“SPT - The SDL Pattern Tool”

Jörg Dorsch

Anders Ek

Reinhard Gotzhein

Fourth SDL and MSC Workshop

Ottawa, Canada

June 2-4, 2004

Topics

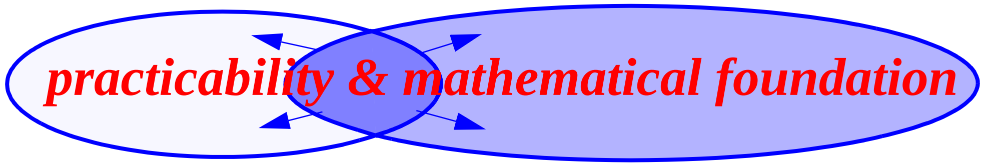
- Survey of the SDL Pattern Approach
- Tool Support
- Pattern based Development with SPT
- Conclusions and Perspectives

Survey of the SDL Pattern Approach

- SDL Patterns -

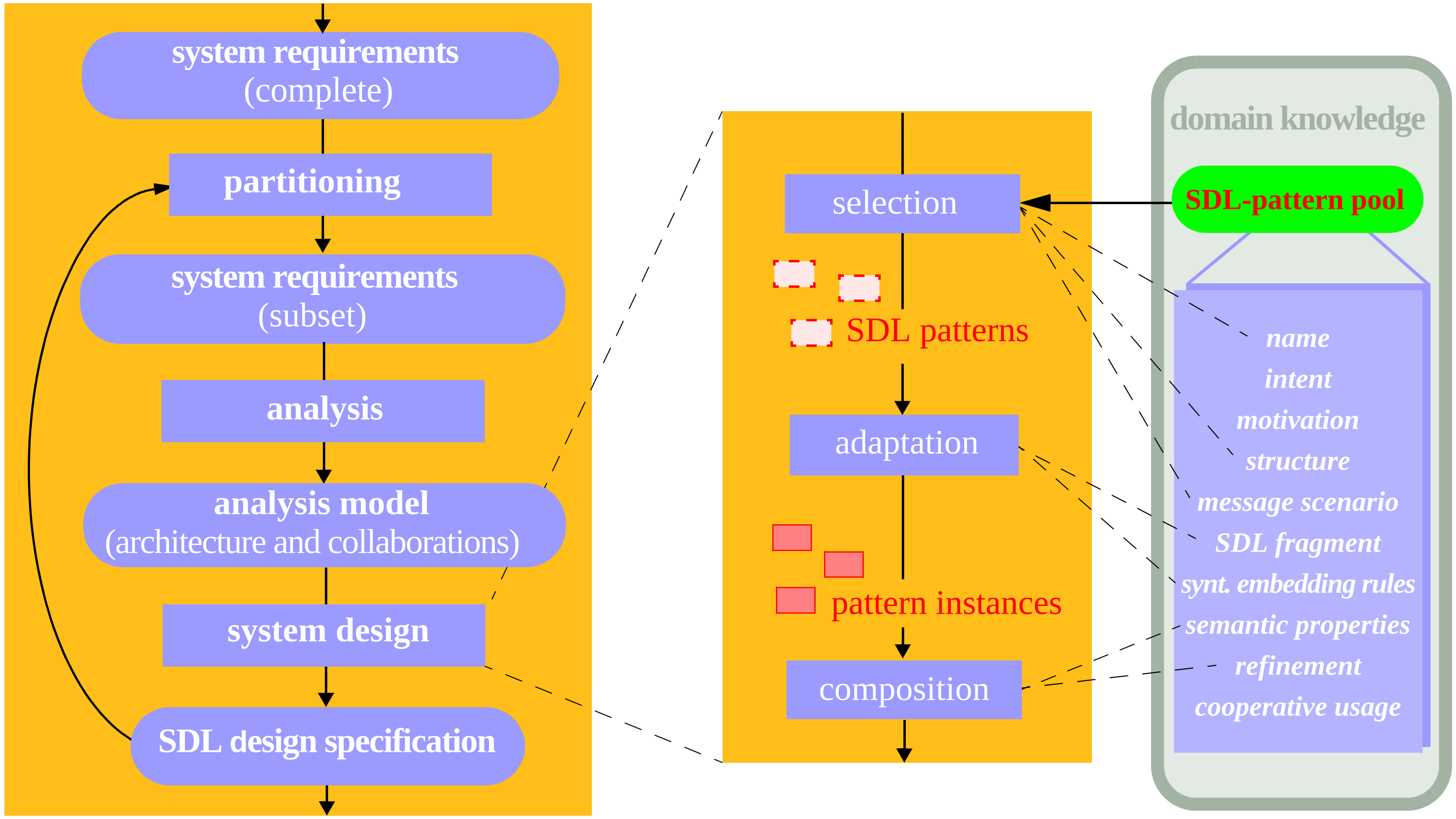
An **SDL Pattern** is a reusable software artifact that represents a generic solution for a recurring design problem with SDL as design language. An SDL pattern is selected from a pattern pool, adapted to a specific description context, and embedded into that context.

+ reduced development effort
+ quality improvements
+ abstract developer vocabulary
+ orthogonal documentation
- informal definition/application ➪ formal design language (SDL) supports the precise description of
  + pattern definitions
  + pattern applications
  + effects on the embedding context



Survey of the SDL Pattern Approach

- Process Model -



Survey of the SDL Pattern Approach

- The SDL Pattern Pool -

Architecture Patterns	Interaction Patterns	Control Patterns	Management Patterns	Interfacing Patterns
• **SERVICE ARCHITECTURE**	• ASYNCHRONOUSNOTIFICATION	• LOSSCONTROL	• EXCEPTIONHANDLING	• CODEX
• SERVICE PROVIDER REFINEMENT	• **SYNCHRONOUSINQUIRY**	• DUPLICATEIGNORE	• CONDITIONEDINPUT	• CAN-INTERFACING
• CLIENTSERVER	• SYNCHRONOUSGROUPINQUIRY	• DUPLICATEHANDLE	• DELAYEDINPUT	• UART-INTERFACING
• CLIENTSERVER REFINEMENT	• MULTIPLEINQUIRY	• RECEIVEFILTER	• MULTIPLEPRIORITY INPUTS	• TCP-INTERFACING
• ...	• PERIODICALUPDATE	• ...	• BUFFERMANAGEMENT	• UDP-INTERFACING
	• GETSTATUSINFORMATION		• ...	• ...
	• MULTIPLENOTIFICATION			
	• CYCLICNOTIFICATION			
	• THREEWAYHANDSHAKE			
	• ...			

Tool Support

- Areas of Tool Support -

☞ **definition** of SDL patterns

- graphical tool support (e.g., for PA-SDL)
- pattern application script generation from pattern definitions (skeletons, traceability)

☞ **application** of SDL patterns

- pattern selection (based on developer dialogues and the analysis model)
- design context identification (based on developer dialogues and the analysis model)
- pattern adaptation (based on developer dialogues and the analysis model)
- embedding of the pattern instance (automatic)

☞ **documentation** of SDL pattern applications

- pattern application log
- documentation of pattern application history
- documentation of the reuse and collaboration structure of the design
- documentation of abstract design decisions
- navigation support (e.g., between pattern instance fragments)

Tool Support

- Implementation of SPT -

☞ implementation status

- prototype tool
- fully integrated with Telelogic Tau Developer TTD G2
- pattern specific Tcl scripts, based on a tool developer API of TTD G2
- support of 3 SDL patterns - selection, adaptation, composition

☞ code structure

• include part	basic dialogue support and basic commands (API of TTD G2)
• menu part	code for the menu bar item and the pull-down menu
• procedure part	common procedures for menus, dialogues and pattern scripts
• tool scripts	e.g., arrangement of symbols, pattern log
• pattern scripts	pattern application

Pattern based Development with SPT

- Informal Requirements “InRes Service” -

- connection-oriented communication service
- reliable exchange of messages between two users
- preservation of the sending order (FIFO)
- asymmetrical service
  - initiator: requests connection and sends data
  - responder: accepts/refuses connection, clears connection, and receives data
- service functionalities

