Automating the Formalization of Product Comparison Matrices

Guillaume Bécan, Nicolas Sannier, Mathieu Acher, Olivier Barais, Arnaud Blouin, Benoit Baudry

Automating the Formalization of Product Comparison Matrices

Guillaume Bécan, Nicolas Sannier, Mathieu Acher, Olivier Barais, Arnaud Blouin and Benoit Baudry
Inria - IRISA
Campus de Beaulieu
35042 Rennes cedex, France
guillaume.becan@irisa.fr, nicolas.sannier@inria.fr, mathieu.acher@irisa.fr, olivier.barais@irisa.fr, arnaud.blouin@irisa.fr benoit.baudry@inria.fr

ABSTRACT

Product Comparison Matrices (PCMs) form a rich source of data for comparing a set of related and competing products over numerous features. Despite their apparent simplicity, PCMs contain heterogeneous, ambiguous, uncontrolled and partial information that hinders their efficient exploitations. In this paper, we formalize PCMs through model-based automated techniques and develop additional tooling to support the edition and re-engineering of PCMs. 20 participants used our editor to evaluate the PCM metamodel and automated transformations. The results over 75 PCMs from Wikipedia show that (1) a significant proportion of the formalization of PCMs can be automated – 93.11% of the 30061 cells are correctly formalized; (2) the rest of the formalization can be realized by using the editor and mapping cells to existing concepts of the metamodel. The automated approach opens avenues for engaging a community in the mining, re-engineering, edition, and exploitation of PCMs that now abound on the Internet.

Categories and Subject Descriptors

D.2.13 [**Software Engineering**]: Reusable Software; D.2.9 [**Software Engineering**]: Management—*Software configuration management*

Keywords

Metamodeling; Product comparison matrices; Domain analysis; Automated transformation

1. INTRODUCTION

In her book "The Art of Choosing", Sheena Iyengar discusses cultural and social factors that lead people to choose their products among many others [24]. In this endeavor of choosing, people rely on their own experiences, dedicated expert reviews, or even general product comparison materials. One common representation that can help to make a choice is to visualize all the products characteristics through a matricial representation, the so-called *Product Comparison Matrices (PCMs)* (see Figure 2 for an example).

The intrinsic *theoretical* good properties of such matrices are notably: (1) simplicity: no particular training is required to understand how it works; (2) synthesis: all the "expected" information is present, without any verbose claim; (3) easy to write and define from existing sources; (4) widely used; (5) independent from any particular domain or support.

Though apparently simple, synthetic, or easy to design, PCMs hide, in *practice*, an important complexity while expressing commonalities and variabilities among products [36, 37]. PCMs can be seen as a special form of spreadsheets and thus share some of their problems [1, 2, 7, 9, 11, 12, 14, 15, 19–23, 32, 35]. Specifically, the underlying reasons of the complexity of PCMs are: (1) ambiguity: PCMs are mainly written in uncontrolled natural language, mixing scopes, granularity, and heterogeneous styles; (2) lack of scalability: as a PCM grows up, its readability dramatically decreases; (3) too much equality: all the information is equally presented as textual assets; (4) lack of support and services: the previous points are emphasized by the limited number of advanced services tackling these limits.

A systematic engineering approach with dedicated tools is needed to improve the current practice of editing, maintaining, and exploiting PCMs. Our initial work [36] proposed a preliminary analysis of PCMs and characterized the nature of the problem. Yet the key challenge remains: providing a more structured and formal expression of PCMs.

The general problem of transforming raw data to formal model has a long tradition [34]. In this paper, we address three research questions (see Figure 1): **(RQ1)** How to formalize data that are contained in large and uncontrolled natural language PCMs? **(RQ2)** How to automatically transform PCMs into more formal representations? **(RQ3)** What tools and services can be built on top of this formalization?

Our approach is to take the *specificities* of PCMs into account – redundant tricks and constructs for documenting supported features of products in a PCM are empirically identified and formalized into a domain-specific modeling language (a metamodel). Then, automated techniques are developed to encode raw data as models conformant to the metamodel. A series of tools are built on top of the metamodel to (1) ease the re-engineering of PCMs or (2) pro-

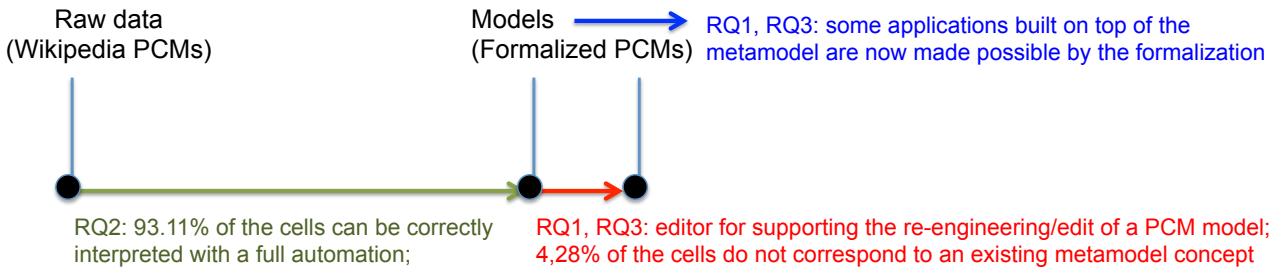

Figure 1: From Raw Data to Models of PCMs (Overview of the Contributions and the Evaluation)

vide services that are now made possible by the encoding of PCMs as models.

The contributions of this paper are as follows:

1. A metamodel that acts as a unifying canvas by proposing a more formal description of PCMs semantics with respect to their content but also to the variability information they contain;
2. A set of automated transformations and analyses that leverage the domain data of PCMs composed of various and uncontrolled concepts. These transformations allow to build from raw PCMs, corresponding PCM models that can be then enriched with further editing and validation services. In particular, we develop a comprehensive PCM editor;
3. An empirical evaluation for validating the two previous contributions. This evaluation is based on a set of 75 PCMs extracted from Wikipedia, a large and open source of general and heterogeneous PCMs. 20 participants use our PCM editor to evaluate the PCM metamodel and automated transformations. The results (see Figure 1) show that (1) a significant proportion of the formalization of PCMs can be automated – 93.11% of the 30061 cells are correctly formalized; (2) the rest of the formalization can be realized by using the editor and mapping cells to existing concepts of the metamodel. We also analyze qualitatively and discuss the missing concepts of the metamodel reported by the participants

We believe the automated approach is a significant step towards engaging a community in the mining, re-engineering, edition, and exploitation of PCMs that now abound on the Internet.

The remainder of the paper is organized as follows. Section 2 gives background information about PCM and formulates the research problem. Section 3, introduces our model-based approach for formalizing PCMs. Section 4 presents the results of our empirical evaluation. Section 5 discusses threats to validity. Section 6 discusses related work. Section 7 concludes the paper and presents future work.

2. FROM RAW DATA TO MODELS: THE CASE OF PCMS

In this section, we describe the current limits of *Product Comparison Matrices (PCMs)*. We first detail the complexity of analyzing the content of PCMs. Then, we explain that raw data representing PCMs offer a low level of formalization hindering their subsequent exploitation.

