

APPLYING PETRI NETS TO REQUIREMENTS VALIDATION

José Reinaldo Silva

Mechatronics Engineering Dept.
University of Sao Paulo, Brazil
reinaldo@usp.br

Eston Almança dos Santos

Mechatronics Engineering Dept.
University of Sao Paulo, Brazil
eston@usp.br

Abstract. *Mechatronics and Information Systems can be assumed as complex, and thus have a prohibitive cost for adjustments out of the specification phase. This emphasizes the early development phases as especially critical for the success of projects, besides being very important to guide implementation decisions. We proposed a requirement validation method suitable to a life cycle represented with the Unified Process and addressed to the specification and design of information systems. The key issue of the proposal is the facility to model information flow formally, by the capture of the requirements into Elementary Petri Nets. A software tool is proposed to perform the transformation capturing the dynamic of the processes. Requirements validation is performed in the net, by simulation or by property analysis (suitable to large systems). Finally, we briefly discuss the achievement of a proper (and minimum) set of viewpoints enrolled in information system projects.*

Keywords. *Requirements validation, Mechatronics, Petri Nets*

1. Introduction

There is a growing demand for disciplined methods of development in modern Systems Engineering, specially for automated systems, which includes not only the critical problem of managing the fusion of hardware and software but also the requirements for the software system.

Conceptual and practical schemas appear frequently in the literature, some of which are associated with software development techniques, or connected with specific areas of Engineering, or even with the knowledge acquired in the development of specific artifacts. However, in spite of the consensus that the early phases of the development process are the most important to assure that the target system works properly, still, most of the tools to support system development address the design and implementation phases.

The present paper focus on the validation of requirements only, supposing that the artifact is related to processes control – meaning the control of business process or industrial processes as introduced in the CIMOSA (Vernadat, 1996) terminology. In both cases the challenge is to find a systematic transformation between requirements and specifications.

In the informal requirement elicitation it is common to face problems like: i) adjust several viewpoints generated by different agents related with the artifact ii) remove discrepancies, contradictions and iii) validate the acquired requirements. Information Systems Design faces the same problems but requires special attention to the validation of the information flow and control signs (considering the supervisory systems as the same line of artifact).

Petri Nets has being used, as a case tool to the modeling, analysis and documentation of this kind of system and it is also a good framework to express requirements in this area. A simple transformation can be applied to transform Use Cases in an Elementary Petri Net, as we will present in this paper. Thus, Use Cases could be considered a metaphor for requirements informally elicited. Besides, in a first approach, the sequence in which the events are planned can be easily tested by direct simulation or property analysis. The later approach is specially promising to large systems.

Also, Petri Nets are flexible enough to allow that detected anomalies in the requirements specifications could be corrected changing the net directly or suggesting a change in the Use Case. Therefore we could end with a set of isomorphic net models representing different viewpoints of the same information system.

It is not in the scope of this paper a complete analysis of generic viewpoints for information systems, but it is not possible to neglect this problem. Therefore, we propose a set of three basic viewpoints: one from the *provider* of the system, one for the *final user* and one for the *developer*. A complete analysis of the formal process of verifying isomorphism among the nets of each viewpoint is also out of the scope of this paper.

The *provider* is the business owner or the controller of the business process, while the *final user* is the set of agents or machines that effectively interact with the system without changing it.

An illustrative (and classical) example is presented, based on the last release of the Rational Unified Process (RUP) (Rational Software Corporation, 2001): an automated bank teller machine. Initial Use Cases were taken from (Rational Software Corporation, 2001) and transformed in a tagged specification, where the tags are specific signals to the Net Translator. The resulting net is sent to a Net simulator where it is possible to choose conveniently an initial marking and following states. Interruptions in the flow of marks can then be inspected as possible mistakes or contradictions in the requirements.

2. Turning Use Cases in a Tagged Representation

Similarly to the development of any artifact, for information systems the requirements represent “what the system does”, or, in other words a generic description of its behavior, which is composed of the integrated behavior of subcomponents. We will call these functionalities business processes, generalizing the meaning of the term proposed in the terminology of CIMOSA (Vernadat, 1996).

