## Software technical description

<table>
<thead>
<tr>
<th>Document Date</th>
<th>September 11, 2007</th>
</tr>
</thead>
</table>
| Organisation and authors | Marcell Szathmári, SZTAKI  
Zsolt Kemény, SZTAKI |
| Abstract | This document complements the architecture description and focuses primarily on required functionalities of the TraSer solution platform, the proposed underlying structure of the solution, as well as the standards and off-the-shelf components selected for the sample implementation to be produced within the TraSer project. |

SZTAKI: Computer and Automation Research Institute, Hungarian Academy of Sciences
1. INTRODUCTION

This document complements the architecture description and focuses primarily on required functionalities of the TraSer solution platform, the proposed underlying structure of the solution, as well as the standards and off-the-shelf components selected for the sample implementation to be produced within the TraSer project.

2. DEFINITION OF TRACKING AND TRACING

The main field of use of the TraSer solution platform is tracking and tracing of individual entities; therefore, an exact definition of these terms is needed. Although tracking and tracing can be used in a wide spectrum of application areas, all share common characteristics which can be specified without exact knowledge of the requirements of the application using tracking and tracing.

**Tracking** is the act of keeping track of one or more selected properties of an entity (in logistics and delivery processes, this is usually the physical location of the item). Usually, the following two technical solutions must be provided to support tracking (note that not all application cases may require both of these prerequisites):

- the properties of interest must be made visible to users or other system components in charge of observation, and
- the properties of interest must be archived in a retrievable form to support operations on historic data.

**Tracing** is the act of keeping track of the relations of a selected entity with other entities (such as a built-into relation of sub-assemblies in composite products), or following a chain of such relations. Tracing resembles tracking with respect to observation and shares, therefore, the above mentioned prerequisites with it. In addition, means of representing relations between individual entities must be provided, e. g., through special attributes which serve as pointers to other items.

3. TARGET SPECIFICATION

The TraSer project aims to deliver a set of specifications which, when followed, make it possible to track and trace uniquely identifiable individual entities with respect to their selected properties within a given organizational unit and across organizational borders as well. The project is also devoted to developing a sample implementation of these specifications in the TraSer solution platform. Based on this sample implementation, the TraSer project will create a set of applications that can be used for simple tracking and tracing problems out-of-the-box and serve as an explanatory example for developers who wish to further customize the operation.

4. NON-FUNCTIONAL REQUIREMENTS AUGMENTING THE ARCHITECTURE DOCUMENT

The solution to be delivered will operate on the Internet, a possibly adverse network, which necessitates secure communication and access control when accessing information. This also induces the need of authentication. Some participants of the network may have no or just intermittent network access, thus it is necessary to use some buffering mechanism. It may be necessary to provide some guarantees of the delivery of messages.
Similarly, the forwarding of messages must be provided for cases where a group of participants is protected by a common firewall which all messages must pass through. As the TraSer project is limited in time and development resources, the needs of new or evolving application domains can be only met by ensuring extensibility of the data model, as well as the set of applicable messages.

5. DRAFT ARCHITECTURE

A TraSer network consists of TraSer nodes (i.e., servers in charge of maintaining item-related information in their database) and TraSer clients (which provide interfaces for querying and updating of item-related information by human operators or other hardware/software components). The requirements listed below determine the architecture proposed for the TraSer network.

- TraSer focuses on items which are unambiguously marked with an identifier. The latter can be entered into TraSer through a reader unit which is either a specific piece of hardware (RFID/barcode reader, alphanumeric terminal etc.) or software (in the case of product data management). Identifiers are static, they cannot be altered by the system. Data migration from one identifier to another is supported by the system, but the method of re-labeling is out of scope of TraSer.

- Data associated with an item (having a given identifier) can be updated or queried through one given node in charge. An administrator is responsible for configuring and maintaining the TraSer node. Although the TraSer project will specify and implement one given software solution for TraSer nodes, other pieces of software may be TraSer-compliant as well, as long as they provide the same services, data and communication interfaces. This is important as some potential industrial customers have already pronounced their preference of re-implementing TraSer components with their own software tools.

- TraSer Clients are entities which do not permanently store item-related data and provide no services related to given items but can contact TraSer nodes to use their data and services. The primary purpose of clients is communication with a user who can be either a human operator or a piece of hardware/software. Although we may specify a given range of possible TraSer clients, there are, theoretically, no constraints concerning the innards of a potential client, as it is only the communication behavior which determines its validity as a TraSer client.

- It is a general principle that only TraSer nodes and TraSer clients can be directly connected to TraSer nodes. Other components, such as already existing ERP systems, can be linked to TraSer nodes through special clients acting as adapters.

Fig. 1 shows possible interactions between the entities listed above—note that the interfaces marked with dotted lines may not necessarily belong to the core specification of TraSer.
DRAFT USE CASES

As an initial working assumption, the set of draft use cases listed below was found typical for the targeted application domain. The cases were grouped to facilitate systematic work in the future and changes or new groups of use cases may be introduced as development work advances.

