Automated workflow for generation of CANopen system monitoring GUI

Heikki Saha, TK Engineering Oy

System monitoring features are needed throughout the systems life cycle, but development and maintenance of such are experienced as lower priority and less important than primary controls. This paper presents an automated workflow for generation of such GUI, based on CANopen projects and targeted for typical embedded displays.

Main challenge is that CANopen projects are not capable of define logical device locations relative to each other. The challenge has been solved by creating a connection to electric schematics in order to determine logical device locations and device interconnections. Further challenge is how to assign device screen coordinates. It is impossible to automate assignment in general and computer aided manual assignment has been developed instead. All information has been merged into a GraphML project file, from which the target specific GUI configuration may be generated.

Main focus has been in a workflow enabling efficient work. Computer aids has been selected to phases, where full automation does not make sense. Schematic parsing has been isolated as a component, enabling flexible adaptation to various schematic formats. The presented workflow intrinsically supports iterative development and efficient information re-use. It is also compatible with the most common CANopen and application development tools in the market.

Introduction

Diagnostics is one of the most important but also most often too far postponed entity in control system design. It is an entity serving the entire development starting from the first unit tests, continuing to the final integration tests and finally in the system operation. During systems life cycle, diagnostics shall be maintained along with the evolving control features and system structure.

As already published in the literature, CANopen diagnostics is a comprehensive set of features, mainly powered by the communication protocols together with the standardized system design process, boot-up procedure and synthesis of resulting information [1] [2] [3] [4]. Certain set of services support also efficient and reliable preparing for [5] and performing [6] series production. The design process may be expanded by data imports from other disciplines [7] or system level simulation model [8], improved adjustment of device parameters [9] and support for optional structures [10]. It has also been proved that information may be re-used in a managed way between the design phases [11], which is often required in practical system development projects. It can be summarized that each design activity inside the CANopen system design may be automated or improved by corresponding design tools.

While application code may be generated from system model [8] only, generation of system structure description requires at least electric schematics containing logical installation order of devices and CANopen project listing the system member devices. CANopen system description [12] has been supplemented by introducing GraphML node list format capable of representing screen coordinates and logical device locations relative to each other [13].

Main challenges in the design and maintenance of a diagnostics entity are, that information content of CANopen system project is not sufficient and there does not exist an interface for adjusting the generated screen coordinates. Logical device locations shall be read from electric schematics, where the physical connections between the devices are described. As a consequence, information of two disciplines shall be combined in a managed way. It is worth of remembering, that any change in either CANopen system design or schematics shall be followed by refresh of the resulting GraphML description.

The automated workflow regarding the diagnostics has already been proved to work in a very basic form. Generation of GraphML format system description from CANopen project and schematics has been implemented by using legacy ProCANopen project as a backup source. Furthermore, diagnostics view components with generation of parameter data set for them from GraphML have been implemented. Main missing things are in the workflow – screen coordinate assignment at the top of system layout is missing as well as automated re-use of the screen coordinates between the design cycles.

There exist already reference designation mechanisms in the industry, which are widely used to provide assembly location specific addressing scheme [14] [15]. Such addressing schemes are typically used to link different disciplines together, commonly hydraulics and electrics. In order to support such linking from CANopen, reference designator entries had to be added into device descriptions [16]. The details, how to combine CANopen and schematics are outside the scope of this paper.

Main scope of this paper is a workflow and interface enabling the screed coordinate adjustment after GraphML generation. Cross-platform, open-source and consortium based tools, Python, Gephi and Inkscape was used to get open and generic implementation. The structure of this paper follows the design process phases within the focus of the paper by starting from collecting nodes information from CANopen project and electric schematics. The detailed description of the synthesis is not within the scope of this paper. Second, a workflow and example tools for screen coordinate editing are explained by keeping typical working scenarios in mind. Third, workflow for exchanging information with IEC 61131-3 development environments is explained. The last topics are discussion around the proposed approach and concluding remarks.