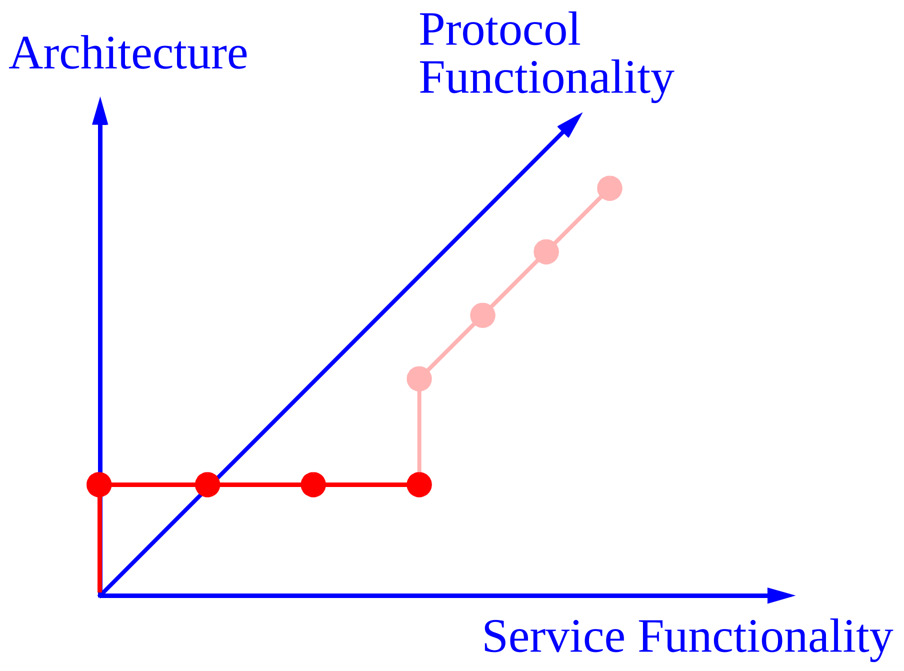
- connection setup (successful):	ICONreq - ICONind - ICONresp - ICONconf
- connection setup (unsuccessful):	ICONreq - ICONind - IDISreq - IDISind
- data transfer:	IDATreq(isdu) - IDATind(isdu)
- connection release:	IDISreq - IDISind

- addressing: implicit (determined by architecture)
- acknowledgement: no
- flow control: no
- reference for the comparison of different approaches to the specification and verification of services (no real service)

Pattern based Development with SPT

- Partitioning: InRes System -

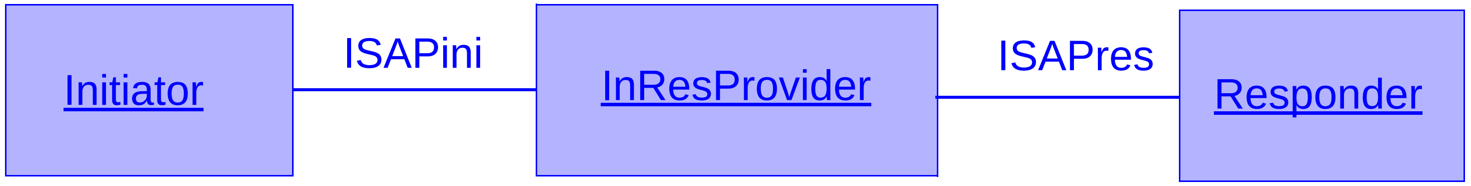
Step I:	**System Architecture**
Step II:	**Connection Setup Phase**
Step III:	Connection Release Phase
Step IV:	Data Transfer Phase
Step V:	Service Provider Refinement
Step VI:	Coding and Decoding of Messages
Step VII:	Flow Control
Step VIII:	Error Control



Pattern based Development with SPT

- InRes System - Step I: Analysis Model -

- Structure

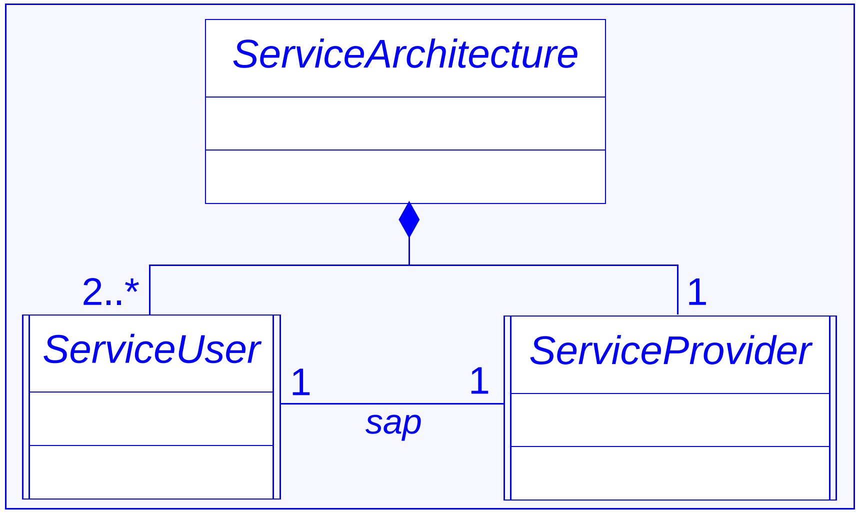


- Message Scenario
  - none

Pattern based Development with SPT

- SDL Pattern SERVICEARCHITECTURE (1) -

- **Name**: SERVICEARCHITECTURE
- **Structure**:
  - The following UML class diagram shows the graphical representation of the structural aspects of the pattern's solution. A service architecture consists of a service provider and 2 or more service users, associated by a service access point:

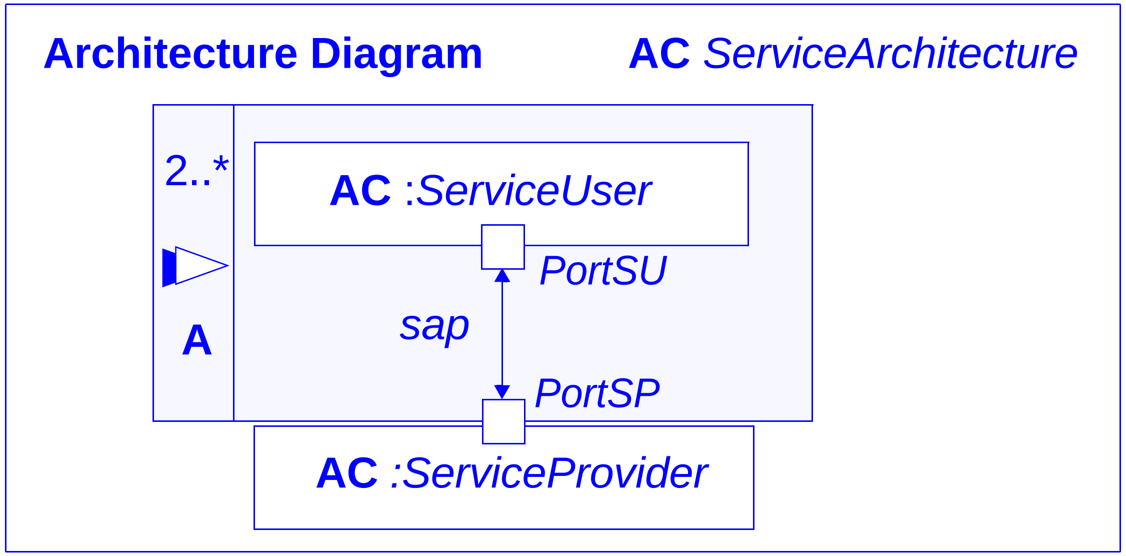


- **Message Scenario**:
  - The pattern addresses structural aspects only and leaves out the behavior of the components. Hence, there are no message scenarios at this stage.