- Information access (either local or remote). Communicating parties need to obtain attribute information (e.g., location) associated with a given item for the sake of transparency, we suggest that both remote and local (i.e., with node and client running on the same physical machine) access requests should be forwarded to the TraSer node in charge through a suitable TraSer client.
- Attribute update (either local or remote). Communicating parties store attribute information (e.g., location) associated with a given item of a TraSer node. Also here, we suggest to carry out all communication through clients. (For these two examples, see also Fig. 2.)
- Gates. Automatic reader stations can be programmed to update the location attribute of an item upon its detection. This case may be grouped, in fact, with other examples of remote attribute update. From the point of view of node/client communication, this may be more or less equivalent to local attribute updates, as explained above (see also Fig. 3).
- Web user interface. The client communicates with the user through a web interface and posts requests to a TraSer node accordingly. This means that a user posts requests as HTTP requests, the client translates these requests, and returns a webpage according to the response of the TraSer node (see also Fig. 4).
- Attaching already existing systems. Already existing systems can be attached to the system through special clients which can handle the protocols of the attached system and forward them as TraSer-compliant requests to the node or vice versa (see also Fig. 5).
- Composing a TraSer network. A TraSer node can propagate either updates or queries to other nodes in the system (see also possible connection to other nodes on Fig. 1). This is needed when items of interest are combined with other entities maintained by other TraSer nodes, and either properties are shared (such as items
moving together with a container), or tracing activities are performed (e.g., over sub-assemblies provided by different suppliers).

Figure 2: TraSer client with local user access—the same topology may apply to remote access, e.g., for a gate with optional human intervention in “fallback” cases

Figure 3: TraSer client with an automatically operating reader

Figure 4: Another example of remote access—web user interface for accessing item data

Figure 5: Attaching already existing systems to a TraSer node
7. DRAFT BLOCK DESCRIPTION

7.1. TraSer nodes (servers)

TraSer nodes (servers) receive encrypted messages, either from a TraSer client or another TraSer node. The node decodes the messages, authenticates the sender, and decides whether the sender is authorized to update/query the piece of information in question. If authorization is negative, the request will be discarded (and a rejection reply issued to the sender), otherwise, the processing goes on:

- If the message is an update, its contents are mapped from the event level to the data level, data are updated and historic values are stored in the system if archiving is required. If defined by the event, other events are created and sent to other nodes.
- If the message is a query, the return message is populated from the data stored in the system and is returned.

![Figure 6: Processing of incoming messages in a TraSer node](image)

7.2. TraSer clients

Even though a variety of TraSer clients can be implemented for a wide spectrum of purposes, from a single-user web interface to an adapter connected to an ERP system, all clients share the characteristics of an interface with built-in intelligence. The communication between the user (or external software/hardware component) and the client has two major steps:

- the user provides information to the client about "which item is meant" and "what to do with the item" (the latter may be hard-coded in a given implementation, e.g., an automatic gate reader), and
- the client returns the requested data, an acknowledgement of the transaction, or a failure/rejection notification.

A typical working cycle of a client begins with an input from which an identifier is to be extracted. For this purpose, the client examines the input for a set of possible syntaxes, and if the input conforms to one of these, the identifier is extracted. Next, the client obtains the
address of the TraSer node in charge of hosting the information about the item which has the identifier in question. If the address of the TraSer node can be resolved, the client contacts the node and sends a message corresponding to the operation requested by the client. When the node reply arrives to the client, it executes the adequate action which can be, depending on the client type, a simple acknowledgement notification towards the user, the displaying of results or forwarding of data to another software or hardware component connected to the client (e.g., an ERP system).

Figure 7: Processing of external or user inputs in a TraSer client

Figure 8: Processing of node messages in a TraSer client

8. OFF-THE-SHELF COMPONENTS USED

Since the above mentioned requirements are not only specific to TraSer but occur frequently in various applications, some or all of them are already covered by standards (in some cases only proposals with the prospect of becoming standards). It is not uncommon that these standards have open implementations, and it is assumed that using some of these will benefit the development efforts of the TraSer project in terms of maintainability and acceptance by prospective users and the open-source community.

For the purposes of TraSer, two families of standards were found to be useful: SOAP and the WS-* set of standards for communication related problems, and the XML related standards for extensibility and interoperability issues.
8.1. SOAP and WS-* standards

SOAP is a widespread communication envelope format for use over several transport channels. SOAP defines a mechanism for extending the envelope with value-adding services, such as security and reliability. The WS-* set of standards defines extensions conforming to the SOAP extension model and is used to implement aspects of communication needed in B2B applications.

Requirement issues of secure communication and reliability guarantees are already covered by WS-* standards WS-Security and WS-ReliableMessaging, both being approved standards of OASIS. There are existing open-source solutions for each of these standards. We chose to use a set of Apache products in TraSer, namely Axis2, Sandesha and Rampart. Axis2 implements SOAP communication and has a well-defined modular architecture. This architecture enables us to later easily deploy our own WS-* proposal if needed, while it also serves as the basis for the mentioned other two products, both of which come in the form of Axis2 modules. Rampart is an implementation of WS-Security, while Sandesha implements WS-ReliableMessaging and defines extension possibilities to buffer messages during communication drop-outs which can be used to cover another requirement to support devices with intermittent network access.

8.2. XML standards

XML standards provide the capability of integrating information from multiple schemes into one document, and a full set of interpreted languages capable of manipulating, transforming and querying this information. Available open-source implementations of XML standards (Apache Xalan, Apache Xerces, Apache Axis2 Object Model AXIOM, Apache Axis2 Data Binding and SourceForge project Saxon along with the aforementioned B2B communication implementations) serve as an excellent basis for the creation of a distributed database, a system which TraSer has several similarities with.

9. REFERENCES