Heterogeneity and Ambiguity. Analyzing the information of a PCM leads to the problem of heterogeneity and ambiguity of the PCM content. PCM stakeholders can master PCMs while the number of features and products is limited. Figure 2 illustrates the ambiguity of content with the matrix that describes several features of different portable media players. We can observe very different kind of values within the cell, from boolean cells, unknown values, cells with comas, parentheses, references, non intuitive and ambiguous values (for instance "Clicker Only"). Heterogeneity and ambiguity traditionally pose a problem for computer-based approaches and tools. As a practical consequence, additional services such as product comparators often propose only a limited number of products/features under comparison as well as a very few advanced services such as features sorting, product ranking, or flexible comparison mechanisms (difference of pricing, dimensions, etc.) [37].

Lack of Explicit Concepts and Relationships. While one considers the matricial representation that characterizes PCMs as a table, he/she manually interprets the content as products, features, and interpret other cells as cross-references between products and the corresponding values for their features.

There is no such precision within current syntaxes like Excel spreadsheets, HTML tables or Wikipedia tables for instance. They focus on the layout and not on the semantics of PCMs. Figure 3 proposes a short snippet of mediawiki syntax, which is used for editing Wikipedia tables and matrices.

The Wikipedia Mediawiki syntax describes wikitables as a collection of rows containing cells. We observe that a wikitable has no knowledge of column and no relationship to cells that belong to one column. For instance, the second cell of a given row is not bound to the second cell of each other rows. They are only part of the same column because of a fortunate layout side effect. Similarly, the Wikitable syntax allows to define a cell as being a header without any constraint on its position.

The syntax is also complemented with a set of references and templates that will add some layout features (colored cells, logos, etc.) For instance the cell "{{Yes|?}}" is part of a template that will colored the cell in green ({{yes}}) with the verbatim "?".

Absence of Unified Formats. Writing such PCMs is a difficult task and many transformations from alternative formats (CSV, Excel, etc.) to Mediawiki have been created[1]. Though easier to edit, Excel spreadsheet suffer from the same limitations while expressing product × feature re-

[1] http://en.wikipedia.org/wiki/Wikipedia:Tools

Name	Navigation	Speaker	Screen size (in)	Screen type	Resolution (px)	Color depth (bpp)	Video out	Video battery life (hr)
Archos 105[1]	D-pad, 7 buttons	No	1.8	OLED	160 × 128	18	Component, Composite	Unknown
Archos 405[2][3]	D-pad, 6 two-way buttons	No	3.5	TFT LCD	320 × 240	Predecessor: 24	Component, Composite	5
Cowon Q5W[4]	Touchscreen	Yes	5	TFT LCD	800 × 480	24	Component, S-video, composite	7
Gigabeat T-Series[5]	D-pad, 4 buttons	No	2.4	TFT LCD	320 × 240	18	No	5
GP2X F-200[6]	8-way d-pad/touchscreen, 9 buttons	Yes	3.5	TFT LCD	320 × 240	16	Component	3.5
iPod classic[7]	Click wheel, 1 center, 4 embedded buttons	Clicker only	2.5	TFT LCD	320 × 240	16	Component, composite	6
iPod nano 6G[8]	Multi-touch screen	No	1.54	TFT LCD	240 × 240		Component, composite	No video support
iPod touch[9]	Multi-touch screen, 3 buttons	Yes (ex. 1st gen.)	3.5	TFT LCD	480 × 320 (4th gen. 960 × 640)	24 (4th gen.) 18 (1st. - 3rd gen.)	Component, composite	6

Figure 2: PCM of Wikipedia about Portable Media Players

```
{| center;" class="wikitable sortable"
|-
! Program
! License
! Simultaneous User Capacity
! Linux
[[Image:Tux.svg|25px|Linux]]
! Mac OS X
...
! Recording capabilities
|-
| {{rh}} | [http://www...]
| [[Proprietary software license|Proprietary]]
| {{Sort|0000500|1-1500 (80,000 w/webcast)}}
| {{Yes|?}}
|...
| {{Yes|?}}<ref name="ReferenceA">
Supports two-way ... integration</ref>
| {{Yes|}}VGA, HQ, HD<ref>[http://www.ad... ]
Retrieved on 2014-02-27.</ref>
| {{Yes|?}}
...
| {{No|X}}
|-
```

Figure 3: Code Snippet of a Wikipedia Table (Mediawiki Syntax)

lationships. Cells are associated to a column and a row, while cells are arbitrary and manually interpreted as features, products or an associated valued cells. As a consequence, PCMs lack a proper description language that can precisely define the concepts and the relationships they contain. Such language would ease the disambiguation and the analysis capabilities on top of more formalized PCMs.

Large Dataset of PCMs. Moreover, heterogeneous, complex and ambiguous PCMs are legions on the internet, independently from the domain or the products they aim at describing. It does not only concern open initiatives like Wikipedia: our previous work [37] reported similar issues for PCMs of commercial tools and services (i.e., comparators and configurators).

Problem Statement. While analyzing a Wikipedia PCM, there is gap between the concrete source code representation of a PCM, and the human manual interpretation of this PCM. This gap is both syntactic and semantic. Providing analysis capabilities upon such PCMs requires more formalization and the integration of parts of the common knowledge that is, for now, handled by users who interpret these tables.

The problem impacts three kinds of users:

- data writers (e.g. Wikipedia contributors): how to add a new product entry when everything around is heterogeneous and there is no standard way to edit data? How to give a proper structure and semantics to the data?
- developers (e.g. data scientists or product analysts) in charge of processing, transforming, and analyzing PCMs;
- end-users that want to understand or interact with PCMs. For instance, in Figure 2, how to filter media players that do support a specific screen type, say TFT LCD, and a screen size greater than 3?

Two challenges arise:

- **metamodeling** (1) to abstract from heterogeneity, reinforce structure and give a clear semantics to data; (2) to enable the specification of raw data transformation into models conformant to a metamodel;
- **model transformations** to automatically (1) parse and encode data into a suitable format, despite heterogeneity (2) normalize data to give a proper semantics (3) devise new editing tools.

The research problem we address in this paper can be summarized as follows: *How to automate the encoding of heterogeneous, ambiguous and large-scale data of PCMs into well-structured, well-typed and well-formalized models?*

3. AUTOMATING THE FORMALIZATION OF PCMS

3.1 Model-based Approach

This section presents our automated model-based approach for formalizing PCMs. Figure 4 provides an overview of our approach, from raw data of PCMs (e.g., as expressed in Wikipedia) to models conforming to a metamodel.

The heterogeneity of data contained in PCMs, supplemented with syntax and semantics ambiguities, requires both a formalization canvas and associated analyses. For this purpose, which corresponds to our RQ1, we first propose a PCM metamodel (see ① of Figure 4). Metamodels provide a definition for the main concepts of a domain and their properties as well as the organization of these concepts by providing a set of associations.

