5th INTERNATIONAL CONFERENCE
TRANSPORT SYSTEMS TELEMATICS TST'05

ZESZYTY NAUKOWE
POLITECHNIKI ŚLĄSKIEJ 2005
TRANSPORT z.59, nr kol. 1691

Logic Linker,
diagnostic scripts,
XML files

Markus BREGULLA[1]
Tomasz JANIK[2]
Leszek CZAPKOWSKI[3]

AUTOMATIC GENERATION OF DIAGNOSTIC SCRIPTS FOR THE DISTRIBUTED SYSTEMS BASED ON COMPONENT TEMPLATES AND RELATIONS BETWEEN THEM

During introduction and implementation new technologies into the modern industry including the vehicle engineering, the previous and already verified solutions and components are very often to be used. That may be difficult though especially for the outer designers, who might not know these structures. Thus, the modification and adaptation of the previous projects into the new requirements and needs is annoying and may cause lots of problems. What is more, further analysis and diagnostic of such systems may also cause some problems. The article deals with pointing the method and showing a tool to solve such problems. Therefore, the authors will introduce a program called Logic Linker. The application automatically generates the diagnostic scripts for the distributed systems basing on the component templates and relations between these components.

AUTOMATYCZNE GENEROWANIE SKRYPTÓW DIAGNOSTYCZNYCH DLA SYSTEMÓW ROZPROSZONYCH W OPARCIU O WZORCE KOMPONENTÓW I RELACJE POMIĘDZY NIMI

Podczas wdrażania nowych technologii w dzisiejszym przemyśle, w tym również przemyśle samochodowym, bardzo często stosuje się wcześniejsze i sprawdzone rozwiązania i komponenty. Jest to czasami kłopotliwe, szczególnie dla projektantów z zewnątrz, którzy nie zawsze znają poprzednie struktury. Modyfikacja wcześniejszych projektów i ich przystosowanie do nowych potrzeb może stanowić pewne problemy. Problemem może być również późniejsza analiza i diagnostyka tych systemów. Tematem artykułu jest zatem wskazanie sposobu i pokazanie narzędzia, które te problemy rozwiązuje – Logic Linker. Aplikacja ta automatycznie generuje skrypty diagnostyczne dla systemów rozproszonych w oparciu o wzorce komponentów oraz relacje pomiędzy tymi komponentami.

[1] University of Applied Sciences, Esplanade 10, 85049 Ingolstadt, Germany, markus.bregulla@fh-ingolstadt.de, tel.:+49 841 9348 389
[2] University of Applied Sciences, Esplanade 10, 85049 Ingolstadt, Germany, tomasz.janik.wi@fh-ingolstadt.de, tel.:+49 841 9348 594
[3] University of Applied Sciences, Esplanade 10, 85049 Ingolstadt, Germany, leszek.czapkowski.wi@fh-ingolstadt.de, tel.:+49 841 9348 594

1. INTRODUCTION

New technologies for the modern transport industry, vehicle engineering, as well as industry in general, are very often applied and implemented basing on the previous and already verified components and solutions. That may cause of course some complications especially for the engineers who might not know those structures at all, or would not get with them well. Therefore, modification and adaptation of the previous projects into the new needs and requirements is demanding and may cause lots of problems. Moreover the further analysis and diagnostic of such systems may also be problematic.

The article deals with pointing the method and showing a tool to solve such problems. Therefore, a program called Logic Linker will be brought in, and a general methodology of implementation will be introduced.

Logic Linker makes an application for the automatic generation of the diagnostic scripts for the distributed systems, also for the transport needs. This is made basing on the templates of components and relations between these components. The application, in general, fulfills the gap in the engineering process between the planning layout and diagnostic server.

The figure below (Fig.1.) shows the general idea of how the method works.

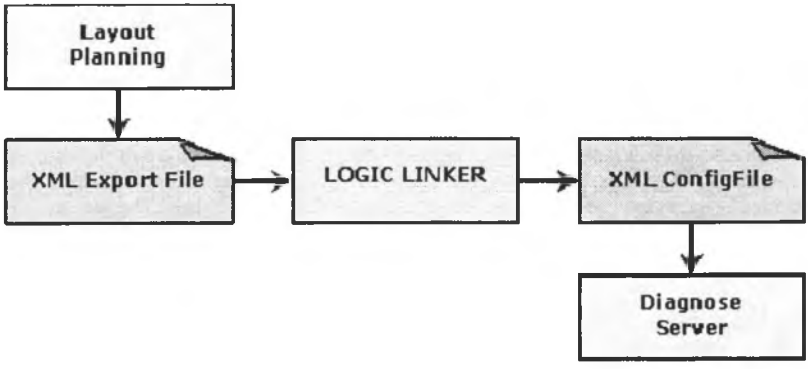

Fig.1. General idea of the method

2. SYSTEM ARCHITECTURE

The system architecture contains of the application itself, two income files and one outcome file. All files need to be of the XML type hence, all components and rules describing dependencies between these components must be described as XML-files.