Pattern based Development with SPT

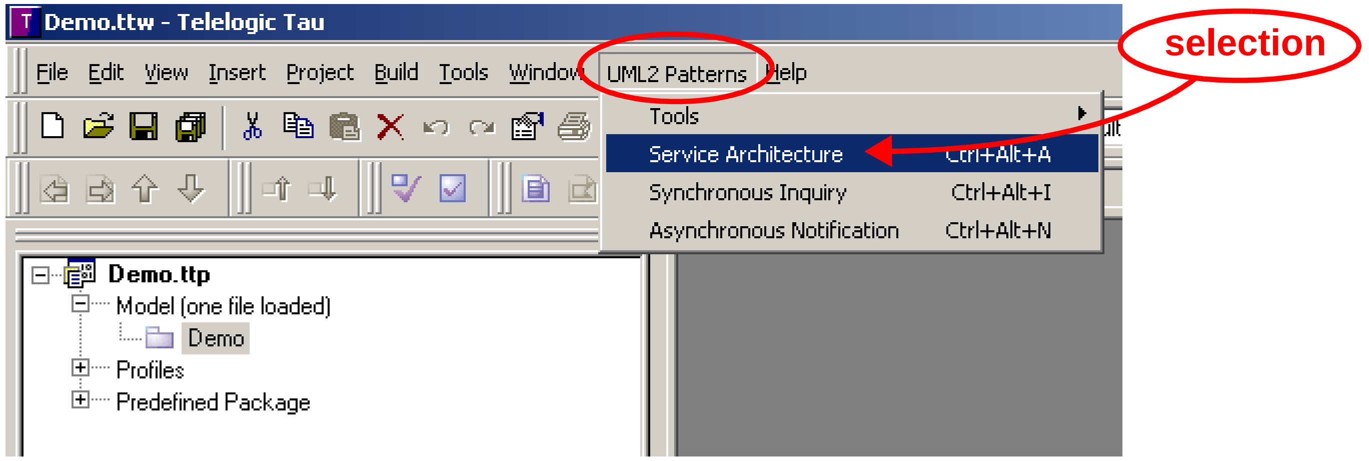
- SDL Pattern SERVICEARCHITECTURE (2) -

- **Design Fragment**:
  - The pattern introduces new design elements that are added to the context specification. In particular, one service provider and 2 or more service users are introduced, connected by bidirectional channels (or signal routes). No signals are associated with these channels at this point, as only structural aspects are addressed by this pattern.



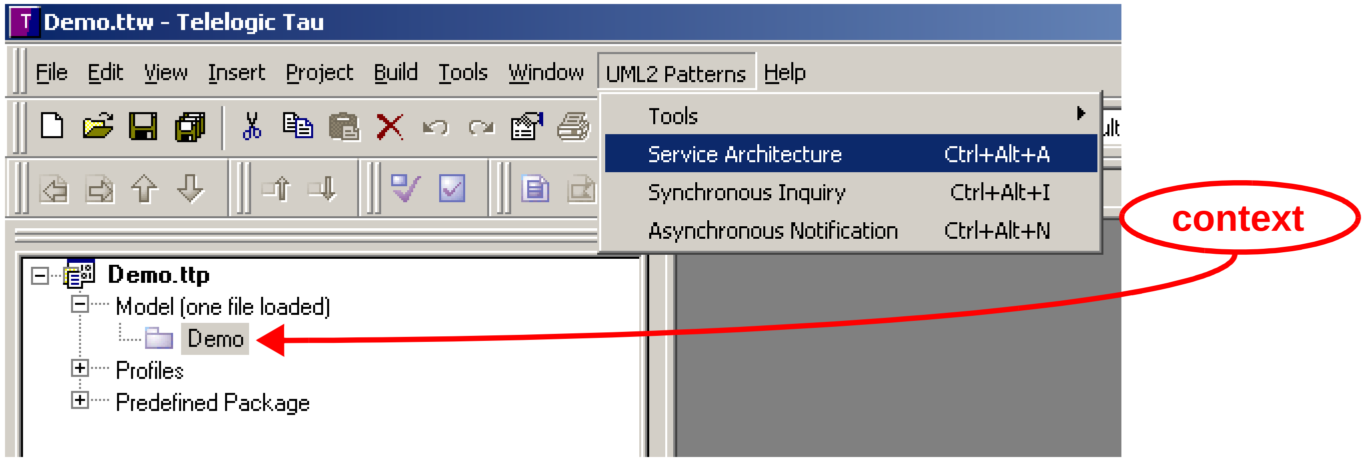
Pattern based Development with SPT

- Pattern Selection -


Demo.ttw - Telelogic Tau
File Edit View Insert Project Build Tools Window UML2 Patterns Help
Tools
Service Architecture Ctrl+Alt+A
Synchronous Inquiry Ctrl+Alt+I
Asynchronous Notification Ctrl+Alt+N
Demo.ttp
Model (one file loaded)
Demo
Profiles
Predefined Package
selection

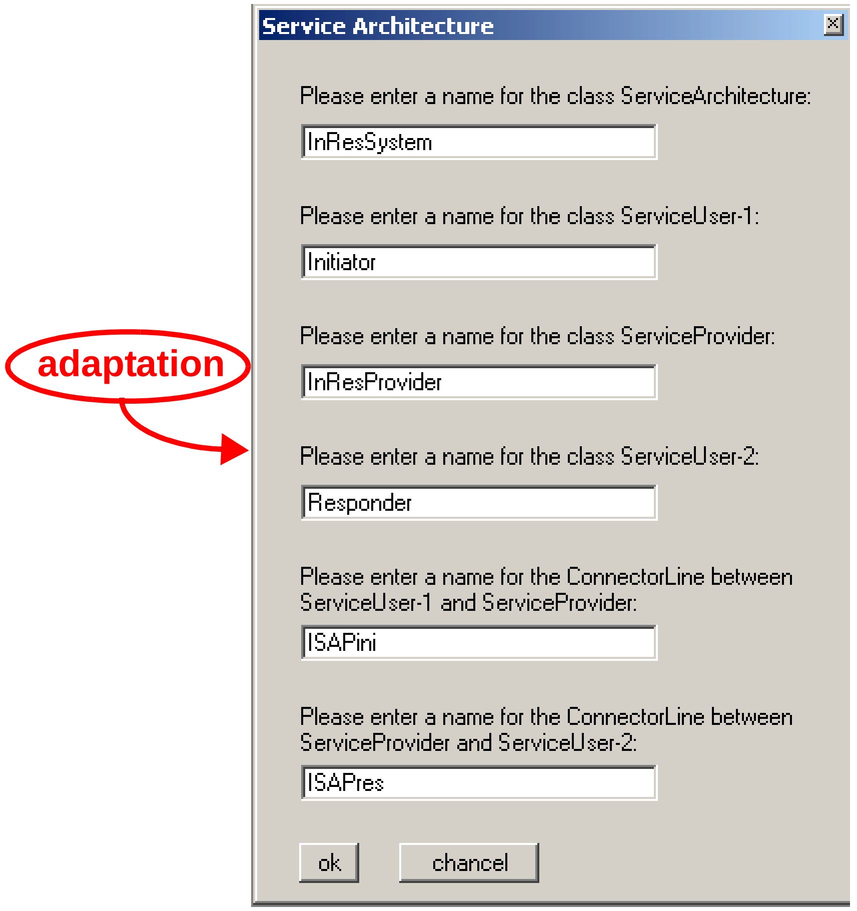
Pattern based Development with SPT

- Context Identification -


Demo.ttw - Telelogic Tau
File Edit View Insert Project Build Tools Window UML2 Patterns Help
Tools
Service Architecture Ctrl+Alt+A
Synchronous Inquiry Ctrl+Alt+I
Asynchronous Notification Ctrl+Alt+N
Demo.ttp
Model (one file loaded)
Demo
Profiles
Predefined Package
context