GraphML generation

GraphML node list has intentionally been designed to be an open, generic format for interchanging more comprehensive structural information of the entire system than before [13]. It naturally contains same core information than nodelist.cpj [12] and thus may be used as an improved CANopen project file.

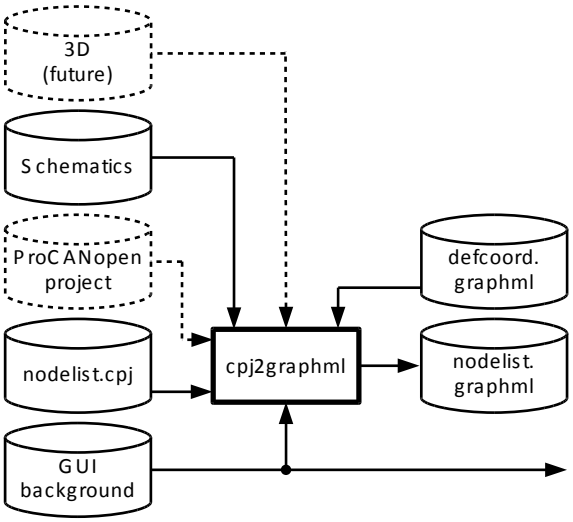

Figure 1: Overview of GraphML node list generation

Main content is generated from CANopen, electric schematics and default layout based on a background graphics size. ProCANopen project was tested as a fallback method to determine installation order, but such requires special guidelines and tool usage and is thus not as consistent source for device locations as electric schematics.

Figure 1 illustrates the generation of a GraphML node list. Member devices of a system are available in the standard nodelist.cpj file of a CANopen project. Installation order of the devices are read from the electric schematics. Devices in both CANopen projects and schematics are

identified by reference designators, which enables the information synthesis.
Screen background image dimensions are used to set screen size and the default device layout on the screen so, that devices are evenly located by default in order to make the locations easy to edit. If a file defcoords.graphml exists, default coordinates are reused from it for devices included by it. When devices are added, only locations of the added devices need to be set manually. There is a reservation for the future to generate the screen background and bring the default device coordinates from e.g. 3D-model as default coordinates.

Edit GraphML screen coordinates

It shall always be possible to manually edit the screen coordinates. As long as the background graphics is manually designed, default coordinates are not available. Even if there were automated generation of background and export of defaults coordinates, manual adjustment may be required. Whenever an automated generation is used in conjunction with manual editing, it shall be possible to maintain results of manual editing by default in order to avoid design overhead and confusion caused by lost changes.

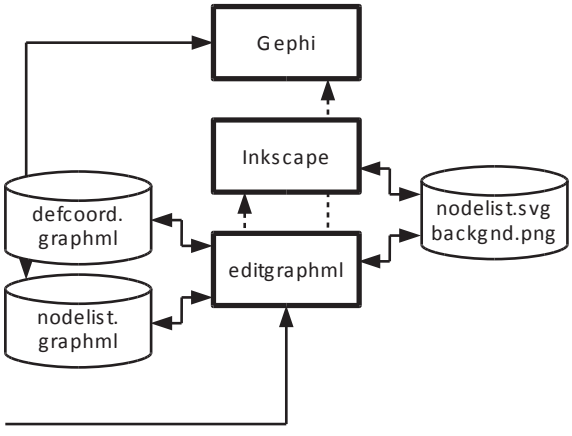

Figure 2: Overview of screen coordinate management

Screen coordinate management principle is show in Figure 2. Gephi is the most obvious tool for editing GraphML files. Its main disadvantage is still a missing support for background image. The feature has been tested in a special fork, but not included in a main branch. The example system layout in Gephi is shown in Figure 3.
Main advantage of GraphML is, that it is based on XML. Thus, implementing a simple wrapper tool was far simpler than writing an entire dedicated editor. Main idea is to convert the GraphML file into an SVG file before editing and the modified SVG file back to GraphML after editing. As an integral part of such conversion, background image may also be converted from any format to PNG to enable problem free usage of it. Furthermore, coordinate system is slightly different. X-axis is equal, but positive direction of Y-axis in GraphML points up and in SVG down. Image height is required for coordinate transformation.

