Aligning Software Configuration with Business and IT Context

Fabiano Dalpiaz[1], Raian Ali[2], and Paolo Giorgini[1]

[1] University of Trento, Italy
{dalpiaz, paolo.giorgini}@disi.unitn.it
[2] University of Bournemouth, United Kingdom
rali@bournemouth.ac.uk

Abstract. An important activity to maximize Business/IT alignment is selecting a software configuration that fits a given context. Feature models represent the space of software configurations in terms of distinguished characteristics (features). However, they fall short in representing the effect of context on the adoptability and operability of features and, thus, of configurations. Capturing this effect helps to minimize the dependency on analysts and domain experts when deriving a software for a given business and IT environment. In this paper, we propose *contextual feature models* as a means to explicitly represent and reason about the interplay between the variability of both features and context. We devise a formal framework and automated analyses which enable to systematically derive products aligned with an organizational context. We also propose FM-Context, a support tool for modeling and analysis.

Keywords: Variability, Product Lines, Business/IT Alignment

1 Introduction

Business/IT (B/I) alignment concerns the effective usage of Information Technology (IT) in a business environment [9]. Homogenizing software with the business and technical context of an organization is a major B/I alignment challenge which requires a joint effort of business managers and IT administrators. This task is particularly difficult due to cultural differences between the involved actors (the so-called B/I gap [14]) and the evident—though fuzzy—mutual impact between IT and business aspects. A main challenge for information systems engineering is to bridge this gap minimizing mismatches and maximizing alignment.

Software product lines [3] and feature models [10] are a development paradigm and a modeling notation that support the configuration of software products from reusable assets. A product is generated for each organization depending on its business and IT profile. Existing configuration approaches, e.g., [11, 5], provide limited automated support to achieve B/I alignment. Thus, they heavily rely on the skills of analysts and domain experts. This calls for the development of systematic approaches and CASE tools to identify a configuration that fits an organizational context both at the business and IT level.

The contribution of this paper is a modeling and analysis framework which enables the configuration of a software aligned with an organizational context. We propose Contextual Feature Models (CFMs), which extend traditional feature models with the concept of *context*. This enables minimizing the efforts of domain experts when configuring a product to a specific environment: CFMs express how the organizational context affects the inclusion/exclusion and the applicability of features. We distinguish between two types of context:

- *Business context* concerns the organizational characteristics of an enterprise and helps to determine whether a feature is required or advisable. For instance, a feature like "LDAP server" may be advisable in a business context where "the company has multiple branches";
- *Technical context* concerns the technical infrastructure required to operationalize a feature. For example, a feature "bug tracking system" may require a technical contexts "MySql" and "PHP v$\geq$5" to be successfully deployed.

We develop automated analysis techniques to examine the interplay between features and context. Our techniques enable maximizing B/I alignment, as they support the selection of a configuration that fits well with a given environment:

- **Configurations**: given an organizational environment, in terms of technical and business contexts, we identify possible software configurations. Among these configurations, the analyst can select the best-quality one;
- **Business-to-IT alignment**: we identify an IT infrastructure that supports all software configurations fitting with a given business context, as well as the minimal IT technical requirements to support the business context;
- **IT-to-Business alignment**: given an IT infrastructure, described in terms of technical contexts, we identify the business contexts that support such infrastructure. This analysis serves to compare candidate IT infrastructures so to maximize the support of the business contexts the client cares about.

We also develop FM-Context, an Eclipse-based CASE tool which allows for drawing CFMs and implements our analysis techniques on top of a Datalog solver. We illustrate our framework using the following scenario.

Motivating scenario. Drupal[3] is an open-source content management system with wide industrial adoption. Several web development companies base their products upon a Drupal-based product line (over 13.000 modules are available for use). The challenge is about identifying a configuration of Drupal which is aligned with the business and technical context of a client organization. We model Drupal configurations in a CFM, and illustrate how our automated analyses support the derivation of a configuration highly aligned with an organizational context. □

The paper is structured as follows. Sec. 2 introduces CFMs and applies them to the Drupal scenario. Sec. 3 describes a formal framework for CFMs and presents automated analysis techniques built on top of it. Sec. 4 details FM-Context and reports on preliminary scalability results. Sec. 5 presents related work, while Sec. 6 draws our conclusions and presents future directions.

[3] www.drupal.org

2 Contextual Feature Models (CFMs)

Feature models are a compact and intuitive modeling language to represent variability—the existence of multiple configurations—in software product lines. Many features, however, are not always applicable or advisable: they are context-dependent. A feature may be *advisable* only in a business context: "semantic search engine" could be advisable only in business context "the customer has a vast catalog". A feature may be *applicable* only in a technical context: "role-based access control" could require technical context "mysql v≥5". Such constraints apply to configurations too, as a configuration is a set of features.