Using this metamodel, we can develop a transformation chain (see ②, ③, and ④ of Figure 4) for producing PCM models. The precision of the tool chain corresponds to our

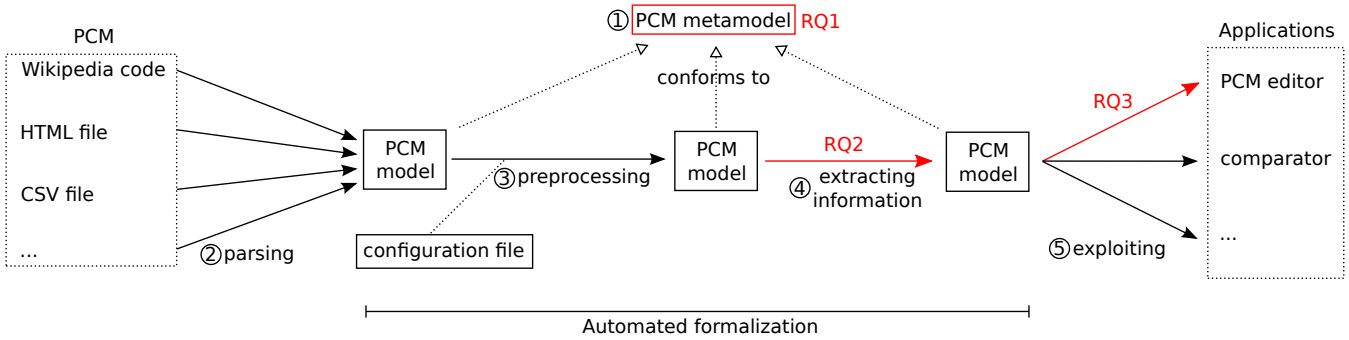

Figure 4: Global View of our Automated Formalization of PCMs

RQ2 and is described in Section 4. We further discuss the metamodel and the transformation chain in the two following subsections. The exploitations of PCM models (see ⑤ of Figure 4) are described in Section 4 and corresponds to our RQ3.

3.2 The PCM Metamodel

Figure 5 presents the PCM metamodel we defined as our unifying canvas. This metamodel was obtained while observing various PCMs either on the internet or real ones in magazines or shops and discussions all along the past year.

This metamodel describes both the structure and the semantic of the PCM domains. In this metamodel, PCMs are not individual matrices but a set of different matrices that contain cells. This happens when comparing a large set of products or features. In order to preserve readability, PCM writers can split the PCM content into several matrices. Cells can be of 3 types: *Header*, *ValuedCell*, and *Extra*. Header cells identify products or features.

In our metamodel, the structure of the PCM is not led by rows or columns but with explicit concepts of products and features. These products (resp. features) can have a composite structure that is used when describing several level of granularity for these products (resp. features) and which are usually represented as product (resp. features) row or column spans.

In Excel or Wikipedia, cell values are associated to products and features because of their relative and fortunate positions. In our approach we have explicit associations between a cell and its related product and feature. In addition, we keep the syntactic layout with the row and column attributes in the *Cell* class.

On the semantic side, PCM express commonalities and differences between products. As a consequence, formalizing such domains necessarily requires to introduce some concepts from the variability and product line engineering community but also to introduce new ones.

We have two main concepts: the *Constraint* class that represents the interpretation of the information contained in a valued cell and the *Domain* class that define the possible values for a feature.

The interpretation of a valued cell is given according to different patterns and information types defined as sub-concepts of *Constraint* in the metamodel:

- Boolean: states that the feature is present or not,
- Integer: integer numbers,
- Double: real numbers,
- VariabilityConceptRef: references a product or a feature,
- Partial: states that the feature is partially or conditionally present,
- Multiple (And, Or, Xor): composition of values constrained by a cardinality,
- Unknown: states that the presence or absence of the feature is uncertain,
- Empty: the cell is empty,
- Inconsistent: the cell is inconsistent with the other cells bound to the same feature

The domain a of feature is represented as a set of *Simple* elements (*Boolean, Integer, Double* or *VariabilityConceptRef*) which defines the valid values for the cells that are related to this feature. The concept of domain allows us to detect invalid values and reason on discrete values such as features but also use the properties of boolean, integers and real values for ranking or sorting operations.

3.3 The Transformation Toolsuite

Having a formalizing canvas with a metamodel is only one mean to a larger end. Formalizing PCMs in their whole diversity and heterogeneity requires a set of transformations steps. These steps include:

- parsing: extracting the PCM from its original artefact (e.g. MediaWiki code)
- preprocessing: normalizing the PCM
- extracting information: interpreting cells and extracting variability concepts and features' domain

The parsing and preprocessing steps aim to handle the complexity from the syntax while the extracting step aim to handle the complexity from the semantics.

3.3.1 Parsing

PCMs are represented in various artefacts and languages. Before analyzing PCMs, we need to represent their matrices in a common form. Our PCM metamodel provide this common form as it can represent PCMs in a purely syntactic way. In this step, we only use the following classes from our metamodel: *PCM*, *Matrix*, *Cell* and the different sub-classes of *Cell*. We developed a parser for CSV files and a parser for MediaWiki code using the Sweble Library[2]. We used this parser to extract 75 PCMs from Wikipedia (see Section 4). This step is fully automated.

[2] http://sweble.org/

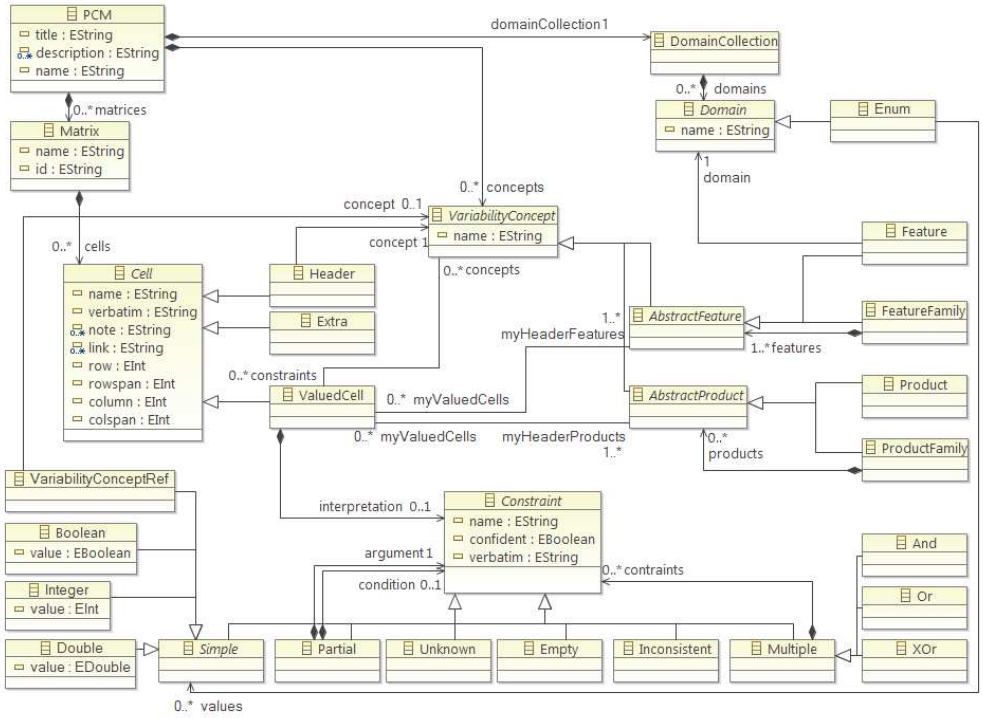

Figure 5: The PCM Metamodel

3.3.2 *Preprocessing*

As the concept of columns is not always present (e.g. a CSV file is a list of rows) and rowspan and colspan may create rows or columns that have different lengths, PCMs may not have rectangular matrices. Therefore, a first step toward the formalization of PCMs is to normalize their matrices. The preprocessing phase add missing cells and duplicate cells with rowspan and colspan. It results in a simple and plain matrix.

