Design

DaCoPAn

Helsinki 31st May 2004

Software Engineering Project

UNIVERSITY OF HELSINKI
Department of Computer Science

UNIVERSITY OF PETROZAVODSK
Department of Computer Science

Course

581260 Software Engineering Project (6 cr)

Project Group

Carlos Arrastia Aparicio
Jari Aarniala
Alejandro Fernandez Rey
Vesa Vainio
Jarkko Laine
Jonathan Brown

Kirill Kulakov
Andrey Salo
Andrey Ananin
Mikhail Kryshen
Viktor Surikov

Customer

Markku Kojo

Project Masters

Juha Taina (Supervisor)
Yury Bogoyavlenskiy (Supervisor)

Turjo Tuohiniemi (Instructor)
Dmitry Korzun (Instructor)

Homepage

`http://www.cs.helsinki.fi/group/dacopan`

Change Log

Version	Date	Modifications
1.0	Put the date here	First version

Contents

1 Introduction

This document defines integration design for DaCoPAn software according to [4]. The document can be considered as a model for sufficient implementation of the requirements, stated in the Requirements specification [3].

The DaCoPAn software is composed by two main subsystems, which are the DaCoPAn Analyzer and the DaCoPAn Animator. The specific design of each of them will be presented in separate documents [1, 2].

The integration architecture of the DaCoPAn software is presented in section 2. The integration subsystems interface and Protocol events file document type definition is described in section 3.

The document is intended mainly for the project development team. Experts from customer's side may analyze this document to be sure that the requirements are going to be implemented sufficiently and efficiently.

This specification may be changed during the implementation phase. All such changes must be shortly described and grounded in a separate document — The Implementation Document.

2 Architecture

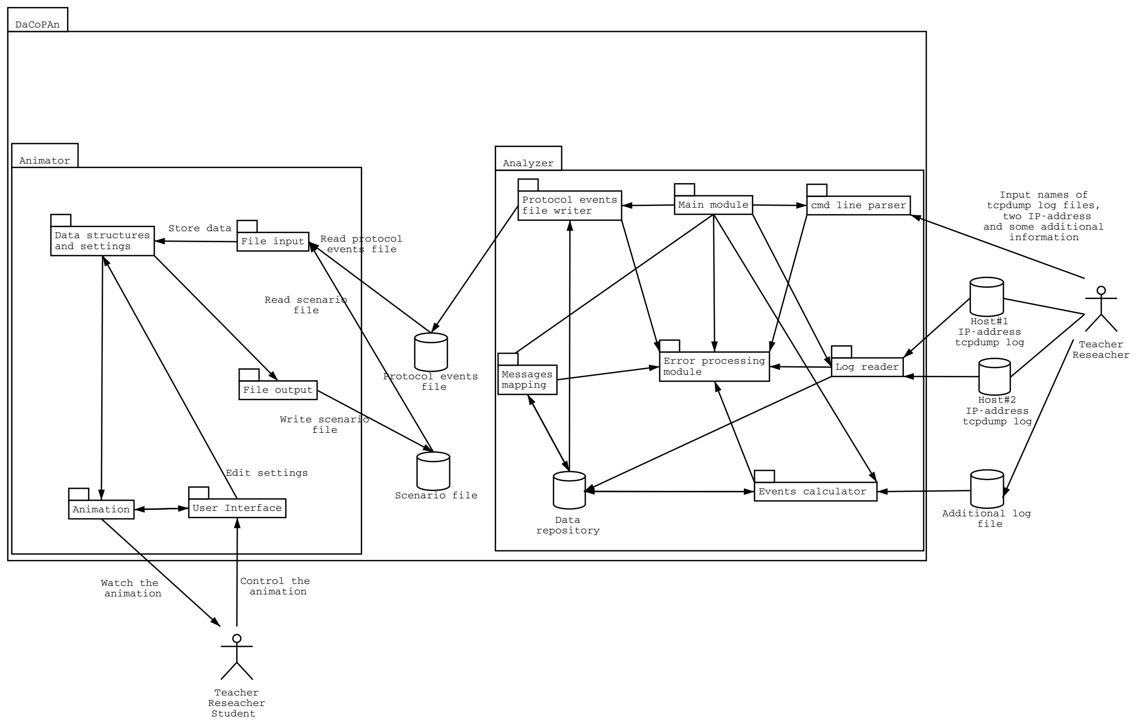

Figure 1: General architecture model of the DaCoPAn product

The high-level architecture model of the DaCoPAn system is presented in Figure 1. This is more detailed architecture for the product than in the *Requirements specification* [3].

The diagram shows interaction between two main subsystems, their modules, interaction between users and the DaCoPAn system, and the input and output data.

The DaCoPAn software consists of two large subsystems: the Analyzer and the Animator, which are connected by a file called *Protocol events file*. The Animator also uses another file for adding user settings and comments to the data produced by the Analyzer. This file is called the *Scenario file*.

Figure 1 shows different groups of users and relations between groups and components of the system. Lecturer and researcher use the Analyzer for creating Protocol events files. This file contains data about the network traffic that will be animated by the animator. Each user group can change any data in Protocol events file manually. Each group use the Animator for observing the behavior of protocols in a network.

3 Subsystems interface

The Protocol Events File (PEF) is the interface between Analyzer and Animator modules. It is the output of the Analyzer, it is read by the Animator, and it contains the necessary packet interchange data. Some of this information may as well have been added manually to the PEF. The use of XML has been preferred in an attempt to make the PEF extensible, human-readable and editable.

