Notes on Call Configurations with Features

Stephan Reiff*

May 8, 2000

Abstract

In this positional paper, we define and analyse **call configurations**. The main result is a theorem for calculating the number of users in an n-feature configuration.

1 Introduction

The evaluation of feature interaction detection and resolution methods requires scenarios. Scenarios are usually created in a rather pragmatic way, whereby the experience and intuition of the scenario designer has a major impact on the quality of those scenarios. For example a typical scenario will contain information about the state of two users (say A and B are offhook, A and B are connected, etc). Motivated by this, we develop a more general approach which in particular depends less on predicted behaviour.

We propose a more abstract and structured approach, resulting in a notion of call configurations. A call configuration is a class of abstract scenarios; a scenario is then an instance of a call configuration. The approach guarantees scenarios that are both minimal and sufficient. By minimal we refer to the fact that adding further users will not result in new behaviour – in contrast it might allow situations where no interactions can occur. Sufficient expresses the fact that the number of users is large enough for all features to be deployed.

During the remainder of this document we will use the terms *call configuration* and *configuration* to denote the same concept.

*Stephan Reiff, Department of Computing Science, University of Glasgow, 8-17 Lilybank Gdns.,Glasgow G12 8QQ, `sreiff@dcs.gla.ac.uk`

Section 2 introduces an abstract, structural view of features, which is used in the precise definition of call configurations in section 3. The method to compute the number of users required in a configuration for possible interactions to occur is developed in section 4. The final section (6) concludes on the achievements and shows how the result can be used to aid testing of features and detection/resolution methods.

2 Arity of Features

The definition of a feature is made explicit in [Rei00]. Informally, a feature is a set of traces where each trace consists of a sequence of messages. For our purpose a message consists of at least an event, a source and a destination.

An abstract view of a feature, suitable for calculating numbers of users in configurations, requires the concept number of users involved in a feature. The number of users involved in a feature is a property of this feature which we call its *arity*.

Example 1 *Terminating Call Screening (TCS) is represented by the following (tau is a null message):*
TCS = {(i_alert, A, B, -)(o_inform, B, A, "screened"),
(i_alert, A, B, -)(tau)}
The second element in each message is the source, the third the destination. There are 2 users involved in TCS, namely A and B.

Definition 1 (Arity of a Feature)
Let f be a feature. Let Sources and Destinations be the sets containing values occurring in the source and destination fields (respectively) of the messages that form the traces of f. Then, $arity(f)$ is the arity of f:

$$arity(f) = |Sources \cup Destinations|$$

From this we can derive the arity for some example features (taken from [KMMR99]):

Feature	Src/Dest	Arity
Teenline	A	$arity(TL) = 1$
Terminating Call Screening	A, B	$arity(TCS) = 2$
Call Waiting	X, Y, Z	$arity(CW) = 3$
Call Forwarding on Busy	id, C, D	$arity(CFB) = 3$
Group Ringing	X, Y, N1, N2	$arity(GR) = 4$

Note that the arity of a feature might change depending on the initialisation of its data, e.g. a consequence could be that 2 or more users are identified. For example CFB might forward to the caller rather than a third party. Here we consider only the case where such exceptional initialisations do not occur.

3 Call Configurations

We have introduced an informal idea of configurations earlier, and in this section we will formalise the notion of configurations.

Definition 2 (Call Configuration)
1. *Let f be a feature of arity n. Then,*
   feature f with users $u_1, ..., u_n$ is a configuration.
2. *Let C be a configuration with features $f_1, ..., f_m$, with users $u_1, ..., u_p$ and f a feature with arity n. Then,*
   $f_1, ..., f_m, f$ with users $u_1, ..., u_k$ where $p \leq k \leq p + n$ is a configuration and in this configuration $n - (k - p)$ is the number of common users of C and f.

In the following examples, we use a graphical notation for configurations. Users are represented by nodes. All users for a given feature are connected. In general the edges denote the potential of message exchange between the users, but what is important is connectivity rather than the number of edges. Common users between features are enclosed by a round cornered rectangle.

First, consider the stepwise construction of an example (figure 1). The final configuration consists of three features. The three figures show the sequence of configurations that evolve as each feature is added to the previous configuration. The number of common users between the existing configuration and the added feature in the second and third graph is 1, for the first graph the number of common users is not defined. $arity(f_1) = 2$, the arities of f_2 and f_3 are equal to 3.

