4.3 System design using Petri Nets


Introduction

We present a Petri Net based design method which supports the initial phases of the Software development process. This method allows for the systematic specification and analysis of system requirements and Software requirements.

Our design method
- is easy to learn and to understand, also for somebody who is not familiar with computers,
- allows for a continuous and systematic development leading from informal to formal specifications,
- supports the problem-oriented structuring of a system,
- allows for a systematic transition from the representation of static components and relations to the description of dynamic behaviour,
- comprises, besides of the usual top-down- and bottom-up- hierarchies also the possibility of embedding or isolating components,
- contains methods to discover inconsistencies and gaps within a design,
- allows to formulate and to verify system properties without regard to the (particular) implementation.

4.3.1 An Example

**Task:** We have to model an industrial assembly line.

Assume the following requirements of our assembly line:

There are three machines M1, M2, M3 and two Operators O1 and O2.

Each task has first to be processed at M1, then either at M2 or M3.

The Operator O1 has access to M1 and M2. O2 has access to M1 and M3
At the beginning we want to model this system using channel-instance-nets (i.e. without dynamic).

As a first approach to the modelling of some real system, it is useful to decompose the system into a few, but significant components. As an example, fig. 4.3.1 shows a very first step towards the design of an industrial assembly line. Some more details of this system are given in fig. 4.3.2 and in fig. 4.3.3 where only the new channels and instances are more detailed labelled.

**Fig. 4.3.1:** Coarse structure of a simple industrial assembly line

**Fig. 4.3.2:** More detailed structure of the assembly line

**Fig. 4.3.3:** Assembly line with an additional channel Operator

The decompositions of fig. 4.3.1 which are shown in fig. 4.3.2 and 4.3.3 are oriented at the objects to be processed and the processing machines. On the contrary fig. 4.3.4 decomposes the system from the operator's point of view new.
Fig. 4.3.4: Decomposition of the system from the operator's point of view

Fig. 4.3.5, finally, represents the system on the detailed level of a place-transition-net (usually called Petri Net), specifying the dynamic behaviour of the system by means of the weak firing rule.
4.3.2 Refinements and Embeddings

The nets of fig. 4.3.1 to fig. 4.3.5 concern all one distinguished system. It will turn out that there nets - and all nets which occur in this paper - are related to each other in a well-defined way. Starting from any of them, the other nets can be constructed systematically be refinements and embeddings.

A net A is refined by replacing an element x (channel or instance) and its surrounding arcs by a whole net B. The result of this operation should, of course, again be a net (in the following called C).

**Definition 4.3.1**

In a net A a channel p is *refined* by a net B if B is substituted instead of p such that for each arrow x \(\rightarrow\) y from B to A' (resp. y \(\rightarrow\) x from A' to B) hold:

1) x is a channel of B and y is an instance of A'.

2) In A exists an arrow p \(\rightarrow\) y (resp. y \(\rightarrow\) p).
Thereby A’ is the net which results from A by deleting of p and its incoming and outcoming arrows.

**Definition 4.3.2**

In a net A an instance t is refined by a net B if B is substituted instead of t such that for each arrow x -> y from B to A’ (resp. y -> x from A’ to B) hold:

1) x is an instance of B and y is an channel of A’.
2) In A exists an arrow t -> y (resp. y -> t).

Thereby A’ is the net which results from A by deleting of t and its incoming and outcoming arrows.

**Definition 4.3.3**

A net C is a refinement of a net A if C results from A by refinements of some channels and instances. Then net A is called a coarsening of net C.

For example, the steps from fig. 4.3.1 to fig. 4.3.2 and from fig. 4.3.3 to fig. 4.3.5 are refinements.

The step from fig. 4.3.2 to fig. 4.3.3 cannot be understood as a refinement: fig. 4.3.3 contains arcs which have no representative in fig. 4.3.2. Respecting the inscriptions of channels and agencies we find that there is no other possibility than to consider "operators" as an additional buffer: The net of fig. 4.3.2 is embedded into the net of fig. 4.3.3 (or net 4.3.3 is an embedding of net 4.3.2).

**Definition 4.3.4**

A net A is embedded in a net C if each channel, each instance, each arrow, and each inscription of A also exists in C.

Then C is called an embedding of A and A is called an extract from C.