Thus, a business process as used in this work is such that: a set of events arranged in a proper schema, that is, a directed sub-graph where there are two distinguished unique elements called start and end, such that all remaining elements belongs to one of the paths connecting these two elements; there is a unique business process to each functionality;

All business process is composed of a set of events that stands for specific operations in a shop floor, or generically, characteristic operation of the business in focus. We call those operations enterprise activities.

Therefore, a requirement elicitation for an information system is a process that identifies all business processes of the target application (be it a manufacturing site, service in the web, etc.) and its components, the enterprise activities. Each business process can be represented by a net (a sequence of events or operations) as well as the whole system, which is a process composed by several (alternative or integrated) business processes.

As a typical example, let us take the Use Case diagram for an ATM machine (Rational Software Corporation, 2001), already used to illustrate the application of RUP method of development. With this example (for instance the functionality Cash Withdraw) we will show how to transform Use Case in a tagged representation and then in a Petri Net. We used the HPSim (Anschuetz, 2001) simulator to analyze the business processes.

We will show finally that the same approach can be applied to different viewpoints of the same functionality, resulting nets that have isomorphic subnets.

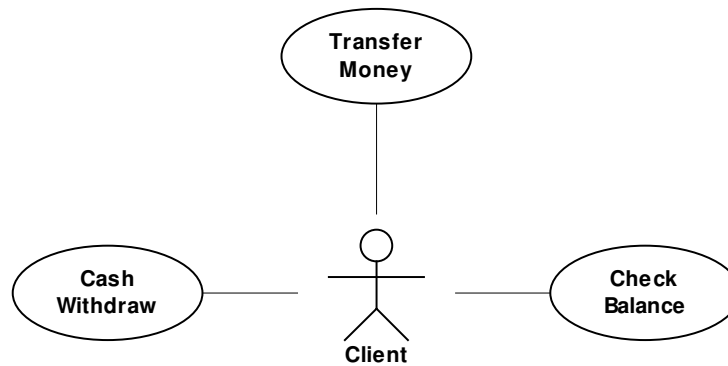


Figure 2-1 Use Cases diagram for ATM

2.1. Use Case for the Cash Withdraw Functionality

Following there are the basic and alternative flows of events for the Cash Withdraw functionality.

2.1.1. Basic Flow of Events

1. Initiate Withdraw – Customer inserts bank’s card in the card reader on the ATM machine
2. Verify Bank Card – The ATM reads the account code from the magnetic strip on the bank card and checks if it is an acceptable bank card
3. Enter PIN – The ATM asks for the customer’s PIN code (4 digits)
4. Verify PIN – The account code and PIN are verified to determine if the account is valid and if the PIN entered is the correct PIN for the account. For this flow, the account is a valid account and the PIN is the correct PIN associated with this account
5. Select Withdraw – The ATM displays the different alternatives available at this ATM. In this flow, the bank customer always selects “Cash Withdraw”
6. Enter Amount – The ATM asks for the amount to withdraw. For this flow the customer selects a pre-set amount (\$10, \$20, \$50, or \$100)
7. Authorization – The ATM initiates the verification process with the Banking System by sending the Card ID, PIN, Amount, and Account information as a transaction. For this flow, the Banking System is online and replies with the authorization to complete the cash withdrawal successfully and updates the account balance accordingly
8. Dispense – The Money is dispensed
9. Receipt – The receipt is printed and dispensed. The ATM also updates the internal log accordingly
10. Return Card – The Bank Card is returned

