Feature specification and static analysis for interaction resolution

Marc Aiguier, Karim Berkani, Pascale Le Gall

Feature specification and static analysis for interaction resolution

Marc Aiguier[1], Karim Berkani[2] and Pascale Le Gall[1]*

[1] Université d'Évry, LaMI CNRS UMR 8042,
523 pl. des Terrasses F-91000 Évry
email : {aiguier,legall}@lami.univ-evry.fr
[2] Laboratoire Heudiasyc, UMR CNRS 6599
BP 20529 60205 Compiègne Cedex
email kberkani@hds.utc.fr

Abstract. While designing a service-oriented system, deciding whether a service interaction is desired or harmful is a subjective choice which depends on the requirements expressed by the user with respect to the service integration. In this paper, we define both a formalism and a methodology which, respectively, allow us to automatically analyse interactions based on specification consistency. For the latter (i.e. the methodology), we take advantage of both specifier expertise and formal methods.
Keywords: pre-post formalism, specification consistency, static analysis, feature integration, feature interaction resolution.

1 Introduction

The work presented in this paper was performed within the French project ValiServ[3] in collaboration with the French telecommunication company FranceTelecom and the LSR team of the university J. Fourier of Grenoble [8]. This project was devoted to service (feature) design for telecommunication purposes. The aim of this project was to better answer both feature system specification and the underlying problems: *feature integration* and *feature interactions*. Indeed, software telecommunication systems are composed of a kernel providing the basic expected functionalities and a set of satellite entities, called features[4]. Each of them aims to modify the set of functionalities characterising the rest of the system (possibly including other already existing features). This project also aimed to develop an assistant tool for integrating new phone services. The interest was to provide support for rapid service-oriented development which is an important issue, especially for telecommunication operators. Indeed, the primary motivation to offer numerous features to users is that the set of offered features differentiates providers, and then becomes a significant source of income. However, if some

* this work was partially supported by the RNRT French project VALISERV and by the European Commission under WGs Fireworks (23531)
[3] the acronym of which means "Validation de Services".
[4] In the following, we will indifferently use the two words feature and service although we are aware of that services also represent more particularly the notion of components such as web services.

behaviours of a telecommunication system do not conform to some feature descriptions offered to customers, this may have calamitous effects on the public image of the concerned provider.

The paper is the continuation of the works developed in [3, 4] by giving the theoretical basis of the methodology and the tool presented respectively in [4] and in [3]. This will be briefly recalled in Section 4. Our purpose is then to formally define an integration methodology allowing to solve interactions resulting from an inconsistent integration of a feature in a system specification, according to expert's point of view. The theoretical foundations will be based on algorithms the correctness of which will be proved (see Theorem 2 and Theorem 3). These algorithms deal with specification consistency. More precisely, interactions are properties which are violated. They may be qualified as desirable or not by an expert who can modify both the considered property, and integration choices to make service integration conform to its judgement. Thus, interaction resolution takes care of interactions which may be introduced during the integration process. To ease the service design, we define an axiomatic formalism (i.e. system behaviour is specified by logical properties) which will be used for detection and resolution. This formalism aims to specify telecommunication systems viewed along phone services at which customers can subscribe. Now, both formalism and methodology can be obviously extended and applied to specify and automatically analyse interaction in systems viewed along services (not necessarily phone services) at which objects can subscribe (e.g. lifts equipped with different services such as the basic service and the service which indicates a priority floor).

The methodology presented in this paper will then take advantage of designer's expertise with an interactive integration activity assisted by static analysis of specification consistency. This static analysis will be based on symbolic techniques dealing with phone variables to deduce the appropriate subscription configuration. Hence, the formalism defined in this paper will manipulate state transition rules (str), invariants and inequations between phone variables. The formalism developed in the paper is then a simple restriction of classic pre -post logical language. The interest of such a language is twofold:

1. it allows to automatically detect inconsistencies after integrating a new feature in a system specification. This is precisely the main goal of the present paper.
2. its syntax is very simple up to some syntactical sugar[5]. Hence, specifications are made readable for the expert what will ease his(her) choices to circumvent inconsistencies. Besides, this has been experimented in the ValiServ project with some experts of our partner France Telecom.

The paper is structured as follows. Section 2 presents the formalism and the notion of specification consistency on which our interactions are based on. Specifications are provided in the form of invariant properties and state transition rules, very much as in [20]. Examples are also provided. For lack of space, we present a simple and pedagogical example which only integrates three features on a basic system. More generally, our method can deal with all services which can be expressed within our specification

[5] which will not be presented in the paper in order not to make heavy the presentation. The interested readers can find them in [3, 10].

formalism and integrated on the intelligent network (see [2] and [14] for more than 20 examples of such service specifications). Section 3 details the algorithms used to check specification consistency in the integration process. Section 4 presents the methodology and our results on usual services using the tool [3] developed in the ValiServ project. Finally, related works are addressed in Section 5.

By lack of space, most of the proofs of propositions and theorems are not given in this paper. However, they can be found in the preliminary version of this paper [1]. Only the proof of Theorem 1 is given because the algorithms described in this paper are based on it.

2 Service specification

Here, we define a formalism dedicated to service (so-called feature) telecommunication systems. Services will be specified along two types of predicates, *subscription* and *status*. By the former, we will specify what and how customers subscribe to a service. For instance[6], $TCS(x, y)$ will mean that x has subscribed to TCS and any call from y to x is forbidden. By the latter, we will specify communication situations such as to be busy, idle, etc... Moreover, telecommunication systems are dynamic systems. Therefore, in order to automatically analyse interactions, the formalism under definition will manipulate sentences of the form $(pre, event, post)$ where pre and $post$ will be finite sets of atoms denoting respectively pre and post-conditions and $event$ will be an event triggering side-effect. Moreover, some invariants roughly defined by free-quantifier first-order formulas (i.e. state-evolution independent) will be stated.

2.1 Syntax

The formalism is devoted to specify features in telecommunication systems. Its syntax is closely related to the one developed in [10]. Vocabularies (so-called signatures) over which pre and post-conditions and invariants will be built on, will then contain two types of predicates: status and subscription predicates.

Definition 1 (Signature). *A* signature Σ *is a triple* (St, Sb, E) *where* St *and* Sb *are two sets of predicates names, and* E *is a set of events names. Each element in* $St \cup Sb \cup E$ *are equipped with an arity* $n \in \mathbb{N}$. St, Sb *and* E *are disjoint sets.*
A signature is said finite *when both* St, Sb *and* E *are finite sets. An element* $p \in St \cup Sb \cup E$ *equipped with the arity* n *is noted* p^n.
St *and* Sb *contain respectively,* status *and* subscription *predicates.*

Note, by the definition of signatures, that variables are the only allowed arguments for the predicates and the events. Hence, variables will necessarily denote terminals.