Pattern based Development with SPT

- Pattern Adaptation -



Please enter a name for the class ServiceArchitecture:
InResSystem

Please enter a name for the class ServiceUser-1:
Initiator

Please enter a name for the class ServiceProvider:
InResProvider

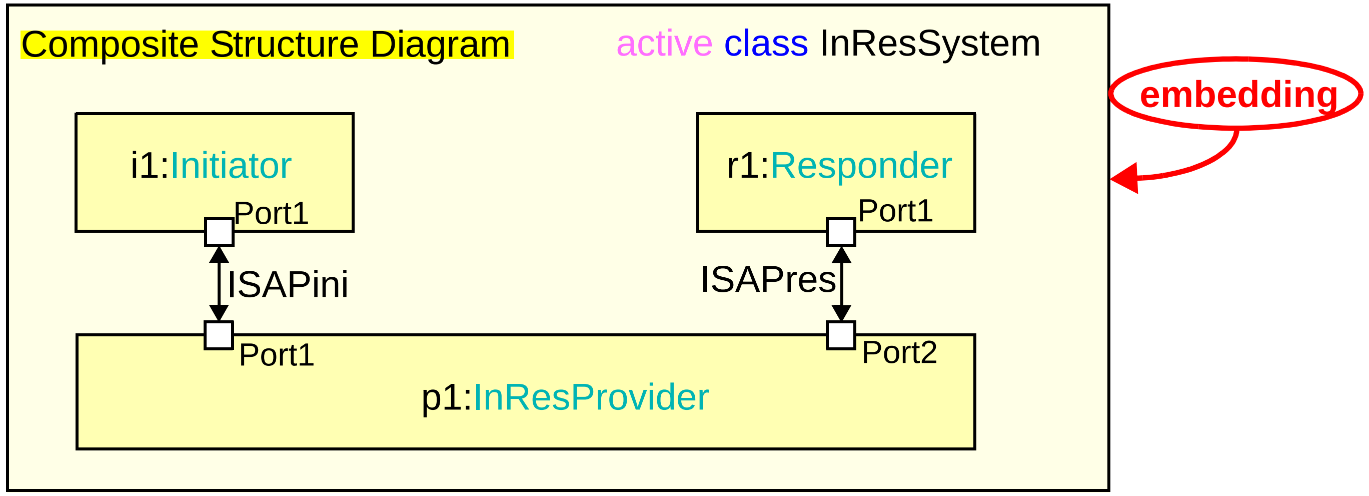
Please enter a name for the class ServiceUser-2:
Responder

Please enter a name for the ConnectorLine between ServiceUser-1 and ServiceProvider:
ISAPini

Please enter a name for the ConnectorLine between ServiceProvider and ServiceUser-2:
ISAPres

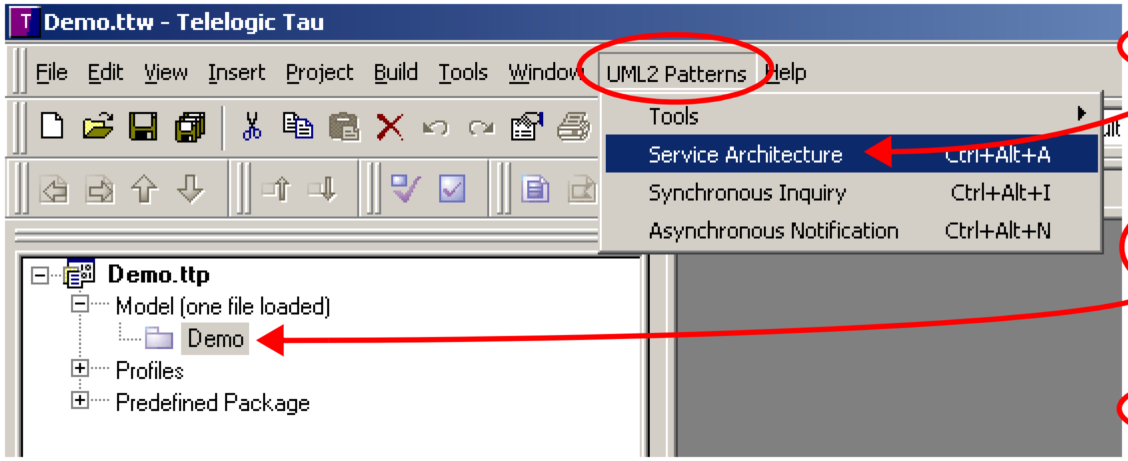
Pattern based Development with SPT

- Pattern Embedding -



Pattern based Development with SPT

- SPT - The SDL Pattern Tool -

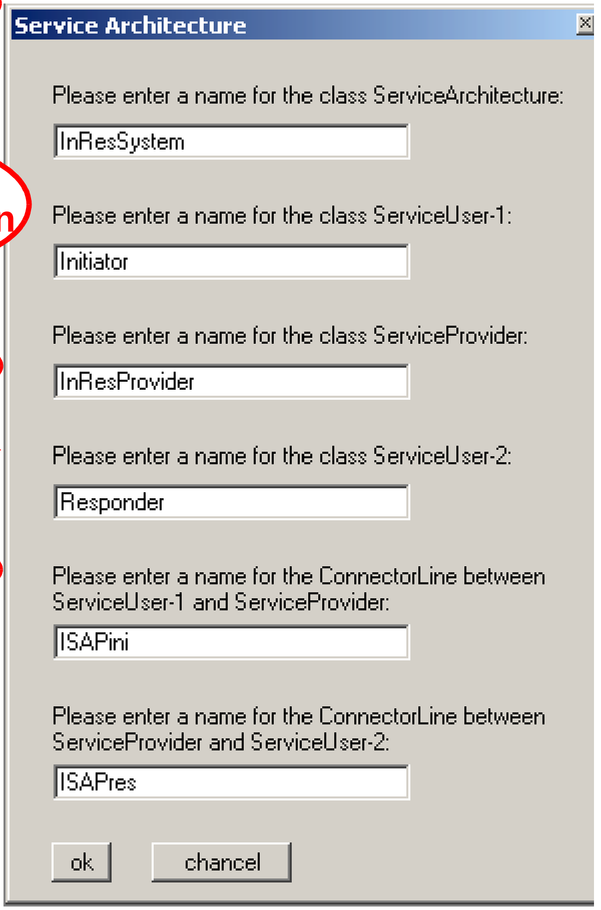

Demo.ttw - Telelogic Tau
File Edit View Insert Project Build Tools Window UML2 Patterns Help
Tools
Service Architecture Ctrl+Alt+A
Synchronous Inquiry Ctrl+Alt+I
Asynchronous Notification Ctrl+Alt+N
Demo.ttp
Model (one file loaded)
Demo
Profiles
Predefined Package

selection

context identification

adaptation

embedding


Service Architecture

Please enter a name for the class ServiceArchitecture: InResSystem

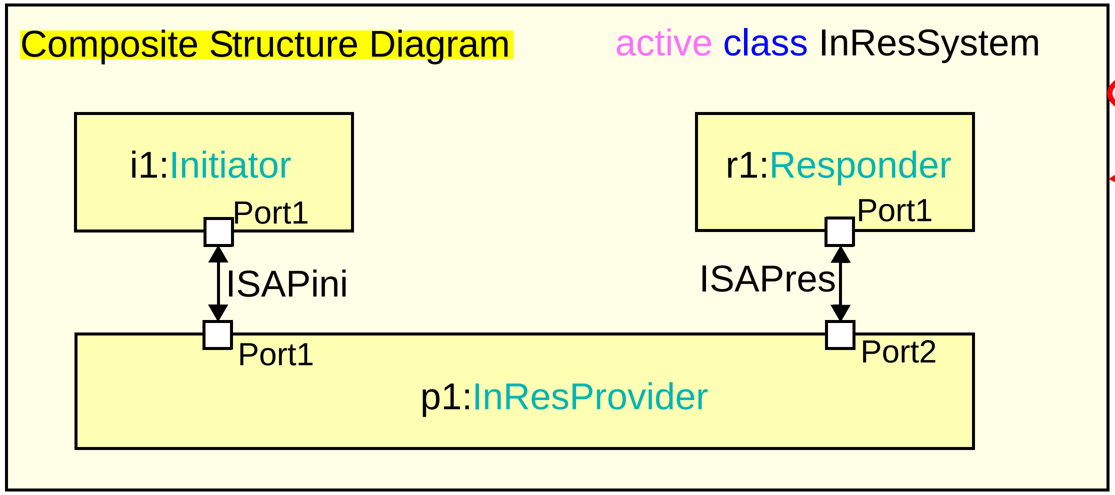
Please enter a name for the class ServiceUser-1: Initiator