2.1.2. Alternative Flows of Events

1. Not a valid Card – In Basic Flow Step 2 - Verify Bank Card, if the card is not valid, it is ejected with an appropriate message
2. ATM out of Money – At Basic Flow Step 5 - Select Withdraw, if the ATM is out of money, the “Cash Withdraw” option will not be available
3. Insufficient funds in ATM – At Basic Flow Step 6 - Enter Amount, if the ATM contains insufficient funds to dispense the requested amount, an appropriate message will be displayed, and rejoins the basic flow at Step 6 - Enter Amount.
4. Incorrect PIN – At Basic Flow Step 4 - Verify Account and PIN, the customer has three tries to enter the correct PIN. If an incorrect PIN is entered, the ATM displays the appropriate message and if there are still tries remaining, this flow rejoins Basic Flow at Step 3 - Enter PIN. If, on the final try the entered PIN number is incorrect, the card is retained, ATM returns to Ready State, and this use case terminates
5. No Account – At Basic Flow Step 4 - Verify Account and PIN, if the Banking system returns a code indicating the account could not be found or is not an account which allows withdrawals, the ATM displays the appropriate message and rejoins the Basic Flow at Step 10 - Return Card
6. Insufficient Funds in Account – At Basic Flow Step 6 - Authorization, the Banking system returns a code indicating the account balance is less than the amount entered in Basic Flow Step 6 - Enter Amount, the ATM displays the appropriate message and rejoins the Basic Flow at Step 6 - Enter Amount
7. Daily maximum withdrawal amount reached – At Basic Flow Step 7 - Authorization, the Banking system returns a code indicating that, including this request for withdrawal, the customer has or will have exceeded the maximum amount allowed in a 24 hour period, the ATM displays the appropriate message and rejoins the Basic Flow at Step 6 - Enter Amount
8. Log Error – If at the Basic Flow Step 9 - Receipt, the log cannot be updated, the ATM enters the “secure mode” in which all functions are suspended. An appropriate alarm is sent to the Bank System to indicate the ATM has suspended operation
9. Quit – The customer can, at any time, decide to terminate the transaction (quit). The transaction is stopped and the card ejected
10. “Tilt” – The ATM contains numerous sensors that monitor different functions, such as power, pressure exerted on the various doors and gates, and motion detectors. If at any time a sensor is activated, an alarm signal is sent to the Police and the ATM enters a “secure mode” in which all functions are suspended until the appropriate re-start / re-initialize actions are taken

2.2. The tagged representation

The tagged representation is constructed from the Use Case identifying the start and end points of processes and possible branches (Santos, 2002). Use the following set of symbols to build the target representation:

Symbol	Description
●	Start of a process (Use Case)
⊙	End of a process (Use Case)
→	Start of an event of basic flow of process
↳	Start of an event of alternative flow of process
◇	Start of a conditional event
~ ⁿ	Jump of current iteration to iteration ⁿ
	Sequence of concurrent events

Table 2-1 Tagged representation of Use Case description

It should be noticed that the construction of the tagged representation does not restrict the elicitation of requirements in Use Case. Thus, the initial process of inception can proceed normally without any constraint. The tagged representation (Santos, 2002), applied to the Cash Withdraw Use Case, is showed in the next subsection.

2.3. Tagged Representation for the Cash Withdraw Functionality

Following there are the basic and alternative flows of events for the tagged representation of the Cash Withdraw functionality.

2.3.1. Basic Flow of Events with Alternative Flows

- | |
|---|
| 1 ● Initiate Withdraw – Customer inserts bank’s card in the card reader on the ATM machine; (|
|---|