Second, we show a further example (figure 2), where each part contains a configuration. Parts a, b and e are configurations involving three features, c, d and f both involve two features. We would like to draw special attention to part b and e: in part b all three features are between three users, in part e the third feature added has two common user with the configuration formed by the previous two. Finally part f represents a configuration where the number of common users is 0.

We are interested in feature interaction detection and it should be obvious that figure 2.f does not represent a configuration where interactions can occur.

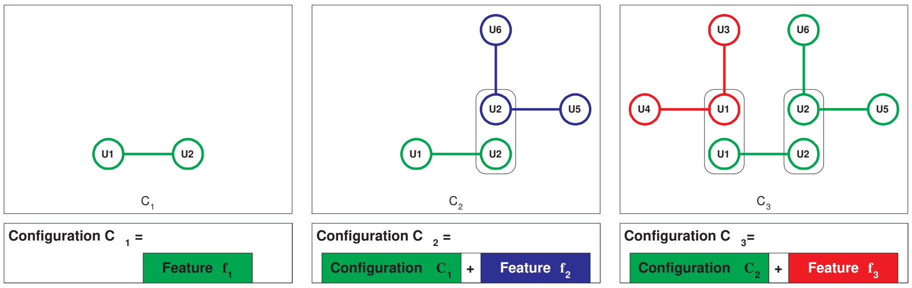

Figure 1: Stepwise Construction of a Configuration

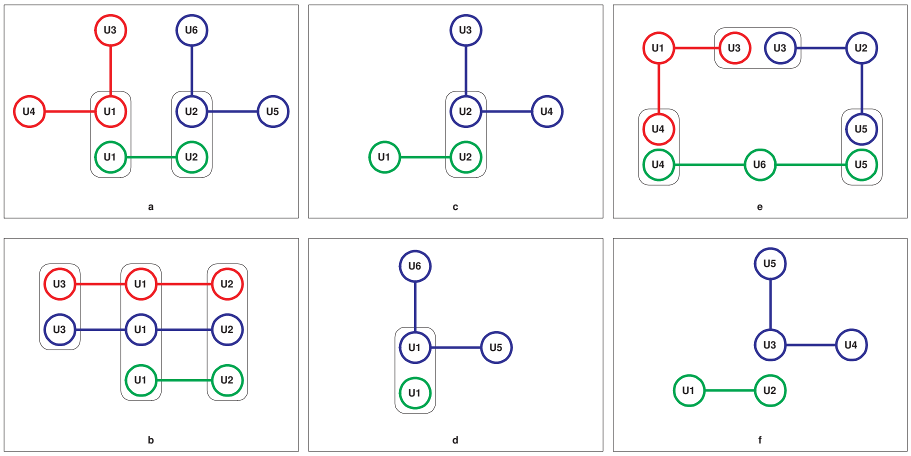

Figure 2: Example Call Configurations

Since our primary interest is feature interactions, configurations of interest to us exclude cases such as 2.f. To exclude them, we define possibly interacting configurations.

Definition 3 (Possibly Interacting Configuration (PIC))
Let $n \geq 2$. Let $C_1, ..., C_n$ be a sequence of configurations built up as follows: $C_i = f_1, ..., f_i$ with users $u_1, ..., u_u(i)$ and C_{i+1} is build from C_i and f_{i+1}, with the requirement that

$$\forall i : 1 \leq i \leq n \rightarrow (\textit{the number of common users of } C_{i-1} \textit{ and } f_i > 0)$$

In the graphical representation, this can be interpreted as: a possibly interacting configuration is a configuration in which all nodes are reachable from any other node.

4 Number of Users in a Configuration

As we have seen, it is possible to choose any number of users for a configuration. However, we are interested in configurations that lead to interactions (if those exist between the features), hence we will consider configurations with 2 or more features only, i.e. possibly interacting configurations. The following result allows us to calculate the optimal number of users for possibly interacting configurations:

Theorem 1 (Number of Users in a PIC)
Let $C = C_1, ..., C_n$ be a possibly interacting configuration.
The number of users u in C_n is given by

$$u = \sum_{i=1}^{n} arity(F_i) - \sum_{i=1}^{n-1} common\ users(C_i, f_{i+1})$$

This result is interesting, as it allows to define sufficient and minimal configurations, hence we claim that the number of users is optimal. To visualise this, consider the following two cases:

1. attempt to construct a configuration with less than u users: In this case we are unable to deploy all features, as at least one feature will involve a user that is not part of the configuration.

