# Tutorial for SAFARE: a tool for SAt-based Feature-oriented dAta aggREgation design

Simin Cai

School of Innovation, Design and Technology, Mälardalen University,
Västerås, Sweden
simin.cai@mdh.se

SAFARE (SAt-based Feature-oriented dAta aggREgation design) [1] is a tool for systematic design of Data Aggregation Processes (DAP). It provides interface to design DAP based on DAGGTAX (Data AGGregation TAXonomy) [1], and integrates the Microsoft Z3 Theorem Prover [2] to check the consistency of the design solution. This document describes the functionalities of SAFARE.

The graphical user interface of SAFARE is shown in Fig. 1. On the top lie three tabs, that is, **DAP Specification**, **Consistency Check**, and **DBMS Specification**, which are the three main functionalities provided by the tool. The **DAP Specification** tab is an interface for designers to specify the DAP of the designed system. Design constraints can be added by domain experts on **Consistency Check**, where the consistency of the DAP specifications can be checked against the constraints. On **DBMS Specification**, designers can specify DBMS transaction support, and associate the verified DAP with the appropriate DBMS specification.

You may need to set the correct PATH to the lib directory.

## 1 DAP Specification

This tab is divided into two parts. On the left side, the designer can specify the DAP of the designed system, while on the right side, the specified DAP can be connected into a multi-level aggregation design.

The designer starts the design with creating a DAP specification, by clicking the button **New Data Aggregation Process**. A panel **Data Aggregation Process** filled with (checkable) boxes and radio buttons are displayed, as shown in Figure 2, representing the features of a DAP in DAGGTAX. Among them, optional features are represented by checkable boxes. Mandatory features, except for the top-level features, are represented by boxes that are automatically checked, if their parent features are selected. A group of alternative features are represented by a group of radio buttons, of which, one and only one button has to be checked. New raw data types, and new data, can be added by clicking the **New Raw Data Type** button, and **New Raw Data** button, respectively. The specified DAP are loaded for analysis when the designer clicks the **Load** button.

Once the DAP are loaded, the designer can specify the dependencies between DAP on the right side of the graphical interface, which is the **Dependency between DAP** panel in Fig. 2. A dependency connects the aggregated data of one DAP with the raw data of another DAP, meaning that the former is used as the latter. A new dependency

[1] `http://www.idt.mdh.se/personal/sica/safare/index.html`

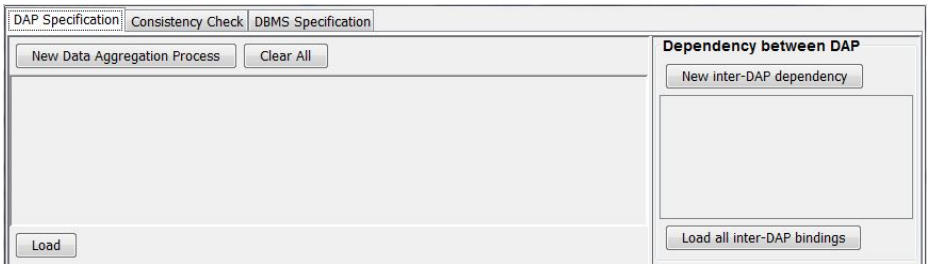


**Fig. 1.** Graphical interface of SAFARE.

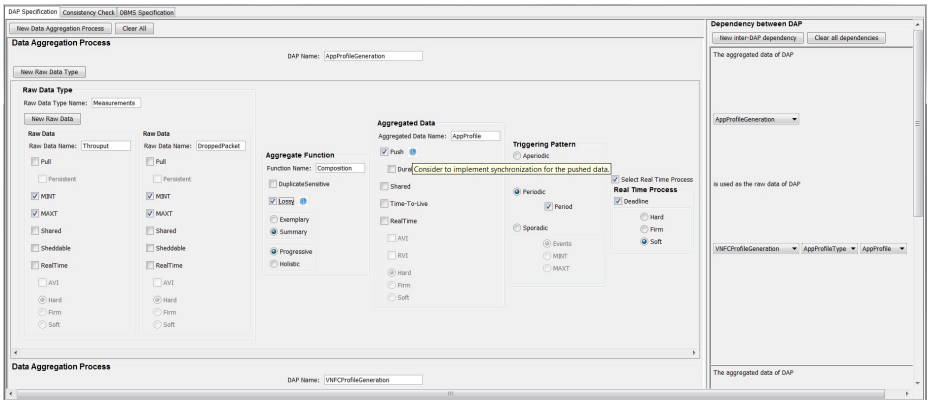

**Fig. 2.** Example: creating DAP and bind them to a multi-level aggregation.

is created by clicking the **New Inter-DAP Dependency** button. Then the designer can select the DAP and the raw data from the list in the combo boxes.

