Complexity metrics for Petri net based logic control algorithms

Georg Frey, Lothar Litz, and Frank Klöckner

Abstract—In the area of automatic control more and more tasks formerly solved by special hardware are performed by software. Hence to talk about quality in automation, there is a need to measure the quality of a software product. Software Quality is a field of mayor interest for researchers and practitioners today. However in the area of Control Engineering it is rarely studied.

In previous publications the authors introduced the concept of transparency and defined corresponding metrics to measure the quality of logic controllers. In addition to these metrics, in this contribution known complexity metrics from Computer Science are adapted to the area of logic control design and it is shown how the concepts of complexity and transparency are related. The metrics are introduced using the Signal Interpreted Petri Net, but they are also valid for other Petri net types.

Index terms—Petri net, PLC, logic control, complexity, transparency, quality metrics.

I. INTRODUCTION

In general, the realization of a logic controller includes hard- and software. With the assumption of standard hardware with well-defined functionality, the realization is the program of the control algorithm—i.e. software. Hence the quality of the controller mainly depends on the software quality. ISO/IEC 9126 [1] standard defines software quality characteristics as:

A set of attributes of a software product by which its quality is described and evaluated. A SW quality characteristic may be refined into multiple levels of sub-characteristics.

The field of metrics to measure these software characteristics is maturing, see e.g. [2] or [3] for an overview. Some of the best known metrics date back to the late 70s [4], [5]. However, experience with those metrics in the area of controller programming is rare.

In contrast to known software quality metrics the concept of transparency originates from the area of controller design. Primary goals in applying formal methods to controller design are the correctness and the transparency of the resulting algorithm. Correctness and transparency are independent properties. An algorithm can be correct or not; and it can be more or less transparent. Transparency of an algorithm is defined as follows [6]:

- At any time it must be easy and clear to see what the controller does in the moment and what it will do in the next step.
- At any time it must be possible to reinterpret the algorithm, i.e. the aim of the algorithm must be recognizable.

The main motivation for transparency are increasing costs for software maintenance. To assure a good maintainability the software should have

- good analyzability if failures occur,
- easy changeability to remove detected design faults or to modify the software,
- high stability, i.e. low risk of unexpected effects due to changes,
- good testability to proof the success of changes with lowest expense.

Petri nets (PN) are a well established formal framework for the specification and analysis of logic control algorithms. In [6] transparency metrics for Signal Interpreted Petri Nets (SIPN) are introduced. Based on the fact that the less complex a net is the more transparent it is—in this contribution—known complexity metrics from Computer Science are investigated and adapted to the SIPN framework. As a result, two new criteria for transparency are introduced in this contribution, the Net Complexity and the Expression Complexity. The metrics are introduced using SIPN. Yet their application is not restricted to SIPN. The net complexity can be calculated for all types of PN and even for similar formal descriptions as Sequential Function Chart according to IEC61131-3 standard, because it only evaluates structural properties of the PN and its reachability graph. The expression complexity naturally is restricted to PN with expressions as firing conditions.

The rest of this contribution is structured as follows. At first, the SIPN is introduced. Following are the definition of the two new measures Net Complexity and Expression Complexity with their corresponding transparency metrics. An illustrative example shows how the metrics are applied. The contribution closes with a short summary and an outlook on further work.

Contact: Georg Frey, University of Kaiserslautern, Institute of Process Automation, PO Box 3049, 67653 Kaiserslautern, Germany, tel.: +49 631 205 3652, fax: +49 631 205 4462, e-mail: frey@eit.uni-kl.de.
II. SIGNAL INTERPRETED PETRI NETS (SIPN)

Signal Interpreted Petri Nets (SIPN) can be used for the formal specification of a control algorithm [7]. With SIPN, logic controllers are modeled by places—setting output signals—and transitions between those places—depending on input signals. SIPN allow formal analysis of a variety of properties, [7]. The strong relation of SIPN to SFC, a standard PLC programming language according to IEC61131-3 standard [8], [9] allow their use in practical applications.