2. attempt to construct a configuration with more than u users: Here we can deploy all features and will achieve a configuration, however there is no guarantee that this configuration is interacting (or if we ensured that we have constructed a possibly interacting configuration, there will be exist one user that is not involved in any feature and hence can be removed).

A justification for sufficient configurations is not required, however the question *Why are we interested in minimal configurations?* might arise. The answer to this question can be given in several forms: Firstly, it is more aesthetic to obtain configurations that do not contain any unnecessary elements. Secondly, for automated testing and verification techniques it is best to work with as small as possible configurations due to well-known problems such as state space explosion.

Application of this result yields the following:

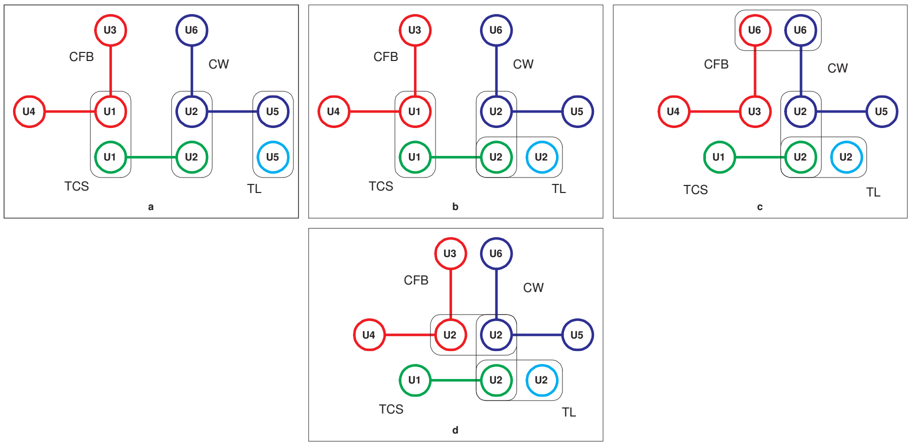

Figure 3: Possible Call Configurations

Exs	No of U Expctd.	Sum of Arities	Sum of Common Users	No of U Calculated
2.a	6	$2+3+3=8$	$1+1=2$	6
2.b	3	$2+3+3=8$	$2+3=5$	3
2.c	4	$2+3=5$	1	4
2.d	3	$1+3=4$	1	3
2.e	6	$3+3+3=9$	$1+1+1=3$	6

5 Discussion

Using 4 features, Teenline (TL), Terminating Call Screening (TCS), Call Waiting(CW) and Call Forwarding on Busy(CFB), we attempt to construct a concrete possibly interacting configuration. From section 2 we know the arities of these 4 features are 1, 2, 3 and 3 respectively. We decide that each feature has one common user with at least one other feature. We can quickly determine that the number of distinct users required is 6 by inserting the given data in the formula to calculate the number of users. Figure 3 visualises 4 examples of the possible configurations.

Once we have established the number of users and the number of common users between the features, we can now decide rôles of the users (subscriber, terminating party, originating party) with respect to each feature. Further we can decide which users are common between two features. These efforts define a minimal and sufficient scenario.

An interesting question arising is whether a configuration can be used to identify that no interaction occurs between two features. Consider for example 3.a where one could assume that CFB cannot interact with TL. However, this question remains open at the current time. An answer would provide a way to reduce the number of possible interactions that have to be searched in order to determine actual interactions.

6 Conclusions

We have shown an abstract structural view of features which helps us to build configurations for feature interactions. The result, posted as Theorem 1, allows for these configurations to be sufficient and minimal.

The practical use is that the feature designer or the developer of a tool for interaction detection and resolution can chose a number of features, consider their arity and decide how many common users each pair of the features has. From this it is very simple to calculate the number of users needed.

Acknowledgements

I thank Muffy Calder for stimulating discussions and helping me spot errors in the detail by making me explain this work to her.

References

[KMMR99] M. Kolberg, E. H. Magill, D. Marples, and S. Reiff. Second feature interaction contest. 1999.

[Rei00] S. Reiff. Identifying resolution choices for an online feature manager. In M. Calder and E. Magill, editors, *Proceedings of the* 6^{th} *International Workshop on Feature Interactions in Telecommunications and Software Systems*, pages 00–00, Glasgow, Scotland, May 2000.