perception of the online dating application. Former research has pointed out that websites should be interested in taking the responsibility for their users’ safety and security [41]. If the feature was not implemented, users might feel insecure and not well supported. Having this in mind, we check the overview of trustworthiness facets for technology [19] (Box C). Our findings are depicted as the attributes of identity verification in the orange boxes in Fig. 4. By having identity verification implemented, the online dating application presents its ability to address the problem. Furthermore, the application thereby helps its users in countering their catfish concern. In addition, it shows that it is accountable and takes care of the safety and security of its users.

Last but not least, we conclude the trustworthiness facets of the service provider if identity verification was implemented in the online dating application. They are depicted as the attributes of identity verification framed in purple in Fig. 4. The facets are in accordance to the ones for the online dating application as a technology (see Fig. 2 attributes of identity verification, orange frames). We came to the conclusion that service providers express their care for users' safety and security when using identity verification. Furthermore, catfishing has been discussed in court concerning online impersonations [41]. Therefore, service providers would demonstrate their responsibility and legal compliance.

For the rest of the feature model, we proceed in a similar way. Due to space constraints, we do not further explain the following steps. Yet, we want to mention that for the statuses on layer four of the feature model (see right sub-tree in Fig. 4, we have not added trustworthiness facets as attributes. Here, we regard them as specifications of their parent feature, which are expressed in detail by their child features. Therefore, we limit the facet allocation on the child features for this specific case. Figure 4 shows the feature model after allocation phase (Step 1.1, Fig. 2).

After the allocation phase, the propagation phase can be performed. Trustworthiness facets that are not yet allocated to parent features are now propagated. This is for example the case for "reputation" from the green check mark feature, which is added as an attribute to the notification feature.

As an outlook for the configuration process of this feature model, the tree structure of the model points out to the optional properties of the verification algorithm. The software engineer needs to determine whether the ID card, phone number, Facebook account or a combination of them shall be used for checking a match with the user profile.

7 Related Work

As previously introduced, the model for the design of nudges (DINU model) [26] provides a catalogue of existing nudges that can be used as input for the method presented in this work. Additionally, the DINU model relates to our work insofar that it guides practitioners in the analysis, design, and evaluation of nudges and their context. In doing so, it focuses on the nudging criteria that we included in the catalogue structure (see Fig. 3). Compared to our work, the DINU-model misses the model-based approach of feature models. It not only allows software engineers to support users in their online trustworthiness assessment but also to validate trust-related software features in a structured way for software product line engineering.

Another related work is from Martinez et al. [42], who invented a feature model tool to select appropriate features based on their attributes. For that, they introduce algorithms on the basis of petri nets. Similar digital tools for configuring software product lines based on feature models are the FeAture Model Analyser (FAMA) [43] or Requiline - a requirements engineering tool for software product lines [44]. These tools are missing the interdisciplinary trust background, but are valuable for complementing our method. By combining such tools with our catalog structure, configurations of trust-related software product lines can be automated.

8 Discussion

This work introduces a method for software engineers that is based on extended feature models. Its intend is to create reusable catalogues for software features that focus on online trustworthiness assessments. Online trustworthiness assessments are especially relevant for social media users of computer-mediated introductions (CMIs). Based on the assessment, CMI users decide whether to tolerate risks associated with the interaction of other parties such as unknown users, the CMI platform and the CMI service provider. However, users are most often not aware of the assessment's importance or find it too difficult to perform. Therefore, this work uses adapted feature models to derive awareness, trigger and empowerment features for trustworthiness assessment. Feature models are adapted by adding trustworthiness facets as attributes to software features for considering them in the features' specification and design.

Applying the adapted feature models for online trustworthiness assessment has shown that they are suitable to derive and specify awareness, trigger and empowerment features. For each of the features, a huge range of different interaction elements can be considered for realizing associated software requirements. Another key element of the adapted feature models is the large collection of trustworthiness facets related to each software feature. The model further differentiates between the trustworthiness facets of the CMI parties "user", "platform" and "service provider". Thereby, software engineers can ensure to implement cues that foster online trustworthiness assessments regarding all three parties. Catalogues of such feature models lead to comprehensive solution approaches for specific problems and reflect a variety of design options.