1.1 ◇ Is the card valid? – Verify Bank Card – The ATM reads the account code from the magnetic strip on the bank’s card and checks if it is an acceptable card;
1.1.1 ↪Send message of invalid card – If the card isn’t an acceptable card, send an appropriate message. ~1.9
1.2 → Enter PIN – The ATM asks for the customer’s PIN code (4 digits); 1.3 ◇ Is PIN Correct? – Verify PIN – The account code and PIN are verified to determine if the account is valid and if the PIN entered is the correct PIN for the account;
1.3.1 ◇ Is the account valid? – Verify account code – The account code is verified;
1.3.1.1 ↪Send message of invalid account – The Banking system returns a code indicating the account could not be found or is not an account which allows withdrawals. ~1.9
1.3.2 ◇ Is the final try? – Checks the number of tries – The customer has three tries to enter the correct PIN;
1.3.2.1 ↪Send message of incorrect PIN – The ATM send an appropriate message. ~1.2
1.3.3 ↪Retain card – on the final try the card is retained, ATM returns to Ready State, and this use case terminates. ~2
1.4 ◇ Have money the ATM? – Select Withdraw – The ATM displays the different alternatives available at this ATM. In this flow, the bank customer always selects “Cash Withdraw”;
1.4.1 ↪Send message ATM out of Money – If the ATM is out of money, the “Cash Withdraw” option will not be available. ~1.4
1.5 ◇ Sufficient funds and does not exceed the daily limit? – Enter amount – The ATM asks for the amount to withdraw. For this flow the customer selects a pre-set amount (\$10, \$20, \$50, or \$100);
1.5.1 ◇ Sufficient funds? – Check sufficient funds – Check if the funds are sufficient;
1.5.1.1 ↪Send message insufficient funds in ATM – The ATM contain insufficient funds to dispense the requested amount. ~1.5
1.5.2 ↪Send message daily maximum withdrawal amount reached – the Banking system returns a code indicating that, including this request for withdrawal, the customer has or will have exceeded the maximum amount allowed in a 24 hour period. ~1.5
1.6 ◇ Authorized withdraw? – Authorization – The ATM initiates the verification process with the Banking System by sending the Card ID, PIN, Amount, and Account information as a transaction. For this flow, the Banking System is online and replies with the authorization to complete the cash withdrawal successfully and updates the account balance accordingly;
1.6.1 ↪Send message Insufficient Funds in Account – The Banking system returns a code indicating the account balance is less than the amount entered. ~1.5
1.7 → Dispense – The Money is dispensed; 1.8 → Receipt – The receipt is printed and dispensed. The ATM also updates the internal log accordingly; 1.9 → Return Card – The Bank Card is returned; 1.10 ◇ Finalized Operation? – Updates log – The ATM updates internal log according the operation;
1.10.1 ↪Cancel Operation – When the log cannot be updated, the ATM enters the “secure mode” in which all functions are suspended. An appropriate alarm is sent to the Bank System to indicate the ATM has suspended operation. ~2
) 2 ● Ends Operation – Use Case ends with the ATM in the Ready State;

2.3.2. Alternative Flows of Events

3 ↪Cancel Operation – The customer can, at any time, decide to terminate the transaction (quit). The transaction is stopped and the card ejected. ~1.9
4 ↪Activate Tilt’s sensor – The ATM contains numerous sensors that monitor different functions, such as power, pressure exerted on the various doors and gates, and motion detectors. If at any time a sensor is activated, an alarm signal is sent to the Police and the ATM enters a “secure mode” in which all functions are suspended until the appropriate re-start / re-initialize actions are taken. ~2

3. Synthesizing a Petri Net for the Requirements

One of the characteristics of Information Systems is the importance of its dynamics, which is, the integration and synchronizing of its business processes components. This is salient in the inception phase as well as in the following phase of elaboration. To validate dynamic frameworks as Petri Nets (Reisig, 1981) have been widely used (Oberweis and Sander, 1996). Besides, Petri Net is a generic schema, and thus suitable to represent abstract specifications of processes (Silva, 1998).

To translate the tagged representation to Elementary Petri Nets we defined a set of basic net elements to fit the tagged elements. Thus we have the translation specified in BNF (Backus Naur Form) (Naur, 1960) as shown in Table 3-1.

Thus, each tagged element is an operation (enterprise activity) and each Use Case a business process. The set of all Use Cases functionalities compose the requirement specification for the whole system.

In Table 3-1 each tagged construct is mapped in a defined token, each of which is related to net element (Santos, 2002). Table 3-2 shows all these elements in its graphical form, where the connection restrictions were already included, that is, it is always possible to represent an information flow by putting together all net elements.

The general structure is guided by the Use Case, in the sense that the resulting net shows the Basic Flow (BF) of events – if no exception occurs – as the core of the net. The element BF defined in Table 3-2 shows the definition of the Basic Flow of events, while the Alternative Flow is defined by the element AF in the same table (Santos, 2002).

Conditional events are represented by the element condition in Table 3-2.

```

<BF>          ::= <initial> { <event> } <final> { <AF> }
<initial>     ::= <label> "●" <title> "-" <description> ";" "("
<label>       ::= <number> { "." <number> }
<number>      ::= <sequential_number>
<title>       ::= <string>
<description> ::= <string>
<string>      ::= <any_caracter> { <any_caracter> }
<event>       ::= [ <eventB> | <eventA> ]
<eventB>      ::= <labelB> [ ( "→" <title> "-" <description> ";" ) |
                           ( "◇" <condition> "-" <title> "-" <description> ";" <eventA>
                           ) ] { <eventB> } |
                 "||" "(" <eventB> { <eventB> } ")" "(" <eventB> { <eventB> } ")" "||" {
                 <eventB> }
<labelB>      ::= <label> "." <number>
<condition>   ::= <string>
<eventA>      ::= <labelA> [ ( "↙" <title> "-" <description> <branch> ) |
                           ( "◇" <condition> "-" <title> "-" <description> ";" <eventA>
                           ) ] { <eventA> }

<labelA>      ::= <labelB> "." <number>
<branch>      ::= "↘" [ <label> | <labelB> ]
<final>       ::= ")" <label> "◎" <title> "-" <description> ";"
<AF>          ::= <eventUA>
<eventUA>     ::= <label> "↘" <title> "-" <description> <branch> { <eventUA> }