One of the income files is the Plant Layout Plan, which comes from the Layout Planning Tool. It encloses a structural description of all components belonging to the system as well as their performance and mutual layout in the structure. The second income file is the Library of Rules. This library holds descriptions of cooperation and dependencies between two or more elements.

The Diagnostic Script File makes the outcome file for the system. It is generated by Logic Linker according to the input files. Logic Linker reads all needed and relevant data from the Export File, and links them with the diagnostic rules corresponded to a general

concept. The Diagnostic Script File may be read as a script realization directly in script processor of Diagnostic Server. Therefore, data are sent to the Diagnostic Server for the system diagnostic purposes.

The general idee of how the system works is shown the scheme below (Fig.2.).

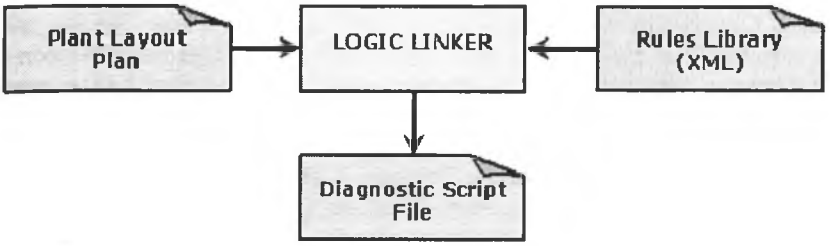

Fig.2. General schema of how system works

2.1. COMPONENTS AND RULES

A component is an object that performs a specific function and is designed to easily operate with other components. They may be single or included in a container of other components. Components can be used for assembly of the system, or make a subsystem. Each of them must be assigned to the Component Class that characterizes each element belonging to it. The Component Class makes therefore a class of all components that may lay in the system, and is used for instancing the components of this system. All Component Classes are included in a special library (Component Class Library). This library contains all Classes of Components that perform specific functions and it is designed to easily operate with other elements.

A rule describes dependencies between the Classes of Components in order to generate errors by failed states. When a Rule is related to the one Component Class only, then it is included in this Class. This is a case of Single Level Logic Rule (SLL Rule). And when a Rule affects two ore more Component Classes, then it calls these Classes from the Component Class Library. This is a case of Multi Level Logic Rule (MLL Rule). Therefore, all the Rules related to more than one Component Class are included in the Library of Rules.

The Library of Rules makes a collection of all classes of Rules set for simplicity of use that Logic Linker can apply. And the Library is applied to store frequently used Rules. The linker automatically looks up in the Library to find an adequate and sufficient Rule. The Library by its nature is Multi Level Logic (MLL) only, and that means that it contains Rules describing cooperation and dependencies between two or more Component Classes.

Each Component Class Library as well as Rules Library is given as a strictly settled structure. The architecture of these structures needs to be kept as it is, in order to the proper performance of the Logic Linker and Layout Planning Tool.

3. LOGIC LINKER

For the automatic implementation of the system diagnosis is used a special application, called Logic Linker. The application reads all needed and relevant data for the system diagnosis from the Plant Layout Plan, and links them with the diagnostic rules corresponded to a general concept. At the output of the system Logic Linker generates the Diagnostic Script File that might be read as a scripts realization directly in the script processor of the Diagnostic Server.

Logic Linker contains of two inner elements: Linker and Interpreter. Linker works as the Single Level Logic, while Interpreter works as the Multi Level Logic. Data that are read from the Plant Layout Plan are next used to create the outcome XML file. For that reason, data from the Plant Layout Plan are read out of Linker and Interpreter. Depending whether it is a component or a relation between the components it will be adequately sent by Logic Linker to the Linker (SLL) or to the Interpreter (MLL). In case when data are sent to the Interpreter, the relation between the components is searched in the Library of Components and if found, then is instanced. It will receive a real name (ID) of the existing components in the real system.