A. Formal Definition

A Signal Interpreted Petri Net is described by a 9-tuple

\( (P, T, F, m_0, I, O, \varnothing, \varphi, \Omega) \)

with:

- \( P \) a set of places
- \( T \) a set of transitions
- \( F \) a set of arcs
- \( m_0 \) an ordinary PN with places \( P \), transitions \( T \), arcs \( F \), and binary initial marking \( m_0 \)
- \( I \) a set of logical input signals
- \( O \) a set of logical output signals with \( I \cap O = \emptyset \)
- \( \varnothing \) a mapping associating every transition \( t_i \in T \) with a firing condition \( \varnothing(t_i) \) = Boolean function in \( I \)
- \( \varphi \) a mapping associating every place \( p_i \in P \) with an output \( \varphi(p_i) \in \{0, 1, -\}^{(O)} \)
- \( \Omega \) the output function of the net combines the output of all marked places. For a formal definition of \( \Omega \) see e.g. [6].

Figure 1 shows the graphical representation of an SIPN.

![Figure 1: SIPN](image)

\[ \begin{align*}
\varphi(t_1) &= i_1 \vee i_2 \\
\varphi(t_2) &= i_1 \vee i_2 \\
\varphi(t_3) &= i_1, i_2 \\
\varphi(t_4) &= i_1, i_2, \neg i_1
\end{align*} \]

B. Dynamic behavior

The dynamic behavior of an SIPN is given by the movement—or flow—of tokens through the net i.e. the change of its marking. This flow is realized by the firing of transitions. The firing of a transition \( t_i \) removes a token from each of its pre-places and puts a token on each of its post-places. For the firing process there are five rules.

1. A transition is enabled, if all its pre-places are marked and all its post-places are unmarked.
2. A transition fires immediately, if it is enabled and its firing condition is fulfilled.
3. All fireable transitions fire simultaneously.
4. The firing process is iterated until a stable marking is reached (i.e. until no transition can fire anymore).
5. After a stable marking is reached, the output signals are recalculated by applying \( \Omega \) to the new marking.

C. Logic Control Semantics

Places in an SIPN mean situations. A situation is a local state of the controller. A situation can be active or non-active. While a situation is active it can influence the controller environment via a corresponding output function.

Transitions specify under what circumstances a situation ends and a new situation gets active. The firing condition of the transition specifies when the change of situations is allowed to happen.

D. Reachability Graph

A marking \( m' \) is said to be reachable from a marking \( m \) if there exists a sequence of input signal settings such that a firing sequence starting from \( m \) reaches \( m' \) as a stable final marking. The reachability graph is a graph with all markings reachable from \( m_0 \) as vertices. An edge from \( m_i \) to \( m_j \) indicates that there exists an input signal setting such that the marking \( m_i \) is the next stable marking reached from \( m_j \). The reachability graph of an SIPN shows which markings (states with corresponding outputs) are reachable in the controller and how they are reachable (transitions with corresponding inputs). The edges of the graph are labeled with the transitions and inputs and the vertices are labeled with the marking and the output of the net (cf. Figure 2).

![Figure 2: Reachability Graph of the SIPN in Figure 1](image)
III. NET COMPLEXITY (NC)

To get a measure for the structural complexity of a PN, four known complexity measures from Computer Science are adapted to PN and combined in a weighted sum. The measure Net Complexity (NC) is composed of the measures: Structural Complexity of PN ($SC_{PN}$), Hierarchical Complexity of PN ($HC_{PN}$), Unstructuredness of PN ($MCC_{PN}$), and Branching of PN ($B_{PN}$). These four components rate different aspects of the structure of a PN and their combination gives a good coverage of the varying aspects of structural PN complexity.

The four components are combined in a weighted sum to give the complexity measure NC (Net Complexity). Finally the measure NC is normalized to one resulting in the transparency metric $\pi(NC)$.