```

Where:

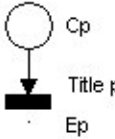
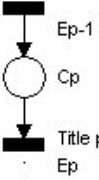
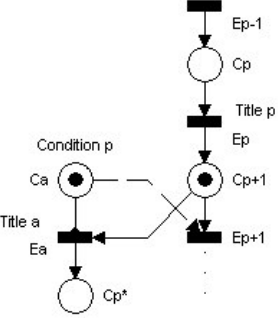
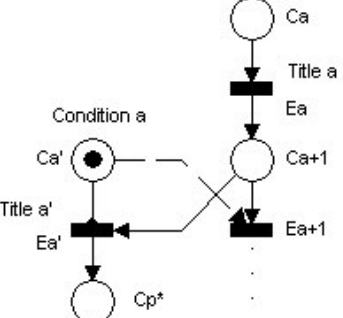
- eventB → Event of Basic flow
- labelB → Event id of Basic flow
- eventA → Alternative Event of Basic flow
- labelA → Alternative Event id of Basic flow
- eventUA → Unconditional Event of Alternative Event
- labelUA → Unconditional Event id of Alternative Event

Table 3-1 Structure definition of the Use Case description

All transformations proposed – from Use Cases to the tagged representation and from these representation to the final representation in Elementary Petri Nets – can be performed by simple algorithms, since are syntactic and do not include any special knowledge about the target system or artifact. Therefore, a simple case tool can be produced to

automate this process. We are developing such tool as an “add on” to be attached to the Rational Suite, since a characteristic of the proposal presented in this paper is to be inside the scope of the RUP (Rational Unified Process).

Applying the mapping of Table 3-2 to the tagged language shown in section 2.3 results in an elementary Petri Net extended to include enabling and inhibitor gates, used to implement the conditional events. The net presented in Figure 3-1 is the provider’s viewpoint of the Cash Withdraw functionality.

Event	BNF notation	Petri net segment
Initial	$p \bullet \text{Title} - \text{Description}; ($	
Basic	$p \rightarrow \text{Title} - \text{Description};$	
Basic conditional and alternative normal ¹	$p \diamond \text{Condition} - \text{Title} - \text{Description};$ $a \hookrightarrow \text{Title} - \text{Description} \sim p^*$	
Alternative conditional and normal	$a \diamond \text{Condition} - \text{Title} - \text{Description};$ $a' \hookrightarrow \text{Title} - \text{Description} \sim p^*$	

¹ Ca represents an external condition, where the arc between the condition Ca and the event Ea is an inhibitor arc, while the arc from Ca to Ep+1 is an enabling arc.

<p>Basic concurrent</p>	<pre> (p₁ → Title - Description; p₂ → Title- Description;) (p₃ → Title - Description; p₄ → Title- Description;) </pre>	
<p>Final</p>	<pre>) p ⊙ Title - Description; </pre>	
<p>Unconditional Alternative</p>	<pre> ai ↪ Title - Description ~P* </pre>	

