# Practical Approaches for Software Components Integration in Telecommunications

S. Rumley [*], D. Savić [**], F. Potorti [†], S. Tomažic [**] and C. Gaumier[*]
[*] Laboratoire de telecommunications, EPFL
TCOM / STI, Station 11, CH-1015 Lausanne, Switzerland
[**] Laboratorij za telekomunikacije, FER
Trzaska 25, SLO-1000 Ljubljana, Slovenia
[†] Istituto di Scienza e Tecnologie, CNR
Via G. Moruzzi 1, I-56124 Pisa, Italy
E-mail: sebastien.rumley@epfl.ch

**Abstract - Nowadays, advances in telecommunication network design and performance analysis often rely on dedicated software tools. Unfortunately, developing new tools is a very time and resources consuming activity. To rationalise development costs, existing applications can be extended. Alternatively, existing software components can be combined and integrated. Integration of heterogeneous components requires many efforts, in particular when the specific input/output data formats have to be adapted. Furthermore, the amount of data exchanged between the components can be huge and needs intermediate processing. To facilitate data exchange between tools, two concepts are presented in this paper: CostGlue and the Multilayer Network Description (MND). Their utilisation modes and the advantages they provide are illustrated through a practical example.**

## I. INTRODUCTION

Present communication networks are complex entities, implying equally complex analysis and/or planning techniques. Software tools are often used to mitigate the complexity of theses techniques. As there is a wide set of communication devices, protocols, or systems, the set of tools related to network modeling or design is also very large [1].

Very often, a tool or a combination of tools developed to address a particular problem might be employed to address a similar but different problem. If possible (only minor changes required), the reuse of one of the numerous existing tools (reusability principle) should be favoured. Alternatively, if a new development has to be undertaken, existing modules of software parts might be recycled, which also reduces the development costs (component oriented design principle) [2].

These two principles are however often difficult to implement in practice, for various reasons:

- A tool might require specific conditions, which makes its integration with other tools difficult. These conditions can be related to the execution environments (e.g. operating system, database, file system or libraries) or to the employed programming language. Different languages indeed reduce the ability to combine separate parts of tools.
- The employed input/output (I/O) formats can still be very heterogeneous.
- Certain types of tools (network simulators in particular) produce huge amounts of data, which are difficult to store and reuse.
- The lack of any kind of data descriptor (metadata) may lead to confusing situations, where dozens of intermediate datasets (set of data resulting from one or more processing step) cannot be differentiated.

All of these issues prevent the reutilization of an output dataset as an input to another component, and more generally reduce the support for data exchanges between components.

This paper presents a combination of two approaches addressing the aforementioned points. The first one is the CostGlue project [3]. It defines ways of storing simulation results efficiently, and describes how raw simulation data, metadata and post-processing data should be structured and exchanged. The CostGlue framework has been setup according to this description. It includes different types of plugins, capable of performing different tasks: data import and data export (using formats for third party tools [8]), data post-processing and data visualization.

The second approach, the Multilayer Network Description (MND), proposes an XML based document format conceived for simple but efficient exchanges between different software tools. An MND document is organised in a variable and unlimited number of layers. This organisation provides a good overview of the contained data.

The rest of the paper is organized as follows. Project CostGlue is presented in the section 2. The Multilayer Network Description (MND) is presented in more details in Section 3. Section 4 lists and comments several situations where MND and CostGlue can be favourably used. Section 5 concludes the document.

## II. HANDLING SIMULATION DATA WITH COSTGLUE

The project CostGlue [3] is intended to facilitate simulation data exchange. It includes the CostGlue data exchange model and its metadata XML description, the CostGlue framework, and a prototype implementation.

### A. Data Exchange Model

CostGlue proposes a data exchange model for structured storage of three types of data: raw simulation data, post-processing data, and the associated metadata. The structure is based on the Open Archival Information System (OAIS) reference model [5] with a more detailed description of simulation data collection in the field of telecommunications. All three types of data are stored in one or more common archives.

When dealing with a large number of archives, it is possible to aggregate them in catalogues. Later, these catalogues can be published in repositories for scientific communities. For this reason, the taxonomy from the Council for the Central Laboratory of the Research Councils (CCLRC) Scientific Metadata Model (CSMD) is employed [6].