Systems will be specified by means of two kinds of formulas: State transition rules (str) and Invariants. Moreover, as we are interested by automatically analysing interactions (which will be defined by properties), manipulated formulas will be constrained on their form.

[6] TCS is an usual acronym for the Terminating Call Screening.

Notation 2 *Let $\Sigma = (St, Sb, E)$ be a signature. Let X be a set of variables. Note $At_\Sigma(X)$ and $\overline{At}_\Sigma(X)$ the two sets defined by:*

1. $At_\Sigma(X) = \{p(x_1, \ldots, x_n) \mid p^n \in St \cup Sb, x_i \in X,\ 1 \leq i \leq n\}$
2. $\overline{At}_\Sigma(X) = \{\neg p(x_1, \ldots, x_n) \mid p^n \in St \cup Sb, x_i \in X,\ 1 \leq i \leq n\}$

Note $Sb_\Sigma(X)$ and $\overline{Sb}_\Sigma(X)$ (resp. $St_\Sigma(X)$ and $\overline{St}_\Sigma(X)$) the two subsets of $At_\Sigma(X)$ and $\overline{At}_\Sigma(X)$ restricted to predicates in Sb (resp. in St).

Definition 3 (Formulas). *Let $\Sigma = (St, Sb, E)$ be a signature. Let X be a set of variables.*

1. *A* str-formula *over Σ is a sentence of the form $< ctr | subs : pre \overset{e(x_1,\ldots,x_n)}{\longrightarrow} post >$ where:*
   - *ctr is a set of inequations $x \neq y$ with $x, y \in X$,*
   - *$subs \subseteq Sb_\Sigma(X) \cup \overline{Sb}_\Sigma(X)$,*
   - *$pre, post \subseteq St_\Sigma(X) \cup \overline{St}_\Sigma(X)$ are two finite sets, and*
   - *$e^n \in E$ and $x_i \in X$ for $1 \leq i \leq n$.*
2. *An* invariant *over Σ is a sentence of the form $< ctr | \varphi >$ where ctr is defined as above, and φ is a quantifier-free first-order formula over $St \cup Sb$.*

In the sequel, quantifier-free first-order formulas will be simply called formulas. We will note $Var(x)$ the set of variables occurring in $x \in \{ctr, subs, pre, post, \varphi\}$.

We have chosen to separate in str-formulas, subscription atoms from pre and post-conditions because events do not modify subscriptions. Hence, subscriptions are necessarily preserved along transitions.

Definition 4 (Service specification). *A* service specification *$\mathcal{F}$ is a 2-tuple (Σ, Ax) where Σ is a signature and Ax is a set of str-formulas and invariants over Σ.*
$\mathcal{F}$ is said finite if both Σ and Ax are a finite signature and finite set of axioms, respectively. In the sequel Ax will be also noted $STR \coprod I$. STR and I will then contain all the str-formulas and invariants, respectively, of Ax.

2.2 Examples

We now provide examples: the specifications of the basic telecommunication system, classically called POTS, and of three common services destined to be plugged on it. The different components of the specifications will be indexed by the specification name. Moreover, elements of the underlying system POTS are implicitly present for the specification of the three services.

example 1: POTS, the Plain Old Telephone Service
St_{POTS} contains $idle(x)$ ("x is idle"), $dialwait(x)$ ("x is in dial waiting state"), $caller(x, y)$ (resp. $callee(x, y)$) ("x is in communication with y as the caller (resp. callee) part"), $ringing(x, y)$ ("x is ringing from the caller y"), $hearing(x, y)$ ("x is hearing the tone of the call to y"), $busytone(x)$ ("x is hearing the busy tone"). By convention, Sb_{POTS} is empty since by default all phones are supposed to subscribe to

the basic service POTS. E_{POTS} contains $offhook(x)$ meaning that x is hooked off, $onhook(x)$ (x is hooked on), $dial(x, y)$ (x dials y). STR_{POTS} contains:

$\phi_1 : \ < \mid idle(A) \stackrel{offhook(A)}{\longrightarrow} dialwait(A) >$
$\phi_2 : \ < A \neq B \mid dialwait(A), idle(B) \stackrel{dial(A,B)}{\longrightarrow} hearing(A, B), ringing(B, A) >$
$\phi_3 : \ < \mid dialwait(A), \overline{idle(B)} \stackrel{dial(A,B)}{\longrightarrow} busytone(A) >$
$\phi_4 : \ < A \neq B \mid hearing(A, B), ringing(B, A) \stackrel{offhook(B)}{\longrightarrow} caller(A, B), callee(B, A) >$
$\phi_5 : \ < A \neq B \mid caller(A, B), callee(B, A) \stackrel{onhook(A)}{\longrightarrow} idle(A), busytone(B) >$
$\phi_6 : \ < A \neq B \mid caller(A, B), callee(B, A) \stackrel{onhook(B)}{\longrightarrow} idle(B), busytone(A) >$
$\phi_7 : \ < A \neq B \mid hearing(A, B), ringing(B, A) \stackrel{onhook(A)}{\longrightarrow} idle(A), idle(B) >$
$\phi_8 : \ < \mid busytone(A) \stackrel{onhook(A)}{\longrightarrow} idle(A) >$
$\phi_9 : \ < \mid dialwait(A) \stackrel{onhook(A)}{\longrightarrow} idle(A) >$

I_{POTS} contains several invariants expressing that status predicates are mutually exclusive when they concern the same variables. For example, it contains:
$< B \neq C \mid \neg(talking(A, B) \wedge talking(A, C)) >$
$< \mid \neg(idle(A) \wedge talking(A, B)) >$
For lack of space, we do not give all such invariants. However, they can be found in [2].

POTS characterises the behaviour of a terminal which has just subscribed to the basic telephone service, when communicating with another terminal with the same subscription. For example, ϕ_5 says that if the call initiator hangs up during a communication, then his party gets a busy tone.

example 2: *TCS*, Terminating Call Screening (this service screens out incoming calls from terminals belonging to the *TCS* subscriber's black list).

Sb_{TCS} contains $Tcs(y, x)$: calls from x to y are forbidden by y. I_{TCS} contains
$\psi_1 : \ < A \neq B \mid Tcs(A, B) \Rightarrow \neg hearing(B, A) >$
while STR_{TCS} contains
$\psi_2 : \ < \mid Tcs(B, A), dialwait(A), idle(B) \stackrel{dial(A,B)}{\longrightarrow} busytone(A) >$

example 3: *CFB*, Call Forward on Busy (this service allows a subscriber to forward all incoming calls to a designated terminal, when the subscriber's terminal is busy).