Starting from small nets with complex or informal inscriptions, we obtain by iterated refinements more detailed nets with less complex or more formal inscriptions. Eventually one may obtain nets with a formally defined firing rule, e.g. place-transition-nets, as given in fig. 4.3.5.
The central concern of this paper is a systematic study of relationship between refinements and embeddings.

4.3.3 Design and Schedule

The Development of large systems according to refinements and embeddings as in chapter 4.3.1 yields a large number of nets such that a survey can easily be lost. Systematic methods are necessary to support the design process and to detect inconsistencies or gaps. Such systematic methods will be derived in the following.

As a first idea for such a method, it is possible to represent, in which way nets related by refinements and embeddings. This can be achieved by a graphical representation, as illustrated in fig. 4.3.6 for the five nets of our example: *Refinements are represented as simple vertical arcs (↓), embeddings are represented as double-lined horizontal arcs (=>). Thereby it is convenient that refinements and embeddings are both transitive operations.*

**Definition 4.3.5**

A set of nets which shows some aspect or parts of a system is called a **design**. The graphical representation of the relations with respect to refinements and embeddings between the nets of a design is called a **schedule**.

Figure 4.3.6 shows an example of a schedule of a design.

```
1
|   
| v
v 2 => 3  4
| /    
| vv
v v
 5
```

**Fig. 4.3.6:** Schedule of the design of our production system

The orientation of arcs shows the direction of information increase.
The schedule of a design may be used to detect inconsistencies and gaps. In order to describe some rules for these investigations, we extend our example. Fig. 4.3.7 shows a coarse structure of our system which, compared with fig. 4.3.1, is *enlarged* by the operators.

![Diagram](image1)

**Fig. 4.3.7:** Enlarged coarse structure of the industrial production system

Fig. 4.3.8 shows a refinement of fig. 4.3.7 which turns out to refine also fig. 4.3.4.

![Diagram](image2)

**Fig. 4.3.8:** Joint refinement of fig. 4.3.3 and fig. 4.3.4
In fig. 4.3.9 we can see how the new nets of fig. 4.3.7 and fig. 4.3.8 fit in the schedule of fig. 4.3.6.

\[
\begin{align*}
1 & \Rightarrow 7 \\
| & \\
v & \\
\downarrow & \\
2 & \Rightarrow 3 \ 4 \\
| & \\
\downarrow & \\
v & \\
8 & \\
| & \\
v & \\
5
\end{align*}
\]

Fig. 4.3.9: Schedule of the enlarged design

### 4.3.4 Consistency of a Design

During the design of large systems there may arise inconsistencies; in particular, if several persons are involved in the design process.

In this chapter, we shall present some rules which help to detect such inconsistencies. To be more precise, we describe some conditions on the schedule of designs which are necessary for **consistency**.

In the following presentations we use broken arcs (- - > and = = >) to denote arbitrary long arc sequences of the corresponding type.

**Rule 1** The schedule of a consistent design contains no figure shaped like

\[
A \\
/ \ \\
/ \ \\
/ \ \\
v \ v \\
B = = > C
\]

As an example, the idea of this rule can be illustrated by fig. 4.3.6. In fig. 4.3.6 we get
which means, that the "operators" are not part of "processing a task", but are an additional component with respect to fig. 4.3.2. Arguing from a formal point of view one might construct a refinement from fig. 4.3.1 to fig. 4.3.3. In this case, the "operators" would be part of the processing unit. Each of the two refinements (1 -> 2 and 1 -> 3) is acceptable, but we cannot allow both of them in one design because this yields a contradiction.

**Rule 2**

The schedule of a consistent design contains no figure shaped like

```
1
  /
 v
 2 = > 3
```

The dotted line (---) denotes either an -> arrow or an = = > arrow which is directed either from B to C or from C to B.

Actually this rule represents four different rules since the dotted line may be interpreted in four different ways. For instance, it is excluded by this rule that a refinement is at the same time regarded as an embedding, or that a net element as well as one of its refinements is contained in the same net.

**Rule 3**

The schedule of a consistent design contains no path sequence which forms a circle.

The soundness of this rules is obvious: Successive refinements and embeddings can never yield the initial net again.

**Definition 4.3.6**

A design is called **consistent** iff its schedule respects the rules 1 to 3.

As fig. 4.3.9 shows, our example is consistent.
4.3.5 Completeness of a Design

If the most refined nets of a design describe a system on the same level, it should eventually be possible to compose them to one most detailed and most comprehensive net.

In this chapter we shall state some rules showing how to supplement a design in order to obtain a complete design.