### B. CostGlue Framework

A framework, which packages simulation output data into a archives, launches post-processing plugins and exports results in various formats, has been defined, based on the data exchange model. It consists of a core, a database, an API (Application Programming Interface), and an arbitrary number of specialized plugins (Fig. 1). The core communicates with the database, and acts as a unified interface for writing to it and reading from it. Several specific functions, such as import and export of data and different mathematical calculations, are represented as a set of self-described plugins which can be loaded if necessary. The plugins access the database by interacting with the core through a well-defined API.

### D. Prototype Implementation

The proposed data exchange model and the framework has been implemented as proof of concept. It constitutes the CostGlue software package, and will be released under GNU Lesser General Public License (LGPL).

According that results of on-the-fields measurements (obtained for instance with a network analyser) are similar to simulator output, both the results of simulations and the results of measurements can be stored in the CostGlue database. This allows an easy comparison between these two types of data.

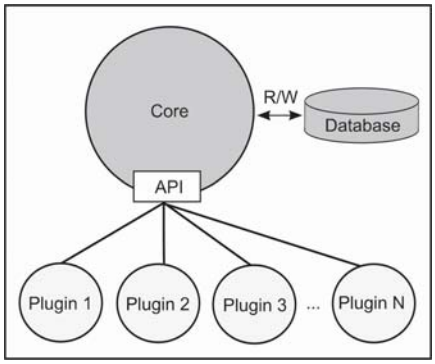


Fig. 1: Architecture of the CostGlue software developing framework for simulation data exchange

## III. MULTILAYER NETWORK DESCRIPTION

The Multilayer Network Description (MND) aims to improve and organize the data exchange between various software components and tools [4]. In addition, it offers a human-oriented presentation of the data. An easy consultation is very important in the context of component integration, as it eases the verification of the data after each component.

Tools can use MND natively, i.e. import from or export to MND documents directly, or can use pre-processing and post-processing modules that act as converters of MND documents into a dedicated format and vice-versa (Fig. 2).

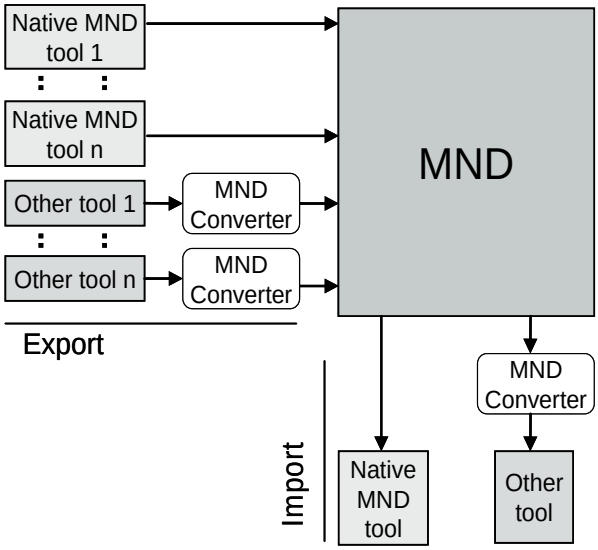


Fig. 2: MND acts as a link between different tools.

The structure of an MND document is organized around three objects: nodes, links and layers. An XML element represents each instance of these objects. Element inheritance and attribute possession are the two main mechanisms of XML. MND employs the inheritance to include an element into another one: a link element is included in a layer element where as a layer element is included into a network element. On the other hand, the attribute possession mechanism permits to associate data to elements. XML attributes are character strings. Outside of this restriction, attributes can be of all types, including references to other objects. For instance, each node element owns an "`id=<integer>`" attribute, while each link element owns two attributes "`orig=<id>`" and "`dest=<id>`", referencing origin and destination nodes by using their ID numbers. This principle permits the inclusion of incidence matrices in MND documents, as well as more complex data structures.

This organization, made of node and links, is well suited for the description of physical topologies, but the link element of the MND document can be also used to describe a logical connection, or simply a relation between two nodes. By grouping links and nodes in layers, many different type of information can be included the structure. Figure 3 shows a typical example of MND document.