Existing feature modeling approaches pay no or limited attention (see Sec. 5) to the interplay between features and context. Most approaches rely on the analyst's expertise or on informal communication with stakeholders and, thus, make B/I alignment hard to achieve. To address these shortcomings, we introduce contextual feature models, which explicitly represent the effect of business and technical contexts on features and configurations.

There is an active discussion concerning the structure of feature models (e.g. graph vs. tree), and the semantics of a feature (e.g. is a feature user-visible?) [15]. Our focus in this paper is to demonstrate the importance of weaving context together with features and, consequently, with configurations variability. Therefore, we adopt a fairly simple structure of feature model and add context to it. Our notation can, however, be extended to support more expressive relations between features, e.g. cardinality constraints over features [6].

Fig. 1. Meta-model of contextual feature models

The meta-model of CFMs is shown in the class diagram of Fig. 1. A Feature is an abstract class with two concrete manifestations: (i) a decomposed feature Dec feature is further decomposed into sub-features, and is not directly implemented; (ii) a Leaf feature is directly implemented and is a leaf in a feature tree.

Features are hierarchically structured via Decomposition relations. A decomposed feature f has exactly one decomposition, which has a type (Dec type): (i) *and*: all sub-features shall be selected to select f; (ii) *xor*: exactly one sub-feature; (iii) *or*: at least one sub-feature. In addition to their hierarchical decomposition, features can be connected via two additional relationships. The requires relation says that the inclusion of a feature mandates the inclusion of another feature. The excludes relation is used to represent mutual exclusion between two features.

Unlike non-contextual feature models, each branch of a decomposition can be constrained by a business context (Bus context). A business context refers to social, business, or organizational characteristics of the deployment environment. Some examples are: "the company is a small enterprise", "the company has branches in different countries", "customers speak different languages", and

"the revenue trend is negative". Business contexts define the information to be acquired to decide if a certain sub-feature is advisable or required. Business contexts do not include technical prerequisites (those are technical contexts). The effect of business context on a sub-feature depends on the decomposition type:

- *and*: the sub-feature is *required* when the context holds;
- *or/xor*: the sub-feature is *selectable* only when the context holds.

Business contexts could be communicated as high-level statements whose validity is hard to judge objectively. For instance, the validity of bc_1= "the company has an international profile" cannot be judged in an objective way: what exactly makes a company profile international? Thus, a business context would need refinement to a formula of objectively observable facts that reifies it. This is essential to avoid misleading judgments of business contexts. For example, while two analysts might give two different answers about bc_1, they would give the same judgment if we refine bc_1 to a formula of observable facts such as $\mathsf{fact}_1 \vee \mathsf{fact}_2$ where fact_1= "the company has branches in different countries" and fact_2= "the company has employees from different countries". We adapt our previous work on *context analysis* [1] to analyze business contexts and ultimately identify a formula of observable facts. Accordingly, we define context as a formula of contextual predicates (Ctx predicate) of statements and facts: statements are hierarchically refinable to formulae of facts which support their evidence:

- Statement: it is a contextual predicate which is not observable per se;
- Fact: it is a contextual predicate which is observable per se.

Leaf features are implemented features and can be associated with a *technical context* (Tech context). Technical contexts specify technical prerequisites for an implemented feature. If not met, they inhibit the operationalization of a leaf feature. Unlike business ones, technical contexts need no context analysis, as the development team of an implemented feature knows—and typically documents—its technical prerequisites. For instance, a technical context for a web-based text editor may be using the Internet Explorer browser. We characterize a technical context by attributes min and max, which specify the range of admitted values. For instance, given the Internet Explorer context, min and max may refer to its version: (min = 6, max = ∞) indicates that the feature requires a version equal to or greater than 6; (min = 6, max = 8) a version between 6 and 8; and (min = 0, max = ∞) no version constraints.

Fig. 2 shows a partial CFM for the Drupal scenario presented in Sec. 1. We focus on features related to content management, i.e. how users can create and modify different types of web pages. To illustrate our approach using a small model, we introduce some features which are not currently available in Drupal.

Fig. 2. Contextual feature model for the Drupal scenario

The root feature content management—a sub-feature in the complete CFM for Drupal—is and-decomposed to three sub-features: content editor to create and modify web pages, content search to search information, and cck (content construction kit) to enable the creation of custom content. cck is context-dependent, as it is required only when business context bc1 holds. The context analysis for bc1 says that custom content is needed if the website has at least ten editors (fact), or the customer has a large business (statement). In turn, large business is and-supported if there are multiple branches and branches are autonomous (there is no organizational policy on how each branch creates content).

The feature content editor is xor-decomposed: only one content editor can be in a configuration. The drupal content editor has no technical contexts, while other editors have prerequisites. For example, word online editor requires internet explorer v≥7; ckeditor requires firefox (v≥3), internet explorer (v≥6), and the ckeditor binaries. In the sub-tree rooted by content search, the features google search and semantic search are mutually exclusive (*excludes* relation). The feature semantic search needs technical contexts php (v≥5) and arcdfstore, as well as the selection of feature cck (*requires* relation). Due to space limitations, business contexts bc3, bc4, and bc5 are not analyzed in Fig. 2. Context bc3 stands for "basic visual editing is needed", bc4 indicates "advanced visual editing is needed", and bc5 means "customer has a large products catalog".