A. Structural Complexity of Petri Net ($SC_{PN}$)

The measure $SC_{PN}$ rates the complexity of the PN structure. The structural complexity decreases if independent parts of the program are grouped into modules.

Definition 1: Module $M$ in a SIPN
A module is an independent part of the PN with exactly one input place and one output place. The minimal module consists of at least two places and one transition. The main PN is also counted as one module.

$SC_{PN}$ is defined analogously to the measure “Structural Complexity” defined for Flowcharts in [10]. It relates the number of arcs in a PN to the number of modules.

Definition 2: Structural Complexity of PN

$$SC_{PN} = \frac{|F|}{M} \text{ with } |F| = \text{number of arcs in the PN} \quad M = \text{number of modules in the PN}$$

In Figure 3 a Petri net with five modules (the main PN, module 1, module 2, and module 3 and module 4; $M=5$) is shown. The total number of arcs in the net is sixteen ($|F| = 16$). Hence in this example $SC_{PN} = 16/5$ holds.

B. Hierarchical Complexity of PN ($HC_{PN}$)

Petri nets offer the possibility to structure an algorithm in several hierarchical levels:

Definition 3: Level $L$

The different steps in the hierarchy of a SIPN are called levels. In the deepest level all modules must be subdivided in places, edges and transitions.

The measure $HC_{PN}$ is defined analogously to the measure “Hierarchy of Complexity” defined for flowcharts in [10]. The complexity decreases if an hierarchy is used to give a better overview of the control algorithm.

Definition 4: Hierarchical Complexity of PN

$$HC_{PN} = \frac{M}{L} \text{ with } M = \text{number of modules in the SIPN} \quad L = \text{number of the levels in the SIPN}$$

Figure 3 shows an SIPN consisting of five modules (main SIPN plus modules one to four; $M = 5$) that are structured hierarchically into three levels ($L = 3$). Hence in this example $HC_{PN} = 5/3$ holds.

C. Unstructuredness of Petri Net ($MCC_{PN}$)

The measure $MCC_{PN}$ refers to the measure “Unstructuredness of Flowgraph” by McCabe [5]. In the following it is redefined for PN. The measure evaluates the number of modules, cycles, and decision places.

![Diagram of different levels and modules of an SIPN](image-url)
Definition 5: Cycle in a PN
A cycle is a sequence of arcs \((p_0, t_0), (t_0, p_1), (p_1, t_1), ..., (p_{n-1}, t_{n-1}), (t_{n-1}, p_n)\) and \(p_n = p_0\). Two cycles are equal if they contain the same arcs.

Definition 6: Decision place in a PN
Places with two or more output arcs are called decision places. The weighted sum \(|D_P|\) counts each decision place with the number of its output arcs \(|p_i \cdot |\) minus one:

\[
|D_P| = \sum_{i=1}^{k} (|p_i \cdot | - 1)
\]

Definition 7: Unstructuredness of PN
\(MCC_{PN} = |D_P| - M + C + 1\)

\(|D_P|\) = Weighted number of decision places in a PN.
\(M =\) number of modules in a PN, and
\(C =\) number of cycles in a PN.

Figure 4 shows an example for the calculation of \(MCC_{PN}\).

To calculate the measure, first its components have to be derived:

1. Both places in the PN of Figure 4 are decision places, place \(p_1\) with two output arcs and place \(p_2\) with three output arcs. Hence, \(|D_P| = 3\) holds.
2. There is only the main PN (\(M = 1\)).
3. The net owns three cycles. The arcs \((p_1, t_2), (t_2, p_1)\) form a cycle, likewise the arcs \((p_2, t_5), (t_5, p_2)\) and the arcs \((p_1, t_1), (t_1, p_2), (p_2, t_3), (t_3, p_1)\). For the number of different cycles \(C = 3\) holds.

The combination results in \(MCC_{PN} = 6\) for the example.

D. Branching (\(B_{PN}\))

The measure Branching refers to the measure \(\varepsilon\) of Schmidt [11]. It rates the degree of nesting of a graph \(G\).

