University of Valladolid
Spain

Applying eODL and SDL-Patterns for Developing TMN Managed Systems

Manuel Rodríguez
Margarita de Cabo

Index of contents

- TMN architecture
- Proposal: Using eODL and SDL-pattern in TMN CORBA-based
  - Step one: Mapping IDL into eODL
  - Step two: Mapping eODL into SDL
  - Step three: Adding behavior
- Example: a generic MO with ARC feature
- Conclusions

TMN (I)

- TMN is a standard infrastructure for managing telecommunication networks
- TMN architecture consists of three parts:
  - Functional architecture (functional blocks)
  - Information architecture (information modeling)
  - Physical architecture (physical blocks)

TMN (II)

- It is distributed in essence → implementation on a DPE is desired
- A framework for developing systems on DPEs exists (ITU-T Rec. X.780)
- Main drawbacks:
  - The target implementation platform has to be CORBA-based
  - It does not include formally specified behavior
- A component-oriented and technology-independent development method would be desired

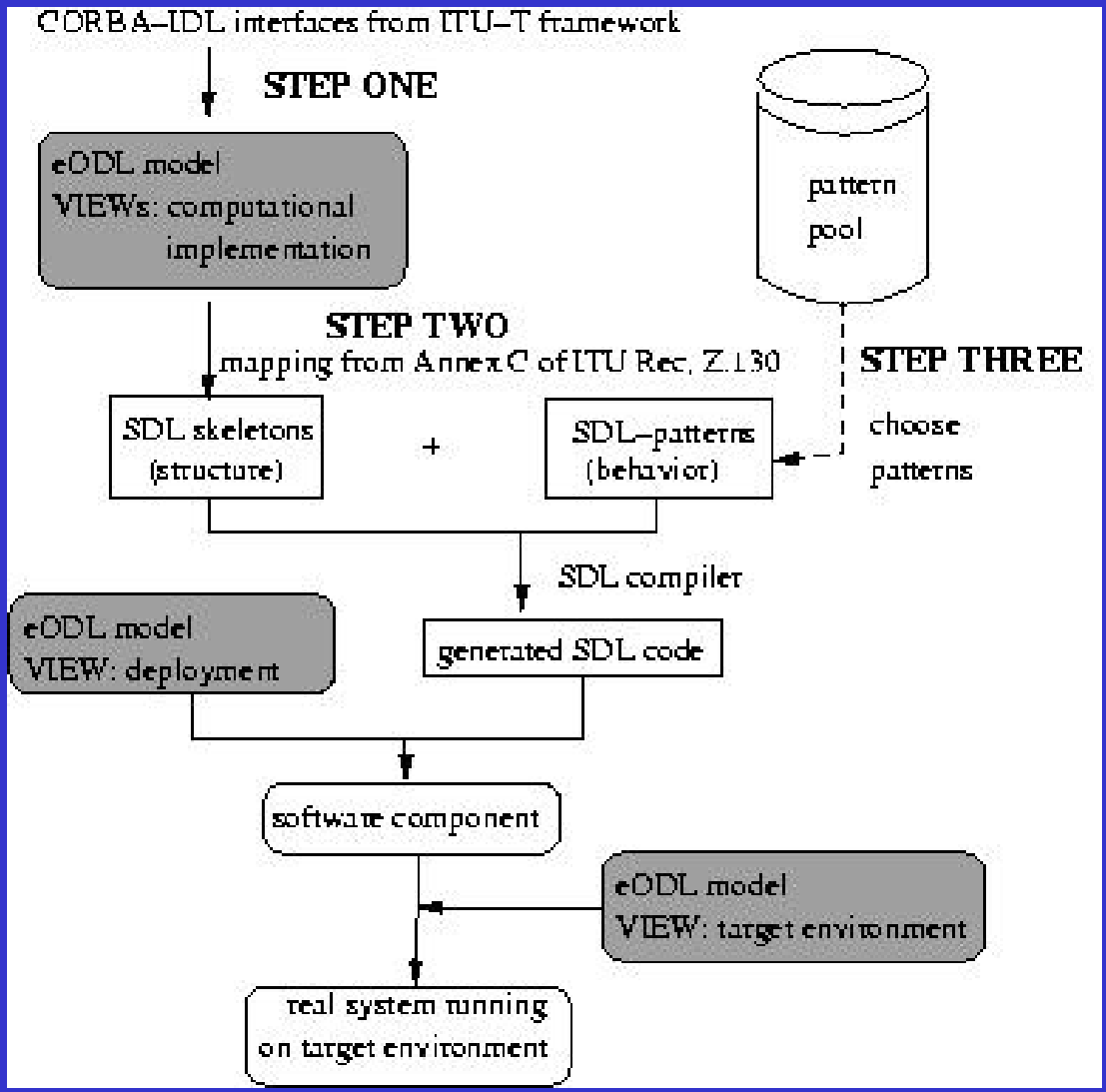
Proposal (I)

