This chapter gives an overview over the Quality of Service Modeling Language (QML), a language which can be used to describe QoS offerings or needs of specified services.

7.1 Motivation

A specification of the quality of a service is an important prerequisite for building dependable systems. During the requirements phase it can be used to gather the quality needs of the end user of a software system. During the design phase it helps in making design decisions and in early, architecture based quality evaluations. During runtime and testing it can be checked by monitoring tools to find insufficient quality characteristics of the running system. Further on, Quality of Service (QoS) negotiations, graceful QoS degradation or self-healing systems rely on specifications of the desired QoS levels.

In order to support the specification of QoS, the Quality of Service Modeling Language (QML) has been developed at HP Software Technology Laboratory by [170]. It is a language which can be used to describe QoS offerings or needs of specified services. A short overview of the most important concepts of the QML is given in the following and further detailed by utilizing a running example.

7.2 Main QML Concepts

QML introduces several concepts to model quality characteristics of services. Dimensions characterize a measurable value and several dimensions are bundled into contract types. For specific contract types multiple contracts using the dimensions of the type can be specified. Finally, profiles bind contracts to interface methods.

Dimensions

QML can be used to model any type of quality characteristic, e.g., it has no explicit quality model. Hence, it allows the specification of any generic quality attribute. Nevertheless, the specification has to introduce the type of the utilized attributes. This is done by using quality dimensions.

A dimension can be seen as the declaration of a domain of a variable. It tells the range of possible values of instances of the declared type. To give an example consider the quality characteristic performance. A typical attribute for performance is the timing related parameter delay. The delay describes the time which passes between the issue of a call and its return to the caller. Hence, it is a non-negative float point number having some kind of unit like seconds or microseconds associated to it. Additionally, in QML...
allows to specify a *direction* of the dimension, e.g., a specification of an ordering of the actual values. This allows to say, whether small or large values are better. For the example of delay small numbers are better as they indicate faster response times. Using QML, the delay dimension would look like code fragment 1.

```
delay : decreasing numeric msec;
```

**Code fragment 1: A QML dimension**

The first parameter (decreasing or increasing) describes whether smaller or larger values are better. The second determines the type of the dimension. Here we use a numeric type allowing float point values. Additional types available in QML are enumerated domains and set domains. Enumerations contain a list of names and a value can have exactly one of the names as content. A set domain is also a set of names, but the values are any possible subset of the set of names. Additionally, in both cases, it is possible to define an order on the set of names to introduce again a goodness relationship. An example for each concept can be found in code fragment 2.

```
cypher_algorithm : enum { RSA, DSA }; 
cypher_strength : increasing enum { 1024, 2048, 4192 } 
    with order { 1024 < 2048, 2048 < 4192 } bits; 
login_mechanisms : set { password, smartcard, fingerprint };
```

**Code fragment 2: QML enumeration and set domains**

Finally, every dimension can have an optional unit specification, like *bits* or *msec* in the examples above.

**Contract Types**

After introducing the dimensions and their associated specification of a partial order the following shows how to further abstract and bundle the dimensions into *contract types*. A contract type is closely related to a quality characteristic as introduced by [246]. Typical characteristics are performance, reliability or security. Note again, that QML does not define any dimension or contract type. This has to be done by the users of QML.

Hence, contract types bundle related dimensions and subsume them using a common characteristic. To give an example, we introduce a performance contract type. Performance can be determined by several dimensions. In this case the specification uses the delay and the throughput of services to describe their performance. The complete contract type can be found in code fragment 3.