**Rule 4** The schedule of a complete design is connected: Each two nets of a schedule are connected by some arcs of arbitrary type and arbitrary orientation.

**Rule 5** In the schedule of a complete design, each figure shaped like

\[
\begin{align*}
A & \rightarrow B \\
| & | \\
| & | \\
v & v \\
C & \text{is supplemented, yielding} & D
\end{align*}
\]

This is an important rule. Whenever a net A is embedded, this rule requires the specification of how all the refinements of A are embedded. It is mainly this rule which guarantees that eventually a most detailed and most comprehensive net exist in a complete design.

**Rule 6** In the schedule of a complete design, each figure shaped like

\[
\begin{align*}
A & \ \\
/ \ & \ \\
/ \ & \ \\
v & v \\
B & C & \text{is supplemented, yielding} & B & C \\
\ \\
\ \\
v & v
\end{align*}
\]

This may include \( B = D \) or \( C = D \).

In fig. 4.3.9 we find this figure, assuming \( A = 7 \), \( B = 3 \) and \( C = 4 \). 3 and 4 represent different refinements of the net \( 7 \). It would not be adequate to assume that only one of them is correct. Rather, they may represent different aspects of the systems. Rule 5 requires only that these aspects fit together in a joint refinement \( D \). In fig. 4.3.9 this is the case with \( D = 8 \).

**Rule 7** In the schedule of a complete design, each figure shaped like

\[
\begin{array}{c}
\text{A} = = \Rightarrow \text{B} \\
\text{||} \\
\text{||} \\
\text{v} \\
\text{C} \\
\end{array} \\
\begin{array}{c}
\text{A} = = \Rightarrow \text{B} \\
\text{||} \\
\text{||} \\
\text{v} \\
\text{v} \\
\end{array}
\]

is supplemented, yielding \( \text{C} \rightarrow \rightarrow \cdots \rightarrow D \).

This may include \( B = D \) or \( C = D \). Each \( \rightarrow \rightarrow \cdots \rightarrow \) arrow represents a path sequence with arcs of arbitrary type but with the indicated orientation.

**Definition 4.3.7**

A design is called **complete** if its schedule respects the rules 4 to 7.

Fig. 4.3.9 shows that our example represents a complete design.

A design is successfully finished if it is consistent and complete.

**Definition 4.3.8**

A design is called **final** if it is consistent and complete.

A central property of our design method is the following:

**Theorem 4.3.9**

In a final design there exists a most detailed and most comprehensive net \( N_0 \).
This means that from each net in the schedule of the design there exists a path to $N_0$. According to rule 3, no arc starts at $N_0$. Fig. 4.3.9 shows that our design is final. The net of fig. 4.3.5 is most detailed and most comprehensive.

The construction of final designs requires often a large number of nets. In many cases, the new nets which must be constructed due to rule 5 result uniquely from the given nets. (In such cases, their explicit construction can be skipped.) We first characterize *unique completions*.

**Theorem 4.3.10**

Let $A$ be a net, let $B$ be an embedding and let $C$ be a refinement of $A$. Then there exists a unique net $D$ such that the diagram

\[
\begin{array}{c}
A \Rightarrow B \\
\downarrow \quad \downarrow \\
C \Rightarrow D
\end{array}
\]

commutes iff no element $x$ of $A$ is connected with a new arc of $B$ and at the same time refined in $C$.

As an example, fig. 4.3.10 shows a unique net $D$, in contrast to fig. 4.3.11.

![Diagram showing unique completion according to rule 5](image)

**Fig. 4.3.10:** Unique completion according to rule 5
Fig. 4.3.11: According to rule 5, D is not unique

One or both of the dotted arcs \( \longrightarrow \) may exist.

In order to show an example, we slightly enlarge our design. Fig. 4.3.12 shows an embedding of the net given in fig. 4.3.7.

Fig. 4.3.12: Embedding of fig. 4.3.7

If this net is added to the design of fig. 4.3.9, the resulting design (fig. 4.3.13) is no longer complete.
According to rule 5, four additional nets W, X, Y, Z are necessary, as indicated in fig. 4.3.14.

*Fig. 4.3.13: Enlargement of schedule of fig. 4.3.9*

We shall not explicitly construct these nets here, because they result uniquely from the rest of the design, as guaranteed by Theorem 4.3.10.

*Fig. 4.3.14: Completion of the schedule of fig. 4.3.13*