3 Reasoning about B/I Alignment with CFMs

We devise the formal framework to represent CFMs (Sec. 3.1) and then we detail and illustrate the proposed techniques that help maximizing B/I alignment in feature-oriented software configuration (Sec. 3.2).

3.1 Formal framework

We rely upon the following assumptions: (i) every feature decomposition is contextual, and such context may be trivially true; (ii) we refer to a single contextual feature model $\mathcal{M}$: in the following, all features, contexts, and relations are part of $\mathcal{M}$; (iii) a business context BC is defined as a set of facts $\{\mathsf{fact}_1, \ldots, \mathsf{fact}_m\}$ that hold; (iv) a technical context $\mathsf{TC}=\{\mathsf{tc}_1, \ldots, \mathsf{tc}_q\}$ is a set of technical contexts $\langle$tcname, min, max$\rangle$; and (v) excludes is a symmetric relation.

A configuration CFG for $\mathcal{M}$ is a set of features that are aligned with the business context (they are adequate to the deployment business environment), with the technical context (the technical infrastructure enables their deployment), and, together, support the root feature of $\mathcal{M}$.

Definition 1 (Configuration). *Given* BC *and* TC*, a set of features* $\{\mathsf{f}_1, \ldots, \mathsf{f}_n\}$ *is a configuration* CFG *for* $\mathcal{M}$ *with respect to* BC *and* TC*, formally* $\mathsf{CFG} \vdash_{\mathsf{BC,TC}} \mathcal{M}$*, if and only if* $\mathsf{root}(\mathcal{M}) \in \mathsf{CFG}$ *and (1-4) hold* $\forall \mathsf{f}' \in \mathsf{CFG}$*:*

1. $\mathsf{dec}(\mathsf{f}', \mathsf{D}=\{\langle \mathsf{f}_1, \mathsf{bc}_1\rangle, \ldots, \langle \mathsf{f}_n, \mathsf{bc}_n\rangle\}, \mathsf{type}) \rightarrow$
   $\nexists\ \langle \mathsf{f}_j, \mathsf{bc}_j\rangle \in \mathsf{D} : \neg\mathsf{support}(\mathsf{BC}, \mathsf{bc}_j) \wedge \mathsf{f}_j \in \mathsf{CFG}$*, and*

- type=and: ($\exists$ 1≤k≤n : $f_k \in$ CFG) $\wedge$ ($\forall$ $\langle f_i, bc_i \rangle \in$ D : support(BC, bc_i) $\rightarrow$ $f_i \in$ CFG)
- type=or: $\exists$ $\langle f_i, bc_i \rangle \in$ D : support(BC, bc_i) $\wedge$ $f_i \in$ CFG
- type=xor: $\exists$!$\langle f_i, bc_i \rangle \in$ D : support(BC, bc_i) $\wedge$ $f_i \in$ CFG

2. $\forall$f'' : requires(f', f'') $\rightarrow$ f''$\in$ CFG
3. $\forall$f'' : excludes(f', f'') $\rightarrow$ f''$\notin$ CFG
4. is-leaf(f') $\rightarrow$ $\forall$ $\langle$tcname, min, max$\rangle \in$ techctx(f')
   $\exists$ $\langle$tcname, min', max'$\rangle \in$ TC : min'≥min $\wedge$ max'≤max □

In other words, CFG is a configuration if it includes the root feature and: (1) if a feature is decomposed, no sub-feature whose business context is not supported is in CFG and, depending on the decomposition type and on the contexts supported by BC, one or more sub-features are in CFG; (2) for any feature f' in CFG, all its required features are in CFG too; (3) if f' is in CFG, all mutually exclusive features are not in CFG; and (4) the technical contexts of leaf features are compatible with TC. The predicate support is defined in our previous work [1] and, roughly, returns true if the formula ($fact_1 \wedge \ldots \wedge fact_n$) gives enough evidence to the truth of bc (as specified via context analysis).

