CADDY: A Highly Integrated Environment to Support Conceptual Database Design

Gregor Engels  
Leiden University  
Dept. of Computer Science  
Niels Bohrweg 2, NL-2333 CA Leiden  
The Netherlands  
email: engels@rulwi.leidenuniv.nl

Perdit a Lohr-Richter  
Technical University Braunschweig  
Dept. of Computer Science  
Gaußstr. 12, D-3300 Braunschweig  
Germany  
email: loehr@idb.cs.tu-bs.de

Workshop Topics: Environments and Tool Integration, CASE Support for Formal Methods, Human-Computer Interface for CASE  
Keywords: Database Design Tools, Tool Integration, Conceptual Data Models

Abstract
This experience report describes the development of the CASE environment CADDY (Computer-Aided Design of non-traditional Databases) which offers an integrated set of tools for specifying, analyzing, and prototyping a database application on a conceptual level. This environment was designed and implemented during the last four years at Braunschweig Technical University (Germany). The process of developing this environment consisted of a sequence of development steps, where each step corresponded to (at least) one of five main tasks. The intention of this report is to describe these five tasks. At the end, we will critically reflect on the achieved results and solved problems, but also on still open questions of the CADDY environment.

1 Task 1: Development of a semantically well-defined theoretical background
We believe that any CASE environment that offers tools for the specification and implementation of a software system should be based on a semantically well-defined theoretical background. This means that syntax and also the semantics of all underlying concepts have to be defined at first.

In our case, the theoretical basis is given by a structured conceptual data model which was developed during the last six years at Braunschweig Technical University ([HNSE 87], [LS 87], [EgoHuelse 92]). As main characteristics, this data model provides integrated concepts to specify the static structure as well as the allowed and desired dynamic behaviour of a database application. For the specification of the
Whenever possible and sensible,

- . . . prefer visual languages to textual ones, and
- . . . adopt formalisms and concepts well-known to
  the database community.

The application of these principles leads to specification
languages which support an immediate comprehen-
sibility of the specification. So, for instance, we
defined visual, diagrammatic languages for the speci-
fication of EER schemata as well as for data flow
descriptions of database transactions [EgoHuelse 92].
On the other hand, the definition of a query language
SQL/EER [HE 90] and a language for integrity con-
straints [HH 91] were heavily influenced by the stan-
dard query language SQL for relational databases.

Due to our structured data model, a conceptual
database schema consists of the interrelated descrip-
tions of the static structure of a database application
and its allowed and desired dynamic behaviour. The
interrelations between the different specification parts
impose a lot of interrelations on the different specifi-
cation languages which in turn have to be observed to
yield a complete consistent specification. For instance,
it has to be guaranteed in the definition of an integrity
constraint that variables are only declared for entity
types which appear in the EER diagram, and that all
used data operations are defined in the corresponding
data type specification.

3 Task 3: Requirements analysis
and specification of an integrated
database design environment

One main motivation for the CADDY project was
to bridge the gap between theory and practice. Thus,
we focussed on the development of an integrated set
of tools on top of the semantically well-founded speci-
fication languages to support a comfortable design
of a conceptual database schema. In this respect, we
planned CADDY from the beginning to capture all
features of a second generation design environment.
These comprehend, for instance,

- a window-based, menu-oriented user interface,
- an easy to handle switch between different tools,
  ie, an integrated tool set,
- an open but integrated environment which can be
  extended by newly developed tools,
- visual resp. textual syntax-oriented editors,
  which check and guarantee contextfree as well
  as contextsensitive correctness of a document as
  soon as possible,
- support for an incremental working style, which
  facilitates, for instance, analyzing and executing
  of still incomplete specifications.

Besides fixing the general requirements for the char-
acteristics of an environment, the necessary tool set
has to be determined, too. Depending on the design
tasks to be supported, such tool sets may consider-
ably differ. For our purposes, ie, the support of the
conceptual design phase, the CADDY tool set consists of

- a set of coupled editors for all specification parts,
  ie, visual editors for EER diagrams and data flow
diagrams and textual editors for algebraic speci-
fications, integrity constraints, and SQL/EER
queries.
- analyzing tools to check complex consistency con-
ditions between different specification parts.
- prototyping tools to gain experiences with the ap-
  plication developed so far, ie,
  
  - an automatic transformer of EER diagrams
    into relational database schemata to install
    a test database on a relational DBMS,
  - interpreters for SQL/EER queries and data
    flow diagrams,
  - a monitor for static and dynamic integrity
    constraints,
  - a visual database browser to navigate
    through the test database.

