5 Form-Based Application Generation

5.1 Overview

In this chapter, an architecture for form-based application generation is presented. The architecture is named \textit{FORMAG}, and features the use of NTs as canonical representations of objects including databases, interactive screens, and output reports. Since database manipulation, screen-based interactive sessions, and report generation are major components of most conventional application systems, FORMAG can provide a sound basis for their modeling and generation.

As mentioned in Chap. 2, the approach described here is related to our previous research on application generation in the context of XDB [158]. The uniqueness of the approach was that application generation was based on a structured application system model. However, in that study, the relational data model was used for modeling database manipulation and report generation, and an informal model was provided to represent the interactive session. In the approach presented in this chapter, application data processing such as database manipulation and interactive data entry is based on NTs. Therefore, the entire data manipulation in the target application system can be specified in a uniform way. Moreover, various new constructs such as the form procedure, NT procedure, and NTPL are introduced for NT handling.

FORMAG includes two important elements. One is a structured model for application systems, and the other is a facility for generating target application systems specified in the model. In the next section, Sect. 5.2, the application system model is defined. Two major concepts for application procedure modeling, namely the form procedure and the NT procedure, are also defined. In Sect. 5.3, an application procedure specification language NTPL is explained. NTPL is used to specify form procedures and NT procedures in this context. However, NTPL itself could be applied to more general procedural specifications of NT manipulations. In Sect. 5.4, the conceptual design of the application generation facility in FORMAG is presented, and an outline of application generation processes is explained with a simple example. Section 5.5 gives a summary of this chapter.

5.2 Application System Model

In this section, we define an application system model in FORMAG. Application systems to be generated in FORMAG are first specified in this model. In other
words, this model determines the range of application systems able to be modeled and generated within FORMAG.

As several studies in software engineering have emphasized, an application system can be modeled from a number of viewpoints. Common viewpoints include functional hierarchy, control flow, and data flow. In our model, an application system is first specified from the viewpoint of functional hierarchy. This corresponds to the hierarchical decomposition of the target system. A tree of functional components called activities results from this analysis. Figure 5.1 shows an example of such a tree. Upper level activities correspond to more abstract functions and lower level activities correspond to more primitive functions. In the decomposition here, leaf activities are to be represented as networks of steps. Typical data manipulations involved in steps are data entry, database browsing, query, database update, and report generation. A more precise definition of the model follows.

5.2.1 Activity Tree and Activity Diagram

An application system $AP$ is modeled as the following tuple:

$$AP = (T = (A, de), D, tf, DB).$$

Here, $T$ is a functional hierarchy called an activity tree, and is denoted by $T = (A, de)$, where $A$ is a set of functional modules called activities and $de: A \rightarrow 2^A$ is an activity decomposition function, meeting the tree condition defined below. An activity $A_i \in A$ is called a unit activity if $de(A_i) = \emptyset$. $D$ is a set of activity diagrams, and $tf: U \rightarrow D$, where $U \subseteq A$ is the set of unit activities, is a function associating an activity diagram with each unit activity. $DB$ is a database of application system $AP$ and is composed of a set of NTs.

Tree condition. Suppose $(A, de)$ is given; then a decomposition graph can be created by representing each activity $A_i \in A$ by a node $n(A_i)$ and drawing a directed arc from $n(A_i)$ to $n(A_j)$ if $A_j \in de(A_i)$. If the decomposition graph forms a tree, $(A, de)$ meets the tree condition.

Using the above formalization, we can represent the functional structure of an application system as an activity tree and associate a collection of primitive