Table 1. Disjunctive datalog rules to enable configurations generation

Id	Rule definition
1	active(F) :- anddecomposed(F), 0=♯count{Fi: dec(F,Fi,Ca), holds(Ca), -active(Fi)}, not noExtraAct(F), dec(F,Fj,Cb), active(Fj), holds(Cb).
2	noExtraAct(F) :- dec(F,Fi,Ca), active(Fi), not holds(Ca).
3	active(F) :- ordecomposed(F), dec(F,Fi,Ca), active(Fi), holds(Ca), not noExtraAct(F).
4	active(F) :- xordecomposed(F), dec(F,Fi,Ca), holds(Ca), active(Fi), not actdiff(F,Fi).
5	actdiff(F,Fi) :- xordecomposed(F), active(Fi), dec(F,Fi,_), dec(F,Fj,_), active(Fj), Fi!=Fj.
6	-active(Fi) :- requires(Fi,Fj), -active(Fj).
7	-active(Fj) v -active(Fi) :- excludes(Fi,Fj).
8	-active(X) :- anddecomposed(X), not active(X).
9	-active(X) :- ordecomposed(X), not active(X).
10	-active(X) :- xordecomposed(X), not active(X).
11	-active(Y) :- dec(X,Y,C), -active(X).
12	holds(TC) :- tc(_,TC), not noPartInactive(TC).
13	noPartInactive(TC) :- tcpart(TC,P,Vmin,Vmax), not istrue(P,Vmin,Vmax).
14	holds(BC) :- anddec(BC), not subUnsat(BC).
15	subUnsat(BC) :- fdec(BC,SUB), not holds(SUB).
16	holds(BC) :- ordec(BC), fdec(BC,SUB), holds(SUB).
17	active(X) v -active(X) :- f(X), tc(X,C), holds(C).
18	active(X) v -active(X) :- f(X), 0=♯count{C: tc(X,C)}.
19	-active(X) :- f(X), tc(X,C), not holds(C).
20	f(X) :- dec(_,X,_), 0=♯count{Z: dec(X,Z,_)}.
21	act(X) :- active(X), f(X).
22	bc(X) :- dec(_,_,X).
23	tch(X) :- tc(_,X), holds(X).
24	bch(X) :- dec(_,_,X), holds(X), X!=true.
25	holds(true).

Based on Definition 1, Table 1 lists a set of rules[4] in disjunctive datalog [7] that form the basis of the analysis techniques which we propose in Sec. 3.2. These rules are a generic framework that, given a set of constraints about technical and business contexts, supports configurations generation.

Rules 1-2 say that an and-decomposed feature is in a configuration (is *active*) if (i) all its sub-features with a holding business context on the respective decomposition branch are active; (ii) no sub-feature whose business context does not hold is active; and (iii) at least one sub-feature is active. Rule 3 handles or-decomposition (at least one sub-feature is active), while rules 4-5 manage xor-decomposition (exactly one sub-feature is active).

Rules 6-7 handle the effect of the requires and excludes relations, respectively: (i) if the required feature is inactive, then the requiring feature has to be inactive too; (ii) one of the mutually exclusive features has to be inactive. Rules 8-10 are technicalities to deal with true negation in disjunctive datalog: if a decomposed feature is not active, then it is inactive. If a decomposed feature is inactive (rule 11), its sub-features has to be inactive too (we support feature trees).

The preconditions of a leaf feature are met (rules 12-13) if all its technical contexts hold. Rules 14-16 deal with business contexts. A business context (or a statement) is and-supported if all statements/facts supporting it hold. A statement is or-supported if at least one statement/fact supporting it holds.

Rules 17-18 say that a leaf feature, in the context of configurations generation, can be either active or inactive if its technical contexts hold (rule 18 handles the case of a leaf feature having no technical context). Rule 19 says that a feature whose technical contexts do not hold has to be inactive.

Rules 20-24 define utility predicates: (i) f for leaf features; (ii) act for active leaf features; (iii) bc for business contexts; (iv) tch for a holding technical context; and (v) bch for a holding business context. A true context always holds (rule 25).

3.2 Reasoning techniques

We present and illustrate reasoning techniques that use the formal framework of Sec. 3.1. These techniques analyze CFMs to answer questions which support the B/I alignment decisions. In the rest of this section, f is the root feature of $\mathcal{M}$, and we list only leaf features in configurations to maximize readability.

Configurations generation

Problem 1 (conf-gen) *Given* TC *and* BC*, return all the configurations* CFG_i *such that* $\mathsf{CFG}_i \vdash_{\mathsf{BC,TC}} \mathcal{M}$.

This is an essential query an analyst would make: given information concerning a prospective deployment environment—the important business contexts to support and the available technical contexts—, which are the candidate configurations of $\mathcal{M}$ that are aligned with such environment? We solve *conf-gen*

[4] We adopt the syntax of DLV [13]: http://www.science.at/proj/dlv/

by extending the datalog formalization of Table 1. For each technical context tc=⟨tcname, min, max⟩ ∈ TC, we add rule istrue(tcname,min,max). For each fact ∈ BC, we add rule holds(fact). The query is active(f)?.