After this normalization, the preprocessing phase identifies the types of the cells. There are 3 types of cells in our metamodel (see Figure 5): *Header*, *ValuedCell* and *Extra*. By default, the top left cell is set as *Extra*, the first row and column is set as *Header* and the remaining cells are *ValuedCell*. A specific mapping can be provided through a configuration file in order to support the various PCM forms. For example, we can ignore some rows or columns that will be represented as *Extra* cells in the model or specify that there is more than 1 row of headers.

This step can be either semi-automated or fully automated depending of the manual input with the configuration file. In our further experiment, we used configuration files.

3.3.3 *Extracting Information*

After the preprocessing phase, the problems related with the syntactic structure of PCMs are resolved. It remains to interpret the cells in order to extract the variability information that a PCMs contain. In this phase, we progressively enrich the model with new information.

This step can be either semi-automated or fully automated depending of the manual input with the configuration file. In our further experiment, it is important to mention we did not use configuration files for this particular task. It was performed in a fully automated way.

Extracting Variability Concepts from Headers. The first information that we extract is the variability concepts from the headers that were specified in the preprocessing phase. By default, the headers in the top rows are considered as features and the headers in the left columns are considered as products. Here again, a different mapping can be provided through a configuration file in order to specify for each header if it represents a feature or a product. At the end of this extraction, the model is enriched with *Feature*, *FeatureFamily*, *Product* and *ProductFamily* elements and the *Header* elements are binded to their respecting variability concept through the *concept* association.

Interpreting Cells. After this extraction, we interpret the cells' contents in order to disambiguate them and extract variability information. Each cell is interpreted as a constraint over the features and the products of the PCM. Only cells of type *ValuedCell* need an interpretation as headers are only labels of features or products and extra cells are simply ignored. This interpretation is led by a list of rules that are either defined by default in our transformation or specified by the user through the configuration file.

The interpretation rules are composed of 4 elements: a list of headers, a type of constraint, a regular expression and parameters. The list of headers allows to restrict the rule to specific rows or columns in the PCM. The type corresponds to one of the *Constraint* sub-classes in the metamodel (see Figure 5) that will be instanciated if the rule is applied. The regular expression defines in which cases the rule should be applied. If it matches the cell content, the rule is applied and the potential groups in the regular expression can be used to separate the cell content in sub-constraints. Finally, the parameters are here to provide additional information for the creation of a constraint. For example, in a rule of type *Partial*, we can specify how the groups of the regular expression will be mapped to the *argument* and *condition*.

For each cell, we iterate over the list of interpretation rules. If a rule matches, it creates its corresponding constraint and the cell is binded to its constraint through the *interpretation* association of the class *ValuedCell*. If the matching rule separates the content in sub-constraints, we recursively interpret them with the same list of rules. If no rule matches, the constraint is simply not created. Therefore, a PCM editor could detect afterwards all the cells that are not interpreted and raise a warning.

Extracting Variability Concepts from Interpreted Cells. Among the constraints that were created during the interpretation, VariabilityConceptRef constraints represent references to products or features. The next step of the formalization is to bind these references to the corresponding products or features through their *concept* association. However, such constraint may reference a feature that is not declared in the headers. In this case, we need to create this implicit feature first.

Extracting Feature Domains. The last step of the formalization is to determine the domains of the previously extracted features. In our metamodel, a feature domain is an enumeration of *Simple* elements (see Figure 5).

To determine the domains, we separate features in two categories: features that are defined in headers and features that are implicitly defined in valued cells. For each feature, we first collect all the *Simple* elements contained in the interpretation of the cells associated to the feature. For implicit features, an empty domain is created as the PCM only deals with their presence.

For features in headers, we group their domain elements according to their types (*Integer* and *Double* elements are grouped together as they both represent numbers). We select the group with the maximum size as it represents, potentially, the most correct type of values regarding the feature.

If it is a boolean or number group, we directly add its values in the domain of the feature. However, if it is a group with *VariabilityConceptRef* elements, we perform hierarchical clustering on it to remove rare values. This selection of valid values for a feature, allow to highlight the cells that are possibility inconsistent with respect to the majority.

4. EVALUATION

In this section, we first present the data set and our experiment over the automatic formalization of PCMs. We then discuss our results with respect to our three initial research question that we recall:

RQ1: How to formalize data that are contained in large and uncontrolled natural language PCMs?

RQ2: How to automatically transform PCMs into more formal representations that are still manageable by PCM stakeholders?

RQ3: What kind of tools and services can be built on top of this formalization to assist PCM stakeholders?

4.1 Experiment Settings

Dataset. We selected 75 PCMs from Wikipedia. Though most of them are related to software (audio player software, browser synchronizers, application launchers, disc images, etc) our sample also covers a very large spectrum of domains. This includes domains such as photography and technology (digital SLRs, embedded computer systems), sport and hobbies (ski resorts, video games), economy (tax preparation, enterprise bookmarking), electronics and hardware (fanless switches, chipsets, radio modules), history (world war II), healthcare (dental practice management, dosimeters) among others.

Formalization of PCMs. Following our automated process depicted by Figure 4, the 75 selected PCMs were first parsed into models that conform our PCM metamodel described in Section 3.2. Then, we manually configured the preprocessing step in order to specify the headers of each matrix. Finally, we executed the extracting information step without configuration and only with default interpretation rules.

These 75 Wikipedia PCMs are made of a total of 211 matrices from various sizes, going from 2 to 241 rows, and 3 to 51 columns for a total of 47267 cells. Among them, 6800 (14.39%) are *Headers*, describing either products or features. Another 992 (2.10%) are *Extra* cells that do not carry any valuable information. Finally 39475 (83.51%) cells are considered as *ValuedCells*.

Participants. To evaluate our research questions, each analyzed PCM was evaluated by at least one person among a panel of 20 persons (mainly researchers and engineers) that were not aware of our work. They never saw the metamodel, neither its tooling before the experiment.

Evaluation Sessions. We organized two evaluation sessions where the evaluators were explained the goal of the experiment. We provided a tutorial describing the tool they will have to use, as well as the concepts they were about to evaluate and related illustrative examples.

The checking process consists of looking at each cell of a PCM to identify cells which computed formalism does not match the expected one. In such a case, the evaluators have to state whether the expected domain value exists in the metamodel, provide a proposition of a new concept, claim that there is no possible interpretation, or declare that he/she does not know at all how to analyze the cell.

In addition to the validation task using the interface, evaluators were allowed to leave comments on an additional menu and to exchange with us.

As the evaluation tool is a webapp, they were also able to continue the evaluation on their own at a different time, though they were encouraged to complete the PCM evaluation before submitting the results. Depending of the size of the PCM (number of matrices, complexity of cells, syntactic wellformedness), evaluating a PCM takes between 5 to 20+ minutes.

Evaluation Scenario. The tool proposes a randomly chosen PCM in a way to assure the global coverage of the 75 PCMs. Consequently, during the group session, no evaluator has the same PCM to evaluate.

Right clicking on a cell displays options to validate or not its proposed formalization. To avoid painful individual validation, evaluators are allowed to make multiple selections for collective validation or corrections. Once evaluated, the cells are colores in green, but it is still possible to modify the evaluation.

Once the evaluation of one matrix is finished, evaluators can check the interpretation of the other ones in order to complete the PCM evaluation. At the end, they submit their evaluation to a database and possibly start again a new evaluation.