Table 3-2 Petri net segment corresponding to Use Case description

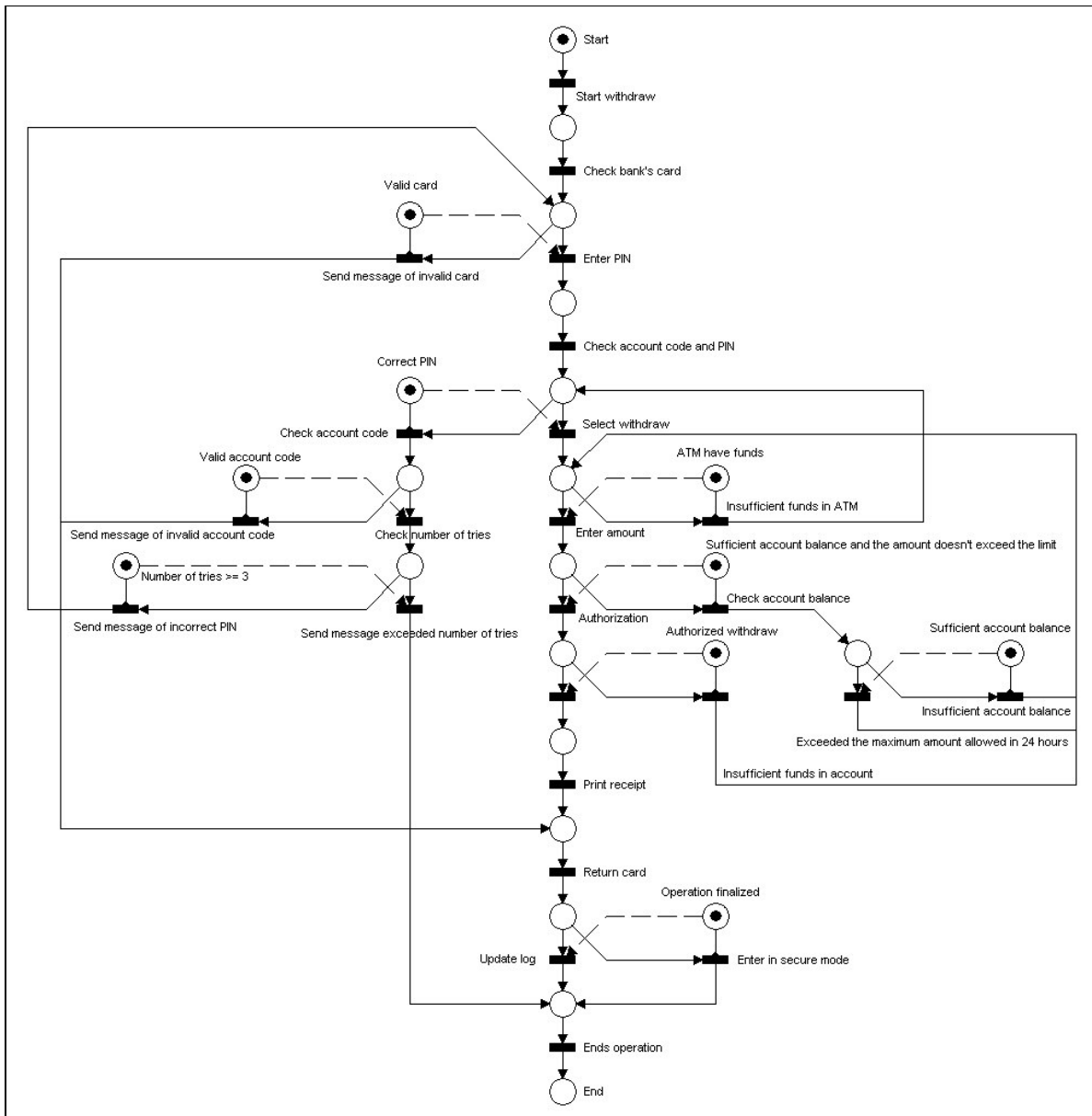


Figure 3-1 ATM Petri Net

4. Analysis and Validation

The net of Fig. 3.1 can be simulated in any token player based in Petri Nets (since it is based in Elementary Nets with extension to include gates). Animation of the token player can be used to validate the process of Cash Withdraw visually. The conditional points are denoted by simple places which marking can stand for uncontrollable events such as entering with the wrong password in the ATM section. There is no way to prevent the “value” of this condition and the praxis would be to evaluate the evolution of the process for each possibility.

That means changing the mark in all those places involved in conditional events, what can be done with any reasonable token player tool.

The proposed approach opens the possibility to represent and validate the merging of different viewpoints (Leite and Freeman, 1991)(Kotonya and Somerville, 1998). For instance, Fig. 4-1 shows a viewpoint of the Cash Withdraw not from the developer viewpoint as we are implicitly doing in the previous sections but from an end user viewpoint, who is more concerned with the main interaction to obtain cash.