Figure 3: Example system layout in Gephi (as GraphML)

Inkscape was selected as an editor, because it is open-source, operation system independent and free tool for editing SVG images. In addition, it supports the use of “rubber-band” connections between graphical blocks.
Overview of a GraphML representation is show in Figure 4. In addition, a separate backround image file is required, which is referenced by a corresponding graph attribute. Such description conforms to GraphML specification and is compatible with corresponding libraries, such as NetworkX [17]. Nodes are represented by node and interconnections as edge elements in GraphML. Required supplemental information may be assigned as attribute entries for graph, node and edge elements.
Structure of SVG representation is shown in Figure 5. The first advantage of SVG over GraphML is, that background image reference is both machine understandable and directly visible, because SVG enables including raster images as integral part of vector images. However, such raster image is still an independent file.

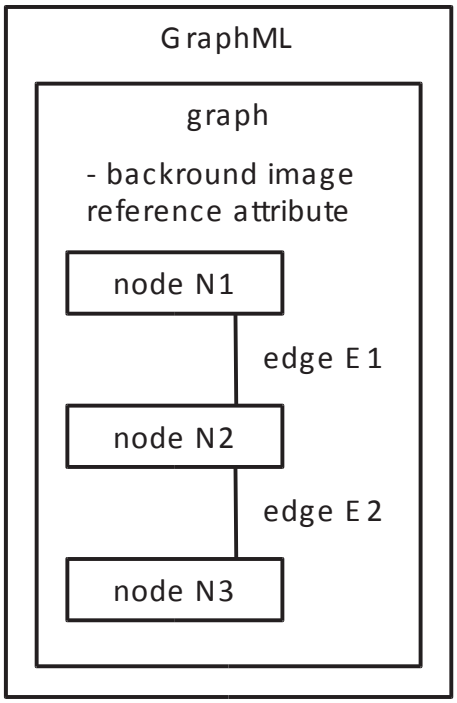

Figure 4: Structural overview of GraphML representation

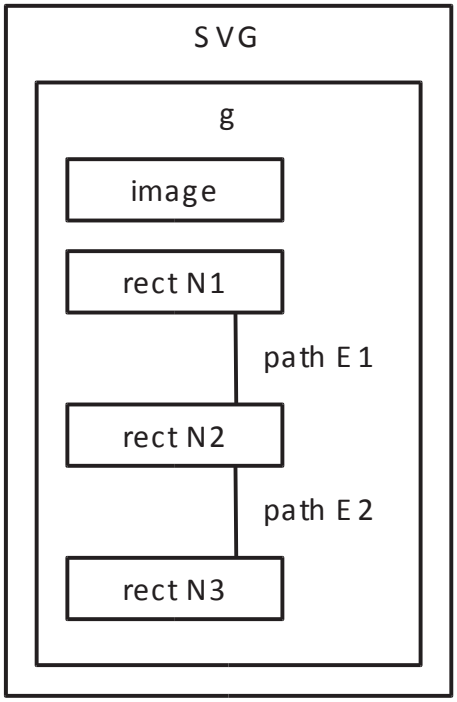

Figure 5: Structural overview of SVG representation

Nodes are represented by rectangle and interconnections as path elements. Required supplemental information may be assigned as attributes of the SVG rect and path elements. The entire layout shall be enclosed in a group, SVG g element.