Evaluated Cells. Among the 39475 cells of the 75 PCMs, 20.83% were ignored (the evaluator declared that he/she

does not know how to analyze the cell) or omitted (no results) by our evaluators. 3.02% were subject to conflicts between evaluators (difference in the correction). As a consequence, 30061 cells are evaluated and present analyzable results. In the following, we will answer the research questions according to these evaluated cells.

4.2 RQ1. Formalizing the Domain of Product Comparison Matrices

In this section, we aim at answering our first research question (RQ1), which is related to the definition of our metamodel, its soundness and completeness.

We define soundness as the effective use of the different metamodel concepts. We evaluate it with the number of occurrences of each concepts of the metamodel.

We define completeness as the fact that there is no missing important concepts in the metamodel. We evaluate it with the analysis of new concepts that our evaluators proposed during the evaluation.

Use of the Metamodel Concepts. During the experiment, 95.72% of the evaluated cells were validated or corrected with concepts of the metamodel. Only 4.28% of cells were not validated and the evaluators proposed a new concept. In Table 1, we present the use of each metamodel concept after the correction by our evaluators. In this table *Multiple* represents the constraints that were not specialized as *And*, *Or* or *XOr*.

First, we observe that all the concepts are used in our metamodel. The most represented concepts are *Boolean*, *VariabilityConceptRef* and *Unknown*. Some concepts are rarely represented such as *Inconsistent* or *XOr* but our evaluators considered them as important for some cells. Yet, we still need to know the quality of the proposed concepts in the metamodel as well as the quality of our classification that we will detail in the following.

New Concepts. As stated previously, 4,28% of the cells were not validated by our evaluators and were corrected with a new concept. The rest of the cells were either validated, corrected with elements of the metamodel, or evaluators are conflicting on the correction. We manually analyzed the new concepts and cluster the answers to form a new set of concepts. We provide the following observations. The very large majority of new concepts occur less than 20 times. A lot of provided concepts are in fact noisy and unexploitable elements as some evaluators provided a concept that is a variant (or a composition) of one already existing concept.

Three new concepts are clearly emerging with more than 200 occurrences: dates, dimensions - units and versions. Dimensions and units of measure are two concepts that can overlap from time to time. Dates and Versions are easier to distinguish though versions can be tagged using dates.

Our classification considers dates as integer numbers, and variability concepts when the date description expresses multiple concepts such as months, days, or trimesters. Our evaluators consider that dates deserve a dedicated concept that allow to handle such expanded expressions. These 3 extensions can be easily applied in our metamodel as they will be considered as specializations of the "Simple" metaclass.

Composing Concepts and Wellformedness. For the sake of concision, PCM authors may also mix different information such as versions and release dates in the same cell. If the distinction between multiple concepts and their interpretation requires little effort for a human analyst, it requires a specific operation from an automatic point of view. Our approach promotes separation of concerns, which can be translated in our example as creating two columns: one for the version and one for the release date.

"Multiple" Specialization. While exchanging with our evaluators, one difficulty that emerged was related to the semantics of the *Multiple* cell that expresses a composition with an AND, OR or XOR operator. Their semantics are simple but not intuitive and hard to determine at the PCM level. These operators are basic operators from a variability modeling point of view. They represent important actors for configuration and for managing combination issues. Yet, their semantics are still to determine from a PCM perspective.

Answer to RQ1. Regarding our research question, our metamodel proposes a set of necessary concepts that will have to be complemented by the three new concepts that have emerged. However, the semantics of the composition of Multiple values with AND, OR, and XOR have to be clarified.

4.3 RQ2. Evaluation of the Automated Formalization Process of PCMs

In this section, we aim at answering our second research question (RQ2), which is related to the degree and the quality of the automation of the transformation toward a PCM model. We will here evaluate the precision of our automated classification with respect to the empirical evaluation of our 20 evaluators and investigate on the errors made by the classifier. The precision will be the ratio of valid cells among all evaluated cells.

Global Results. Tables 1, 2 and 3 detail the data collected during the experiment. We note that in Table 2, the precisions of *Inconsistent* and *XOr* concepts are not available as they are not generated by our default interpretation rules.

Our automated approach have been evaluated with a global precision of 93.11%, with different precision depending of the concept (see Table 2). If we consider that the cells corrected with new concepts are not related to the precision of our automated techniques, then the precision reaches 97.27%.

Quality of the Transformation for PCMs. Table 2 shows that our interpretation rules are evenly precise except for the *Double* concept. This is due to the ambiguity of the interpretation of numbers which makes difficult the development of such rules. In contrast, we note that *Boolean* and *Unknown* interpretation rules have almost 100% of precision.

Table 3 provides a distribution of errors within PCMs. The objective is to know if our errors are very localized or scattered all along the PCMs.

Formalization Errors. Though the formalization step works well for correctly formed data, it is much less evident for complex data such as partial and multiple answers. If we consider the errors from the classification, we observe that they arise from 4 main areas.

- **Overlapping Concepts.** One way of precising one information within the cell is to provide this extra information between parentheses. However, this pattern is also much used to express constraints or partial answers. As a consequence, we observe the same data pattern for two different concepts, and it still requires a human analysis to determine the correct concept among the two.

Table 1: Use of the Metamodel Concepts

Boolean	Integer	Double	VCRef	Partial	Unkn.	Empty	Incons.	And	Or	XOr	Multiple
43.96%	3.20%	0.64%	20.21%	2.69%	12.25%	8.62%	0.07%	8.04%	0.29%	0.02%	0.01%

Table 2: Precision of our Automated Techniques in the Evaluation of 75 PCMs

Total	Boolean	Integer	Double	VCRef	Partial	Unkn.	Empty	Incons.	And	Or	XOr
93.11%	99.83%	79.62%	45.21%	86.25%	82.75%	99.62%	91.37%	N/A	87.07%	91.38%	N/A

- **Missing Concepts.** The concepts that have to be added and their related interpretation rules have decreased the precision of our automated transformation.
- **Missing Interpretation Rules.** We may not have been able to have sufficient generic rules in the catalog.
- **Bad Rules.** We may have written incorrect interpretation rules that lead to bad classifications or misses.

Data Patterns and Specific Transformations. We recall that our transformation rules provide a catalog of generic data patterns but also allows the customization with PCM-specific data patterns. For instance, the Yes/No Boolean pattern can be replaced for a particular column with the pattern X/"", where the empty cell would stand for a "No" in one specific column.

For the experiment, we did not used specific patterns for our evaluation of 75 Wikipedia PCMs, considering a full degree of automation. Customizing with PCM-specific rules can be very useful for re-engineering specific PCMs we want to model as part of managing legacy data, though it lowers the automation degree but increases the precision. Further work is required to determine when specific rules can be generic enough to be incorporated into our generic catalog. Another further work will be to assess the necessary effort to provide these specific rules.

Answer to RQ2. Regarding a fully automated approach (excepting the very first preprocessing steps described in the protocol), our automated approach has been able to qualify our data set with a precision of 93.11%.

Though the precision differs depending of the kind of information in the cell (as reported in table 2, it is worth noticing that these results are obtained without any customization with PCM specific rules. Using such rules would have increased the precision but decrease the degree of automation for complex PCMs.

4.4 RQ3: Tools and Services

In this section, we evaluate how the formalization of PCMs through our automated techniques and metamodel can ease the development of different tools and services in comparison to the use of classic applications supporting PCMs (e.g. spreadsheets, databases and websites). Throughout the section, we illustrate some of the possible operations on formalized PCMs in the editor in Figure 6.