Sb_{CFB} contains $Cfb(x, y)$: when x is not idle, forward incoming calls to y. I_{CFB} contains
$\chi_1 : \ < \mid \neg Cfb(A, A) >$
$\chi_2 : \ < B \neq C \mid Cfb(A, B) \Rightarrow \neg Cfb(A, C) >$
and STR_{CFB} contains
$\chi_3 : \ < B \neq C \mid Cfb(B, C), dialwait(A), \overline{idle(B)}, idle(C) \stackrel{dial(A,B)}{\longrightarrow} hearing(A, C), ringing(C, A) >$
$\chi_4 : \ < B \neq C \mid Cfb(B, C), dialwait(A), \overline{idle(B)}, \overline{idle(C)} \stackrel{dial(A,B)}{\longrightarrow} busytone(A) >$

example 4: *INTL*, IN Teen Line (this service allows a user to restrict outgoing calls during a specified daily period. The restriction can be over-ridden by entering a pin. If the given pin is the right one, then a normal call can be initiated, else the user is requested to abort his call.)

S_{INTL} contains S_{POTS} and specific predicates: $time(x)$ characterises the time slot where a pin is required from the user x to perform outgoing calls, $waitpin(x)$ means that the user x should now dial its personal pin, and $invalid(x)$ means that the dialled pin is not valid. Sb_{INTL} contains *Intl*(x): x is subscribing for the *INTL* service.

E_{INTL} contains two new events related to the pin dialling: $dialgoodpin(x)$ for "x is dialling the expected correct pin", and $dialbadpin(x)$ for "x is dialling a wrong pin". I_{INTL} contains new invariants expressing that the status $invalid$ and $waitpin$ are exclusive with the POTS status $idle$, $dialing$, ... and are also mutually exclusive. STR_{INTL} contains:

κ_1 : $< \mid Intl(A), time(A), idle(A) \stackrel{offhook(A)}{\longrightarrow} waitpin(A), time(A) >$

κ_2 : $< \mid waitpin(A) \stackrel{dialgoodpin(A)}{\longrightarrow} dialwait(A) >$

κ_3 : $< \mid waitpin(A) \stackrel{dialbadpin(A)}{\longrightarrow} invalid(A) >$

κ_4 : $< \mid invalid(A) \stackrel{onhook(A)}{\longrightarrow} idle(A) >$

κ_5 : $< \mid waitpin(A) \stackrel{onhook(A)}{\longrightarrow} idle(A) >$

Specifications are restricted to service specificities. They implicitly refer to the underlying system. For example, the TCS specification contains a service invariant characterising a newly prohibited situation (the subscriber terminal cannot be put in communication with a terminal from its screening list) and a limited behavioural description (what happens when a forbidden terminal attempts to call the subscribing terminal).

2.3 Semantics

Definition 5 (Models). *Let $\Sigma = (St, Sb, E)$ be a signature.*

*A Σ-*model *$\mathcal{A} = (U, S, (e^{\mathcal{A}})_{e \in E})$ is a set U (terminals) and a set $S \subseteq \mathcal{P}(At_{\Sigma}(U))$ (states) equipped for every $e^n \in E$ and every $(u_1, \ldots, u_n) \in \underbrace{U \times \ldots \times U}_{n \text{ times}}$ with a bi-nary relation*[7] *$e^{\mathcal{A}}(u_1, \ldots, u_n) \subseteq S \times S$.*
$\mathcal{A}$ is deterministic *if and only if for every $e^n \in E$ and every $(u_1, \ldots, u_n) \in \underbrace{U \times \ldots \times U}_{n \text{ times}}$, $e^{\mathcal{A}}(u_1, \ldots, u_n)$ is a partial function.*

Definition 6 (Formula satisfaction). *Let $\Sigma = (St, Sb, E)$ be a signature. Let $\mathcal{A}$ be a Σ-model. A state $s \in S$ and an interpretation $\iota : X \to U$* satisfy *a formula φ, noted $(\iota, s) \models \varphi$, if and only if:*

- $(\iota, s) \models p(x_1, \ldots, x_n) \Longleftrightarrow p(\iota(x_1), \ldots, \iota(x_n)) \in s$
- *propositional connectives are handled as usual.*

$\mathcal{A}$ satisfies *a formula φ, noted $\mathcal{A} \models \varphi$, if and only if for every $s \in S$ and every $\iota : X \to U$, $(\iota, s) \models \varphi$.*

Definition 7 (Transition satisfaction). *A Σ-model $\mathcal{A}$* satisfies *for $s \in S$ and $\iota : X \to U$ a str-formula φ of the form $< ctr | subs : pre \stackrel{e(x_1, \ldots, x_n)}{\longrightarrow} post >$, noted $\mathcal{A} \models_{\iota, s} \varphi$, if and only if, if for every $x \neq y \in ctr$, $\iota(x) \neq \iota(y)$ then:*

[7] We note $s \; e^{\mathcal{A}}(u_1, \ldots, u_n) \; s'$ to mean that $(s, s') \in e^{\mathcal{A}}(u_1, \ldots, u_n)$.

$$\textit{if } (\iota, s) \models \bigwedge_{\alpha \in subs \cup pre} \alpha \textit{ then } \forall s' \in S, s\ e^{\mathcal{A}}(\iota(x_1), \ldots, \iota(x_n))\ s' \Rightarrow (\iota, s') \models \bigwedge_{\alpha \in post} \alpha)$$

Definition 8 (Invariant satisfaction). *A Σ-model $\mathcal{A}$* satisfies *for $s \in S$ and $\iota : X \to U$ an invariant $< ctr|\varphi >$, noted $\mathcal{A} \models_{\iota,s} < ctr|\varphi >$, if and only if, if for every $x \neq y \in ctr, \iota(x) \neq \iota(y)$ then $(\iota, s) \models \varphi$*

Definition 9 (Specification satisfaction). *A Σ-model $\mathcal{A}$* satisfies *a service specification $\mathcal{F} = (\Sigma, STR \coprod I)$ if and only if it satisfies for every $s \in S$ and every $\iota : X \to U$ each formula of Ax.*
A service specification is said consistent *if and only if there exists a non-empty Σ-model $\mathcal{A}$ which satisfies it and such that the cardinality of its set U of terminals satisfies:*

$$|U| \geq max\{Var(ctr) | \exists < ctr|subs : pre \xrightarrow{e} post > \in STR \vee \exists < ctr|\varphi > \in I\}$$

The last condition on the carrier cardinality of Σ-models prevents trivial Σ-models. A trivial Σ-model is such that the number of terminals in U is not sufficient to satisfy each inequation occurring in the ctr part of each formula in STR and I.