Any MND document must conform to the structure depicted in Fig. 4. To permit reconstruction of the incidence matrix, it is required for the node element to have the "`id`" attribute defined. Besides other rules listed in [4], the MND structure is open and can be extended to

fulfil future requirements. For instance, more attributes can be added to the elements, or sub-elements can be added to node and link elements if the attributes are not sufficient (Fig. 3). In special cases, information can be also stored outside of the layer/node/link structure. This is however not recommended, as it diminishes the generality of the MND documents.

```
<?xml version="1.0" encoding="UTF-8"?>
<network>
  <main_description>
    <layer id="physical">
      <node id="0" pos_x="306" pos_y="466">
        <ports type="duplex" rate="10"/>
      </node>
      <node id="1" pos_x="413" pos_y="482"/>
      <node id="2" pos_x="307" pos_y="393"/>
      <node id="3" pos_x="376" pos_y="354"/>
      <node id="4" pos_x="440" pos_y="402"/>
      <link dest="0" orig="1" capacity="1"/>
      <link dest="0" orig="2" capacity="1"/>
      <link dest="2" orig="3" capacity="2"/>
      <link dest="3" orig="4" capacity="2"/>
    </layer>
    <layer id="connections">
      <link dest="3" orig="0" rate="3"/>
      <link dest="4" orig="1" rate="1"/>
    </layer>
  </main_description>
</network>
```

Fig. 3: A basic example of an MND structure

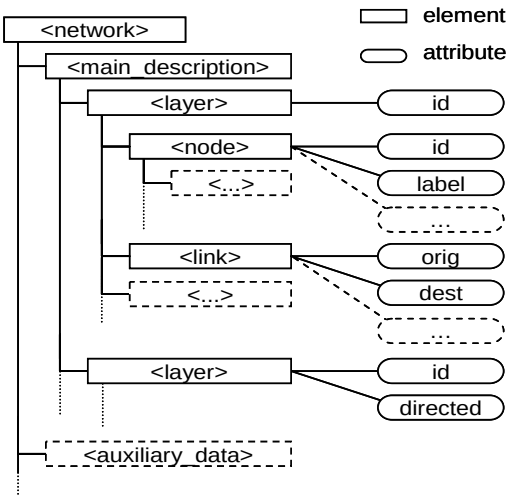


Fig. 4: Generic structure of a MND document.

MND acts as a bridge between components. It tries to balance three issues:

- keep enough flexibility to accept a large variety of input or output data and to guarantee compatibility among large number of applications,
- reach a high level of generality, in order to directly connect components without having to adjust the pre-processing and post-processing operations and,
- propose an acceptable level of complexity, permitting some database-inspired operations over the data (e.g., selection, extraction).

In Fig. 5, MND is compared with tab-separated text documents, proprietary documents, and hypothetical universal format. Each of these three presents two qualities and one drawback. MND has been conceived as an intermediate solution.

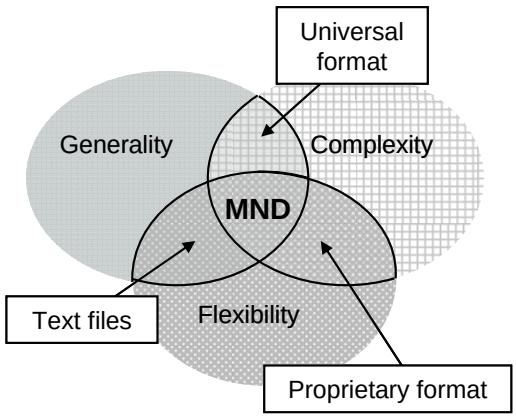


Fig 5: Comparison with other alternatives. Tab-separated files are two-dimensional and may lack of complexity. Proprietary format can have the desired flexibility and complexity, but is not general at all. A hypothetical universal format for networks would lack of flexibility.

## IV. TOOL INTEGRATION EXAMPLES

To illustrate the utility of the presented approaches, several scenarios are depicted in Fig. 6 and 7. Fig. 6 shows typical cases of "what-if" analysis: starting from an initial scenario, different simulations are driven, in order to measure the consequences of precise changes. For instance, in fig. 6(a), variant 1 can be the reference situation, while in variant 2, one link has been removed to the reference topology, in variant 3, one link has been added, and in variant 4, no link is removed nor added, but capacities are adapted.