Editing and Formalizing PCMs. A first application of our metamodel is to help the user in editing and formalizing a PCM. Thanks to our metamodel, we could easily build an editor for our experiment described in Section 4.1 (see Figure 6). Such editor allows to check that the formalization of a PCM is as expected in order to ensure its further exploitation (see Figure 6, A). In case the formalization is incorrect, the user can directly edit the cell content or its interpretation on a familiar tabular view while conserving a model conformed to the PCM metamodel.

Providing Guidance and Good Practices. Guidance capabilities can be developed thanks to the *Inconsistent* concept in our metamodel and the computation of a domain for each feature (see Section 3 for further details). For example, an editor can warn the user that a cell content is considered as inconsistent or is not existing in its corresponding feature's domain (see orange cells in Figure 6, B). These warnings can be used by a user during the edition of a PCM from scratch or during a refactoring scenario to speed up the detection of incorrect cells. For instance, spell errors can be easily detected in a PCM without carefully reading thousands of cells.

Comparing Products. As explained in Section 1 and 2, the complexity of existing PCMs hinders their main objective: providing a simple and intuitive way of comparing products over different features. Formalizing PCMs with our automated techniques and our metamodel allows to achieve this objective by providing all the necessary constructs and precision to a comparator.

A first advantage of our metamodel over spreadsheet applications (e.g. Excel), database or websites is that it contains explicit notions of products and features. With our metamodel, a comparator can directly reason in terms of these variability concepts. It does not have to care about the structural information (rows and columns) and the representation of the cell content. This eases the development and quality of comparators.

A second advantage is that the clear semantics of the cells enables the development of advanced reasoning facilities. The constraints defined by the cells can be easily encoded into state-of-the-art reasoners input format (e.g. CSP or SMT solvers). Such reasoners expect formatted and consistent data that cannot be provided without formalization.

Based on these two previous advantages, comparators can build advanced filtering capabilities working on multiple criteria. The absence of structural constraints in the metamodel allows to reorganize products and features in order to visualize only the most important ones according to a user. This can reduce the cognitive effort required by a user to analyze a PCM. The reasoning facilities also allows to filter the products based on user-defined constraints (see Figure 6, C) or empirical data (e.g. best-selling product).

Producing New Artifacts. In addition to the edition of PCMs and the comparison of products, the models generated by our automated techniques can be used for producing new artefacts. As we explained previously, thanks to our metamodel, the semantics of a PCM can be encoded in a formula. Such formula can be used to generate a feature model that represents the variability of the product line described by the PCM. This feature model can provide a compact and global view of the features and their constraints. We already applied synthesis procedures on a subset of Wikipedia PCMs, focusing on constructs amenable to Boolean logic [4].

Table 3: Distribution of the Precision over 75 Analyzed PCMs

Precision	0-9%	10-19%	20-29%	30-39%	40-49%	50-59%	60-69%	70-79%	80-89%	90-99%	100%
PCMs	1.33%	0%	0%	1.33%	0%	4.0%	0%	8.0%	13.33%	45.33%	26.67%

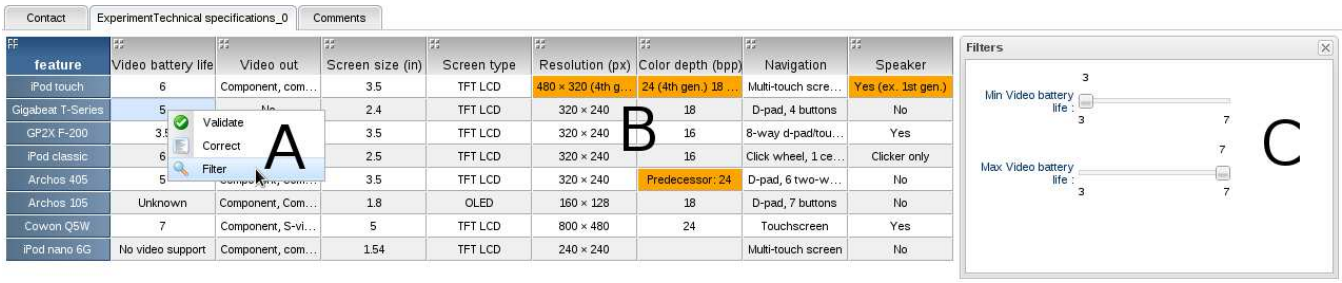

Figure 6: Example of a PCM Editor with Warnings and Filtering

Finally, our metamodel eases the development of generic visualization tools (e.g. a graph illustrating how the features are correlated). The formalization of PCMs into our metamodel provides a uniform format for developing such tools. Therefore, the developers can focus on the visualization of the different types of values (*Integer*, *Double*, *Boolean*) without handling the parsing phase that is required for raw data.

Answer to RQ3. Regarding our research question, the development of the editor in Figure 6 shows that the clear semantics of cell contents and the explicit notions of products and features in our metamodel can ease the development of tools and services based on PCMs. In comparison to spreadsheets applications, databases or websites, our metamodel avoids developers to manage the structure of the PCM and the parsing of the content cells while reasoning on the PCM. Further evaluation of our metamodel in a exploitation scenario are needed to quantify the benefits of using our metamodel over non-formal techniques.

5. THREATS TO VALIDITY

Construct Validity. In order to avoid a bias while evaluating our PCMs, we removed from the evaluation set all the training material, including the PCMs used during our previous work, and the PCMs we used to build and test our interpretation rules.

Size and well-formedness are important factors in the PCM analysis. During the experiment, large matrices (more than 1000 cells) and badly designed PCMs are often source of reject. Several (4) of our evaluators mentioned that the largest PCMs were painful to read and analyze and one of them has voluntary skipped them. If it highlights and justifies the necessary work toward better PCMs, and PCM editors, it also sets the question of the optimal size of an exploitable (standard or model-based) PCM for a human.

Internal Validity. For this evaluation, we obtain heterogeneous results since our classification is evaluated by different persons. Though we made our possible to evaluate each PCM twice or more, the random access to PCM cannot guarantee that participants evaluate all PCMs equally.

External Validity. A threat to validity is the applicability of our metamodel and automated techniques to other PCMs (outside Wikipedia). We progressively defined the metamodel according to our analyses of several PCMs found here and there on the internet, i.e., we did not restrict ourselves to PCMs from Wikipedia (see [37] for more details). The Wikipedia dataset is publicly available but challenging since it covers various domains and exhibits a high degree of heterogeneity. The mining of other PCMs is an additional challenge to address before diversifying our dataset.

Our evaluators are engineers or researchers (PhD students) in software engineering. Consequently, they are familiar with computer interfaces, modeling/typing that can be potentially not intuitive during the evaluation. We have carefuly documented and tested our tutorial in order to provide sufficient and sound examples of typical "Constraints" for the semantics of valued cells so that it can be reused outside this community.

This empirical study does not detect false negatives: matrices that cannot be interpreted as PCMs, for instance Wikipedia timelines, have been either removed or detected as such by participants.

6. RELATED WORK

PCMs as Specific Spreadsheets. Considerable research effort has been devoted to the study of spreadsheets [19, 35]. All studies have the same observation: errors in spreadsheet are common but non trivial [2, 12, 20–23]. Automated techniques have been developed for locating errors; guidelines on how to create well-structured and maintainable spreadsheets have been established, etc. Herman *et al.* reported that the current state of spreadsheet use still leads to numerous problems [22].

