## 1.1 Expressiveness Assessment

In this section, we assess the expressiveness of SDSN in terms of sharing, variation, and change. In particular, as the main goals of this dissertation are to support tenant-specific variations and changes in the SIMT model, we consider the types of sharing and variation, and the types of change that our approach allows.

### 1.1.1 Support for Types of Change

This evaluation is to demonstrate the capability of the SDSN approach in supporting the different types of change to a multi-tenant service network that realizes an SIMT composite cloud application. The impact of a change to an element of a service network can lead to further changes to the service network. Below, we first present the experiment details including different change scenarios, and then analyse the results of the scenarios.

**Experiment Setup**. We have formulated a set of common change scenarios for SIMT cloud applications based on some product line research works [145, 242]. We implement these scenarios with our roadside

**Table 11.3.** Change scenarios for the roadside assistance case study

| No: | Types of Changes (High-level) | Example |
|---|---|---|
| CS1 | Add/Remove a mandatory feature | *Reimbursement* feature (to be used by each tenant) |
| | | Response time < 30 min and max-throughput=150 for all assistance cases |
| CS2 | Add/Remove an optional feature | *Accident Tow* feature (to be used by some of the existing tenants and the new tenant AsiaBus) |
| | | Response time < 2d and throughput = 10 for a reimbursement |
| CS3 | Add/Remove feature to a feature group | *TaxiHire* feature to/from the features *RentalVehicle* and *PublicTransport* |
| | | 4d repair duration in addition to the existing 2d and 3d for *Repair* feature |
| CS4 | Change Feature Type | Make the optional feature *Accident Tow* a mandatory feature |
| | | Make *Or* repair durations 2d, 3d, and 4d *Alternative (XOR)* options |
| CS5 | Add/Remove Feature Dependency | The dependency *Major Repair excludes Accommodation* |
| | | The dependency *Repair time=3d includes Tow duration = 4h* |
| CS6 | Modify Feature Implementation | Extend an implementation of the *Repair* feature to use an external parts supplier if the required parts are not available internally. |
| | | Add an external assessor to a repairing implementation, which increases repair time by 6 hours. |
| CS7 | Add/Remove mandatory feature implementations | One realization of *Reimbursement* to/from its other realizations |
| | | One realization of *Accident Tow* for duration=3h to/from its other realizations |
| CS8 | Add/Remove optional feature implementations | One realization of *Accommodation* to/from its other realizations |
| | | One realization of *Reimbursement* for response time =1d to/from its other realizations |
| CS9 | Add/Remove a feature implementation to a feature implementation group | One realization of *TaxiHire* feature to/from its other realizations |
| | | One realization of *Repair* for repair time=2d to/from its other realizations |
| CS 10 | Change feature implementation type | Make one of the two realizations of *Reimbursement* optional |
| | | Change the relationship between two realizations for *Repair* duration=2d from *And* to *Alternative (XOR)* |
| CS 11 | Add/Remove feature implementation dependency | The dependency *Accident Tow* with MarkTow excludes *Repair* with MacRepair |
| | | The dependency *Major Repair* duration=3d with AutoRepair *includes VehicleRental* duration=3d with SilverVehicles |

assistance case study (see Table 11.3). These scenarios consider the changes such as addition, removal, and update at the high-level features and their (multiple) implementations. For each scenario, we create the organization and operational management policies that capture the differences between the initial system configurations and the expected configurations after the realization of the change scenario. The created management policies are enacted at runtime on the original system. To validate the implementation of a change scenario, we first analyse log traces for all the expected changes. Second, we enact each virtual service network (VSN) in the service network, and collect the response messages and the log traces generated during the enactment of VSNs. We also manually create the expected service network configurations, deploy them to create a service network, and collect the same data by consuming each VSN in it. Then, we compare the collected data for the two cases to validate the implementation of the change scenario. The resources used by each change scenario are in the case study resources [239].

**Results and Analysis**. We have conducted a change and impact analysis for each change scenario to identify the types of change occurred at the service network level. We intentionally kept the complexity of each scenario low to make it easier to identify changes. The scenarios together cover each type of change at the service network level at least one time (see Table 11.4). We present some results for these scenarios to highlight different types of changes and impacts at the service network level.

The scenarios CS1-CS6 assume that a given feature only has a single implementation (a local collaboration between a set of services in the service network). The scenarios CS6-CS11 assume that a given feature have multiple implementations. Each scenario has a functional option and a performance option. For each option, the addition and the removal are considered. The removal of an option generally rollbacks the changed made by the addition of the same option. The case study also includes the addition and removal of include/exclude dependencies, the modifications to a feature implementation, and the type change of a feature and a feature implementation (e.g., mandatory to optional).

