Specification, Detection and Resolution of IN Feature Interactions with Estelle

J. Bredereke and R. Gotzhein

University of Kaiserslautern, Postfach 3049, D-67653 Kaiserslautern, Germany
{bredereke,gotzhein}@informatik.uni-kl.de

Keyword Codes: C.2.2; D.2.1
Keywords: Network Protocols; Requirements/Specifications

1. Introduction

We present an approach for the treatment of Feature Interactions in Intelligent Networks (IN). It is based on the formal description technique (FDT) Estelle and consists of three steps. 1) A *specification style* supporting the integration of additional features into a basic service is introduced. 2) Feature interactions are *detected* which result from the integration of additional features (before their deployment). 3) Previously detected feature interactions are *resolved*. We emphasize that these three aspects interfere and therefore have to be treated together. In particular, the FDT and the specification style have a strong influence on the detection criteria and on the resolution of feature interactions.

Several authors have proposed classifications of feature interactions in telecommunication systems. In [DaNa93], a distinction is made between *technical interference* and *policy interference*. Technical interference leads to phenomena such as deadlocks, livelocks and congestion. The detection of these phenomena has been studied extensively. Policy interference occurs if features interact such that the policy of one or more of the involved features is violated. Our approach is designed to be general, but we will especially concentrate on policy interference.

The approach sketched in this extended abstract is presented in more detail in [BrGo94], and it is applied to the Basic Call Service and several supplementary interacting features.

2. Specification of IN Features with Estelle

Adding (further) features to a system means extending its specification. The number of spots in the syntax and the semantics which are touched should be as small as possible. The following stylistic rules will allow the use of our detection and resolution procedure below: 1. Extend on a *coarse-grained* level. 2. Modify only by *adding* text.

We will look at a single module instance. For Estelle modules, the coarsest grain of behavioural description is the transition. So if we modify the behaviour, we work on the level of entire transitions only. Furthermore, we only add transitions. Because of this, every possible execution sequence in the hitherto existing module instance remains possible in the extended module instance[1]. The existing behaviour is not touched. (The justification for this proposition will be sketched in Section 3.) Of course there also may

[1] Under some premises which mainly concern the notational abbreviations that are possible in Estelle, and under the premise that there are no priorities defined among the transitions. The latter can be overcome easily by a straightforward extension of our resolution method below.

be cases where the existing behaviour needs to be changed. Then, we don't change the definition of the corresponding transition, but we disable[2] the old transition and add a new one. It is easier to detect problems resulting from a few such transition replacements than those from many small changes on the level of single Pascal-like statements inside the transition bodies. How the disabling of transitions is done will be described in Section 4.

3. Detection of Feature Interactions

We designed our stylistic rules in such a way that the only modification to the automaton which is the semantics of an Estelle module instance is the addition of further transitions (and states). Therefore, every existing execution sequence will remain possible for the extended automaton. It is safe to say that adding to the specification of a system will not lead to any interactions with existing features if the visible behaviour of the system is not changed in any way. Therefore, a change in the visible behaviour of the system is a necessary condition for feature interactions to occur. This leads us directly to the following necessary condition for feature interactions to occur:

Rule: A feature interaction may occur only if there is a (reachable) state in which at least two transitions belonging to different features can be fireable simultaneously, i.e. if there is non-determinism among different features.

For orthogonality, we also denote the basic service as a feature. Note that this formal criterion is a necessary condition for feature interactions to occur, but no sufficient condition. It may be possible that our detection rule points out a spot which proves to be harmless by manual inspection. The advantage of our approach is that it provides us with a relatively short list of critical spots. If we investigate each of them and resolve the non-determinism (as described below), then no unwanted feature interaction will be left.

It has to be stressed that only non-determinism between the transitions of *different* features is an indication of possible feature interactions. Non-determinism among the transitions of a single feature is a common means to express abstraction and has nothing to do with our topic. Therefore, we have to differentiate between these two kinds of non-determinism. This is done by an unambiguous association of each transition to one feature.

4. Resolution of Feature Interactions

One technique to resolve feature interactions is to define, for each pair of features, which one will take precedence over the other. This step can be seen as a high-level design decision that requires background knowledge about policies of the telephone administration, and therefore cannot be automated. The results of this step can be represented in the form of a precedence matrix, with an entry for each pair of features.

The resolution of feature interactions as defined by a precedence matrix can be incorporated into the Estelle specification in three steps:

Step 1. From the precedence matrix, derive a precedence function prec associating a precedence value with each feature such that f takes precedence over f' implies $\mathrm{prec}(f) < \mathrm{prec}(f')$[3].

Step 2. Associate each transition of the Estelle specification with one feature.

[2] To be exact, our method will leave the transition *enabled*, but not *fireable*.

[3] In Estelle, smaller values denote higher priorities. We will use the value of the precedence function as the Estelle priority below.

Step 3. For all transitions t: if t is associated with feature f, then add a priority clause "`priority` p_f", where $p_f = \mathrm{prec}(f)$.

The association between transitions and features from Step 2 should also be done syntactically, according to the following provision: for each feature, one constant identifier is defined. In Step 3 we will need a priority constant for each feature anyway, so we will define one priority constant per feature. This way, the priority clause from Step 3 will provide us with the syntactical association between transitions and features.

A simple algorithm for the computation of the precedence function prec from a precedence matrix is presented in [BrGo94].

Sometimes, it is not sufficient to define a precedence between two conflicting features. It may be necessary to introduce completely new behaviour for this special case to handle the complex situation. The addition of this new behaviour is done in our approach formally like the addition of any other feature. This "feature" must be selected exactly if both of the conflicting features are selected, and it must have a (slightly) higher precedence than both of them.

We point out that our formal detection criterion is simple enough to be calculated by tools quickly even for large specifications; the same applies to the algorithm we use in the resolution step which computes priority clause constants automatically from the precedence matrix. Therefore, our approach scales well to the complexity of practical systems.

5. Future Work

The incorporation of further features into the Basic Call Service is currently under investigation. The refinement from the service to a protocol is an obviously promising step to go, introducing more structure into the system architecture. The capacity of our approach should be investigated with respect to the classes of further feature interaction classifications. Furthermore, we will continue the development of supporting tools.

Addition of features and thus extension of the Basic Call Service (BCS) has been achieved by adding specification text, i.e. on a purely syntactical level. We expect that it will be possible to develop a theoretical foundation together with suitable (syntactical) rules for incremental specialization such that the following holds: if the specification of the BCS augmented by additional features is an incremental specialization of the BCS, then it also extends the BCS semantically. Some work in this direction has been reported in [GoBo94], where semantical relations for specializing Estelle module definitions and a notion of incremental specialization based on these relations have been introduced.

References

[BrGo94] Bredereke, J. and Gotzhein, R. *A case study on specification, detection and resolution of IN feature interactions with Estelle*. Tech. Rep. 245/94, Univ. of Kaiserslautern, Dept. of Comp. Sce. (May 1994).

[DaNa93] Dahl, O. C. and Najm, E. *Specification & detection of IN service interference using LOTOS*. In Tenney, R. L. et al., editors, "Formal Description Techniques, VI", Boston, Mass. (26–29 Oct. 1993). North-Holland.

[GoBo94] Gotzhein, R. and von Bochmann, G. *Specialization in Estelle*. In Vuong, S. T. and Chanson, S. T., editors, "Proc. of PSTV, XIV", Vancouver, Canada (7–10 June 1994).