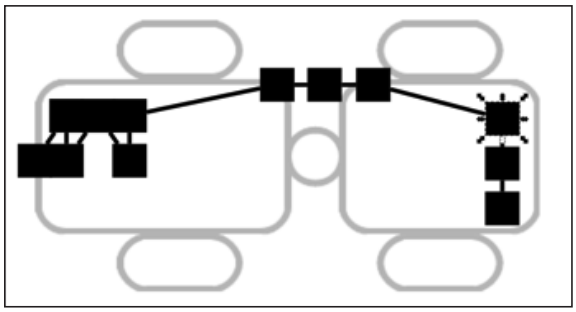

Figure 6: Example syste layout in Inkscape (as SVG)

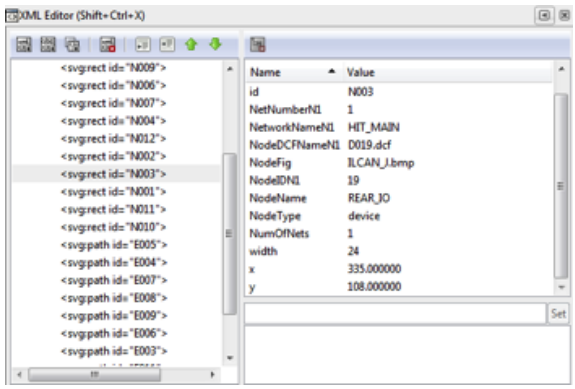

Figure 7: Details of the selected device rectangle element in XML Editor window of Inkscape

Inkscape contains so called “Connector tool” [18] enabling rubber band connections between graphical elements, e.g. rectangles. A special attribute inkscape:connector-type=“polyline“ of a path together with initial path end points located to the center point of source and destination node rectangles makes Inkscape to adapt element connections according to the element location changes.

An example of a system layout edit view in Inkscape is shown in Figure 6, where device REAR_IO has been selected. Advantage is, that with the explained options the device rectangles may be moved around screen layout so, that the interconnections scale and move accordingly.

Minor disadvantage of the SVG representation is, that device names should be included as separate text elements, which cannot be locked to the corresponding rectangles. Fortunately Inkscape contains an SVG Editor window, which shows the actively selected SVG element with it’s attributes, including device name as shown in Figure 7. The most interesting information within the context of device location

adjustment is the device name. Despite of the ability to change all information, all other attributes than screen location shall be adjusted only in the CANopen system tool to keep the project consistent.
As a final step, the edit manager asks, if the most recent changes shall be saved also to the default coordinates, where they automatically apply to the next export from the CANopen project and schematics.
Without such behavior, the changes do not systematically apply to the structure description updates and locations of all devices need to be manually assigned again.
Such workflow issue leads into the mandatory use of an edit manager tool anyway, because without such a tool a save to the default coordinates file would be up to the user and thus prone to the human mistakes.

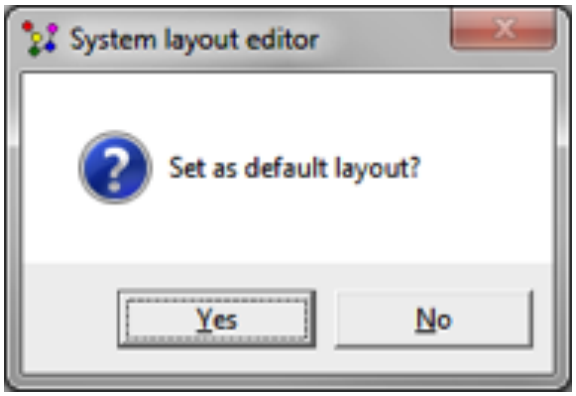

Figure 8: Optional workflow enhancement enables saving new layout as default

The manager tool has been implemented so, that it may be integrated as a part of make procedure of a CANopen system tool. However, it can also be executed at any time for a given CANopen network project outside the scope of any make procedure.

IEC 61131-3 integration

After adjustment of the screen coordinates it makes sense to export the system monitoring information to e.g. IEC61131-3 development environment to deploy a system monitoring view [1] [13]. A reuse mechanism for coordinate quick fixes in an IEC61131-3 project is practically required to avoid to set too tight process constraints, which are commonly leading into ghost processes messing up the design work. Almost always GUI needs to be fine tuned during application development. As a generic structure description file, nodelist.graphml provides open and easy integration interface, which is illustrated in Figure 9.