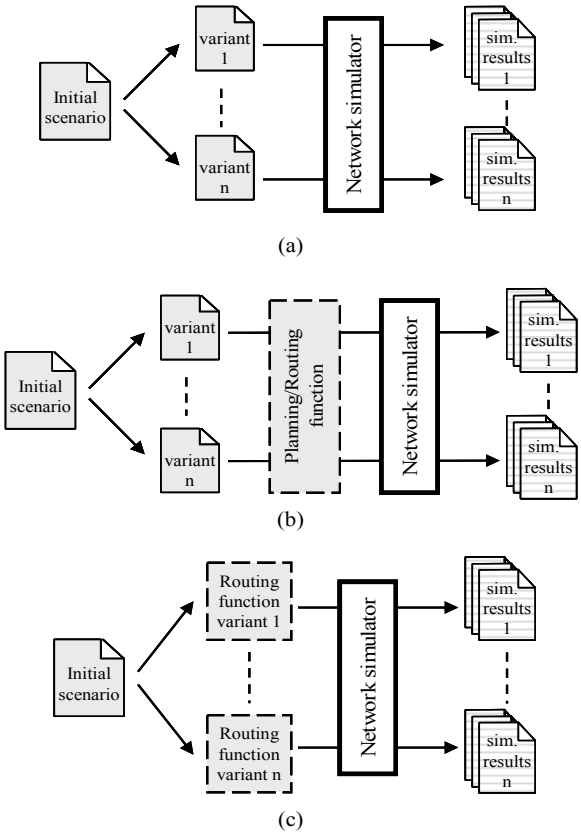


Fig. 6: In many situations, multiple simulations are driven. For each simulation, the resulting output must be archived and processed.

In Fig. 6(b), multiple simulations are driven to test the reaction of a network planning or routing function, confronted to different but similar situations. Again, starting from an initial scenario (network topology + traffic demands), multiple variants are derived, and then passed to the routing function. The resulting data (topology, traffic + routing information) are in turn transmitted to the simulator. By comparing the simulation results, the influence of changes can be measured.

In Fig. 6(c), the routing/planning procedure itself is submitted to a "what-if" analysis. In variant 1, for instance, routing is made using a shortest-path method, while in variant 2, routing can be made to balance the loads over different links.

In all cases depicted in Fig. 6, multiple simulations are driven, implying multiple simulation output dataset. By introducing these results into CostGlue, a batch processing of the *n* dataset corresponding to the *n* variants can be achieved. Additionally, the simulator input data can be stored along to its corresponding output, as metadata. This permits to recover easily the result of a simulation corresponding to a specific input. Finally, as the result of each driven simulation is kept, the use of CostGlue prevents the situations where long lasting simulations have to be executed twice or more times, because results have been overwritten, deleted, or even lost.

Fig. 7 depicts situations where simulation results are used as input for other tools, or reversely where the output of other tools in used as simulation input. The tool can be simply a visualisation tool, like in Fig. 7(a), as it is very intuitive to express graphically certain aspects of the results (for instance, displaying overloaded link in red). It can also be, as showed in Fig. 7(b), an iterative network dimensioning tool, which uses the simulation results to evaluate the intermediate solutions.

Fig. 7(c) represents a situation where a simulator is used together with two other independent tools. The first one generates successive samples of traffic matrices, following given statistical properties. These sample series simulate a traffic load varying over time. The second tool implements a traffic engineering (TE) aware routing algorithm. Using the results of the simulation as feedback, this algorithm reroutes portions of traffic to unload critical links and use network capacities better. In this case, the input of the simulation is constituted by: a) a network topology (links and link capacities, fixed over time), b) the traffic matrix (variable over time), c) the routing information outputted by the TE algorithm (variable). The routing algorithm takes the simulation results as input, as well as the previous routing information.