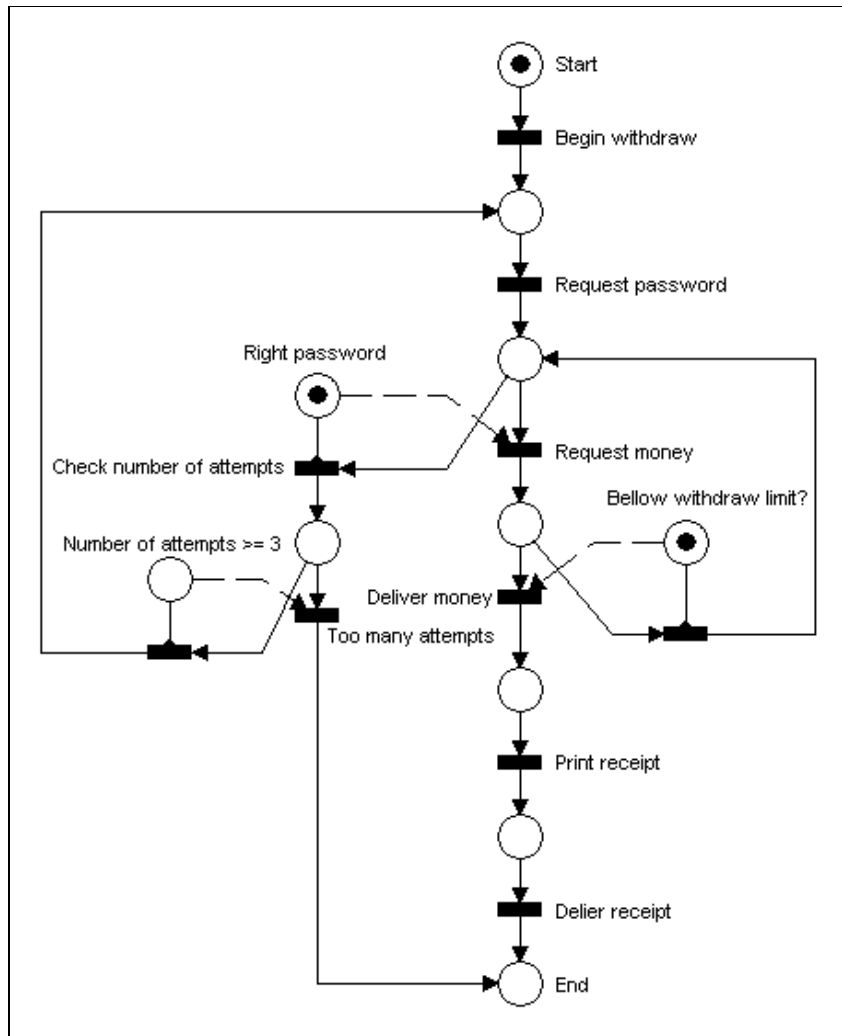


Figure 4-1 Final user viewpoint of Cash Withdraw

By visual inspection is possible to see, in Fig. 4-1, that the net resulting from the user viewpoint is isomorphic to a sub-net of the Fig. 3-1 that represents the developer viewpoint. Thus, the current approach can be used as a guide in the merging of viewpoints, by identifying similarities and collapsing the nets that represent converging viewpoints.

Again we are thinking in a direct and informal matching. Automating that matching with a case tool can be possible due to the performance of approximated algorithms to decide about isomorphism between nets. However, the complete process is not in the scope of this article.

Combinations of conditions can then be tested and validated as the reaction of the system to (conditional) events that are not controllable but that can be predicted.

The complete system can be validated as a set of business processes (BP). If these processes are all independent, the validation is the result of the analysis for each one of the BP components. However, if we have an integrated system, that is, where the BPs are all dependent, then we will need some integrating places (places that belong to both BP or places that can be collapsed from two or more BPs to result in one unique net). Again a token player tool can be used to validate the general behavior.