Overall, extending feature models by trustworthiness facets provides a solid basis for validating the trustworthiness assessment and trust building. However, the trustworthiness facets may impact more constructs than trust building, which could be taken into consideration for the validation process as well. An example could be the halo effect. The halo effect is a cognitive bias that describes an error in reasoning based on one known trait leading to an overall impression of further traits [45]. In terms of the facet allocation phase for example, a software engineer could assign the facet benevolence to a feature and, based on that alone, simultaneously associate the feature with the facet usefulness. Future work needs to consider the halo effect of trustworthiness facets in the validation process from two perspectives. How does the halo effect may have impacted the software engineer to identify irrelevant facets during the facet allocation phase? How might the halo effect impact CMI users to assess further trustworthiness facets by being exposed to a software feature? Maybe some of the trustworthiness facets impacted by the halo effect have not even been identified by the software engineer in the facet attribution process.

Unidentified trustworthiness facets pose another challenge for validation. Currently, the validation process checks on facets that have been identified by the software engineer in the facet attribution process. Future work needs to consider how feature models can be validated in terms of unidentified facets that are nonetheless relevant for software features.

Unidentified trustworthiness facets and the halo effect point to a limitation of the method introduced here. Feature model creation is subject to the subjectivity of the software engineer applying the method. In order to reduce mistakes, we propose to perform the method in an agile team. Agile methods increase the flexibility and efficiency of software development [46]. Thereby, software features and trustworthiness facets can be easily discussed within the team and changes can be done throughout the whole method. In addition, future work needs to validate how the method is accepted by practitioners. By empirical studies, practitioners might give feedback on how the method can be further improved.

Another aspect that future research needs to tackle regarding the trustworthiness facets is the difficulty to distinguish between facets of the CMI platform and of the CMI service provider. During the facet allocation phase, we recognized the similarity of resulting facets for both types. A reason for that might be that users are oftentimes affected by brand image when it comes to their perception of the software application [47]. Future work could examine the relationship between users' perception of these two facet types. This in turn could provide insights for the facet allocation phase and support engineers in performing it.

In terms of configuration, a main challenge of Software Product Line Engineering is handling the variability of a model [48]. Although validated trustworthiness facets support software engineers in improving users' online trustworthiness assessment, it increases the complexity of configuration. Software feature tools like FAMA [43] or Requiline [44] (see Sect. 7) could aid software engineers in this process of systematically selecting those software features that address desired trustworthiness facets. However, future work might focus on the questions whether including as much trustworthiness facets as possible within software enhances online trustworthiness assessments or whether configuring a certain set of facets provides a better support.

The selection of certain sets of features can further be related to the question of diversity and commonality. Finding a balance between the diversity and commonality of software product lines is tackled in variability management [33]. It is about weighing the reduction of complexity within the product for easy usage and the differentiation of a provider's products from competitors based on underlying business strategies [33]. Future work needs to examine whether a balance of variability and commonality is crucial for trust-related software features as well. Usability testing could be a useful approach for finding answers.

Overall, the great scope of trustworthiness facets as attributes of software features allows engineers to model various options of how to support users in their trustworthiness assessment. Yet, drawbacks can be observed in the overwhelming size of models based on the large amount of features and facets. Handling large-scale variability models is a well-known challenge in Software Product Line Engineering [48]. To counter the problem of cluttered feature models, we again propose to use digital tools as FAMA [43] or Requiline [44] for model creation and configuration. These tools support operators to keep an overview.

9 Conclusion

This work focuses on how software engineers can best support social media users in assessing the trustworthiness of other users, the application, and the service provider by means of software features. For that reason, three different types of software features were identified: awareness, trigger and empowerment features. In order to document such software features for specific problems of social media use, feature models were extended by trustworthiness facets to address users' trust building. Feature models for online trustworthiness assessments can serve as reusable catalogues in order to consider and validate different design options and interaction elements of trust-related software feature. Furthermore, they can be used for the configuration of software product lines in social media. For future work, additional validation techniques need to be developed to evaluate the extent to which software features address trustworthiness facets and impact users' trustworthiness assessments.