2.4 Fundamental results

We first define a Σ-model which will be useful to us in the next section. Let $\Sigma = (St, Sb, E)$ be a signature. Let U and $S \subseteq \mathcal{P}(St_\Sigma(U))$ be two sets of terminals and states, respectively. Let STR be a set of str-formulas over Σ. Therefore, define the Σ-model $\mathcal{G}(U, S) = (U, S', (e^{\mathcal{G}(U,S)})_{e \in E})$ as follows:

- S' is the set inductively defined by $S' = \bigcup_{i<\omega} S_i$ with:
  - $S_0 = S$
  - $s' \in S_n \iff \begin{cases} \exists < ctr|subs : pre \xrightarrow{e(x_1,\ldots,x_n)} post > \in STR, \exists \iota : X \to U, \\ \exists s \in S_{n-1}, (\forall x \neq y \in ctr,\ \iota(x) \neq \iota(y)) \wedge \\ (\iota, s) \models \bigwedge_{\alpha \in subs \cup pre} \alpha \wedge \\ s' = (s \setminus \{p(\iota(y_1), \ldots, \iota(y_m)) | \neg p(y_1, \ldots, y_m) \in post\}) \\ \qquad \cup \\ \qquad \{p(\iota(y_1), \ldots, \iota(y_m)) | p(y_1, \ldots, y_m) \in post\} \end{cases}$
- For every $e^n \in E$ and every $(u_1, \ldots, u_n) \in \underbrace{U \times \ldots \times U}_{n \text{ times}}$, $e^{\mathcal{G}(U,S)}(u_1, \ldots, u_n)$ is defined: $s\ e^{\mathcal{G}(U,S)}(u_1, \ldots, u_n)\ s' \iff$

$$\begin{cases} \exists < ctr|subs : pre \xrightarrow{e(x_1,\ldots,x_n)} post > \in STR, \exists \iota : X \to U, \\ (\forall 1 \leq j \leq n, \iota(x_j) = u_j) \wedge \\ (\forall x \neq y \in ctr,\ \iota(x) \neq \iota(y)) \wedge \\ (\iota, s) \models \bigwedge_{\alpha \in subs \cup pre} \alpha \wedge \\ s' = (s \setminus \{p(\iota(y_1), \ldots, \iota(y_m)) | \neg p(y_1, \ldots, y_m) \in post\} \\ \qquad \cup \\ \qquad \{p(\iota(y_1), \ldots, \iota(y_m)) | p(y_1, \ldots, y_m) \in post\}) \end{cases}$$

Let us point out that when X, U, S and STR are finite sets, then $\mathcal{G}(U,S)$ is computable. Let us consider Σ a signature, X a set of variables over Σ and I a set of invariants. Define

$$E_\Sigma(X) = \{s \subseteq At_\Sigma(X) |\ \forall \iota : X \to X, \forall < ctr|\varphi > \in I, \\ (\forall x \neq y \in ctr, \iota(x) \neq \iota(y)) \Rightarrow (s,\iota) \models \varphi)\}$$

then define $I_\Sigma(X) = \{s \in E_\Sigma(X) |\ \nexists s' \in E_\Sigma(X), s' \subseteq s\}$.

Proposition 1. *When Σ is a finite signature and X and I are finite sets, then $E_\Sigma(X)$ and $I_\Sigma(X)$ are computable.*

Theorem 1. *Let $\mathcal{F} = (\Sigma, STR \coprod I)$ be a service specification. $\mathcal{F}$ is consistent if and only if $\mathcal{G}(X, I_\Sigma(X))$ satisfies all the axioms of $\mathcal{F}$.*

Proof. The *if part* is obvious.

The only if part. Suppose that $\mathcal{F}$ is consistent but $\mathcal{G}(X, I_\Sigma(X))$ does not satisfy it.

Obviously, the consistency of $\mathcal{F}$ means the consistency of STR and of I. By construction, the consistency of I implies that $I_\Sigma(X)$ is not empty. In the following, the question of (the verification of) the consistency of I will be simply denoted by **InvCons**.

Therefore, if $\mathcal{G}(X, I_\Sigma(X))$ does not satisfy $\mathcal{F}$ then by construction of $\mathcal{G}(X, I_\Sigma(X))$ which relies on str-formulas, either two str-formulas with the same event lead to two incompatible states or a str-formula leads to a state violating the invariants. These two cases are denoted by respectively **NonDet** for non-deterministic str-formulas and **ViolInv** for the non preservation of the invariants by str-formulas. Then, let us prove that both **NonDet** and **ViolInv** lead to a contradiction.

1. **NonDet** there exists[8] $\psi =< ctr|subs : pre \overset{e(x_1,\ldots,x_n)}{\longrightarrow} post >\in STR$, $s \in S'$ and $\iota : X \to X$ such that for every $x \neq y \in ctr$, $\iota(x) \neq \iota(y)$, $(s,\iota) \models \bigwedge_{\alpha \in subs \cup pre} \alpha$, but there exists s' and[9] $p(y_1,\ldots,y_m) \in post$ such that $s\ e^{\mathcal{G}(X,I_\Sigma(X))}(\iota(x_1),\ldots,\iota(x_n))\ s'$ and $p(\iota(y_1),\ldots,\iota(y_m)) \notin s'$. By definition, this means that there exists $\psi' =< ctr'|subs' : pre' \overset{e(z_1,\ldots,z_n)}{\longrightarrow} post' >\in STR$ and $\iota' : X \to X$ such that for every $x' \neq y' \in ctr'$, $\iota'(x') \neq \iota'(y')$, $(s,\iota') \models \bigwedge_{\alpha \in subs' \cup pre'} \alpha$, $\neg p(w_1,\ldots,w_m) \in post'$ and $\iota(y_i) = \iota'(w_i)$ $(1 \leq i \leq m)$. As $\mathcal{F}$ is consistent, there exists a Σ-model $\mathcal{A}$ which satisfies it. Let $\iota'' : X \to U$ be an interpretation in $\mathcal{A}$ such that for every $x \neq y \in ctr$ and $x' \neq y' \in ctr'$, $\iota''(x) \neq \iota''(y)$ and $\iota''(x') \neq \iota''(y')$, and $\iota''(x_i) = \iota''(w_i)$ $1 \leq i \leq n$. By the property on the carrier cardinality of Σ-models, ι'' exists.
   By construction of $\mathcal{G}(X, I_\Sigma(X))$, there exists a state s'' in $\mathcal{A}$ such that $\iota''(s) \subseteq s''$. We then have for every $\alpha \in subs \cup subs' \cup pre \cup pre'$ that $(s'', \iota'') \models \alpha$. Therefore, there exists s^3 in $\mathcal{A}$ such that $s''\ e^{\mathcal{G}(X,I_\Sigma(X))}(\iota''(x_1),\ldots,\iota''(x_n))\ s^3$. But, we have both $p(\iota''(y_1),\ldots\iota''(y_m)) \in s^3$ and $p(\iota''(y_1),\ldots\iota''(y_m)) \notin s^3$ what is impossible.