In both cases the validation process is based on the animation of the token player, that is, in a visual process of following the schematized behavior of the Petri net, once an initial state is selected. Thus, the method of validation is very suitable for small systems or at least for those where the resulting graph can be easily visualized². These kind of system can also be interpreted, that is, it is not difficult to attribute a semantic meaning for some or all of the places and events, such as “request money”, “check the number of attempts”, etc. and construct an analogy between the evolution of the schema and a real process. Unfortunately that is not the case for larger systems.

² Larger systems can also be treated using a folded or high level net and basing the analysis in property evaluation instead of visual token player. In this work we choose to address the first case.

To larger systems the recommended approach would be to analyze the properties of the nets, principally those directly deduced from its state equation. The state equation is an iterative equation that denotes all reachable states of the net, once defined an initial one. Thus, a generic state can be given by

$$M_i = M_0 + A^T \sum_0^{i-1} \sigma_i$$

where A is called the incidence matrix of the net and denotes the existence of arcs between the passive elements (conditions) and the active ones (events).

Several properties can be deduced from this equation including necessary conditions for the existence of deadlock, that in the case of requirement validation would mean an impossible process for the selected value of the uncontrollable conditions.

However, at least theoretically the proposed method above could not admit an analytical or numerical property analysis since the use of internal gates connected to places with persistent tokens makes the incidence matrix inconsistent.

On the other hand there are other possibilities if another kind of net is used. For instance, the GHENeSys (General Hierarchical Enhanced Net System)(del Foyo and Silva, 2003) is an extended object oriented net system where a specific place is used to represent uncontrolled conditions: the pseudobox. Also this place is not considered as belonging to the core net but as the relationship between the net system and its environment. Thus it is possible to recover a state equation for it even with the presence of gates.

Also, pseudoboxes can be used as collapsed places to connect components in a more generic schema. The same approach presented here could be more suitably represented in the GHENeSys system.

5. Conclusions and further work

We presented a very simple schema for validating requirements, transforming Use Case in Net systems. The approach was inspired in the tedious and informal work that has to be done for information systems where a large set of conditional branches.

For small and medium size systems the approach has worked fine with the advantage of being adherent to the RUP without inserting any new constraint to the inception phase, but, on the other hand contributing for a disciplined elaboration of specifications.

A Case tool is being developed as an “add on” to the RequisitePro (Rational Suite) tool as an alternative to validate requirements. However that tool would work fine only to small and medium size systems. The reason for that was already explained in the previous section.

As a further development we are trying to use another net framework called GHENeSys, which solve the problem of representing formally the gates used to model conditional events.

6. References

- Vernadat, F., Enterprise Modeling and Integration: Principles and Applications, Chapman & Hall, 1996
- Rational Software Corporation. Rational Unified Process: version 2001A.04.00. s.l.: s.ed., 2001
- Anschuetz, H. “A tool for design and simulation of Petri nets: HPSim version 1.1”, s.l.: s.ed., 2001, Available in: <[http:// home.t-online.de/home/henryk.a/petrinet/e/sim_form.htm](http://home.t-online.de/home/henryk.a/petrinet/e/sim_form.htm)>. Accessed in: 13 jul. 2002
- Reisig, W., Introduction to Petri Nets, Springer Verlag, 1981
- Oberweis, A. and Sander, P., “Information system behavior by high level Petri nets”, ACM Transactions on Information Systems, v.14, n.4, October, 1996
- Silva, J.R., “Interactive Design of Integrated Systems”, In Intelligent Systems for Manufacturing: Multi-agent Systems and Virtual Organizations, Kluwer Academic Pub., p. 567-578, 1998
- Naur, Peter (ed.), “Revised report on the algorithmic language ALGOL 60”, Communications of the ACM, v. 3, n.5, 1960, p. 299-314
- del Foyo, P.M.G. and Silva, J.R., “Towards a Unified View of Petri Nets and Object Oriented Modeling”, to appear in 17th International Congress of Mechanical Engineering, São Paulo, Brazil, 2003
- Santos, E.A., Systems Requirements Verification by Using Petri Nets, Master thesis, Escola Politécnica da USP, in Portuguese, December, 2002
- Leite, J.C.P., Freeman, P.A., “Requirements validation through viewpoint resolution”, IEEE Transaction on Software Engineering, v.17, n.12, December, 1991, p. 1253-1269
- Kotonya, G. and Somerville, I., Requirements Engineering, John Wiley & Sons, 1998