Definition 8: Decision Vertex
A vertex of the reachability graph of a PN is called decision vertex, if it has two or more output edges. For decision vertices \(v_i \cdot \geq 2\) holds. For a graph the weighted sum of decision vertices is defined as follows:

\[
|D_V| = \sum_{i=1}^{k} (|v_i \cdot | - 1), \text{ with } |v_i \cdot | \text{ is the number of output edges of the vertex } v_i.
\]

Subgraphs result from the existence of decision vertices in the reachability graph \(RG\) of a PN.

Definition 9: Subgraph
Elements of a subgraph \(RG_U\) are a decision vertex and all vertices and edges after the decision vertex. The subgraph ends if a terminal vertex, the vertex corresponding to the initial marking or a vertex that is already part of \(RG_U\) is reached.

The measure relates the weighted sum of decision vertices of all subgraphs of a PN reachability graph to the weighted sum of decision vertices in the reachability graph. If there are no decision vertices, then \(B_{PN}\) is zero.

Definition 10: Branching

\[
B_{PN} = \begin{cases} 
\frac{\sum_{k=1}^{m} |D_V(RG_U)|}{|D_V(RG)|}, & \text{if } |D_V(RG)| > 0 \\
0, & \text{if } |D_V(RG)| = 0 
\end{cases}
\]

\(k\) is the number of subgraphs \(RG_U\) in the reachability graph \(RG\) of the PN.

As an example Figure 5 shows a reachability graph with two subgraphs:

\(RG_{U1} = (\{m_1, m_3, m_5\}, \{(m_1, m_3), (m_3, m_5)\})\) and

\(RG_{U2} = (\{m_1, m_2, m_4\}, \{(m_1, m_2), (m_2, m_4)\})\).

Figure 5: Subgraphs in \(RG\)
E. Combination to the complexity measure NC (Net Complexity)

The combined complexity measure NC is the weighted sum of the four presented measures. The four aspects may be weighted according to the problem under consideration through the factors ci. In the following ci = 1 is assumed for all factors which showed good results:

**Definition 11: Net Complexity (NC)**

\[
NC = c_1 \text{ SCPN} + c_2 \text{ HCPN} + c_3 \text{ MCCPN} + c_4 \text{ BPN}
\]

To derive a transparency metric NC is normalized to one. The function \(z(\text{NC})\) is a mapping of NC into the Interval \([0; 1]\) (\(z : \text{NC} \rightarrow [0; 1]\)).

\[
z(\text{NC}) = \frac{30}{(\max(10; \text{NC}) + 20)}
\]

According to the function the transparency increases, if the value of NC becomes smaller, however if the value of NC is smaller than 10 the transparency could not be improved anymore. This is because programs under a certain level of complexity are quite easy to understand. The constants in \(z(\text{NC})\) are derived empirically. If the weighting factors in NC are changed, then these constants also have to be adjusted.

IV. EXPRESSION COMPLEXITY (EC\text{SIPN})

This measure rates the expression complexity of the firing conditions that are associated with the transitions of an SIPN. According to the measure “Expression Complexity” of Rechenberg [11] the following values of the expression complexity are allocated to the elements of the firing conditions.

<table>
<thead>
<tr>
<th>Elements</th>
<th>Expression Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Signal</td>
<td>0</td>
</tr>
<tr>
<td>Not</td>
<td>1</td>
</tr>
<tr>
<td>And, Or</td>
<td>2</td>
</tr>
<tr>
<td>NAND, NOR</td>
<td>3</td>
</tr>
</tbody>
</table>

*Table 1: Expression complexity of firing conditions*

The expression complexity of a transition \(ec(t_i)\) is given by the sum of all the expression complexities according to Table 1 of the elements of its firing condition. For each term in brackets the complexity value of the bracketed term is multiplied with 1.5.