[8] S' is the set of state of $\mathcal{G}(X, I_\Sigma(X))$

[9] Without any loss of generality, we only consider the case of a positive literal $p(y_1,\ldots,y_m)$ in $post$. The case of a negative literal $\neg p(y_1,\ldots,ym)$ can be handled in a similar way.

2. **ViolInv** there exists $s' \in S' \setminus I_\Sigma(X)$ [10], an invariant $< ctr|\varphi >$ and an interpretation $\iota : X \to X$ such that for every $x \neq y \in ctr$ $\iota(x) \neq \iota(y)$ but $(s', \iota) \not\models \varphi$.
By definition, this means that there exists $s \in I_\Sigma(X)$, n str-formulas $< ctr_i|subs_i : pre_i \overset{e_i(x^i_1,\ldots,x^i_{n_i})}{\longrightarrow} post_i >$ in STR, n interpretations $\iota_i : X \to X$ and $n+1$ states s_i with $s_1 = s$ and $s_{n+1} = s'$, such that for every $1 \leq i \leq n$ and every $x \neq y \in ctr_i$ $\iota_i(x) \neq \iota_i(y)$, $(s_i, \iota_i) \models \bigwedge_{\alpha \in subs_i \cup pre_i} \alpha$, $s_i\ e_i^{\mathcal{G}(X, I_\Sigma(X))}(\iota_i(x^i_1), \ldots, \iota_i(x^i_{n_i}))\ s_{i+1}$ and $(s_{i+1}, \iota_i) \models \bigwedge_{\alpha \in post_i} \alpha$. By construction of $\mathcal{G}(X, I_\Sigma(X))$, this then means there exists for every $1 \leq i \leq n$, $p_i(y^i_1, \ldots, y^i_{m_i}) \in At_\Sigma(X)$ such that $\iota(y^i_j) = \iota_i(y^i_j)$ for every $1 \leq j \leq m_i$, and $p_i(\iota(y^i_1), \ldots, \iota(y^i_{m_i})) \in s$ but $p_i(\iota(y^i_1), \ldots, \iota(y^i_{m_i})) \notin s'$. As $\mathcal{F}$ is consistent, there exists a Σ-model $\mathcal{A}$ which satisfies it. Let $\iota' : X \to U$ be an interpretation in $\mathcal{A}$ such that for every $x \neq y \in ctr \cup \bigcup_{1 \leq i \leq n} ctr_i$, $\iota'(x) \neq \iota'(y)$. By the property on the carrier cardinality of Σ-models, ι' exists. By construction of $\mathcal{G}(X, I_\Sigma(X))$, for every $1 \leq i \leq n+1$, there exists in $\mathcal{A}$ a state s'_i such that $\iota'(s_i) \subseteq s'_i$. Moreover, for every $1 \leq i \leq n$, $s'_i\ e_i^{\mathcal{A}}(\iota'(x^i_1), \ldots, \iota'(x^i_{n_i}))\ s'_{i+1}$. We the have for every $1 \leq i \leq n$ and every $\alpha \in subs_i \cup pre_i$ that $(s'_i, \iota') \models \alpha$, and then $(s'_{i+1}, \iota') \models \bigwedge_{\alpha \in post_i} \alpha$. Whence we deduce that for every $1 \leq i \leq n$, $p_i(\iota'(y^i_1), \ldots, \iota'(y^i_{m_i})) \in s'_{n+1}$ and $p_i(\iota'(y^i_1), \ldots, \iota'(y^i_{m_i})) \notin s'_{n+1}$ what is impossible.

Let us note that the proof of Theorem 1 highlights the 3 questions to solve in order to show specification consistency. They have been noted **InvCons**, **NonDet** and **ViolInv**. The two last ones will be solved by the two algorithms given in Section 3. The first question will be tackled in Section 4.2.
Let us remark that str-formula determinism is sufficient to ensure specification consistency but in no case it is necessary.

2.5 Service integration

The key question now is how to define service integration provided with an adequate semantic counterpart. A first answer might be to consider the union of axioms issued from different service specifications. However, this is not a good solution. Indeed, recall that a service is defined as possibly modifying the behaviour of the existing system on which it will be plugged on. Hence, any system obtained by the union of axioms of its different services would be lucky enough to be inconsistent. Therefore, in order to avoid to introduce inconsistencies during integration steps, choices are needed about which axioms are preserved, lost, modified and added. Hence, the integration of two services will be parameterised by choices. In this paper, we propose an interactive methodology based on algorithms introduced in Section 3 to determine these choices. These algorithms will automatically check consistency of service specifications. When inconsistencies (i.e. interactions) are detected, they are presented to an expert who makes

[10] S' is the set of state of $\mathcal{G}(X, I_\Sigma(X))$.

integration choices (see Section 4 for more explanations on how this methodology is worked up).

3 Interactions

We have seen in the proof of Theorem 1 that the inconsistency of a service specification may be the result of: the inconsistency of invariants **InvCons**, or the non-determinism of some events such as specified in the service specification **NonDet**, or because some str-formulas question some invariants **ViolInv**.

The first step, that is the question of invariant consistency **InvCons**, boils down to a classical boolean satisfiability problem[11]. The way we reduce **InvCons** to the boolean satisfiability problem will be handled in Section 4.2. Below, we detail the algorithms which solve the two last questions **NonDet** and **ViolInv**.

3.1 Non-determinism

Input A finite specification $\mathcal{F} = (\Sigma, I \coprod STR)$ such that I is consistent. A finite set of variables X. Two str-formulas in STR, $< ctr_1 | subs_1 : pre_1 \overset{e(x_1,\ldots,x_n)}{\longrightarrow} post_1 >$ and $< ctr_2 | subs_2 : pre_2 \overset{e(y_1,\ldots,y_n)}{\longrightarrow} post_2 >$

Initialisation Compute $I_\Sigma(X)$ and $\mathcal{G}(X, I_\Sigma(X))$. Note S' the set of states of $\mathcal{G}(X, I_\Sigma(X))$ and X^X the whole set of endofunctions from X to X. $Tmp := S'$ and $answer := true$.