PCMs can be seen as a special form of spreadsheets with specific characteristics and objectives. A shared goal of this line of research and our work is to improve quality of spreadsheets (i.e., PCMs).

Some works aim at tackling programming errors or code smells in spreadsheets [11]. We describe in Section 4.4 the ability of our tools to detect *warnings*. General rules exposed in [11] can be implemented (at the moment we implement only a subset of the rules). Specific rules that apply to specific concepts of PCMs can also be considered. In both cases, the formalization of PCMs eases the realization.

Reverse Engineering Spreadsheets. As spreadsheets are subject to errors and ambiguity, some works propose to synthesize high-level abstractions or to infer some information [1, 7, 9]. For instance, Chambers and Erwig [7] describe a mechanism to infer *dimensions* (i.e., units of measures). These works typically operate over formulas of spreadsheets – a concept not apparent in PCM – or target general problems that are not necessarily relevant for PCMs. Some of the techniques could be reused or adapted. Another research

direction is to elicitate the domain information stored in spreadsheets. For instance, Hermans *et al.* proposed to synthesize class diagrams based on the analysis of a spreadsheet [21]. Section 4.4 describes the application of synthesizing feature models from PCMs [4]. Our work is highly facilitated since the relevant constructs (e.g. Boolean) for the synthesis are made explicit. Applying state of the art synthesis techniques [3] from raw data would lead to unexploitable and unsound results. It should be noted that the synthesis techniques we have considered so far assume a possible encoding into Boolean logic. An interesting research direction would be to other kinds of logics for numeric or unknown values.

Model-based Spreadsheets. Constructive approaches for ensuring that spreadsheets are correct by construction have been developed in order to prevent typical errors associated with spreadsheets. ClassSheet [15] introduces a formal object-oriented model which can be used to automatically synthesize spreadsheets. MDSheet is based on ClassSheet and relies on a bi-directional transformation framework in order to maintain spreadsheet models and their instances synchronized [10]. Francis *et al.* [17] develop tools to consider spreadsheets as first-class models and thus enable the reuse of state of the art model management support (e.g., for querying a model).

We follow a similar model-based approach and the benefits also apply to our work. A notable difference is that we consider the specific nature of PCMs, i.e., we target a specific class of spreadsheets (PCMs). We elaborate a metamodel as well as automated techniques to transform data as models. Services built on top of the metamodel are domain specific.

Metamodeling. A major contribution of this paper is the elaboration and evaluation of a metamodel of PCM. The metamodel is central to (1) the automated encoding of raw data into models (2) the development of services around PCMs through the exploitation of models. Numerous formalisms, techniques, or guidelines have been proposed to design a metamodel [6, 8, 16, 26–29, 31, 38, 39].

Example-driven approaches have been developed to, e.g., ease the elaboration of metamodels [6, 8, 25, 31]. For instance, De Lara *et al.* [8,31] propose an interactive approach in which domain experts specify example model fragments that are then exploited for constructing a metamodel. Our large-scale experiments over 30061 cells can be seen as an approach by examples with much more examples under consideration compared to existing approaches. Another noticeable property of our work is that our tool, based on the metamodel, ease in gathering examples.

Cabot *et al.* [26] promote the collaborative definition of domain-specific modeling languages (metamodels). Collaboration tools are more and more emerging in global software engineering [30]. We also promote collaboration through the validation of the interpretations of data or the proposal of alternatives in our tool.

Product Descriptions and Variability. Davril *et al.* presented a fully automated approach, based on prior work [18], for deriving feature models from publicly available product descriptions found in websites such as SoftPedia and CNET [13]. The product descriptions on such sites are generally incomplete, lack of a unifying structure, apart from the product header, and propose informal descriptions using uncontrolled natural language.

PCMs also aim to describe products (some cell values represent features of product). Contrary to informal product descriptions considered in [13], PCMs are semi-structured. We found there is an opportunity to formalize the practice of editing, maintaining, and exploiting PCMs. We developed an approach to automate the formalization of existing content. The editor as well as the design of the metamodel also aim to reduce the informality and incompleteness of data. Berger *et al.* reported that spreadsheets are widely used in industry to model variability [5]. Our work can be exploited since variability concepts are apparent in existing PCMs.

7. CONCLUSION

Product Comparison Matrices (PCMs) are simple, convenient, easy means to provide a chooser lots of rich and useful information. However, they present many drawbacks such as heterogeneity and ambiguity, lack of formalization, lack of tool support and efficient services. We addressed three different questions which are related to the formalization of PCMs, the quality of an automated formalization and possibles services and tools that could be possible to ease the life of PCM stakeholders from both reading, editing and analyzing perspectives.

We proposed a metamodel that offers a more formal canvas for PCM edition and analysis. This metamodel has been evaluated with an empirical study over 75 Wikipedia PCMs that highlighted the soundness of its concepts but also 3 emerging concepts that we will have to add and further research direction in order to clarify the semantics of others.

We also proposed an automated approach that convert PCM raw data into PCM models that conforms to the proposed metamodel. The quality of the transformation has been evaluated with a precision of 93.11% for a fully automated classification process. We used our own PCM editor, based on our metamodel, for collecting large scale data. For the divergent values, the PCM editor can be used to further edit (correct) the PCMs. The editor also provides detection of warnings regarding potential bad-smells in the PCMs and some initial reasoning capabilities on feature domains in order to enable sorting or filtering on particular features.

The elaboration of the metamodel is a significant step towards the definition of a unified format for PCMs and the development of innovative services (e.g., configurators). The automated techniques substantially reduce the user effort and can be used to mine the numerous PCMs that now abound on the Internet. Our dataset, empirical results, and the set of tools (including the editor) are available online `http://tinyurl.com/PCMFormalization`. We are in the process of engaging a Web community in the mining, re-engineering, edition, and exploitation of PCMs.

The engineering challenges are many. For instance, the usability of the developed tools should be evaluated. A tradeoff between rigorousness and creativity should ideally be found. Uncertainty and incompleteness of PCMs, though now explicitly identified in our metamodel, should be addressed as well by reasoning services.

Acknowledgements

This work is partially supported by the French BGLE project CONNEXION and the European ITEA2 project MERgE (ITEA2 11011).

8. REFERENCES

[1] R. Abraham and M. Erwig. Type inference for spreadsheets. In *Proceedings of the 8th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, PPDP '06, pages 73–84, New York, NY, USA, 2006. ACM.

[2] R. Abraham and M. Erwig. Ucheck: A spreadsheet type checker for end users. *J. Vis. Lang. Comput.*, 18(1):71–95, 2007.

[3] N. Andersen, K. Czarnecki, S. She, and A. Wasowski. Efficient synthesis of feature models. In *Proceedings of SPLC'12*, pages 97–106. ACM Press, 2012.

[4] G. Bécan, M. Acher, B. Baudry, and S. Ben Nasr. Breathing ontological knowledge into feature model management. Rapport Technique RT-0441, INRIA, oct 2013.

[5] T. Berger, R. Rublack, D. Nair, J. M. Atlee, M. Becker, K. Czarnecki, and A. Wasowski. A survey of variability modeling in industrial practice. In *VAMOS'13*, page 7. ACM, 2013.