Example 1 (conf-gen). Consider company α that supports browser internet explorer v≥8, php v5, and arcdfstore. α needs advanced visual editing (bc4), has a large catalog (bc5), and more than 10 editors. There are three configurations aligned with the business and technical context of such a company:

- CFG_1={cck, drupal search, word online editor, semantic search}
- CFG_2={cck, drupal search, faceted search, word online editor, semantic search}
- CFG_3={cck, drupal search, faceted search, word online editor}

These configurations share features cck, drupal search, and word online editor, while they differ in the type of search features: faceted, semantic, or both. The analyst can suggest either the configuration with minimal cost (associating a cost with features), the one that maximizes a set of qualities, or the one company α prefers. Previous work by Benavides [2]—which enriches feature models with attributes related to cost and quality and reasons about such models using constraint programming—can be used in conjunction with our framework to rank the derived configurations based on cost and qualities. □

Business-to-IT alignment

Configurations generation is useful to explore the space of configurations that match the business and technical context of an enterprise. We present now techniques to identify an IT infrastructure which accommodates configurations that are aligned with a given business context. Specifically, we address the problems of determining alternative IT infrastructures (Problem 2), choosing a minimal infrastructure that enables all software configurations which are aligned with a certain business context (Problem 3), and identifying the core infrastructure elements supporting at least one configuration (Problem 4). These techniques can be complemented by cost analysis to rank alternative infrastructures.

Problem 2 (bus-to-it) *Given* BC, *return all the configurations and technical contexts* $\langle CFG_i, TC_i \rangle$ *such that* $CFG_i \vdash_{BC,TC_i} \mathcal{M}$.

This problem presumes that the analyst has information about the business context that a customer wants to support. The analyst aims to identify the possible configurations for such business context and to compare the technical prerequisites, so to recommend a configuration to the customer which could be based on what infrastructure is already available in the customer's organization. We solve *bus-to-it* by adding the following rules. Each technical context can be selected or not: istrue(X,Min,Max) v -istrue(X,Min,Max) :- tcpart(Y,X,Min,Max), tc(Z,Y), act(Z); for each fact ∈ BC, add rule holds(fact). The query is active(f)?.

Example 2 (bus-to-it). Through context analysis, company β has identified facts that support business contexts bc4 (advanced editing features) and bc5 (vast product catalog). Given this input, six solutions exist (S1-S6 in Table 2):

Table 2. Solutions to the *bus-to-it* problem of Example 2

Feature / Tech context	S1	S2	S3	S4	S5	S6
f1 = drupal search	✓	✓	✓	✓	✓	✓
f2 = google search	✓	×	✓	×	✓	✓
f3 = faceted search	✓	✓	✓	✓	×	×
f4 = word online editor	✓	✓	×	×	✓	×
f5 = ckeditor	×	×	✓	✓	×	✓
tc1 = firefox	≥3	×	≥3	≥3	≥3	≥3
tc2 = ckeditor	×	×	✓	✓	×	✓
tc1 = ie	≥7	≥7	×	≥6	≥7	≥6

The analyst can now compare the technical contexts to determine the most adequate configuration. He may conduct cost analysis that takes into account the existing IT infrastructure as well as the cost of buying, upgrading, and deploying the infrastructure to satisfy missing technical prerequisites. □

We describe two techniques that provide insights about Business-to-IT alignment, especially when many solutions are identified by solving Problem 2.

Problem 3 (high-variability) *Given* BC*, which is the minimal set of technical contexts that enables the deployment of a high-variability product including all configurations for* BC*?*

The problem is to determine the requirements for a high-variability product, that includes all possible configurations for the business context BC. The deployment of high-variability products is useful either to create *adaptive* systems—which are able to switch to an alternative configuration when the current one fails or is ineffective—or to support *customization* to different users, preferences, and profiles. We solve Problem 3 by post-processing the output of Problem 2: all technical contexts are returned, and the minimum-maximum values should satisfy all solutions. If a technical context appears with value range [3,∞] in a solution, and with range [2,5] in another, the minimal range to solve Problem 3 will be [3,5]. In case such set is inconsistent (e.g. a same technical context appears with ranges [0,2] and [3,4]), manual analysis is required.

Example 3 (high-variability). Take the solution of Example 2. The minimum set of technical contexts for a high-variability product is {ckeditor $[0, \infty]$, firefox $[3, \infty]$, ie $[7, \infty]$} □

Problem 4 (core-tech-ctx) *Given* BC*, identify the core sets of technical contexts. Each core set* CS *enables at least one configuration* CFG *for* $\mathcal{M}$ *(*CFG $\vdash_{\mathsf{BC,CS}} \mathcal{M}$*), and is such that, removing any technical context* tc *from* CS*, there exists no configuration* CFG' *such that* CFG' $\vdash_{\mathsf{BC,CS}\setminus\{\mathsf{tc}\}} \mathcal{M}$.

Each core set of technical contexts defines the minimal requirements for an IT infrastructure supporting business context BC. Notice that many of these sets may exist, each defining a candidate technical environment to support BC. Due to space limitations, we do not describe the algorithm, which is implemented in our CASE tool FM-Context (see Sec. 4).