## 2 Consistency Check

The interface of consistency check is presented in Fig. 3. Domain experts can import design constraints into the constraint library, via the **Import Customized Constraints** button. The customized constraints are specified in SMT-LIB format, which is the standard format for specifying logical formulas in Z3 [2]. For instance, the following code is an example constraint in SMT-LIB format:

**Program 1.1.** constraint example in SMT-LIB

```
(declare-const VNFCProfileGeneration_AppProfile_RealTime_Soft Bool)
(declare-const AppProfileGeneration_AppProfile_RealTime_Hard Bool)
(declare-const AppProfileGeneration_AppProfile_RealTime_Firm Bool)
(declare-const AppProfileGeneration_AppProfile_RealTime_Soft Bool)

(define-fun rule () Bool
(=> VNFCProfileGeneration_AppProfile_RealTime_Soft
(or AppProfileGeneration_AppProfile_RealTime_Hard
```

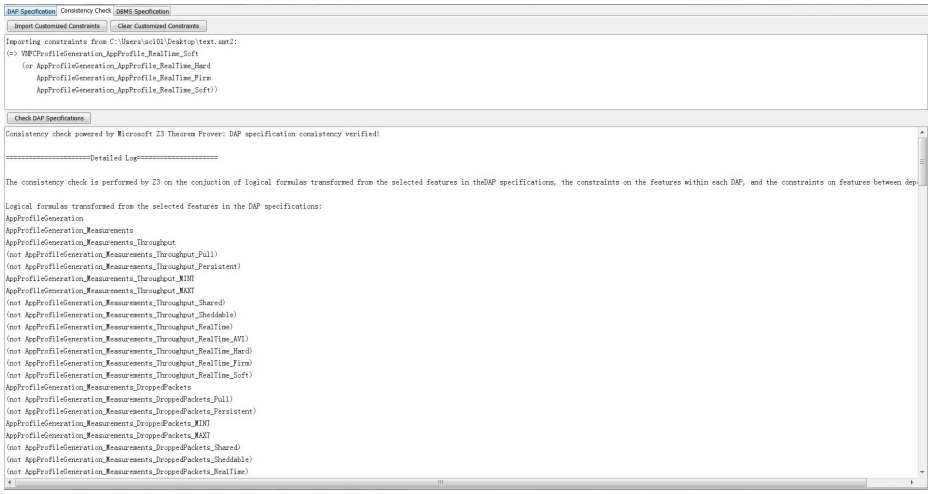


**Fig. 3.** Consistency check using SAFARE.

```
AppProfileGeneration_AppProfile_RealTime_Firm
AppProfileGeneration_AppProfile_RealTime_Soft)))

(assert rule)
```

This piece of code specifies the following rule:
*VNFCProfileGeneration_AppProfile_RealTime_Soft* $\Rightarrow$
*( AppProfileGeneration_AppProfile_RealTime_Hard* $\vee$
*AppProfileGeneration_AppProfile_RealTime_Firm* $\vee$
*AppProfileGeneration_AppProfile_RealTime_Soft )*.
Several general constraints have been imported by default into the tool.

The designer can check the consistency of the specifications by pressing the **Check DAP Specifications** button. The specifications and constraints are transformed into logical formulas, and automatically checked by Z3. In case a specification violates the specified constraints, the tool will return a failure message, and display the violation. If all constraints are satisfied, the tool returns a success message.

## 3 DBMS Specification

The designer can specify DBMS specifications in the **DBMS Specification** tab (Fig. 4). In the current version of the tool, a specification of DBMS only includes the transactional support for atomicity, consistency, isolation, durability, and timeliness. A new DBMS specification can be added via the **New DBMS** button. When all potential DBMS have been specified, the designer loads these DBMS to the tool by pressing the **Load All DBMS** button. Then the designer can select the DBMS for the specified DAP. Checked by the tool, only the DBMS providing the necessary transactional support for the DAP can be selected. When a DAP/DBMS bindings is loaded, the tool

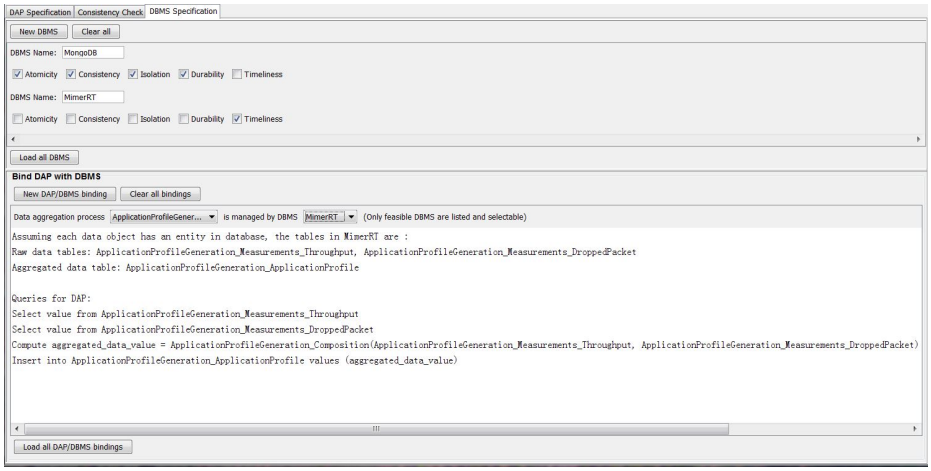


**Fig. 4.** DBMS specification, and binding DAP with DBMS.

generates a set of database tables for the data involved in the DAP, as well as SQL-like queries for processing these data.

## References

1. Cai, S., Gallina, B., Nyström, D., Seceleanu, C., Larsson, A.: Tool-supported design of data aggregation processes in cloud monitoring systems. Journal of Ambient Intelligence and Humanized Computing pp. 1–17
2. De Moura, L., Bjørner, N.: Z3: An efficient smt solver. Tools and Algorithms for the Construction and Analysis of Systems pp. 337–340 (2008)