- Enhancing CORBA-based TMN framework by means of eODL and SDL-2000 patterns
  - Using eODL instead of CORBA IDL
    - Advantages
      - Model driven approach, several mappings to developing languages
      - Any target platform can be chosen
      - Several views of the system can be described
    - Drawbacks
      - Lack of constructions for behavior definition

Proposal (II)

- Using SDL-patterns
  - Each pattern is a schema of a well-known solution and the rules to apply it
  - Main advantages
    - SDL-based → same benefits
    - Improve reuse and sharing of expertise
    - Development is less time-consuming
  - A new notation is needed: PA-SDL

Steps of the proposal



Step 1: Mapping IDL into eODL

- Mapping of CORBA-IDL elements
  - Straightforward:
    - CORBA-IDL is a subset of eODL
    - Its elements can be used without change
- New elements have to be added
  - CO (computational view)
    - One CO for every class in ITU-T information model
    - Artifacts (implementation view)
    - One artifact for every IDL interface
  - Deployment and target environment views can not be extracted from IDL

Step 2: Mapping eODL into SDL

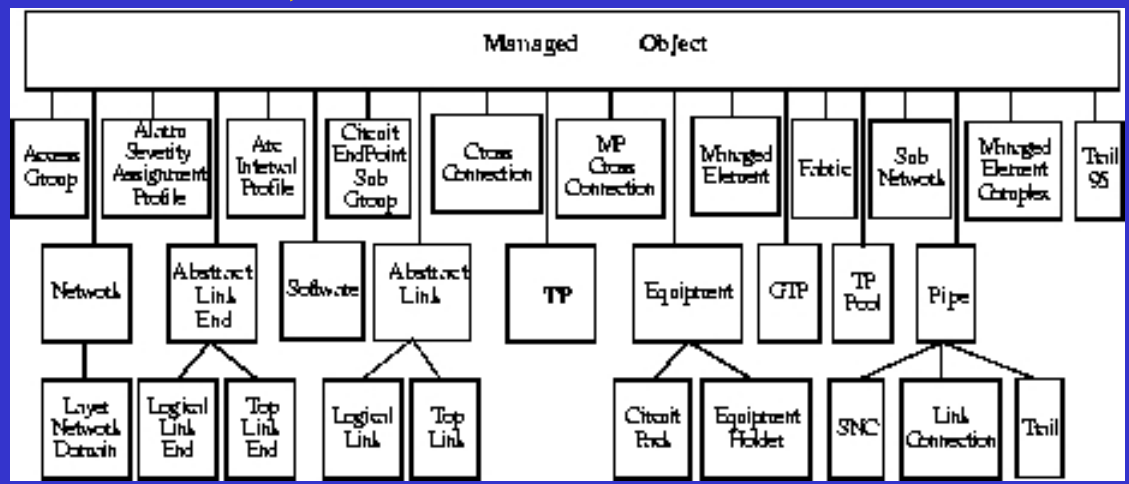
- Following the guidelines in Annex C of Z.130 Rec.
- Result: SDL skeletons (structure)
  - *interface* package
  - *definition* package
    - block type
      - process types
- Behavior has to be added later