The outcome data are generated then based on the rules from the Library of Rules and components from the Linker or Interpreter. They might be created either at the SLL level or MLL level, and are next aggregated as the outcome XML file. Therefore, the file that is transferred to the Logic Linker is there merged properly with the Library of Rules. And as a result, the Diagnostic Script File is generated. This is the outcome file that contains diagnostic scripts, and is transferred to the Diagnostic Server.

A scheme of the process for the application is shown on Fig.3.

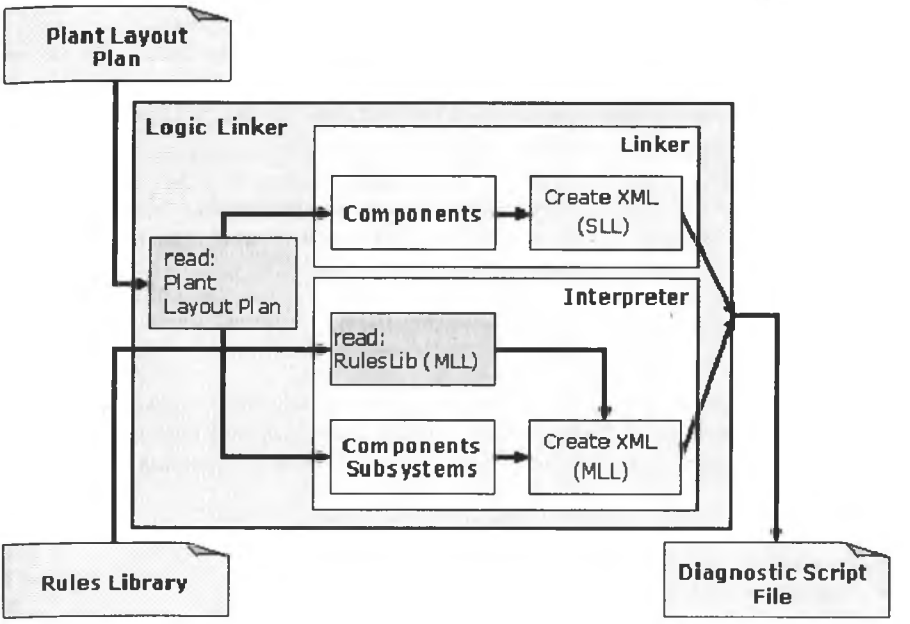

Fig.3. Logic Linker - scheme of the process

3.1. REQUIREMENTS

The goal of the Logic Linker is in general generating the outcome file (Diagnostic Script File) from the income files (Rules Library and Plant Layout Plan). To perform such a procedure, the application must fulfil and fulfils the following requirements:

- **Technology independence**
  When reading the income file (Plant Layout Plan) Logic Linker must recognize and identify the layout of the system. This is done for the automation components applied to the forthcoming technologies. Proper data need to be passed on to the Diagnose Server throughout the Diagnostic Script File. These data are necessary to accomplish and realize the process of diagnosis. A technology of such realization is not significant for the Logic Linker, and that is because Logic Linker must only pass data on in a proper and specialized form.
- **Flexibility**
  Logic Linker should have a possibility of working and cooperating with any XML structure that describes the rules and the system layout. It must take into account new solutions, unknown issues, and ones not yet considered.
- **Possibility of matching (adaptation)**
  During the phase of qualifying and succeeding rules to the components or components to the rules, the possibility of analyzing and processing the Component Classes is required. The process is being repeated as long as specific and defined conditions of matching (adaptation) are fulfilled. Hence, matching the diagnostic rules with the components related to them is possible and reasonable. As long as the specifications are fulfilled, Logic Linker must be able to work easily in a loop and try to match components to the rules as well as rules to the components.

- **Alternativeness**
  The type, size, and realization of the technical device of automation are variant and unconventional sometimes, and that may have an impact on deepness of the hierarchy of subsystems. Logic Linker must be then flexible and adaptable that it will manage any potential kind of the device.
- **Expansibility**
  Logic Linker (Diagnostic Script Generator) should be possibly adaptable to the new conceptions of the diagnostic systems as wall as developable ones.