The expression complexity \(EC\text{SIPN}\) of an SIPN is the ratio of the expression complexities \(ec(t_i)\) of all its transitions to the number of transitions

**Definition 12: Expression Complexity EC\text{SIPN}**

\[
EC\text{SIPN} = \frac{\sum_{i=1}^{T} ec(t_i)}{|T|}
\]

The corresponding transparency metric \(z(\text{EC\text{SIPN}})\) is defined via a decreasing exponential:

\[
z(\text{EC\text{SIPN}}) = e^{-\frac{\text{EC\text{SIPN}}}{10}}
\]

V. EXAMPLE

The following example shows the application of the presented reachability graph \(\text{RG\text{SIPN}}\) is shown in Figure 7.
From the figures the values needed for the calculation of the complexity metrics can be derived (Table 2):

\[
|F| = 22; M = 5; L = 3; C = 3
\]

\[
|D_p| = \sum_{i=1}^{5} |p_i \bullet -1| = 0 + 0 + 1 + 1 + 0 + 0 + 0 + 0 = 2
\]

\[
RG_{U1} = \{(m_2, m_3), \{(m_2, m_3, m_8)\}
\]

\[
RG_{U2} = \{(m_2, m_4, m_5, m_6, m_7), \{(m_2, m_3), (m_3, m_4), (m_4, m_6), (m_3, m_5), (m_5, m_7)\}\}
\]

\[
RG_{U3} = \{(m_3, m_5, m_7), \{(m_3, m_5), (m_5, m_7)\}\}
\]

\[
RG_{U4} = \{(m_3, m_5), \{(m_3, m_5), (m_5, m_7)\}\}
\]

\[
\sum_{i=1}^{4} |D_V(RG_{U3})| = 1 + 2 + 1 + 1 = 5
\]

\[
|D_V(RG)| = \sum_{i=1}^{5} |v_i \bullet -1| = 0 + 0 + 1 + 0 + 0 + 0 + 0 + 0 + 0 = 2
\]

Table 2: Values of the describing constants.

With the data given in Table 2 the different measures and metrics can be calculated the results are shown in Table 3:

\[
SCPN = |F|/M = 22/5
\]

\[
HC_{PN} = M/L = 5/3
\]

\[
MCC_{PN} = |D_p| - M + C + 1 = 2 - 5 + 3 + 1 = 1
\]

\[
B_{PN} = \sum_{i=1}^{3} |D_V(RG_{U3})| = 5/2 = 2.5
\]

\[
NC = SCPN + HC_{PN} + MCC_{PN} + B_{PN} = 9.57
\]

\[
EC_{SPN} = \sum_{i=1}^{4} \text{ec}(t_i) = 0 + 3 + 3 + 3 + 2 + 3 + 4 + 2 + 0 + 7 + 3 + 3 = 30
\]

\[
x(\text{NC}) = 30/(\max(10; \text{NC}) + 20) = 1
\]

\[
x(\text{EC}_{SPN}) = e^{-\frac{30}{10}} = e^{-\frac{3}{11}} = 0.76
\]

Table 3: Results for the example

The results show that the presented example is not very complex and therefore very transparent.

### VI. Conclusions and Outlook

In this contribution two complexity metrics for Petri nets are introduced. The *Net Complexity* includes different aspects of the PN structure. It can be calculated for all PN types and for other models as Sequential Function Chart. The *Expression Complexity* is not restricted to PN but in the PN framework it can only be applied to nets with logical expressions like the presented Signal Interpreted Petri Net. Since high complexity makes it harder to understand an algorithm, it results in low transparency.

The presented metrics show good results in laboratory examples. Besides the fact, that they are adaptations and combinations of known and validated metrics from different areas of Computer Science. The practical evaluation and validation of the new combinations—especially in case of the Net Complexity metric—is still a task for the future.

The next step in the presented work is the implementation of algorithms for the automatic calculation of the new metrics. With this automatic calculation larger applications can be studied.

The scope of the presented work was the adaptation of known metrics to Petri Nets. Further work will be done in the research for complexity metrics that evaluate special PN properties such as concurrency.

### VII. References