Example 4 (core-tech-ctx). Take the solution of Example 2. Two core sets that support business contexts bc4 and bc5 are identified: $\mathsf{CS}_1 = \{\mathsf{ie}{\geq}7\}$ and $\mathsf{CS}_2 = \{\mathsf{ie}{\geq}6, \mathsf{ckeditor}, \mathsf{firefox}{\geq}3\}$. If the analyst aims at supporting all business contexts, he will select one or more configurations that support either CS_1 or CS_2, upon obtaining feedback from the customer about the feasibility and cost of each core set of technical contexts. □

IT-to-Business alignment

We propose techniques that, given an IT infrastructure, provide insights about the business contexts supported by such infrastructure. These analyses are useful if a customer is not willing to change its infrastructure, as well as to conduct *what-if* studies about the business efficacy of alternative infrastructures. To improve readability, the output of these techniques is shown in terms of holding business contexts (bc_1, bc_2, ...) rather than of facts.

Problem 5 (it-to-bus) *Given* TC*, return all the configurations and business contexts* $\langle \mathsf{CFG}_i, \mathsf{BC}_i \rangle$ *such that* $\mathsf{CFG}_i \vdash_{\mathsf{BC}_i,\mathsf{TC}} \mathcal{M}$.

The analyst knows the technical infrastructure of the customer, and aims to understand which are the configurations and business contexts that such infrastructure supports. We solve *it-to-bus* by extending the formalization of Table 1. For each technical context, we add a rule istrue(name,min,max). Then, we add a rule stating that every business context has to either hold or not: holds(Y) v -holds(Y) :- bc(Y), and one saying that if the sub-feature on a contextual decomposition branch is not active, then the business context is not active: -holds(Y) :- bc(Y), 0=♯count{X: dec(_,X,Y), active(X)}. The query is active(f)?.

Example 5 (it-to-bus). Company γ has many customers with legacy web browsers. As a consequence, γ wants to make as few assumptions as possible about the browser customers use: its technical infrastructure includes only php v5 and arcdfstore. Given this technical context, six solutions exist (see Table 3):

Table 3. Solutions to the *it-to-bus* problem of Example 5

Feature / Business context	S1	S2	S3	S4	S5	S6
f1 = drupal search	✓	✓	✓	✓	✓	✓
f2 = drupal content editor	✓	✓	✓	✓	✓	✓
f3 = faceted search	×	✓	✓	×	✓	×
f4 = cck	×	×	✓	✓	✓	✓
f5 = semantic search	×	×	✓	✓	×	×
bc1 = custom content needed	×	×	✓	✓	✓	✓
bc2 = no visual editing needed	✓	✓	✓	✓	✓	✓
bc5 = large product catalog	×	✓	✓	✓	✓	×

The solutions are quite different one from another. For example, S1 supports only business context bc2, while solutions S3-S5 support bc1, bc2, and bc5. Notice that, given the technical contexts in input, business contexts bc3 and bc4 are never supported. If those contexts are important to the management of γ, the analysis could be repeated by extending the technical infrastructure. That will, however, require γ to impose more technical prerequisites to its customers. □

Problem 6 (bus-support) *Given* TC *and a ranking* R=bc_i>bc_j>bc_k>... *that defines the relative priority of the business contexts in* $\mathcal{M}$*, which are all the supported maximal sets of business contexts? Which is the relative order of these maximal sets according to* R*?*

The problem is to identify the maximal sets of business contexts supported by the technical infrastructure. Since there may be many of these sets, the analyst is asked to rank the relative priority of all the business contexts in BC. This problem is solved by post-processing the output of Problem 5 and returning all maximal sets of business contexts (those not included in any other set). These sets are ranked according to R: for any constraint bc>bc', a set including bc and not including bc' is ranked before any set including bc' and not including bc.

Example 6 (bus-support). Company θ supports only technical context ie v≥7; let R=bc1>bc4>bc3>bc5>bc2. There are two maximal sets: {bc1, bc5, bc4} and {bc1, bc2, bc5}. These maximal sets allow for supporting either bc2 (no visual editing needed) or bc4 (advanced editing needed), but not both. Given that R states that bc4 is preferred to bc2, the former maximal set is recommended. □

Our reasoning techniques are a basic toolset to address problems about B/I alignment in software configuration, and can be complemented with cost analysis, elaborated quality-based reasoning, and what-if analysis.

4 FM-Context: a support tool for CFMs

FM-Context is our developed CASE tool for CFMs. The tool enables the graphical creation of CFMs, and implements algorithms to solve the Problems 1-6 of Sec. 3. FM-Context is an Eclipse Rich Client Platform application—built on Eclipse Galileo 3.5.2—implemented according to the meta-model driven development framework provided by the Eclipse Graphical Modeling Project[5]. Its

[5] http://www.eclipse.org/modeling/gmp/

reasoning engine is based on the disjunctive datalog solver DLV. FM-Context currently supports Windows and Linux machines both 32 and 64 bits. FM-Context is freely available from the web at http://goo.gl/wx3Vl.

Figure 3 shows a screen-shot of FM-Context while prompting the user for input to solve the *bus-to-it* problem. FM-Context allows for a more flexible input than that described in Problem 2: the analyst can choose also non-analyzed business contexts. In the foreground window, the analyst has selected not only active facts, but also non-analyzed contexts bc3 and bc4. These selectable lists are populated automatically by analyzing the CFM at hand (in the background window). The central part of the background window is occupied by the contextual feature model of Fig. 2. On the right side is the drawing palette that allows for adding elements and relations to the diagram canvas. At the bottom are details about the output obtained by a previously ran automated analysis.