The MND format has been conceived to be used in situations illustrated on Fig. 6 and 7. On the first side, its human oriented presentation permits to compose rapidly different variants, like in Fig. 7(a) and (b). On the second side, it offers one unique structure to store various values and parameters. In the case 7(c), the MND file will typically include several layers containing heterogeneous information:

- network topology layer,
- traffic demands layer,
- routing layer, and
- simulation statistics layer.

Concerning the simulation statistics layer, an intermediate processing step should be performed to compute statistical values from the large amounts of simulation outputs. This intermediate tool can be CostGlue, as depicted on Fig. 8. CostGlue will perform the post-processing operations over simulation output, and compute, for instance, the average utilisation of each physical link, or the average packet loss ratio of each traffic demand. Using a specific MND plugin, it can later store the results inside the initial MND document. The same MND document can also be used as a descriptor for simulation results.

Up to now nothing has been mentioned about the simulator, which can be a dedicated one, using natively the MND format, or an existing one, like the NS-2 simulator [7]. As a proof of concept, a MND→NS-2 converter has been written and NS-2 simulation scripts have been successfully generated from MND documents. CostGlue framework includes a plugin permitting to analyse NS-2 output traces.

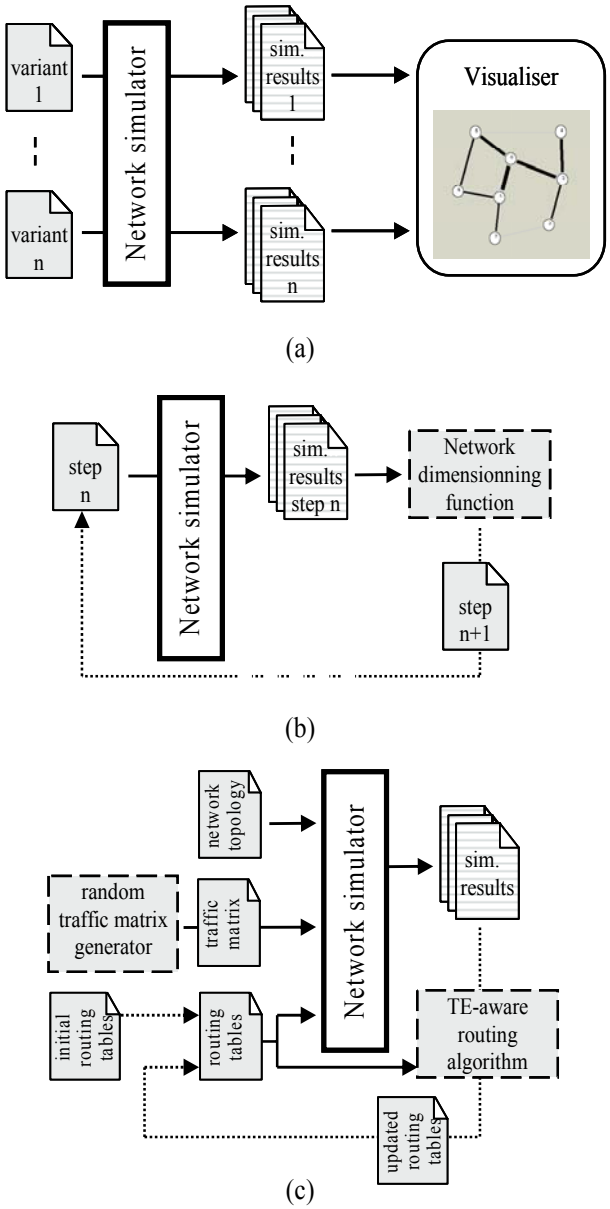


Fig. 7: Network simulators can be combined with various tools. It is therefore very important to specify a common format which permits to integrate tools easily.