Please enter a name for the class ServiceProvider: InResProvider

Please enter a name for the class ServiceUser-2: Responder

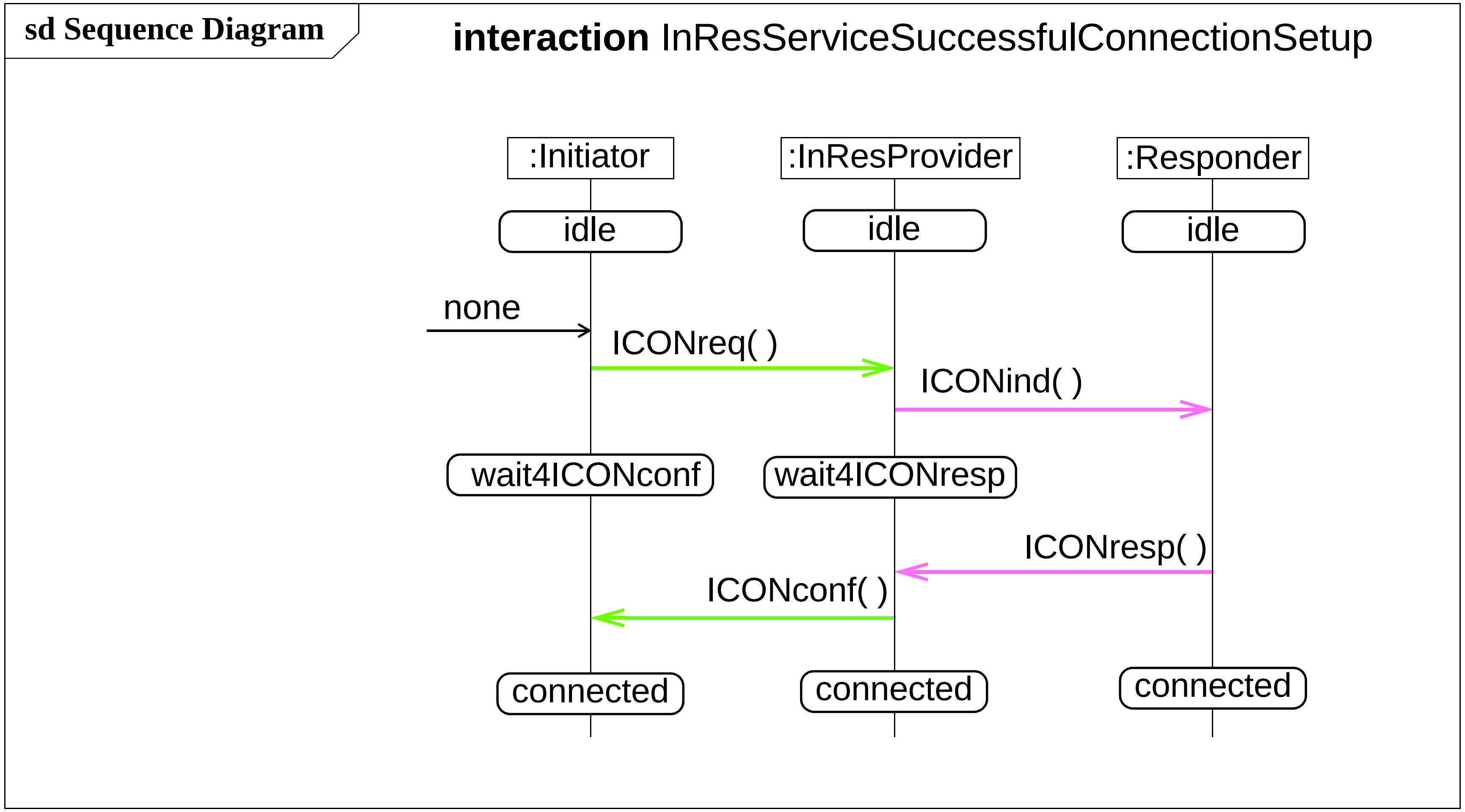
Please enter a name for the ConnectorLine between ServiceUser-1 and ServiceProvider: ISAPini

Please enter a name for the ConnectorLine between ServiceProvider and ServiceUser-2: ISAPres