The DTD (Document Type Definition) to which any PEF must conform follows. It contains information about:

- **Hosts**: include an id for internal use, its IP address, and maybe a hostname.
- **Flow**s: include an id, 2 host ids, and their 2 ports.
- **Link**s: include an id and 2 host ids.
- **Layer**s: include an id and name; inside each layer definition, the protocols particular to that layer can be found. **Protocol** definition includes an id and a name.
- **Variables**: contain constant and dynamic variable definitions. **Constant**s include a name, and maybe a host id, a link id and/or a protocol id. The value of the constant variable is specified after the definition. **Variable**s (those variables that change from unit to unit, for instance) include name, protocol id and scope (flow/unit/unit-field). Their values are specified later inside the units.
- **Events**: different types of units compose the actual packet data. **Unit sent**s include id, source id, destination id, protocol id and may include flow id, time and children id list. Values of dynamic variables are specified inside them. **Unit received**s include an id corresponding to a unit sent id, and the time when the unit was received in destination. **Unit dropped**s have an id and the time when the unit is dropped.

```
<!--
    Document type declaration (DTD) for the protocol events file format.
    $Id: events.dtd,v 1.7 2004/05/07 13:56:25 aarniala-dacopan Exp $
  -->

<!-- Root element of a PEF document -->
<!ELEMENT protocol_events (hosts, flows, links, layers, variables, events)>

<!-- List of hosts present in the scenario -->
<!ELEMENT hosts (host*)>

<!ELEMENT host EMPTY>
<!-- Host id format: H1, H2 etc. -->
<!ATTLIST host
    id ID #REQUIRED
    ip CDATA #REQUIRED
    hostname CDATA #IMPLIED>

<!-- List of distinguishable flows (e.g. TCP connections) in the scenario -->
<!ELEMENT flows (flow*)>

<!ELEMENT flow EMPTY>
<!-- Flow id format: F1, F2 etc. -->
<!ATTLIST flow
    id ID #REQUIRED
    host1 IDREF #REQUIRED
    port1 NMTOKEN #REQUIRED
    host2 IDREF #REQUIRED
    port2 NMTOKEN #REQUIRED>

<!-- A network link between two hosts -->
<!ELEMENT links (link*)>
<!ELEMENT link EMPTY>
<!ATTLIST link
    id ID #REQUIRED
    host1 IDREF #REQUIRED
    host2 IDREF #REQUIRED>

<!-- Network layers -->
<!ELEMENT layers (layer*)>
<!ELEMENT layer (protocol*)>
<!-- Layer id format: L1, L2 etc. -->
<!ATTLIST layer
    id ID #REQUIRED
    name NMTOKEN #REQUIRED>

<!-- Protocol definition -->
<!ELEMENT protocol EMPTY>
<!-- Protocol id format: P1, P2 etc. -->
<!ATTLIST protocol
    id ID #REQUIRED
    name NMTOKEN #REQUIRED>

<!-- Variables section with constants and dynamic variable definitions -->
<!ELEMENT variables (constant*, variable*)>

<!--
    Variable with a constant value. May be specific to a host,
    link, protocol or a combination of thereof.
```

```
-->
<!ELEMENT constant (#PCDATA)>
<!ATTLIST constant
    name NMTOKEN #REQUIRED
    host IDREF #IMPLIED
    link IDREF #IMPLIED
    protocol IDREF #IMPLIED>

<!-- Dynamic variable definition (value is unit-specific) -->
<!ELEMENT variable EMPTY>
<!ATTLIST variable
    name NMTOKEN #REQUIRED
    protocol IDREFS #REQUIRED
    scope (flow|unit|unit-field) #REQUIRED>

<!-- Actual protocol events data -->
<!ELEMENT events (unit_sent|unit_received|unit_dropped)*>

<!-- The event for a sent unit -->
<!ELEMENT unit_sent (value*)>
<!-- Unit id format: U1, U2 etc. -->
<!ATTLIST unit_sent
    id ID #REQUIRED
    flow IDREF #IMPLIED
    source IDREF #REQUIRED
    destination IDREF #REQUIRED
    protocol IDREF #REQUIRED
    time NMTOKEN #IMPLIED
    children IDREFS #IMPLIED>

<!-- The value of a dynamic variable -->
<!ELEMENT value (#PCDATA)>
<!ATTLIST value
    name CDATA #REQUIRED>

<!-- The event for a received unit -->
<!ELEMENT unit_received EMPTY>
<!-- The "id" attribute references the corresponding unit_sent -tag -->
<!ATTLIST unit_received
    id IDREF #REQUIRED
    time NMTOKEN #IMPLIED>

<!-- The event for a dropped unit -->
<!ELEMENT unit_dropped EMPTY>
<!ATTLIST unit_dropped
    id IDREF #REQUIRED
    time NMTOKEN #IMPLIED>
```

References

1 DaCoPAn Software Engineering project, *Design: Analyzer*. Release 1.0. Universities of Helsinki and Petrozavodsk, April 2004.

2 DaCoPAn Software Engineering project, *Design: Animator*. Release 1.0.

Universities of Helsinki and Petrozavodsk, 2004.

3 DaCoPAn Software Engineering project, *Requirements specification*. Release 1.0. Universities of Helsinki and Petrozavodsk, March 2004.

4 Taina J., Korzun D., Tuohiniemi T., Alanko T., Bogoyavlenskiy Y., *Software Engineering Project: Distributed Approach*. Release 1.0. Universities of Helsinki and Petrozavodsk, January 2004.