Loop while $Tmp \neq \emptyset$ and $answer = false$ do:

1) choose s in Tmp and $Tmp := Tmp \setminus \{s\}$;

2) $Tmp' := X^X$;

3) **Loop** while $Tmp' \neq \emptyset$ and $answer = false$ do:

3.1) choose ι in Tmp' s.t. $\iota(x_i) = \iota(y_i)$ $(1 \leq i \leq n)$, and $Tmp' := Tmp' \setminus \{\iota\}$;

3.2) if $\forall x \neq y \in \bigcup_{j=1,2} ctr_j, \iota(x) = \iota(y)$ then if $\forall \alpha \in \bigcup_{j=1,2} subs_j \cup pre_j, (\iota, s) \models \alpha$ then $answer := true$;

end of loop

end of loop

Output $return(answer)$

Theorem 2. *Let $\mathcal{F} = (\Sigma, I \coprod STR)$ be a specification where Σ is a finite signature, and I and STR are finite sets. Let X be a finite set of variables which contains all variables occurring in $I \coprod STR$. Then, $\mathcal{G}(X, I_\Sigma(X))$ is deterministic if and only if for every pair of str-formulas in STR, $< ctr_1 | subs_1 : pre_1 \overset{e(x_1,\ldots,x_n)}{\longrightarrow} post_1 >$ and $< ctr_2 | subs_2 : pre_2 \overset{e(y_1,\ldots,y_n)}{\longrightarrow} post_2 >$, the above algorithm terminates and answers false.*

[11] The boolean satisfiability problem is solved by SAT solvers, e.g. GRASP [18] and Chaff [19].

3.2 Invariant preserving

Let $\mathcal{F} = (\Sigma, STR \coprod I)$ be a specification. Let X be a set of variables which contains all variables occurring in $\mathcal{F}$. By definition, we have:

$$\forall \Psi \in I, \forall s \in I_\Sigma(X), \forall \iota : X \to X, \mathcal{G}(X, I_\Sigma(X)) \models_{\iota,s} \Psi$$

The question **ViolInv** is equivalent to the following one: *are invariants preserved for states in* $S' \setminus I_\Sigma(X)$*?* where S' is the set of states of $\mathcal{G}(X, I_\Sigma(X))$.
When STR, I and X are finite sets, the above problem is computable as expressed by the following algorithm:

Input A finite specification $\mathcal{F} = (\Sigma, I \coprod STR)$. A finite set of variables X which contains all the variables that occur in axioms of $\mathcal{F}$. An invariant $< ctr|\varphi >$ in I.
Initialisation Compute $I_\Sigma(X)$ and $\mathcal{G}(X, I_\Sigma(X))$. Note S' the set of states of $\mathcal{G}(X, I_\Sigma(X))$ and X^X the whole set of endofunction from X to X. $Tmp := S' \setminus I_\Sigma(X)$ and $answer := false$.
Loop while $Tmp \neq \emptyset$ and $answer = false$ do:
1) choose s in Tmp and $Tmp := Tmp \setminus \{s\}$;
2) $Tmp' := X^X$;
3) Loop while $Tmp' \neq \emptyset$ and $answer = false$ do:
3.1) choose ι in Tmp' and $Tmp' := Tmp' \setminus \{\iota\}$;
3.2) if $\forall x \neq y \in ctr$, $\iota(x) \neq \iota(y)$
then $answer := not((\iota, s) \models \varphi)$
end of loop
end of loop
Output $return(answer)$

Theorem 3. *$\mathcal{G}(X, I_\Sigma(X))$ satisfies I if and only if for every invariant in I, the above algorithm terminates and answers false.*

4 Methodology and experiments

When integrating a new feature, we enter upon the problem of how to apply the algorithms and in which order, to ensure the consistency of the resulting specification.

4.1 The Design Phase Process

We have seen in Section 2.5 that to avoid introducing inconsistency during integration, choices are needed about which formulas are preserved, lost, modified or added. We propose an interactive approach based on the algorithms introduced before. Interactions are detected and presented to an expert who makes integration choices.

A service specification $\mathcal{F}$ provides modifications with respect to an implicit system. From a formal point of view, the integration of $\mathcal{F}$ on a system $\mathcal{S}ys$ is a composition which is parameterised by the required choices, i.e., it is abstractly denoted by $\mathcal{S}ys +_{choices} \mathcal{F}$. It generally leads to some modifications of $\mathcal{S}ys$; thus, we do not easily

get the addition of two services together to a system ($\mathcal{S}ys +_c \{\mathcal{F}_1, \mathcal{F}_2\}$) from the addition of each of them, $SYS +_{c1} \mathcal{F}_1$ and $SYS +_{c2} \mathcal{F}_2$. Indeed, it would suppose not only to confront the specifications $\mathcal{F}_1$ and $\mathcal{F}_2$, but also to re-examine c_1 and c_2 because c_1 was thought on $\mathcal{S}ys$ and not on $\mathcal{S}ys$ modified by c_2, and conversely. Thus our approach is to integrate services one by one. Therefore, given *POTS* and services $\mathcal{F}_1, \ldots, \mathcal{F}_n$, we build an integration $(\ldots(POTS +_{c_1} \mathcal{F}_{i_1}) +_{c_2} \cdots +_{c_n} \mathcal{F}_{i_n})$, where the order $i_1, \cdots, i_n$ is significant with respect to the choices $c_1, \cdots, c_n$.

Note $Sys_{j-1} = (\Sigma_{j-1}, STR_{j-1} \coprod I_{j-1})$ the system specification resulting from $(\ldots(POTS +_{c_1} \mathcal{F}_{i_1}) +_{c_2} \ldots +_{c_{j_1}} \mathcal{F}_{i_{j-1}})$ and $\mathcal{F}_{i_j} = (\Sigma_{i_j}, STR_{i_j} \coprod I_{i_j})$. In order to determine the next choice $+_{c_{i_j}}$ for integrating $\mathcal{F}_{i_j}$, the following process is applied:

1. (a) Checking the invariant consistency of $I_{j-1} \cup I_{i_j}$ using the algorithm **ConsInv** by considering one by one the invariants of I_{i_j}
   (b) Solving inconsistency as soon as it occurs by modifying one of the involved invariants and starting again Point 1.a after each encountered inconsistency.

   This first step generates a consistent set $I_{\Sigma_{j-1} \cup \Sigma_{i_j}}$, or more simply I, of invariants which will be used in the reference specification for the next two following points.
2. (a) Performing the algorithm **NonDet** on every pair (ψ_1, ψ_2) where $\psi_1 \in STR_{j-1}$ and $\psi_2 \in STR_{i_j}$ such that ψ_1 and ψ_2 satisfy the condition of the input part of the algorithm (i.e. the event which occurs in ψ_1 and ψ_2 is the same).
   (b) Solving non-determinism conflicts as soon as they occur, and starting again Point $2.a$ after each one of them. This gives rise to a new set of str-formulas STR.
3. (a) Performing the algorithm **ViolInv** on every invariant in I with respect to the $\Sigma_{j-1} \cup \Sigma_{i_j}$-model $\mathcal{G}(X, I_{\Sigma_{j-1} \cup \Sigma_{i_j}}(X))$ computed from I and the set STR resulting from Point 2. above.
   (b) Solving inconsistency conflicts as soon as they occur.
4. Point 3. possibly modifies both sets STR and I, and then gives rise to two new sets STR' and I'. If this is the case, then starting again the above process for I' and STR'. Otherwise, the process is terminating.