The scenario CS1 uses a collaboration between the 24by7 support centre and a motorist to implement the feature *Reimbursement*. 24by7 is already in the service network. This collaboration coordinates the new (or unused) reimbursement capability of 24by7, and the send notification capability of the motorist. The motorist requests the reimbursement from 24by, which fulfils the request and notifies the motorist once the reimbursement is done. The task *tReimburse* is added the role SC, and the tasks *SC.tReimburse*, *MM.tNotifyReimburseStatus* are added to the role MM. The interaction terms *iReimburse*, and *iNotifyReimbursementStatus* are introduced to the contract SC_MM to represent the new interactions between 24by7 and motorist. To reflect the task-interaction dependencies, the new tasks are updated with the references to the new interaction terms. To capture the configuration design of the reimbursement collaboration, a behaviour unit *bReimbursement* is added to with the references to the tasks *tRequestReimbursement*, *tReimburse* and *tNotifyReimburseStatus*. Two synchronization regulation rules are created at the roles SC and MM to execute tasks *tReimburse* and *tNotifyReimburseStatus*. Two routing rules are added to the roles MM and SC to route the request for reimbursement (*tRequestReimbursement)* and the response of the task *tReimburse*. Two pass-through rules are introduced at the contract SC_MM to process the interaction messages for the new interaction terms. To capture the regulation design of the reimbursement collaboration, the regulation unit *ruReimbursement* is created by including references to these new regulation rules. Since the feature *Reimbursemen*t is mandatory, each process in each VSN need to use it. Thus, the configuration and regulation designs of each process are updated to include the references to the behaviour unit *bReimbursement* and the regulation unit *ruReimbursement*, respectively.

To support the performance feature in the scenario CS1, the service network state type *ResponseTime* and its instances (*AssistanceProcessTime*) for each VSN are added. Then, the existing admission control rules (routing) for the instantiation request *iRequestAssist* at the role MM are updated to include a maximum threshold of 150 for each current VSN. Next, two pass-through rules are added at the contract SC_MM to monitor the response time being 30 minutes. The existing regulation units *ruAdmissionV1*, *ruAdmissionV2*, and *ruAdmissionV3* for the tenants HappyTours, EuroCars, and AnyTrucks already include the above admission control rules.  A regulation unit *ruCaseProcessingMonitoring* is created with the references to the new pass-through rules. The regulation design of each VSN is updated by including the references to this new regulation unit. All VSNs share the same monitoring rules, as the monitoring requirements are the same.

In the above scenario, a routing rule at the role MM is introduced to limit the admission of the instantiation requests across the three tenants (360 new assistance cases). A regulation unit *ruSnLevelAdmission* is created with the reference to the routing rule, and the regulation design across VSNs is updated by including the created regulation unit. Additionally, within in the same scenario, we have verified the addition, removal, and update of service network events (the event *eMMNotify* to/from all instances of the process AnyTrucksP1).

The implementation of the functional feature *AccidentTow* in the scenario CS2 creates a new collaboration between the motorist, 24by7, and AusLaw (law firm). As AusLaw is a new service, a new role that represents it and a set of new contracts that represent its interaction relationships with other relevant services in the service network are also created. Similar to the previous scenario, this one also includes the changes at tasks, interaction terms, behaviour units, regulation rules, and regulation units. The feature *AccidentTow* is an optional feature, and only the existing tenant HappyTours and the new tenant AisaBud bus seller want it. The configuration design and the regulation design for the process in the HappyTours's VSN are updated by adding a reference to the behaviour unit and the regulation unit for the accident tow collaboration. For the new tenant, the VSN AsiaBus is created.  The process AsiaBusP1 is added to the created VSN.  The configuration design of the process is created with the references to the behaviour units for the collaborations accident tow, repairing, vehicle rental, and case handling. The regulation design of the process is created with the references to the regulation units for these collaborations and their interactions. The regulation design of the VSN is also configured with the regulation units that capture the admission control for the VSN, and the instantiation of the instances of the VSN.

The realization of the performance feature in the scenario CS2 is similar to that in the scenario CS1. An admission control rule and two monitoring rules are added, and the regulation unit *ruReimbursementPerformanceV1* is created with the references to these rules.  Compared with the performance feature in CS1, this performance feature *"response time < 2d and throughput=10"* for the reimbursement is optional, and is only used by some tenants. Thus, the regulation designs for the processes in the VSNs of the tenants (HappyTours and EuroCars) who want the performance feature are updated by including the references to the new regulation unit.