Fig. 3. Screen-shot of FM-Context: the user is prompted for the input to Problem 2

In Table 4 we outline preliminary results about the scalability of the automated reasoning in FM-Context. Specifically, we ran the configurations generation technique (to solve Problem 1) on CFMs of growing size. Starting from the Drupal CFM, we constructed five CFMs of different sizes (in terms of features *Feat*, statements *Stat*, facts *Fact*, and technical contexts *Tech*) and hypothesized that all contexts (business and technical) hold; for each CFM, we executed four tests: (i) *1-var*: all decompositions are of type "and" (a single variant); (ii) *Low-var*: prevalence of xor-decompositions; (iii) *Mid-var*: a balanced number of xor- and or-decompositions; and (iv) *High-var*: all or-decompositions.

Table 4. Preliminary performance evaluation of the configurations generation reasoning with FM-Context. Time in milliseconds

					1-var	Low-var			Mid-var			High-var		
Feat	Stat	Fact	Tech	Nodes	Time	Vars	Time	$\frac{time}{var}$	Vars	Time	$\frac{time}{var}$	Vars	Time	$\frac{time}{var}$
7	4	5	2	18	16	7	22	3.14	9	24	2.67	31	31	1.00
16	10	13	6	45	20	127	59	0.47	319	101	0.31	2047	487	0.24
24	15	17	8	64	25	251	96	0.38	1799	508	0.28	18431	6712	0.36
32	20	26	12	90	33	1023	648	0.63	8191	3400	0.42	32767	19968	0.61
64	40	52	24	180	51	3615	2954	0.81	14415	12518	0.86	54960	66048	1.21

For each experiment we report the number of variants (*Vars*), overall time (*Time*), and time per variant (*time/var*). Since experiment *1-var* includes only one variant, we show only overall time. We repeated each test three times and present the average time in Table 4. The results of our evaluation are promising: the main criteria to increase reasoning time is the number of variants, and not

the number of nodes. In a realistic environment, moreover, the analyst would not typically obtain so many variants as in our example, given that the deployment environment would constrain the possible configurations. A CFM with much non-contextual variability would make the configuration task very difficult.

5 Related work

The relation between context and features has been studied in literature. Lee and Kang [12] propose to analyze usage contexts to derive the factors—qualities and technical constraints—that drive feature selection. They rely on a set of feature diagrams that represent the product line, usage context, qualities, and technical constraints. Similarly, Hartmann and Trew [8] complement feature diagrams with a context variability model, so to support multiple product lines. The two models are linked via constraints (the context constrains applicable features). These models are merged into a feature model for a specific context. We share the same motivation and vision. However, we rely upon richer conceptual models to represent context: business contexts are specified via context analysis, while technical contexts are expressed as prerequisites to leaf features. Moreover, our reasoning helps maximizing B/I alignment during product configuration.

Tun et al. [16] address the configuration problem based on a refinement approach that consists of multiple feature models, which analyze requirements, problem world contexts, specifications, and expression of quantitative constraints. We do not focus on the refinement process here: our purpose is to identify and reason about the relationships between features, business, and technical contexts.

The original feature modeling notation of FODA [10] does not explicitly support context modeling. However, it documents decision points with "issues", specific questions that an analyst should answer in order to choose the most adequate configuration. Thurimella et al. [17] extend FODA by adopting a rationale management framework to express issues associated with decision points. Our approach goes one step further and uses a formal—as opposed to a textual—language to document such rationale; moreover, we focus on B/I alignment.

Czarnecki et al. [5] propose a method—called staged configuration—wherein the best configuration is achieved via stepwise specialization of feature models, with multiple people involved in such process. This approach could be used in conjunction with CFMs to guide the selection process.

Our approach is in the spirit of supporting automated reasoning on feature models (Benavides et al. [2]). They propose a framework that supports attribute-extended feature models to express technical constraints. Unlike them, we consider the business context and propose a different set of reasoning mechanisms.

In their empirical study, Chan et al. [4] provide valuable insights about the relationship between planning (among which, information systems configuration), and B/I alignment. In particular, their field study evidences that planning sophistication does not directly improve B/I alignment, while it improves shared domain knowledge. Such knowledge, in turn, improves B/I alignment. There exist s, thus, an indirect relationship between planning and alignment. While

CFMs provide directions about the B/I alignment problem, their output still necessitates interpretation by analysts and domain experts.

In previous work [1], we have proposed contextual goal models to represent and reason about the interplay between requirements at the intentional level and context. That modeling framework helps to detect whether there exists a set of system requirements that satisfies the goals of stakeholders in a specific context. Differently, CFMs apply when a software product family already exists, and help to derive a configuration that fits well in a prospective deployment environment.

