Characterizing Implicit Communal Components as Technical Debt in Automotive Software Systems

Andreas Vogelsang[1], Henning Femmer[2], Maximilian Junker[2]

1 Introduction

Automotive software systems are often characterized by a set of features that are implemented through a network of communicating components. It is common practice to implement or adapt features by an ad hoc (re)use of signals that originate from components of another feature. Thereby, over time some components become so-called *implicit communal components*. These components increase the necessary efforts for several development activities because they introduce feature dependencies. Refactoring implicit communal components reduces these efforts but also costs refactoring effort. In this paper, we provide empirical evidence that implicit communal components exist in industrial automotive systems. For two cases, we show that less than 10% of the components are responsible for more than 90% of the feature dependencies. Secondly, we propose a refactoring approach for implicit communal components, which makes them explicit by moving them to a dedicated *platform component layer* (PCL). Finally, we characterize implicit communal components as technical debt, which is a metaphor for suboptimal solutions having short-term benefits but causing a long-term negative impact. With this metaphor, we describe the trade-off between accepting the negative effects of implicit communal components and spending the necessary refactoring costs. The full paper can be found in [VFJ16].

Terms and Definitions: We use this definition of *feature dependency* [VF13]: A feature *depends* on another feature if at least one component associated with the feature reads a signal that originates from a component associated with the other feature. In recent studies, we found that not only almost every feature depended on another feature, we have also seen that there is a 50% chance that a developer is not aware of a specific feature dependency [VF13, VTG12]. A *communal component* is a component that exchanges signals (sending or receiving) with components of features different from the feature of the communal component. We distinguish between *implicit* and *explicit* communal components depending on whether a component is associated with a feature or with a dedicated *platform component layer* (PCL). The purpose of this dedicated platform layer is to bundle components that implement functionality important for a number of features. This may include components that provide some general signals (e.g., vehicle speed) but also components that collect and process signals for one specific actuator (e.g., different brake demands). An *implicit communal component* is a communal component associated with a feature and not with the PCL. This means that the component contributes to a feature dependency by exchanging signals with a component that is associated with another feature. In contrast, we call a communal component *explicit* if it is associated with the PCL.

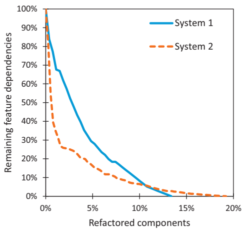
[1] Technische Universität Berlin, DCAITI, andreas.vogelsang@tu-berlin.de
[2] Technische Universität München, Institut für Informatik, {femmer,junkerm}@in.tum.de

2 Study and Results

RQ1: How many implicit communal components exist in automotive systems? Tab. 1a shows that in the examined systems more than a third of all components are implicit communal components. Thus, refactoring all implicit communal components at once is not realistic and, therefore, it is useful to consider implicit communal components as technical debt that should be removed if the refactoring pays off.

Characteristics:	**System 1**	**System 2**
Type	Small truck	SUV
Features	57	94
Components	269	380
Results for RQ1:		
Implicit communal components	97 (36%)	175 (46%)
Feature dependencies	136	1451

(a) Study objects



(b) Feature dependencies after refactoring implicit communal components.

RQ2: What is the distribution of feature dependencies over implicit communal components? Fig. 1b shows the remaining percentage of feature dependencies within the two systems after successively refactoring the component that contributes to the largest number of feature dependencies. For both systems, the number of feature dependencies decreases strongly after refactoring only a few components. In fact, to remove 90% of the feature dependencies, we need to refactor less than 10% of the components.
RQ3: What are the interest and refactoring costs for implicit communal components? From interviews with 3 experts, we extracted a set of cost factors. A cost factor impacts a development activity either in the role of refactoring costs or in the role of increased costs in the future. A cost factor may be influenced by contextual parameters that increase or decrease the severity of the cost factors.
Conclusions: We conclude that implicit communal components constitute technical debt in automotive systems that is worth to be managed. To implement or adapt features, automotive developers (re)use signals that are produced by components of other features. This leads to a large number of implicit communal components (36% and 46% of all components in our cases). When developers do this, they introduce feature dependencies, which, according to our interview participants, increase costs for several activities in the future (e.g., impact analysis, safety cases, testing, deployment). For most of the mentioned activities, their costs depend on the number of feature dependencies for which a communal component is responsible. In the examined systems refactoring some communal components has a much higher impact on feature dependencies than refactoring others. Therefore, for some communal components a refactoring saves more costs compared with others.

References

[VF13] Vogelsang, A.; Fuhrmann, S.: Why Feature Dependencies Challenge the Requirements Engineering of Automotive Systems: An Empirical Study. In: RE. 2013.

[VFJ16] Vogelsang, A.; Femmer, H.; Junker, M.: Characterizing Implicit Communal Components as Technical Debt in Automotive Software Systems. In: WICSA. 2016.

[VTG12] Vogelsang, A.; Teuchert, S.; Girard, J.: Extent and characteristics of dependencies between vehicle functions in automotive software systems. In: MiSE. 2012.