Step 3: Adding behavior

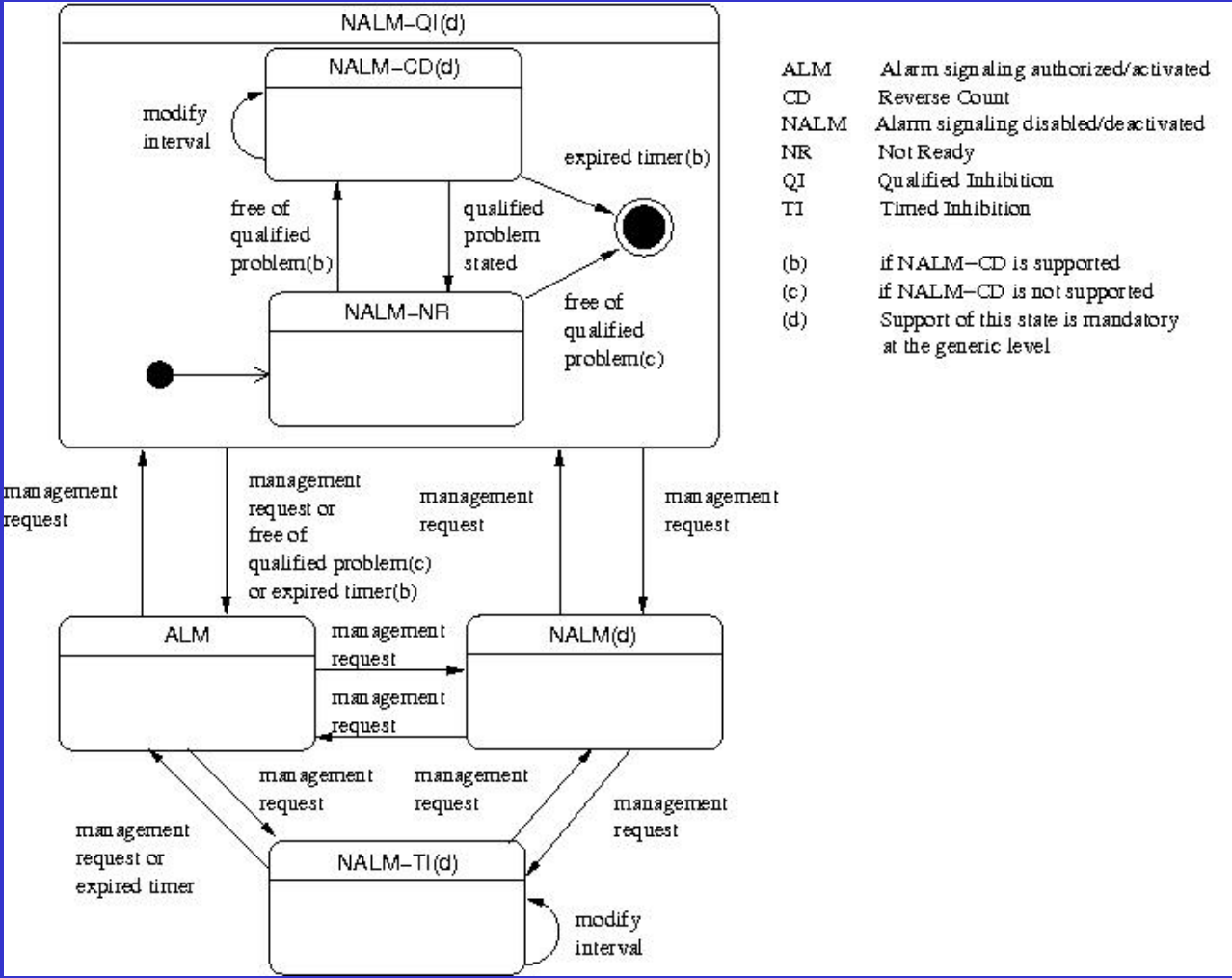
- Choosing SDL-patterns
  - From a pattern-pool if exists
  - Develop a new one, if there is not an adequate pattern
- Find the context for the pattern
- Apply the rules in pattern template to obtain a correct SDL-2000 description

Example: A generic MO with ARC feature

- Managed Object (MO) is the most simple object in the network-element hierarchy proposed by ITU (M.3120 Rec.)