References

1. Obada-Obieh, B., Somayaji, A.: Can I believe you? Establishing trust in computer mediated introductions. In: Proceedings of the 2017 New Security Paradigms Workshop, pp. 94–106 (2017)
2. Jozsa, K., Kraus, A., Korpak, A.K., Birnholtz, J., Moskowitz, D.A., Macapagal, K.: "Safe behind my screen": adolescent sexual minority males' perceptions of safety and trustworthiness on geosocial and social networking apps. Arch. Sex. Behav. **50**(7), 2965–2980 (2021). https://doi.org/10.1007/s10508-021-01962-5
3. Yi, J., Yuan, G., Yoo, C.: The effect of the perceived risk on the adoption of the sharing economy in the tourism industry: the case of Airbnb. Inf. Process. Manage. **57**(1), 102–108 (2020)
4. Couch, D., Liamputtong, P.: Online dating and mating: perceptions of risk and health among online users. Health Risk Soc. **9**(3), 275–294 (2007)
5. Son, J.Y., Kim, S.S.: Internet users' information privacy-protective responses: a taxonomy and a nomological model. MIS Q. **32**, 503–529 (2008)
6. Hang, L., Kim, D.H.: SLA-based sharing economy service with smart contract for resource integrity in the internet of things. Appl. Sci. **9**(17), 3602 (2019)
7. Becerra, M., Lunnan, R., Huemer, L.: Trustworthiness, risk, and the transfer of tacit and explicit knowledge between alliance partners. J. Manage. Stud. **45**(4), 691–713 (2008)

8. Lewicki, R.J., Wiethoff, C.: Trust, Trust Development, and Trust Repair. The Handbook of Conflict Resolution: Theory and Practice **1**(1), 86–107 (2000)
9. Bialski, P., Batorski, D.: From online familiarity to offline trust: how a virtual community creates familiarity and trust between strangers. Social Computing and Virtual Communities, pp. 179–204 (2010)
10. Bonnefon, J.F., Hopfensitz, A., De Neys, W.: The modular nature of trustworthiness detection. J. Exp. Psychol. Gen. **142**(1), 143 (2013)
11. Ding, S., Yang, S.L., Fu, C.: A novel evidential reasoning based method for software trustworthiness evaluation under the uncertain and unreliable environment. Expert Syst. Appl. **39**(3), 2700–2709 (2012)
12. Borchert, A., Díaz Ferreyra, N.E., Heisel, M.: Building trustworthiness in computer-mediated introduction: a facet-oriented framework. In: International Conference on Social Media and Society, pp. 39–46 (2020)
13. Cassell, J., Bickmore, T.: External manifestations of trustworthiness in the interface. Commun. ACM **43**(12), 50–56 (2000)
14. Mishra, A.K.: Organizational responses to crisis. Trust in organizations. Front. Theor. Res. **3**(5), 261–287 (1996)
15. Mcknight, D.H., Carter, M., Thatcher, J.B., Clay, P.F.: Trust in a specific technology: an investigation of its components and measures. ACM Trans. Manage. Inf. Syst. **2**(2), 1–25 (2011)
16. Mayer, R.C., Davis, J.H., Schoorman, F.D.: An integrative model of organizational trust. Acad. Manag. Rev. **20**(3), 709–734 (1995)
17. Büttner, O.B., Göritz, A.S.: Perceived trustworthiness of online shops. J. Consum. Behav. **7**(1), 35–50 (2008)
18. McKnight, D.H., Chervany, N.L.: What trust means in e-commerce customer relationships: an interdisciplinary conceptual typology. Int. J. Electron. Commer. **6**(2), 35–59 (2001)
19. Borchert, A., Heisel, M.: The role of trustworthiness facets for developing social media applications: a structured literature review. Information **13**(1), 34 (2022)
20. Hsi, I., Potts, C.: Studying the evolution and enhancement of software features. In: icsm, p. 143 (2000)
21. Anton, A.I.: Goal identification and refinement in the specification of software-based information systems. Georgia Institute of Technology (1997)
22. Glinz, M.: On non-functional requirements. In: 15th IEEE International Requirements Engineering Conference, pp. 21–26. IEEE (2007)
23. User Interface Design Patterns. www.cs.helsinki.fi/u/salaakso/patterns/. Accessed 11 Apr 2022
24. Welie.com - Patterns in Interaction Design. www.welie.com/patterns/index.php. Accessed 11 Apr 2022
25. Zetterholm, M., Elm, P., Salavati, S.: Designing for pandemics: a design concept based on technology mediated nudging for health behavior change. In: 54th Hawaii International Conference on System Sciences, pp. 3474–3483 (2021)
26. Meske, C., Potthoff, T.: The DINU-model-a process model for the design of nudges (2017)
27. Acquisti, A., et al.: Nudges for privacy and security: understanding and assisting users' choices online. ACM Comput. Surv. **50**(3), 1–41 (2017)
28. Fogg, B.J.: A behavior model for persuasive design. In: Proceedings of the 4th International Conference on Persuasive Technology, pp. 1–7 (2009)
29. Thaler, R.H., Sunstein, C.R.: Nudge: Wie man kluge Entscheidungen anstößt. Ullstein eBooks (2009)