This chosen variety of tools [EHHL 89] forms a con-
venient basis to assist the designer in performing the
conceptual design of an application.

4 Task 4: Design of an integrated
database design environment

To guarantee the above mentioned general features
(openness, integration, incrementality, etc.), the over-
all software architecture of the whole environment has
to be determined in advance before starting with the
implementation of any tool. Consequently, the soft-
ware architecture reflects all relevant design decisions
from which we want to present here only the most
important ones.

The requirements for an integrated tool set are cov-
ered by the use of a central project database. It stores
all generated documents in terms of one uniform inter-
 nal representation, to which the tools commonly ac-
cess. This avoids unnecessary transformation steps be-
tween tool-dependent internal representations of doc-
uments.

The intended working style of the tools heavily
influences the choice of the concrete internal represent-
ation of the documents. Because we always emphasized
the integration aspect of our environment, we em-
ployed abstract syntax graphs for this purpose [ES 89].
Those abstract syntax graphs are extensions of usually
used abstract syntax trees. Here, besides contextfree
informations, all contextsensitive interrelations within
a document and also between different documents are
explicitly expressed by additional edges. The tools use
this information to execute in a highly integrated man-
ner. This approach of using abstract syntax graphs
was originally developed within the IPSEN project

...
and now successfully applied in the CADDY project. As already mentioned, the central project database stores these internal representations of documents and acts as a special purpose database kernel supporting very fast accesses to these graphs [LS 88].

To enable the integrated applicability of the tools, we introduced a control component (CADDY Control) to the CADDY architecture which encapsulates the necessary synchronisation of tools, transmits external events to the tools and controls the access to the abstract syntax graphs. Besides these tasks, CADDY Control is planned to support different design styles, i.e., incremental or corporated design.

Finally, we decided to support an easy portability of CADDY. Thus, the implementation bases on standard platforms like UNIX as the operating system, X Window to build the window-oriented user interface and a relational DBMS (INGRES) for the test database.

5 Task 5: Implementation

The implementation of CADDY has begun in 1988. Up to now, we invested about 30 years of person power and generated more than 200 000 lines of C code for the CADDY prototype. So in the meantime, we succeeded in developing a running prototype implementation. We successfully demonstrated earlier versions of this prototype on several conferences 1 in Europe. The technical background for the implementation comprises SUN-Workstations as hardware platforms, the special purpose database kernel named GRAS [LS 88] for the project database, and the relational DBMS INGRES as underlying test database.

6 Critical Reflection

During the process of developing CADDY, we concentrated on the construction of a sound prototype in order to ascertain the applicability of our concepts. By this proceeding, we learned a lot of the nature of our approach.

In this respect, we experienced the usefulness of a well-defined theoretical background. Its existence ensures clear definitions of languages, tools, and interfaces and simplifies the division of design and implementation work.

Similarly, we recognized areas where concepts are necessary but still missing. For instance, our data model lacks of clearly defined module-like structuring mechanisms. As real applications tend to become very large, it is necessary to structure them in more abstract units and, thus, to offer well-structured surveys about the current state of the design work.

Another shortcoming occurs in that we have no well-defined design method for our specification languages. Thus, we realized a simple, general purpose design process in CADDY which only incompletely reflects possible method facilities. But this topic is part of current and future research work [Lo 92].

Besides the missing concepts, we succeeded to reach our goal of a highly integrated environment. Due to our approach with a common internal representation via abstract syntax graphs, we gain a set of completely integrated tools and additionally, preserve extensibility for future tools. Complementary, the uniform appearance of the tools’ interface completes the intended tool integration.

Summing up, we obtained a prototype for CADDY which contains some shortcomings in functionality but with which we achieved to prove our concepts practicable and which is skillfull enough to conveniently support the process of performing conceptual database design.

Acknowledgements

We gratefully acknowledge our (former) colleagues Christiane Beer, Uwe Hohenstein, and Klaus Hülsmann for their contributions to the development of CADDY. Accordingly, we thank all students working hard for the implementation of CADDY. Furthermore, acknowledges are due to the research group of Prof. Nagl, RWTH Aachen, which supported our work by putting the graph storage at our disposal.

References