Example: ARC States (Rec. M.3120)



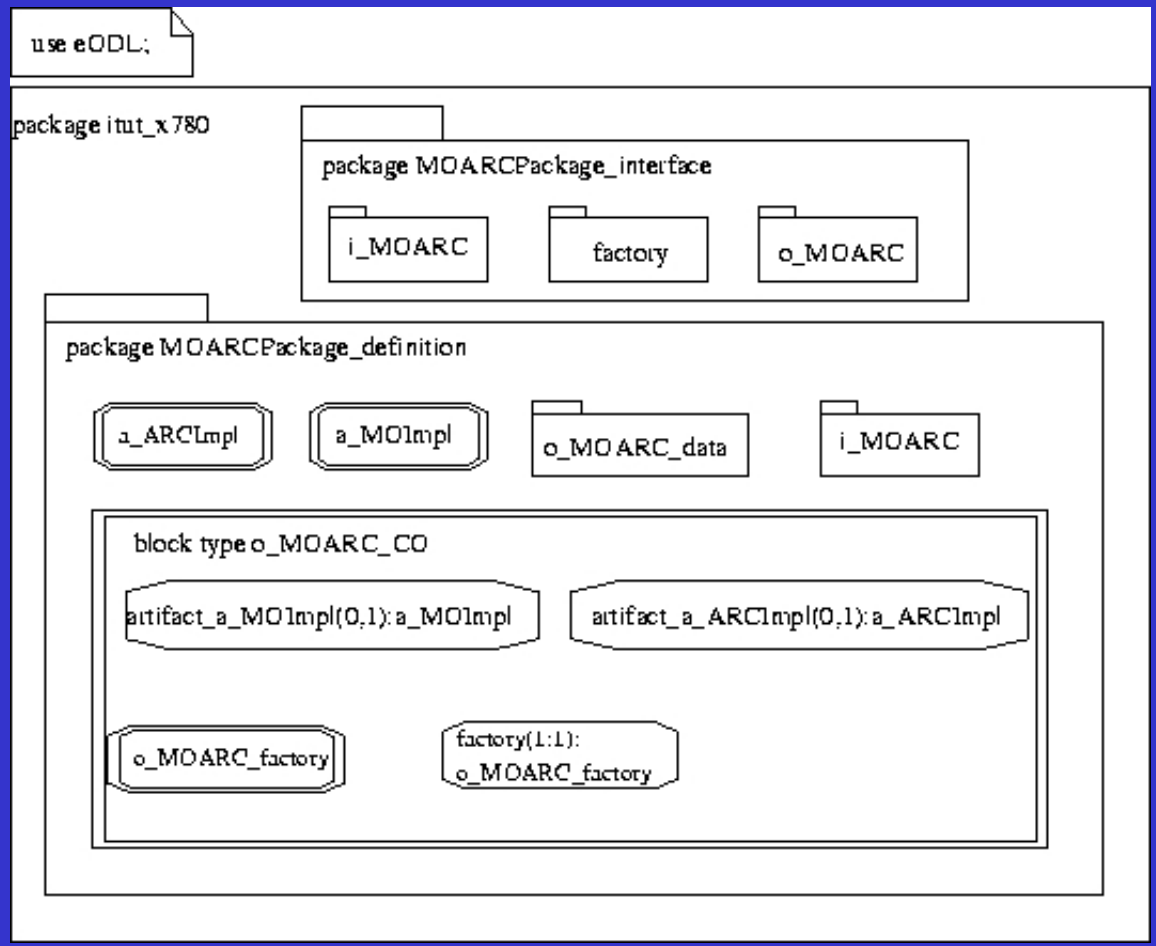
Example: eODL definition for MO with ARC (I)

```
module itut_x780{
    interface i_MO{
        NameType nameGet () raises
            (ApplicationError); …
    };
    artefact a_MOImpl {
        nameGet implements supply
        i_MO::nameGet …
    };
```