To ensure termination of the above process, a strategy is to impose that:

1. for every pair (ψ_1, ψ_2) where ψ_1 and ψ_2 are two str-formulas satisfying conditions of Point $2.a$, if all non-determinism conflicts for ψ_1 with all str-formulas of STR_{j-1} have been already handled (i.e. ψ_2 is a str-formula which has been added or modified during some previous steps of the first algorithm) then the choice of the expert to solve the non-determinism conflict between ψ_1 and ψ_2 (when it exists) necessarily rests on ψ_2.
2. when a consistency conflict occurs on invariants by the algorithm **ViolInv**, the choice of the expert necessarily rests on invariants (i.e. str-formulas of STR are preserved).

4.2 Implementation

In the Valiserv project framework, the process presented in the previous section has been implemented. We have then defined a prototype to help the expert for specifying

and validating service-oriented telecommunication systems. To produce more efficient implementations of algorithms, a first step of the above process has been to restrict the cardinality of the set of variables X occurring in axioms of the specification under consideration. This has allowed to reduce the invariant consistency **InvCons** to a propositional satisfiability problem of reasonable size and to decrease the complexity of the step **3)** in both algorithms **NonDet** and **ViolInv** in Section 3. The point is to translate a set of invariants into an equivalent single invariant. To achieve this purpose we first transform any axiom into its Skolem form. To simplify, let us consider an invariant of the form $< ctr|\phi >$ where X is its vector of variables occurring in ctr and ϕ. Obviously, such a formula can be written under its equivalent Skolem form: $\forall X, \bigwedge_{x_i \neq y_i \in ctr} x_i \neq y_i \Rightarrow \phi$. If we consider two such formulas ψ_i of the form $< ctr_i|\phi_i >$ for $i = 1, 2$ with X_i their respective variable set and provided that $X_1 \cap X_2 = \emptyset$, a naive approach consists on putting $\forall X_1 \cup X_2$ as a global universal variable vector quantifier. But such a solution has the main drawback of building formulas with too many variables. Under the hypothesis that the size of X_1 is less or equal to the one of X_2, in order to minimise the number of variables, we search for substitutions $\iota : X_1 \rightarrow X_1 \cup X_2$ such that every inequality on two variables of X_1 is preserved by ι in $X_1 \cup X_2$. There necessarily exist such substitutions (e.g. the identity). In fact, we are looking for such substitutions which minimise for the size of the set $\iota(X_1) \cup X_2$. When such a substitution is found, then $\iota(X_1) \cup X_2$ will become the variable vector used to universally quantify the resulting Skolem formula $\forall \iota(X_1) \cup X_2, \bigwedge_{x_i \neq y_i \in \iota(ctr_1) \cup ctr_2} x_i \neq y_i \Rightarrow \phi$. The computation of an optimal substitution is done by means of systematic study of all substitutions compatible with the inequality constraints. By iterating such a variable factorisation between all invariants, we can control the whole number of variables to be considered. The boolean satisfiability problem corresponding to a formula $\forall X, \bigwedge_{x_i \neq y_i \in ctr} x_i \neq y_i \Rightarrow \phi$ is then simply given by the propositional formula $\bigvee_{\sigma : X \rightarrow X, \forall x_i \neq y_i \in ctr, \sigma(x_i) \neq \sigma(y_i)} \sigma(\phi)$ where the atoms $p(x_1, \ldots, x_n)$ occurring in $\sigma(\phi)$ are viewed as simple propositional variables.

4.3 Case study

The above methodology has been applied on many telecommunication examples. Among other, it has been applied on the example presented in Section 2. Here, we give the report of this case study. Its interest is it is significant enough but short enough to be presented in this paper. We incrementally integrate several services yielding the system $(((POTS +_{c_1} TCS) +_{c_2} CFB) +_{c_3} INTL)$. The main steps have been the following:

- $POTS +_{c_1} TCS$: a non-determinism has been detected between ϕ_2 and ψ_1. We have modified ϕ_2, intuitively giving the priority to *TCS* on *POTS*.
- $((POTS +_{c_1} TCS) +_{c_2} CFB)$: a non-determinism has been detected between ϕ_3 and χ_3. We have modified ϕ_3. We have then detected that χ_3 violates the *TCS* invariant ψ_1. We have corrected it by adding $\neg Tcs(C, A)$ to the subscription set of χ_3. Then, we add the following str-formula for the case we have $Tcs(C, A)$:

$\chi'_3 : \ < B \neq C \mid Cfb(B,C), Tcs(C,A), dialwait(A), \overline{idle(B)}, idle(C) \xrightarrow{dial(A,B)}$
$busytone(C), idle(C) >$

Thus, *TCS* has the priority on *CFB* and *CFB* has the priority on *POTS*.

- (((*POTS*$+_{c_1}$*TCS*)$+_{c_2}$*CFB*)$+_{c_3}$*INTL*): a non-determinism has been detected between ϕ_1 and κ_1. We have modified ϕ_1, intuitively giving the priority to *INTL* on *POTS*, *TCS* and *CFB*.

The specification of *POTS*, *TCS*, *CFB* and *INTL* together contains twenty formulas. During the integration process, we have modified four of them and introduced a new one. The ValiServ tool automatically detects current interactions, presents the detected interactions to the expert under a detailed form and allows the expert to modify the related specification part so that the considered interaction is suppressed according to its judgment. Such an approach allows to manage the intrinsic complexity of service-oriented systems since the expert only intervenes to solve interactions according to their subjective status. Thus, our service integration method may be viewed as a sequence of expert choices in a set of resolution options, each of these expert choices coming from an automatic feature interaction detection.

5 Related Work

Several previous works have been interested by feature integration and interaction detection issues from a high level of abstraction. In particular, new architectures have been designed for telecommunications systems in order to facilitate the addition of a new service. [5] or [13] present such approaches, useful for designing and implementing new services but not to found rigorous interaction detection methods. [23] gives a general framework to systematically combine services together. Only consistent combinations of services are considered. When an inconsistency is detected for a given combination of services, this means that there exists an interaction between combined features. However, the paper is not concerned by the need of providing theoretical and methodological help in order to combine service in presence of interactions. Some other works, like [9], are also based on the use of model-checking tools in order to detect interactions. This allows to consider general temporal properties. The main drawback of all these approaches is that they require to instantiate *a priori* different configurations to build all the interesting subscription patterns among a small number of fixed phones.