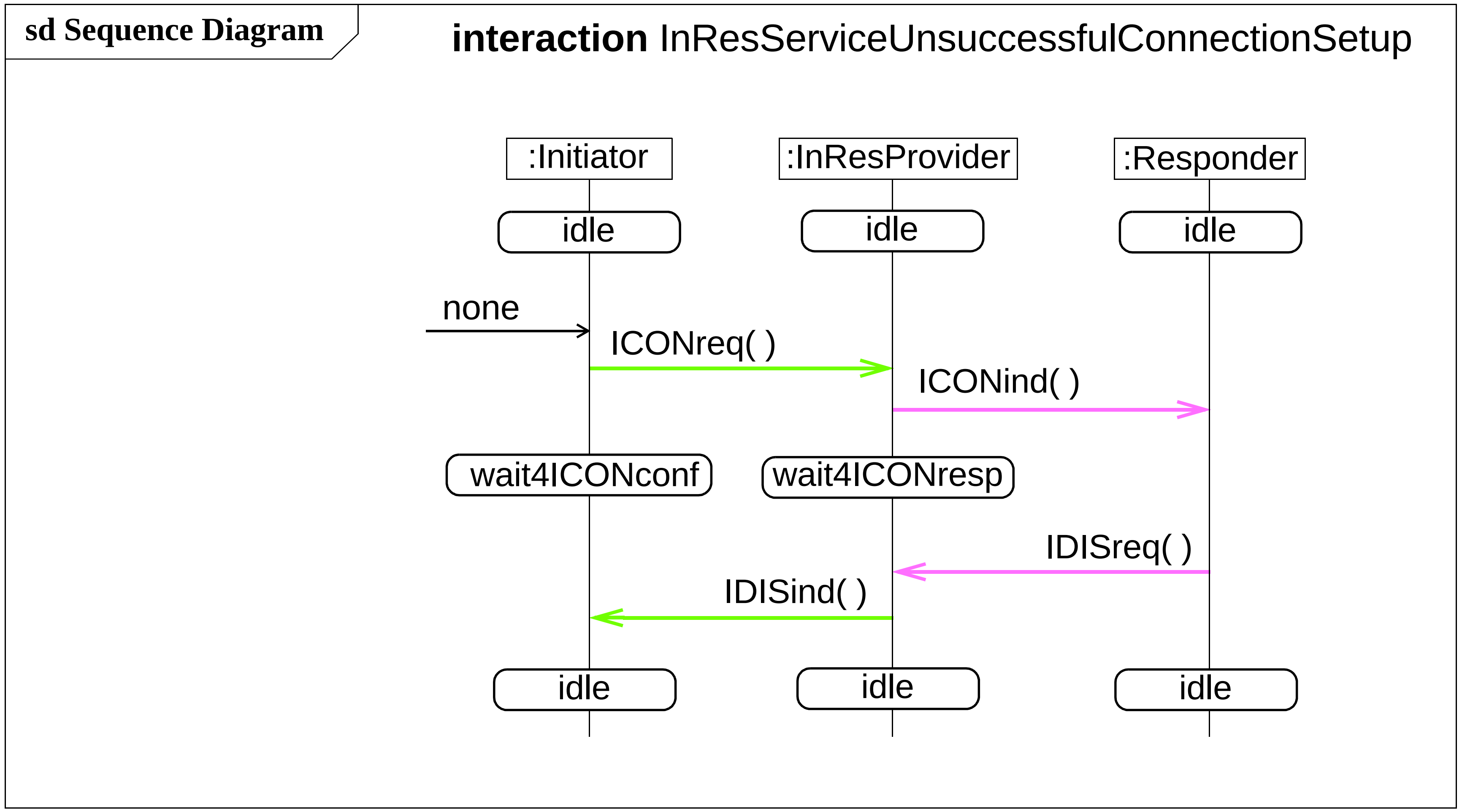
Pattern based Development with SPT

- InRes System - Step II: Analysis Model (1) -



Pattern based Development with SPT

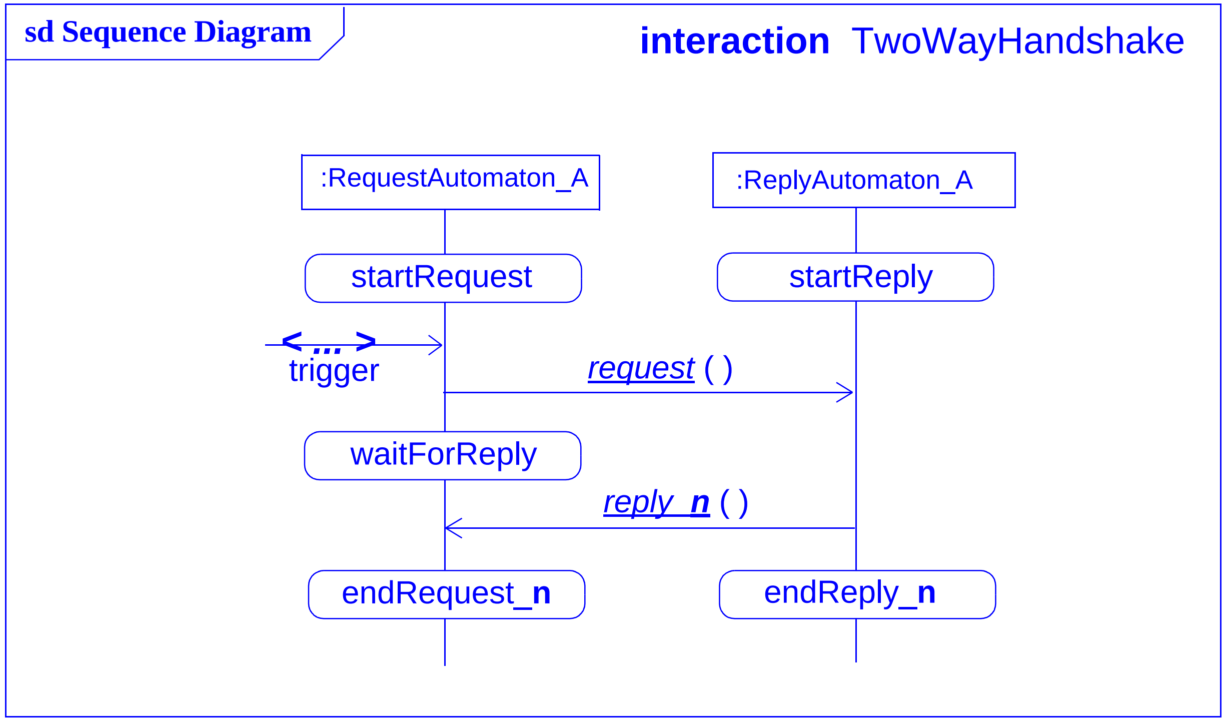
- InRes System - Step II: Analysis Model (2) -



Pattern based Development with SPT

- SDL Pattern SYNCHRONOUSINQUIRY (1) -

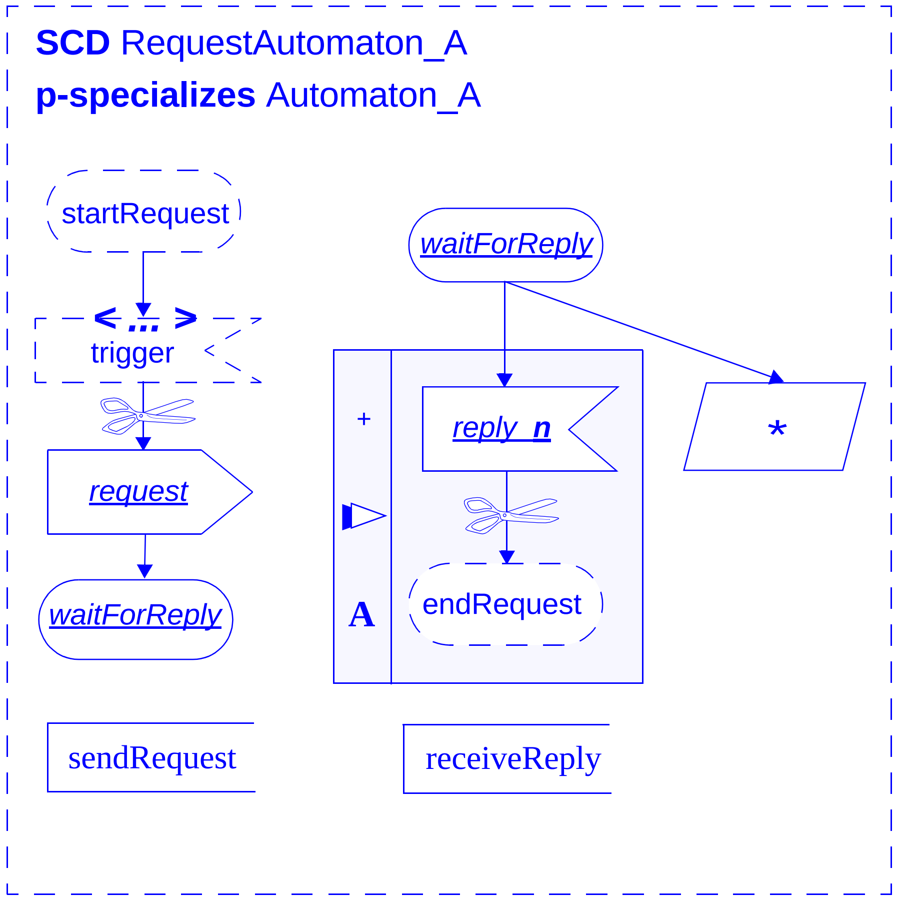
- **Name**: **SYNCHRONOUSINQUIRY**
- **Message Scenario**: The following shows a typical generic usage scenario between two communicating peers following the SYNCHRONOUSINQUIRY pattern.