Example: eODL definition for MO with ARC (II)

```
interface i_ARC{
        boolean arcControl() raises
        (ApplicationError,NOarcPackage); …
        };
artefact a_ARCImpl{
        arcControl implements supply i_ARC::arcControl; …
        };
CO o_MOARC{
        supports i_MO, i_ARC;
        provide i_MO mo; provide i_ARC arc;
        /*requires nothing*/
        implemented by a_MOImpl with Singleton, a_ARCImpl with
    Singleton;
    };
```

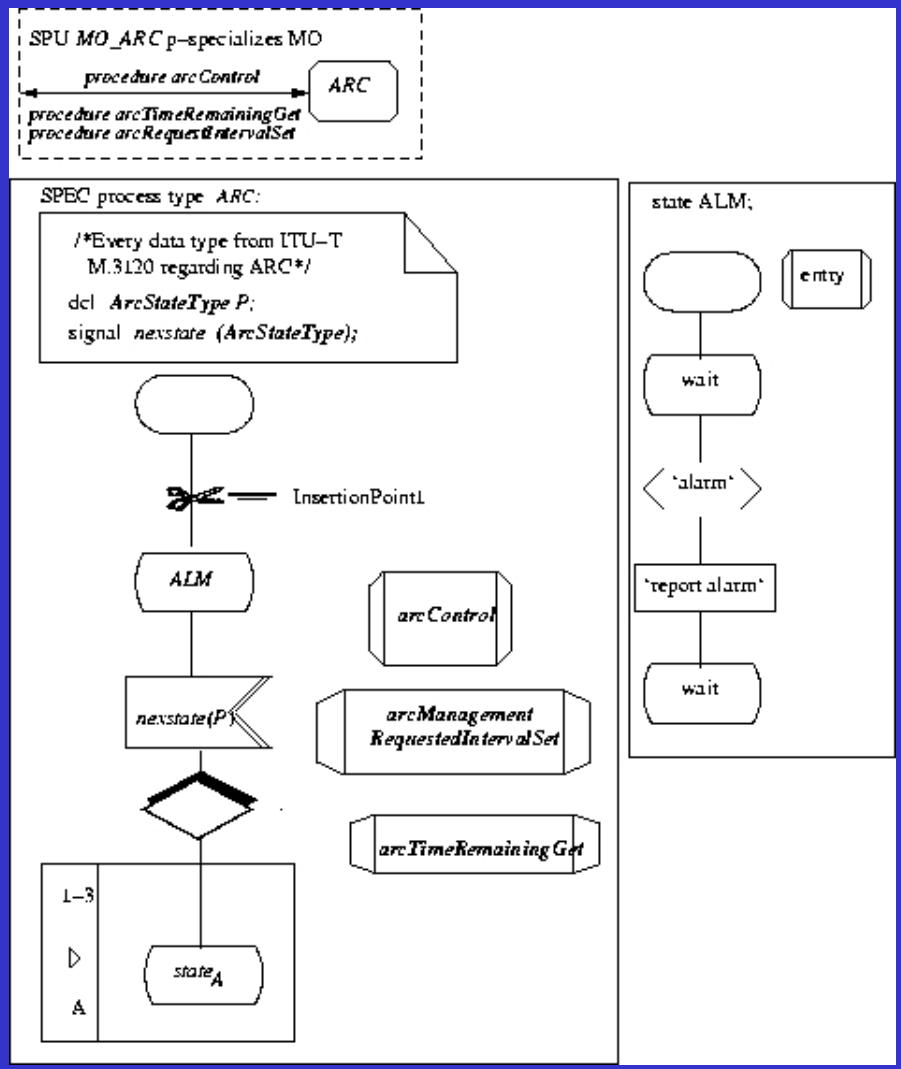
Example: SDL skeleton for MO with ARC



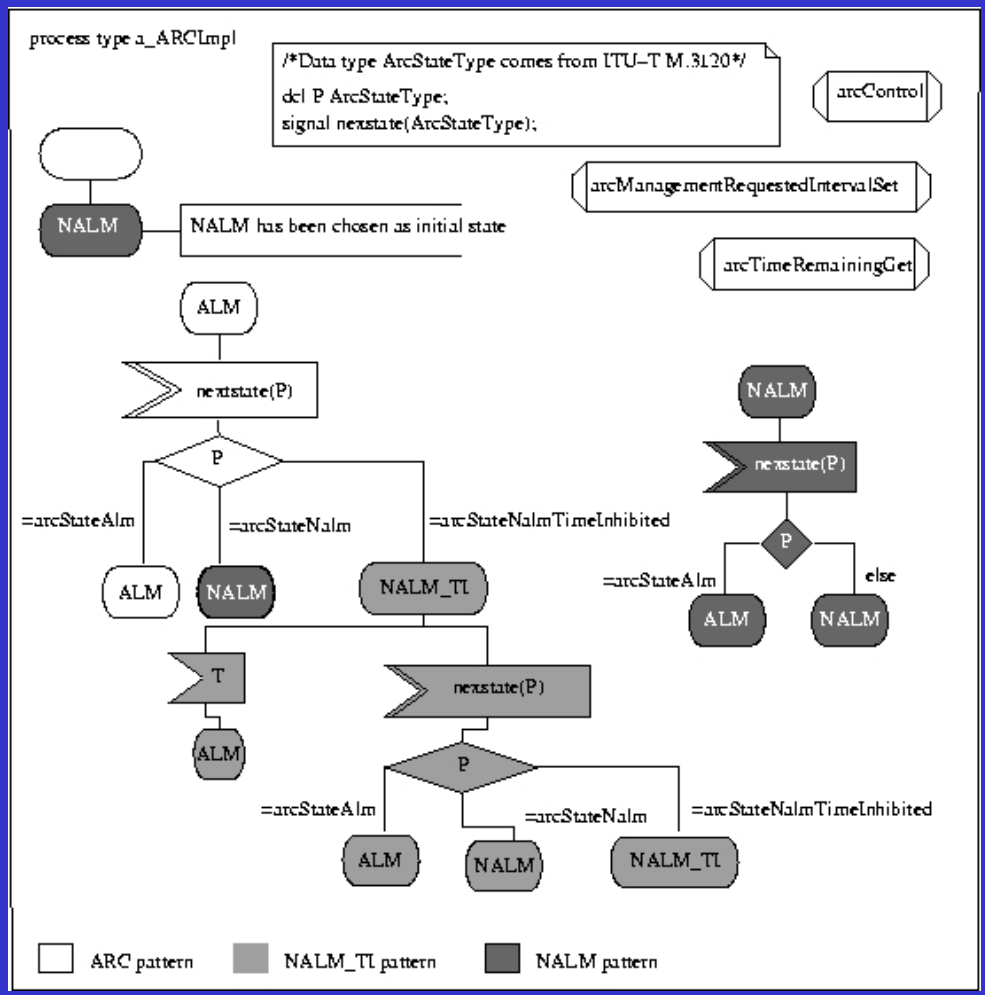
Example: Adding behavior

- Adding ARC feature
- Using ARC pattern plus NALM-TI and NALM patterns
  - ARC pattern is the basic one for ARC feature
  - NALM-TI and NALM are optional
  - There are two other optional patterns:
    - NALM-QI and NALM-CD
  - There must be at least two different patterns one being ALM

Example: ARC pattern (excerpt)



Example: ARC with ALM, NALM and NALM-TI



Conclusions (I)

- eODL and SDL-patterns complement each other for obtaining enhanced models
- eODL gives a well-defined metamodel
  - easy translation into different languages
  - several views of the same model
- SDL-patterns allow:
  - Formal specification of behavior
  - Reusing and sharing of gained expertise in other projects

Conclusions (II)

- Lack of tool support
  - Too few SDL-2000 features supported in CASE tools
  - Tool support for SDL pattern-based design needed
- Need of new options in eODL to SDL mapping
  - Concurrent execution of artifacts implementing a CO