We claim that the use of symbolic techniques for dealing with phone variables is the key to deduce interactions built over an appropriate number of phones equipped with their subscriptions. Some other works manipulate generic variables to represent phones, without restricting the number of phones to be considered. In particular, several approaches rely on STR-like specifications. [10] precisely explains the interest of using STR formulas, invariants and inequality preconditions. The authors were already concerned with providing guidelines to integrate a service on the basic call system and hints on how to perform non-determinism checks. Unfortunately, the described detections are mainly guided by hand-waving and thus, there was no study of how to systematically support this process. Our framework which is largely inspired by their process, addresses this weakness. [20, 24] have proposed specialised techniques for interaction

detection based on STR-like specifications. From a given initial state, they analyse properties of reachability graphs in terms of non-determinism or deadlock or contradictions raised by the simultaneous application of two STR sentences ... Works introduced in [26, 25] discuss the advantage of dealing with static methods, without building any intermediate graph. They introduce techniques for finding interactions from non-determinism criteria or from elicitation knowledge between two services. They compute a lot of interactions, but as they do not look for service integration, they do not exploit their presence to compose services in an adequate way. Moreover, as they do not use invariants, they cannot help the specifier in designing STR specifications. Let us remark that we handle the preservation of invariants as in [6]. However, the underlying proof-based techniques require too much expertise to our point of view. [22, 7, 11] introduce systematic mechanisms of service composition avoiding a lot of interactions. Roughly speaking, all of these works are based on some precedence relations between services: the last integrated feature seems to have the highest priority level. However, if undesirable interactions subsist, then it is not possible to review the integrated system, except if a new design process is managed from the beginning.

6 Conclusion and perspectives

We presented a methodology for service-oriented development that takes interaction and integration issues into account. We introduced a dedicated formalism taking into account subscriptions. and manipulating two kinds of formulas, state invariants and state transition rules. We gave algorithms allowing the specifier to check the consistency of the specification under consideration. The service integration results from the incremental insertion of formulas preserving at each step the consistency of the target specification. Each detected consistency problem represents an interaction and requires an expert decision to modify, and to replace the formula(s) causing inconsistency. The whole methodology has been validating by the industrial partner France Telecom of the project ValiServ.

This work can be pursued in several ways. We want to study state reachability issues to ensure that each detected non-determinism case corresponds to a real interaction case. We also want to study how it is possible to introduce different types on variables to capture different rôles (users, phone numbers or IP addresses) in order to apply our algorithms and methodology to application domains such as voice over IP [15]. From a methodological point of view, we aim to strengthen expert assistance by minimising choices and backtrack at design step. Such improvement should rely not only on theoretical consideration but also on expertise about the telecommunication domain.

References

1. M. Aiguier, K. Berkani, and P. Le Gall. Feature specification and static analysis for interaction resolution. Technical report, University of Evry, 2005. available at www.lami.univ-evry.fr/∼aiguier.
2. K. Berkani. *Un cadre méthodologique pour l'intégration de services par évitement des interactions*. Phd thesis, France, October 2003.

3. K. Berkani, R. Cave, S. Coudert, F. Klay, P. Le Gall, F. Ouabdesselam, and J.-L. Richier. An Environment for Interaction Service Specification. [17], 2003.
4. K. Berkani, P. Le Gall, and F. Klay. An Incremental Method for the Design of Feature-oriented Systems. In *[12]*, pages 45–64. Springer, 2000.
5. J. Bredereke. Maintening telephone switching software requirements. *IEEE Communications Magazine*, Nov, 2002.
6. D. Cansell and D. Méry. Abstraction and Refinement of Features. In *[12]*, pages 65–84. Springer, 2000.
7. D.P. Guelev, M.D. Ryan, and P.Y. Schobbens. Feature Integration as Substitution. [17], 2003.
8. L. du Bousquet, F. Ouabdesselam, J.-L. Richier, and N. Zuanon. Feature Interaction Detection using Synchronous Approach and Testing. *Computer Networks and ISDN Systems*, 11(4):419–446, 2000.
9. A. Felty and K. Namjoshi. Feature specification and automated conflict detection. *ACM Transactions on Software Engineering and Methodology*, 12(1):3–27, 2003.
10. A. Gammelgaard and J.E. Kristensen. Interaction Detection, a Logical Approach. In *Feature Interactions in Telecommunications Systems II*, pages 178–196. IOS Press, 1994.
11. C. Gaston, M. Aiguier, and P. Le Gall. Algebraic Treatment of Feature-oriented Systems. In *[12]*, pages 105–124. Springer, 2000.
12. S. Gilmore and M. Ryan, editors. *Langage Constructs for Describing Features*. Springer-Verlag, 2000.
13. M. Jackson and P. Zave. Distributed feature composition: A virtual architecture for telecommunications services. *IEEE Transactions on Software Engineering*, 24(10):831–847, 1998.
14. H. Jouve. *Caractérisation et détection automatique d'interactions de services à partir de spécifications graphiques*. Phd thesis, France, October 2003.
15. H. Jouve, P. Le Gall, and S. Coudert. An Automatic Off-Line Feature Interaction Detection Method by Static Analysis of Specifications. [21], 2005.
16. K. Kimbler and L.G. Bouma, editors. *Feature Interactions in Telecommunications and Software Systems (FIW'98)*. IOS Press, 1998.
17. L. Logrippo and D. Amyot, editors. *Feature Interactions in Telecommunications and Software Systems (FIW'03)*. IOS Press, 2003.
18. J. P. Marques-Silva and K. A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 1999.
19. M. W. Moskewicz, C. F. Madigan, Y. Zhao, and S. Malik. Chaff : Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC'01)*, 2001.
20. M. Nakamura. *Design and Evaluation of Efficient Algorithms for Feature Interaction Detection in Telecommunication Services*. Phd thesis, Japan, June 1999.
21. S. Reiff-Marganiec and M.D. Ryan, editors. *Feature Interactions in Telecommunications and Software Systems (FIW'05)*. IOS Press, 2005.
22. D. Samborski. Stack Service Model. In *[12]*, pages 177–196. Springer, 2000.
23. B. Schätz and C. Salzmamm. Service-based systems engineering : Consistent combination of services. In LNCS Springer, editor, *ICFEM 2003,Fifth International Conference on Formal Engineering Methods*, volume 2885, 2003.
24. K. Tatekuwa and T. Ohta. Automatic deleting specification errors which cause miss-detection of feature interactions. In *[21]*, pages 320–326, 2005.
25. T. Yoneda, S. Kawauchi, J. Yoshida, and T. Ohta. Formal approaches for detecting feature interactions, their experimental results, and application to voip. [17], 2003.
26. T. Yoneda and T. Ohta. A Formal Approach for Definition and Detection of Feature Interaction. In *[16]*, pages 202–216. IOS Press, 1998.