30. Sunstein, C.R.: Nudging: a very short guide. J. Consum. Policy **37**(4), 583–588 (2014)
31. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (FODA) feasibility study. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst. (1990)
32. Czarnecki, K., Eisenecker, U.W.: Generative programming (2000)
33. Pohl, K., Böckle, G., Van Der Linden, F.: Software product line engineering: foundations, principles, and techniques, vol. 1. Springer, Heidelberg (2005)
34. Riebisch, M.: Towards a more precise definition of feature models. Model. Variability Object-Oriented Prod. Lines 64–76 (2003)
35. Benavides, D., Trinidad, P., Ruiz-Cortés, A.: Automated reasoning on feature models. In: Pastor, O., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 491–503. Springer, Heidelberg (2005). https://doi.org/10.1007/11431855_34
36. Xiong, J.: New Software Engineering Paradigm Based on Complexity Science: An Introduction to NSE. Springer, NY (2011). https://doi.org/10.1007/978-1-4419-7326-9
37. Arnowitz, J., Arent, M., Berger, N.: Effective Prototyping for Software Makers. Elsevier (2010)
38. Simmons, M., Lee, J.S.: Catfishing: a look into online dating and impersonation. In: Meiselwitz, G. (ed.) HCII 2020. LNCS, vol. 12194, pp. 349–358. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49570-1_24
39. Kaskazi, A.: Social network identity: Facebook, Twitter and identity negotiation theory. In: iConference 2014 Proceedings (2014)
40. Mcdougall, S.J., Curry, M.B., De Bruijn, O.: Measuring symbol and icon characteristics: norms for concreteness, complexity, meaningfulness, familiarity, and semantic distance for 239 symbols. Behav. Res. Meth. Instrum. Comput. **31**(3), 487–519 (1999). https://doi.org/10.3758/BF03200730
41. Koch, C.M.: To catch a catfish: a statutory solution for victims of online impersonation. U. Colo. L. Rev. **88**, 233 (2017)
42. Martinez, C., Díaz, N., Gonnet, S., Leone, H.: A Petri net variability model for software product lines. Electron. J. SADIO (EJS) **13**, 35–53 (2014)
43. Benavides, D., Segura, S., Trinidad, P., Cortés, A.R.: FAMA: tooling a framework for the automated analysis of feature models. VaMoS (2007)
44. von der Maßen, T., Lichter, H.: RequiLine: a requirements engineering tool for software product lines. In: van der Linden, F.J. (ed.) PFE 2003. LNCS, vol. 3014, pp. 168–180. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24667-1_13
45. Thorndike, E.L.: A constant error in psychological ratings. J. Appl. Psychol. **4**(1), 25 (1920)
46. Campanelli, A.S., Parreiras, F.S.: Agile methods tailoring-a systematic literature review. J. Syst. Softw. **110**, 85–100 (2015)
47. Yang, T., Bolchini, D.: Branded interactions: predicting perceived product traits and user image from interface consistency and visual guidance. Interact. Comput. **26**(5), 465–487 (2014)
48. Metzger, A., Pohl, K.: Software product line engineering and variability management: achievements and challenges. In: Future of Software Engineering Proceedings, pp. 70–84 (2014)