Similarly, the realization for the scenario CS3 adds, removes, and updates elements in the service network. However, multiple variant features introduce additional variations because more compositions of features are possible. The alternative compositions generally introduce regulation-level changes (e.g., routing and synchronization rules). To support the multiple performance options with a single feature implementation (i.e., the same service), we have added a regulation mechanism (*SimpleMarker*) that can classify and mark the assistance cases (messages) for differentiated treatment by services. This mechanism simply adds a message header with the expected performance level (based on the configured values). The synchronization regulation rules that use this mechanism, as well as the regulation units that capture those rules are also introduced. The processes of the VSNs are updated with the references to the created regulation units.

In the scenario CS4, a change of a feature type generally results in regulation-level changes and VSN/process definition level changes. For example, when the optional feature *AccidentTow* becomes mandatory, all processes that have not used it need to be updated by including the behaviour unit and the regulation units for the accident tow collaboration. As a process can use both an accident tow collaboration and an ordinary tow collaboration, a new routing rule is added to select the accident tow if there is an accident. By altering the regulation units used by a VSN instance dynamically, the path of the instance and the regulation applied to the path can be changed (content-based dynamic routing in the service network). In the performance feature of the scenario CS4, the number of possible performance variations decreases because a single process can only uses a single repair performance option (*OR* to *XOR*). Thus, some regulation rules and units are removed, and the regulation designs of the affected processes are updated.

The changes to the exclude/include dependency between features (scenario CS5) also changes the number of possible feature sets, and the features in a given feature set. The processes need to alter their use of features by including new features or excluding currently used features.  Thus, the affected processes are updated by adding or removing the references to the behaviour units and regulation units corresponding to the affected features.  The next scenario CS6 modifies the configuration and regulation designs of the collaboration that implements a feature. As the impacts of these changes, the behaviour and performance of the collaborations can change, and thus the processes that use those collaborations need to be updated.

**Table 11.4.** Support for types of change and impact (service network level)

| Service Network Element | Types of Changes | | |
|---|---|---|---|
| | Add | Remove | Update |
| Service | CS: 2-3, 6-9 | CS: 2-3, 6-9 | CS: 1-3, 6-9 |
| ***Configuration Design Elements*** | | | |
| Role | CS: 2-3, 6-9 | CS: 2-3, 6-9 | CS: 1-3, 6-9 |
| Contract | CS: 1-3, 6-9 | CS: 1-3, 6-9 | CS: 1-3, 6-9 |
| Task | CS: 1-3, 6-9 | CS: 1-3, 6-9 | CS: 1-3, 6-9 |
| Interaction Term | CS: 1-3, 6-9 | CS: 1-3, 6-9 | CS: 1-3, 6-9 |
| Behaviour Unit | CS: 1-3, 6-9 | CS: 1-3, 6-9 | CS: 1-3, 6-9 |
| VSN | CS2 | CS2 | All |
| Process | CS2 | CS2 | All |
| Process Configuration Design | CS2 | CS2 | All |
| ***Regulation Design Elements*** | | | |
| Regulation Mechanism (Type) | CS3 | CS3 | CS3 |
| Regulation Rule | CS: 1-4, 6-10 | CS-1-4, 6-10 | CS-1-4, 6-10 |
| Regulation Unit | CS: 1-4, 6-10 | CS: 1-4, 6-10 | CS: 1-4, 6-10 |
| Regulation Design (Process) | CS2 | CS2 | All |
| Regulation Design (VSN) | CS2 | CS2 | All |
| Regulation Design (Across VSNs) | CS1 | CS1 | CS1 |
| Service Network Event | CS1 | CS1 | CS1 |
| Service Network State (Type) | CS: 1-2 | CS: 1-2 | CS: 1-2 |
| Service Network State (Instance) | CS: 1-2 | CS: 1-2 | CS: 1-2 |

In the scenarios CS7-CS11, the types of high-level changes are similar to the above scenarios (CS1-CS6). However, these scenarios consider the multiple realizations of a single feature, the type of a feature implementation, and the exclude/include dependencies between feature implementations. In general, these types of changes are similar to those of the scenarios CS1-CS5. However, the multiple realizations require regulation mechanisms that can select a subset of feature realizations (collaborations) at runtime, e.g., a load balancer and a content-based router.

Overall, the case study scenarios have demonstrated the feasibility of our SDSN approach to support common change types at the feature-level (and in an SIMT cloud application). They have also evaluated the types of service network changes covered by our SDSN approach (see Table 11.4).