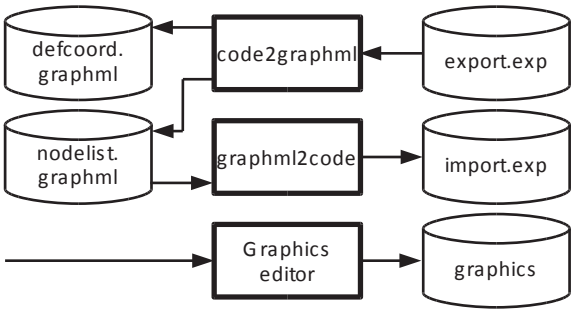

Figure 9: IEC 61131-3 integration powered by GraphML description file

GUI development environments may require graphic elements in certain format(s) and color depths. Thus, a dedicated task may be needed to adjust the screen background figure to the target specific format. Adjusting the background graphics may be integrated to the integral part of the system monitoring parameters generation. The details depend on the target constraints.

IEC 61131 software development

System monitoring parameter structures may be imported to and exported from IEC development environment. It depends on the product, whether graphics is developed within IEC environment only or compiled with dedicated tool and elements referenced from IEC application. The result may be one or two binary files, depending on the target HMI.
As explained in existing literature [1], the basic functional elements of a system monitoring view may be standardized as a library functions. Imported parameters just configure the operation of such library functions.
While SW integration has been implemented for IEC61131-3 first, support for other frameworks may be easily added later on. The use of nodelist.graphml as an interface provides full freedom for future developments.

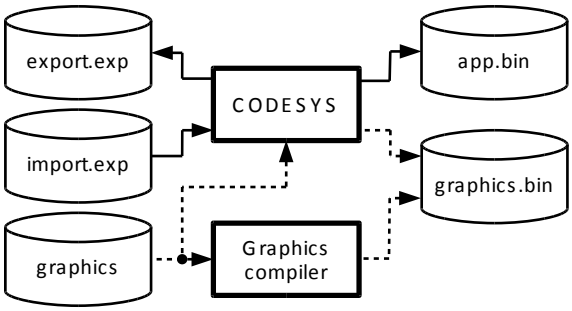

Figure 10: Overview of the IEC 61131-3 application development

Figure 10 provides an overview, how SW integration may work in CODESYS based platforms. The use of graphics elements depend on the selected architecture.

Discussion

GraphML is an excellent format capable of describing the entire network structures, including physical appearance. Gephi is the most natural open and free editor, but due to its constraints regarding background image, an alternative approach by using transformation to SVG during edit and Inkscape as an editor has been developed.
A systematic workflow is essential and definition of such enabled systematic integration of various state-of-the-art tools. Special emphasis has been put on information re-use, because many processes are defined too tightly and do not support practical ways of working, mostly by means of back-annotation of changes and re-use of information.
Focus has been in utilization of standardized interface concepts enabling utilization of the best tools in the market for each purpose. Standardized interfaces also enable updating the selection of the tools, when required. In the cases where standardized interfaces were not available, de-facto interfaces were used instead.
Advantage of the presented approach is, that GraphML is the standardized interface and enables practically limitless tool adaptations withing its scope of applicability. As in the presented approach, the tool specific adaptations may be realized in a generic process tools hiding the complexity from the users. It is up to the process implementation, which information connections will be activated.

After getting the screen coordinate management implemented, there exists a complete computer aided workflow and solid example tool chain from model based design to the system assembly and field service for both system configuration and application behavior, including system level exception management. Supporting tools enable efficient entering of new information, when automatic generation do not apply.
The use of open formats and third party SW tools resulted several advantages. The use of Python for own developments enable running of the tools on any operating system, independent of any single company. The selection of consortium-based tools Inkscape and Gephi ensures, that large consortia develop further important functions, which are not within the main scope of the design process development but which are providing great improvement by means of functionality and performance. It is also important, that free-of-charge tools may be used by anybody, without any commercial constraints.
Presented workflow and edit concept is not restricted to CANopen, it may be used for other bus systems, too. Some standardized bus systems do not cover configuration descriptions, which reduces the level of generality.
Some future research topics were recognized. First interesting topic will be, how node locations and background image could be automatically generated from e.g. system's 3D-model. Second interesting topic will be, how PLCopen XML based interfacing of IEC 61131-3 development environments could be utilized instead of the legacy import and export interfaces.