4. SAMPLE OF USE

On the figure below (Fig.4) is shown a simple of the schematic conveyor belt line. The line makes an industry transportation system that is being tested by Logic Linker. Therefore, the line as a real system is built of the following components, belonging to the particular Component Classes:

- Conveyor belts: Con, Con1, Con2, Con3, Con4, Con5;
- Sensors: Sen01 … Sen52;
- Separators: SP1, SP2, SP3, SP4, SP5, SP6;
- Track switches: S1, S2, S3, S4, S5;
- Trace sensor: TrSen.

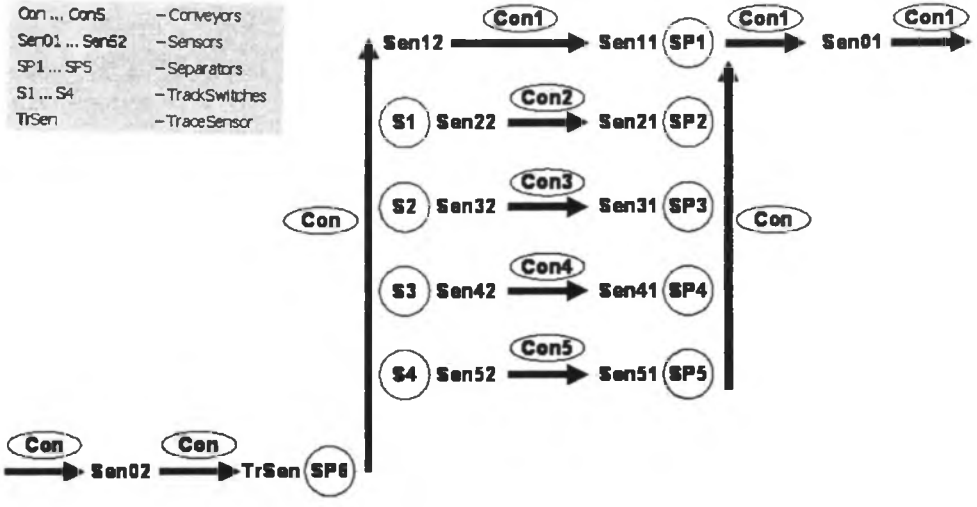

Fig.4. Schematic conveyor belt line [3]

For a given layout it is possible to generate many different Rules as well as descriptions of the Component Classes. The figure below (Fig.5) shows a simple of the MLL Rule that describes cooperation between two conveyor belts. The given Rule verifies if Successor is

ready to receive elements conveyed by Predecessor. If not, then returns a message about the failure.

```
- <Rule Template="No" ID="5">
    <Target ComponentClass="Ref-Conveyor" Name="Predecessor" />
    <Target ComponentClass="Ref-Conveyor" Name="Successor" />
  - <Condition Param="Predecessor.ON.OUT.Var_On" Op="=" Value="1"
      ValueIsRef="0">
    - <True>
      - <Condition
          Param="Predecessor.Abgabebereit.OUT.Var_Abgabebereit"
          Op="=" Value="1" ValueIsRef="0">
        - <True>
          - <Condition
              Param="Successor.Aufnamebereit.OUT.Var_Aufnamebereit"
              Op="=" Value="0" ValueIsRef="0">
            - <True>
              - <Return>
                - <Var Name="ERROR" Value="">
                    <Var Name="ID" Value="" />
                    <Var Name="DESC"
                      Value="Successor not ready" />
                  </Var>
```

Fig.5. Sample of the MLL Rule

5. CONCLUSIONS

The system of the automatically generated diagnostic scripts that has been introduced in this paper is a powerful and universal application. It may be applied to many different fields of industry where it is possible to see and consider the industry systems by the components of such systems, and the rules describing cooperation as well as dependencies between these components. According to this, it needs to be mentioned that transport in general and transportation systems make such a field of industry.

Logic Linker has been still developed and proved for the transportation systems at the in-house industry transport (production lines at Audi).

BIBLIOGRAPHY

[1] BREGULLA M., GROßMANN D., Automatische Generierung systemweiter Diagnosenfunktionen auf der Basis von Layout-Informationen, ATP 47 (2005), Germany

[2] BREGULLA M., GROßMANN D., Open interface for distributed access to diagnostic information, 4th International Conference "Transport Systems Telematics", 4-6.11.2004, Katowice, Poland

[3] DÖBELE M., Konzept zur automatischen Generierung von Diagnosefunktionalität auf Basis von Layoutinformationen, Diplomawork at ITM, Technical University of Munich, 2004

Reviewer: Ph. D. Jerzy Mikulski