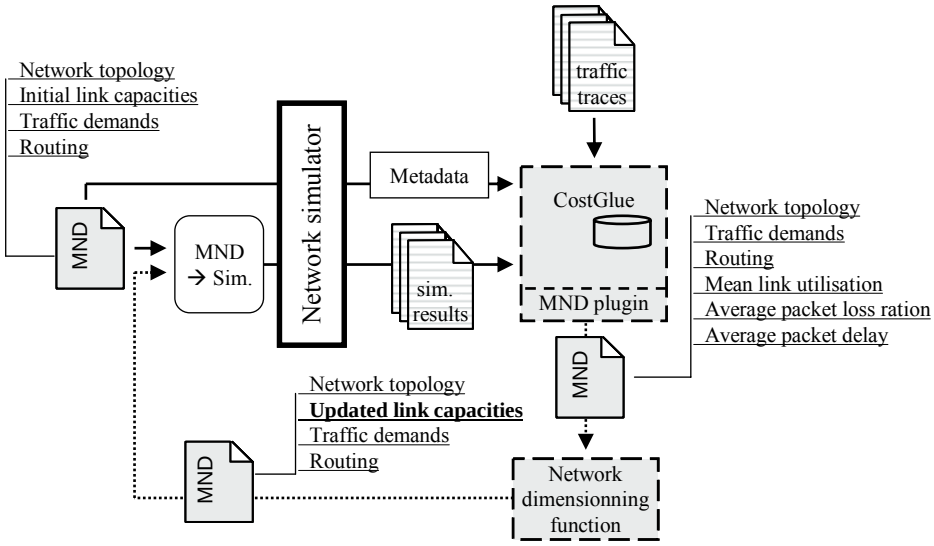


Fig. 8 : In this block scheme, an iterative network dimensioning tool is used together with a network simulator. The simulator tests the successive intermediate solutions generated by the dimensioning tool, until reaching an adequate level. CostGlue is used to store and process simulation results, while MND documents serve as intermediate between tools. Remark that 1) MND document is used as meta-data for simulation results, 2) traffic traces can also be processed by CostGlue, whose results can later be stored in the MND document, 3) as the results of each successive simulation is stored in the GostGlue archive, it is possible to retrace the evolution, and therefore to assess the quality of the network dimensioning function.

## V. CONCLUSION

In various situations, it is more valuable to combine existing tools rather than write new ones. However existing software components have been conceived in an independent way. Therefore they often provide very heterogeneous input and output formats. To address this problem, features offered by CostGlue and MND have been presented.

MND is a light-weight document format which rationalizes the data exchanges between components. It furthermore organizes the data in a way which eases the consultation, and permits a graphical visualization as depicted in Fig. 7.

CostGlue offers structured storage for raw simulation data, metadata and the post-processing data. As certain tools like network simulators output large amount of data, this structured storage is of great importance to realize the integration process. Additionally, CostGlue presents post-processing capabilities.

Practical scenarios, where multiple simulations are driven and/or where different components are evolved, have been presented to illustrate the validity of the proposed approaches.

## VI. ACKNOWLEDGMENTS

The research presented here has been undertaken by members of the European Cooperation in the field of Scientific and Technical Research (COST) Action 285 "Modeling and Simulation Tools for Research in Emerging Multi-service Telecommunications" and Action 291 "Toward Digital Optical Networks". It has been supported by the Swiss Secretariat for Education and Research

## REFERENCES

[1] A.W. Bragg, "Which network design tool is right for you?" *IT Professional*, vol.2, no.5, pp.23-32, Sep/Oct 2000

[2] M. Lackovic, C. Bungarzeanu, "A Component Approach to Optical Transmission Network Design", *Modeling and Simulation Tools for Emerging Telecommunication Networks,* pp. 335-355, Springer, 2006.

[3] CostGlue project web site: http://lt.fe.uni-lj.si/costglue/

[4] S. Rumley, C. Gaumier, "Multilayer Description of Large Scale Communication Networks", *Recent Advances in Modeling and Simulation Tools for Communication Networks and Services,* pp. 121-135, Springer, 2008.

[5] CCSDC 650.0-r-2: Reference Model for an Open Archival Information System, Blue Book, Issue 1, ISO 14721, 2003

[6] S. Sufi, B. Mathews, CCLRC scientific metadata model: Version 2, CCLRC technical report DL-TR-2004-001, September 2004

[7] Network simulator NS-2 web site: http://www.isi.edu/nsnam/ns/

[8] D. Savić, M. Pustisek, F. Potorti. A tool for packaging and exchanging simulation results. V: First International Conference on Performance Evaluation Methodologies and Tools Valuetools, Pisa, Italy, October 11-13, 2006.