Conclusions

Final major missing block from the entire generation process of a system monitoring view has been deployed. Thanks to the use of free-of-charge tools, commercial bottlenecks does not exist. OS independence of the related tools improve the usability and maintainability further.
It has been proved, that CANopen enables excellent support for managing the system level structural and configuration information throughout the design process, starting

from model-based design and ending to the assembly line and field service. Systematic configuration management improves both productivity and quality of the development by speeding up the design itself and drastically reducing number of design failures.

Following open interface standards enable both the use of generic processes and freedom to select optimal tools for each purpose, scale of operation and organization without breaking the process. Such freedom also helps in maintenance and further development of the process.

However, openness of CANopen is not enough. If the efficiency needs to be improved further, openness and open, standardized interfaces shall be improved also in the tools of other disciplines, CAD software and IEC 61131-3 SW development environments.

Dr. Heikki Saha
TK Engineering Oy
Hovioikeudenpuistikko 13 as 3
FL-65100 Vaasa
www.tke.fi

References

[1] Saha H., Exception management in CANopen systems, CAN-Newsletter 2/3013, CiA, 2013, pp. 12-17
[2] Saha H., Experimental CANopen emergency error code (EEC) management, CAN-Newsletter 1/2013, CiA, 2013, pp. 12-18
[3] Saha H., Improving development efficiency and quality of distributed IEC 61131-3 applications with CANopen system design, Proceedings of the 13th iCC, CiA, 2012
[4] Saha H., SI unit and scaling management in CANopen, CAN-Newsletter 3/2013, CiA, 2013, pp. 30-34
[5] Saha H., Accelerated transfer of CANopen projects into assembly and service, CAN Newsletter 4/2012, CiA, 2012, pp. 16-20
[6] Saha H., CANopen in series production, CAN Newsletter 3/2015, CiA, 2015, pp. 8-11
[7] Saha H., SI unit and scaling management in CANopen, CAN-Newsletter 3/2013, CiA, 2013, pp. 30-34
[8] Saha H., Model-based design of distributed mechatronic systems, CAN Newsletter 4/2014, CiA, 2014, pp. 38-45
[9] Saha H., CANopen device configuration editors, CAN-Newsletter 4/2013, CiA, 2013, pp. 30-33
[10] Saha H., Optional structures in CANopen projects, CAN-Newsletter 1/2014, CiA, 2014, pp. 32-35
[11] Saha H., Systematic re-use of information, CAN-Newsletter 2/2014, CiA, 2014, pp. 20-25
[12] Electronic Device Description, Part 3: Network variable handling and tool integration, CiA-306-3
[13] Helminen M., Salonen J., Saha H., Nykänen O., Koskinen K.T., Ranta P., Pohjolainen S., A new method and format for describing CANopen system topologies, Proceedings of the 13th iCC, CiA, 2012
[14] Industrial systems, installations and equipment and industrial products – Structuring principles and reference designations – Part 1: Basic rules, IEC 81346-1, IEC, 85p.
[15] Industrial systems, installations and equipment and industrial products – Structuring principles and reference designations – Part 2: Classification of objects and codes for classes, IEC 81346-2, IEC, 45 p.
[16] Electronic Device Description, Part 1: Electronic Data Sheet and Device Configuration File, CiA-306-1, Version 1.3.12, CiA, 2018
[17] Hagberg A., Schult D., Swart P., NetworkX Reference, Release 1.8.1, 2013, 470 p.
[18] Wybrow M., Connector Tool Tutorial, http://wiki.inkscape.org/wiki/index.php/Connector_tool_tutorial (30.10.2019)