6 Conclusion and future directions

In this paper, we introduced contextual feature models, which enrich traditional feature models with information about contextual constraints—both at the business and at the technical level. Importantly, we formally represent the interplay between contexts and features. Our automated reasoning techniques can be exploited to configure the product line while maximizing B/I alignment. Both modeling and reasoning are supported by our CASE tool FM-Context.

Our approach is a means to support analysts in selecting a configuration that fits well with a prospective deployment environment. CFMs can be used as a sophisticated planning tool that increases shared domain knowledge [4]. However, they do not replace the role of designers. The suggestions about configurations and contexts of our analyses are used by analysts and domain experts, who will take the final decision about which configuration to deploy.

This paper opens the doors to several research directions that will be part of our future work: (i) devise a methodology analysts can use to fully benefit from CFMs; (ii) refine and extend the modeling notation (e.g. support cardinality and more complex technical contexts); (iii) develop and implement new algorithms to support analysts; and (iv) evaluate the applicability of the approach by conducting experimentation on industrial case studies.

Acknowledgment

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grants no 257930 (Aniketos), 256980 (Nessos), and 258109 (FastFix), and by the Science Foundation Ireland under grant 10/CE/I1855.

References

1. Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini. A Goal-based Framework for Contextual Requirements Modeling and Analysis. *Requirements Engineering*, 15(4):439–458, 2010.

2. David Benavides, Pablo Trinidad, and Antonio Ruiz-Corts. Automated Reasoning on Feature Models. In Oscar Pastor and Joo Falco e Cunha, editors, *Proceedings of the 17th International Conference on Advanced Information Systems Engineering (CAiSE'05)*, volume 3520 of *Lecture Notes in Computer Science*, pages 381–390. Springer Berlin / Heidelberg, 2005.
3. Jan Bosch. *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach*. Addison-Wesley Professional, 2000.
4. Yolande E. Chan, Rajiv Sabherwal, and Jason B. Bennet Thatcher. Antecedents and outcomes of strategic IS alignment: an empirical investigation. *IEEE Transactions on Engineering Management*, 53(1):27–47, 2006.
5. Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. Staged Configuration Using Feature Models. In Robert Nord, editor, *Software Product Lines*, volume 3154 of *Lecture Notes in Computer Science*, pages 162–164. Springer Berlin / Heidelberg, 2004.
6. Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. Formalizing Cardinality-based Feature Models and their Specialization. *Software Process: Improvement and Practice*, 10(1):7–29, 2005.
7. Thomas Eiter, Georg Gottlob, and Heikki Mannila. Disjunctive Datalog. *ACM Transactions on Database Systems*, 22(3):364–418, 1997.
8. Herman Hartmann and Tim Trew. Using Feature Diagrams with Context Variability to Model Multiple Product Lines for Software Supply Chains. In *Proceedings of the 12th International Software Product Line Conference (SPLC'08)*, pages 12–21, 2008.
9. John C. Henderson and N. Venkatraman. Strategic alignment: Leveraging information technology for transforming organizations. *IBM Systems Journal*, 32(1):4–16, 1993.
10. Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. Spencer Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Carnegie Mellon University, 1990.
11. Kyo C. Kang, Sajoong Kim, Jaejoon Lee, Kijoo Kim, Euiseob Shin, and Moonhang Huh. FORM: A Feature-oriented Reuse Method with Domain-specific Reference Architectures. *Annals of Software Engineering*, 5:143–168, 1998.
12. Kwanwoo Lee and Kyo Kang. Usage Context as Key Driver for Feature Selection. In Jan Bosch and Jaejoon Lee, editors, *Proceedings of the 14th International Software Product Lines Conference (SPLC'10)*, volume 6287 of *Lecture Notes in Computer Science*, pages 32–46. Springer Berlin / Heidelberg, 2010.
13. Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV System for Knowledge Representation and Reasoning. *ACM Trans. Comput. Log.*, 7(3):499–562, 2006.
14. Joe Peppard and John L. Ward. "Mind the Gap": Diagnosing the Relationship between the IT Organisation and the Rest of the Business. *The Journal of Strategic Information Systems*, 8(1):29–60, 1999.
15. Pierre-Yves Schobbens, Patrick Heymans, and Jean-Christophe Trigaux. Feature Diagrams: A Survey and a Formal Semantics. In *Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)*, pages 136–145, 2006.
16. Thein Than Tun, Quentin Boucher, Andreas Classen, Arnaud Hubaux, and Patrick Heymans. Relating Requirements and Feature Configurations: a Systematic Approach. In *Proceedings of the 13th International Software Product Line Conference (SPLC'09)*, pages 201–210, 2009.

17. Anil Kumar Thurimella, Bernd Bruegge, and Oliver Creighton. Identifying and Exploiting the Similarities between Rationale Management and Variability Management. In *Proceedings of the 12th International Software Product Line Conference (SPLC'08)*, pages 99–108, 2008.