[6] K. Bąk, D. Zayan, K. Czarnecki, M. Antkiewicz, Z. Diskin, A. Wąsowski, and D. Rayside. Example-driven modeling: Model = abstractions + examples. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 1273–1276, Piscataway, NJ, USA, 2013. IEEE Press.

[7] C. Chambers and M. Erwig. Automatic detection of dimension errors in spreadsheets. *J. Vis. Lang. Comput.*, 20(4):269–283, Aug. 2009.

[8] J. S. Cuadrado, J. de Lara, and E. Guerra. Bottom-up meta-modelling: An interactive approach. In R. B. France, J. Kazmeier, R. Breu, and C. Atkinson, editors, *MoDELS*, volume 7590 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2012.

[9] J. Cunha, M. Erwig, and J. Saraiva. Automatically inferring classsheet models from spreadsheets. In *Visual Languages and Human-Centric Computing (VL/HCC), 2010 IEEE Symposium on*, pages 93–100, Sept 2010.

[10] J. Cunha, J. a. P. Fernandes, J. Mendes, and J. a. Saraiva. Mdsheet: A framework for model-driven spreadsheet engineering. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, pages 1395–1398, Piscataway, NJ, USA, 2012. IEEE Press.

[11] J. Cunha, J. a. P. Fernandes, H. Ribeiro, and J. a. Saraiva. Towards a catalog of spreadsheet smells. In *Proceedings of the 12th International Conference on Computational Science and Its Applications - Volume Part IV*, ICCSA'12, pages 202–216, Berlin, Heidelberg, 2012. Springer-Verlag.

[12] J. Cunha, J. Visser, T. L. Alves, and J. Saraiva. Type-safe evolution of spreadsheets. In *FASE'11*, volume 6603 of *LNCS*, pages 186–201, 2011.

[13] J.-M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, and P. Heymans. Feature model extraction from large collections of informal product descriptions. In *ESEC/FSE'13*, 2013.

[14] W. Dou, S.-C. Cheung, and J. Wei. Is spreadsheet ambiguity harmful? detecting and repairing spreadsheet smells due to ambiguous computation. In *ICSE'14*, 2014.

[15] G. Engels and M. Erwig. Classsheets: Automatic generation of spreadsheet applications from object-oriented specifications. In *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, ASE '05, pages 124–133, New York, NY, USA, 2005. ACM.

[16] M. Faunes, J. J. Cadavid, B. Baudry, H. A. Sahraoui, and B. Combemale. Automatically searching for metamodel well-formedness rules in examples and counter-examples. In Moreira et al. [33], pages 187–202.

[17] M. Francis, D. S. Kolovos, N. D. Matragkas, and R. F. Paige. Adding spreadsheets to the mde toolkit. In Moreira et al. [33], pages 35–51.

[18] N. Hariri, C. Castro-Herrera, M. Mirakhorli, J. Cleland-Huang, and B. Mobasher. Supporting domain analysis through mining and recommending features from online product listings. *IEEE Transactions on Software Engineering*, 99(PrePrints):1, 2013.

[19] D. Hendry and T. Green. Creating, comprehending and explaining spreadsheets: a cognitive interpretation of what discretionary users think of the spreadsheet model. *International Journal of Human-Computer Studies*, 40(6):1033 – 1065, 1994.

[20] F. Hermans, M. Pinzger, and A. v. Deursen. Detecting and visualizing inter-worksheet smells in spreadsheets. In *ICSE'12*, pages 441–451. IEEE, 2012.

[21] F. Hermans, M. Pinzger, and A. van Deursen. Automatically extracting class diagrams from spreadsheets. In *ECOOP'10*, volume 6183 of *LNCS*, pages 52–75. Springer-Verlag, 2010.

[22] F. Hermans, M. Pinzger, and A. van Deursen. Supporting professional spreadsheet users by generating leveled dataflow diagrams. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 451–460, New York, NY, USA, 2011. ACM.

[23] F. Hermans, M. Pinzger, and A. van Deursen. Detecting code smells in spreadsheet formulas. In *ICSM*, pages 409–418. IEEE, 2012.

[24] S. Iyengar. *The Art of Choosing*. Twelve, 2010.

[25] J. L. C. Izquierdo and J. Cabot. Discovering implicit schemas in json data. In F. Daniel, P. Dolog, and Q. Li, editors, *ICWE*, volume 7977 of *Lecture Notes in Computer Science*, pages 68–83. Springer, 2013.

[26] J. L. C. Izquierdo and J. Cabot. Enabling the collaborative definition of dsmls. In *Proceedings of the 25th International Conference on Advanced Information Systems Engineering*, CAiSE'13, pages 272–287, Berlin, Heidelberg, 2013. Springer-Verlag.

[27] F. Javed, M. Mernik, J. Gray, and B. R. Bryant. MARS: A metamodel recovery system using grammar inference. *Information and Software Technology*, 50(9–10):948 – 968, 2008.

[28] G. Karsai, H. Krahn, C. Pinkernell, B. Rumpe, M. Schneider, and S. Völkel. Design guidelines for domain specific languages. In M. Rossi, J. Sprinkle, J. Gray, and J.-P. Tolvanen, editors, *Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling (DSM'09)*, pages 7–13, 2009.

[29] S. Kelly and R. Pohjonen. Worst practices for

domain-specific modeling. *IEEE Software*, 26(4):22–29, 2009.

[30] F. Lanubile, C. Ebert, R. Prikladnicki, and A. Vizcaíno. Collaboration tools for global software engineering. *IEEE Softw.*, 27(2):52–55, Mar. 2010.

[31] J. López-Fernández, J. S. Cuadrado, E. Guerra, and J. de Lara. Example-driven meta-model development. *Software & Systems Modeling*, pages 1–25, 2013.

[32] H. Miyashita, H. Tai, and S. Amano. Controlled modeling environment using flexibly-formatted spreadsheets. In *ICSE'14*, 2014.

[33] A. Moreira, B. Schätz, J. Gray, A. Vallecillo, and P. J. Clarke, editors. *Model-Driven Engineering Languages and Systems - 16th International Conference, MODELS 2013, Miami, FL, USA, September 29 - October 4, 2013. Proceedings*, volume 8107 of *Lecture Notes in Computer Science*. Springer, 2013.

[34] S. Nestorov, S. Abiteboul, and R. Motwani. Inferring structure in semistructured data. *SIGMOD Rec.*, 26(4):39–43, Dec. 1997.

[35] R. R. Panko. Thinking is bad: Implications of human error research for spreadsheet research and practice. *CoRR*, abs/0801.3114, 2008.

[36] N. Sannier, M. Acher, and B. Baudry. From comparison matrix to variability model: The wikipedia case study. In *ASE*, pages 580–585. IEEE, 2013.

[37] N. Sannier, G. Bécan, M. Acher, S. Ben Nasr, and B. Baudry. Comparing or configuring products: Are we getting the right ones? In *Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems*, VaMoS '14, pages 9:1–9:7, New York, NY, USA, 2013. ACM.

[38] M. Voelter, S. Benz, C. Dietrich, B. Engelmann, M. Helander, L. C. L. Kats, E. Visser, and G. Wachsmuth. *DSL Engineering - Designing, Implementing and Using Domain-Specific Languages*. dslbook.org, 2013.

[39] D. Zayan, M. Antkiewicz, and K. Czarnecki. Effects of using examples on structural model comprehension. In *ICSE'14*, 2014.