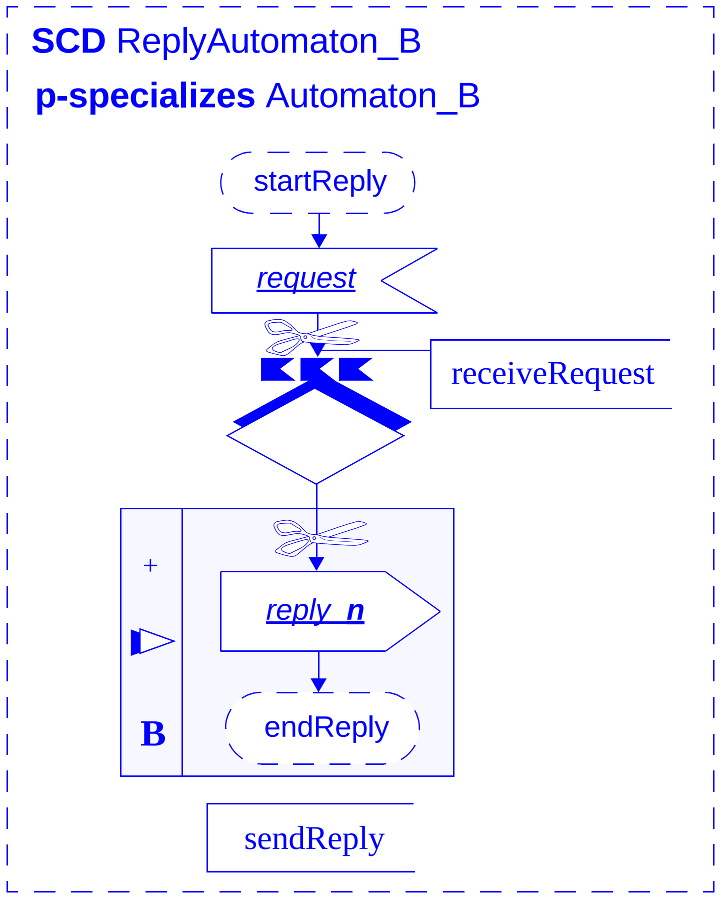


Pattern based Development with SPT

- SDL Pattern SynchronousInquiry (2) -

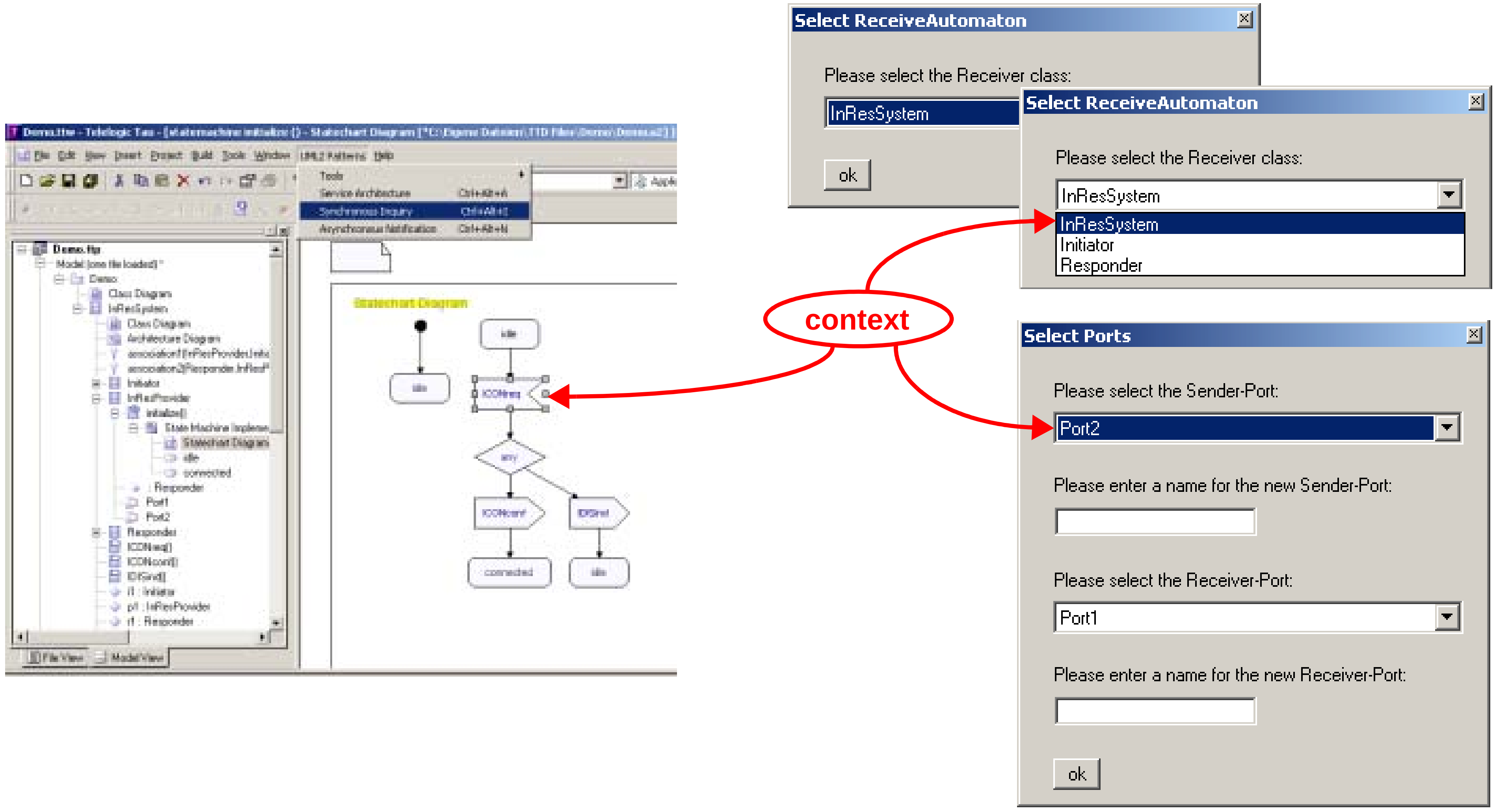
- **Design Fragment (Excerpt)**:





Pattern based Development with SPT

- Context Identification -



Select ReceiveAutomaton

Please select the Receiver class: InResSystem

Select ReceiveAutomaton

Please select the Receiver class: InResSystem

InResSystem
Initiator
Responder

Select Ports

Please select the Sender-Port: Port2

Please enter a name for the new Sender-Port:

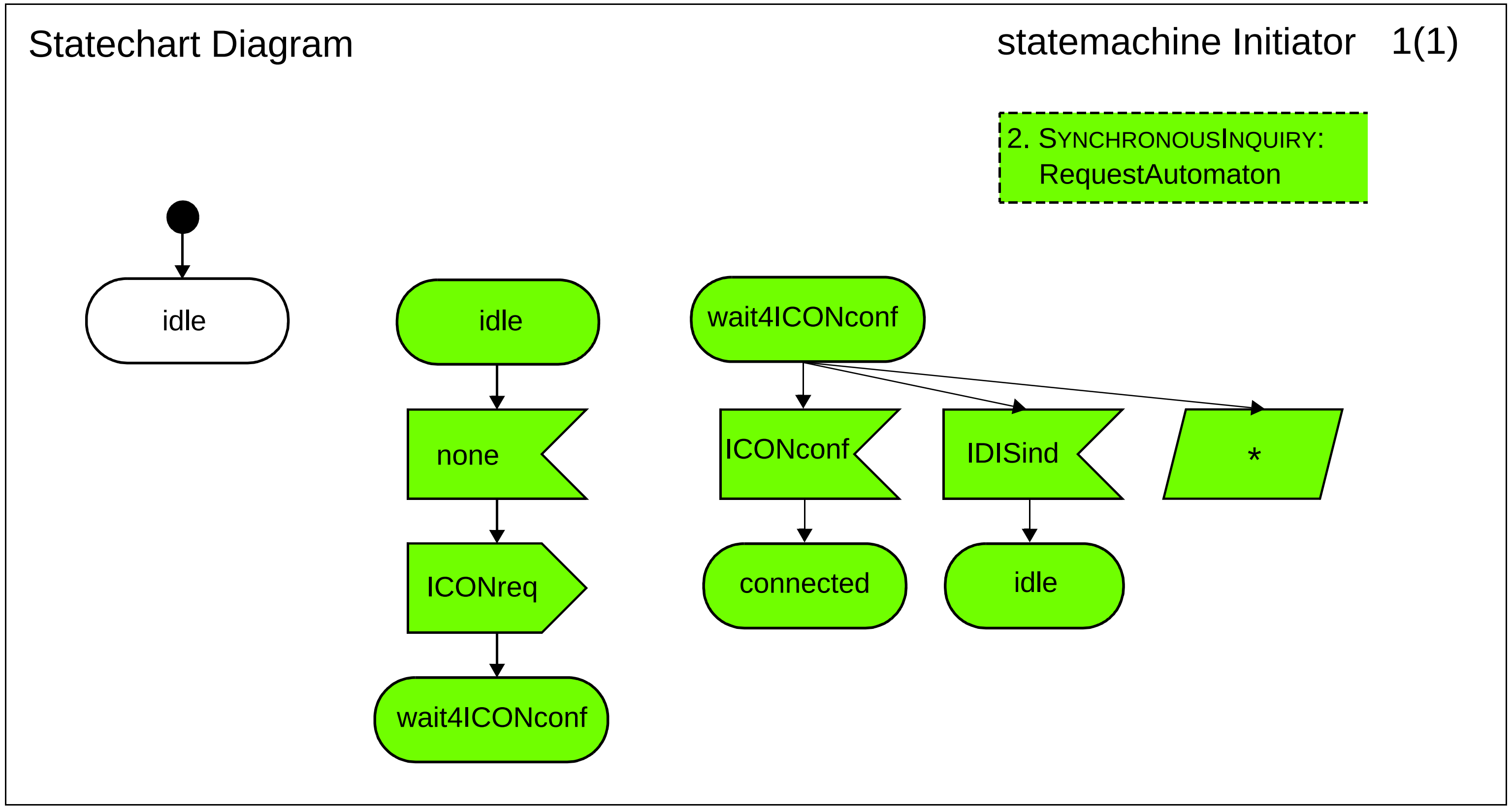
Please select the Receiver-Port: Port1

Please enter a name for the new Receiver-Port:

Pattern based Development with SPT

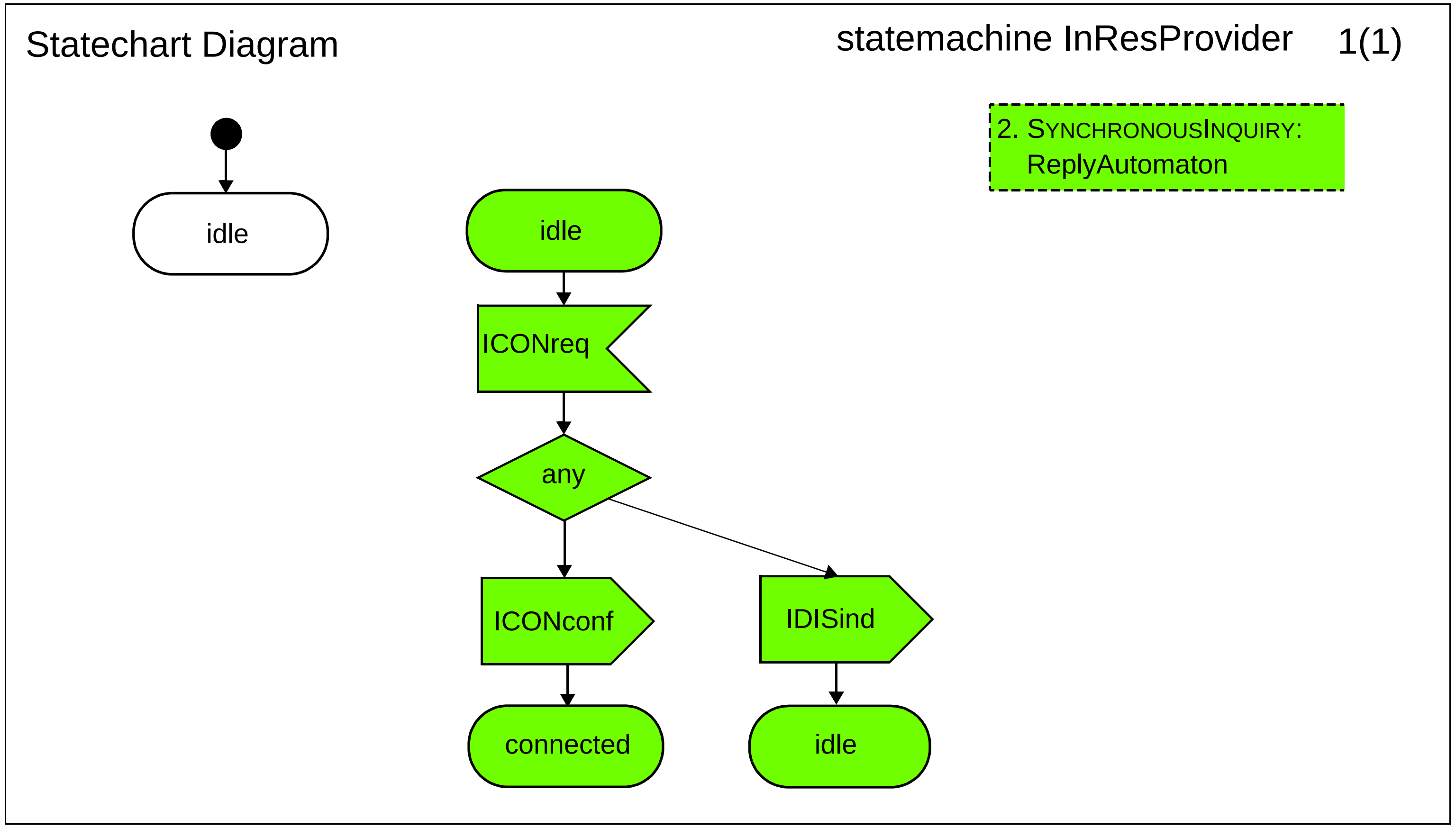
- InRes System - Step II: Design Specification (1) -

- Pattern Selection: SYNCHRONOUSINQUIRY (2X)



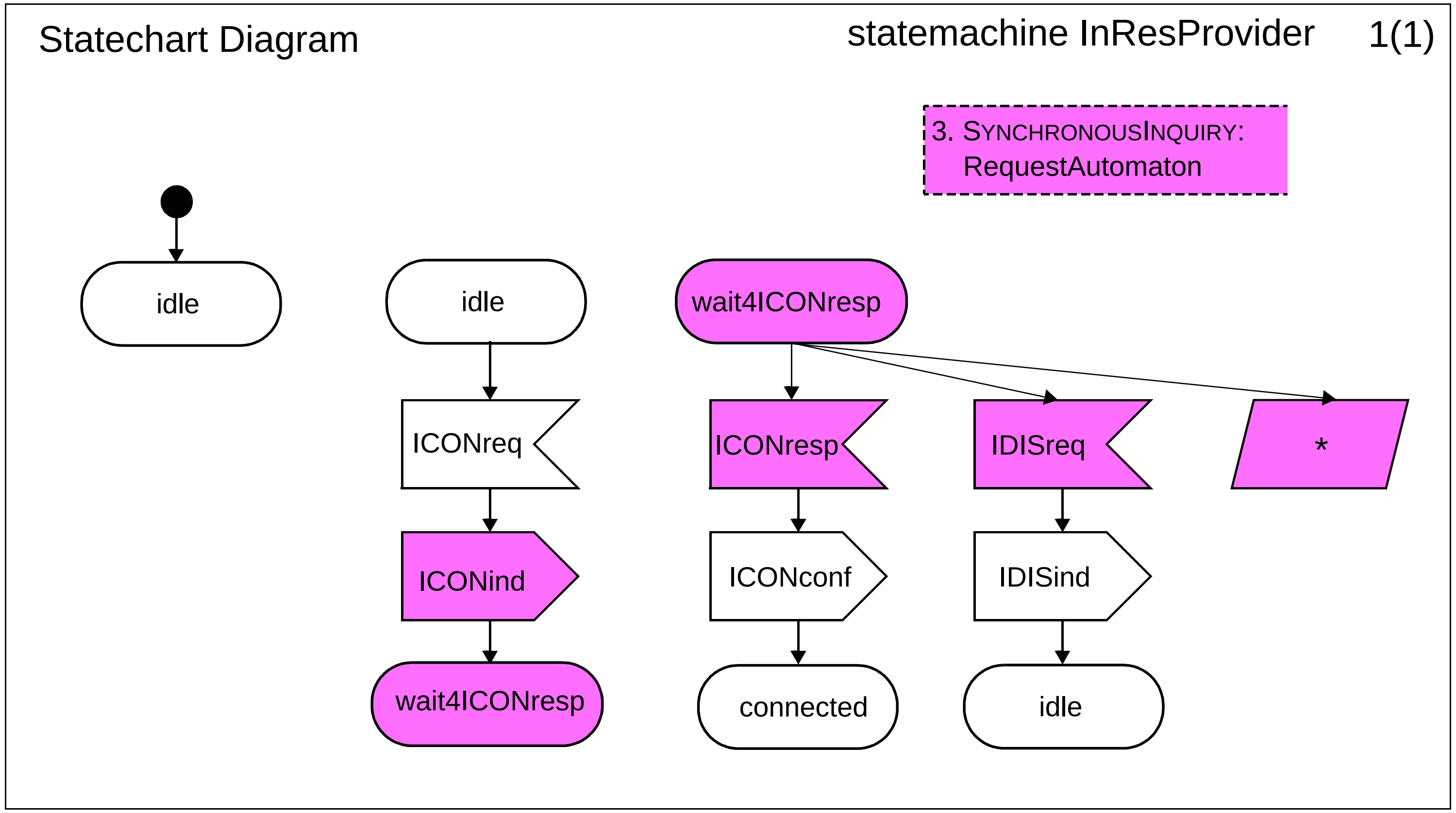
Pattern based Development with SPT

- InRes System - Step II: Design Specification (2) -



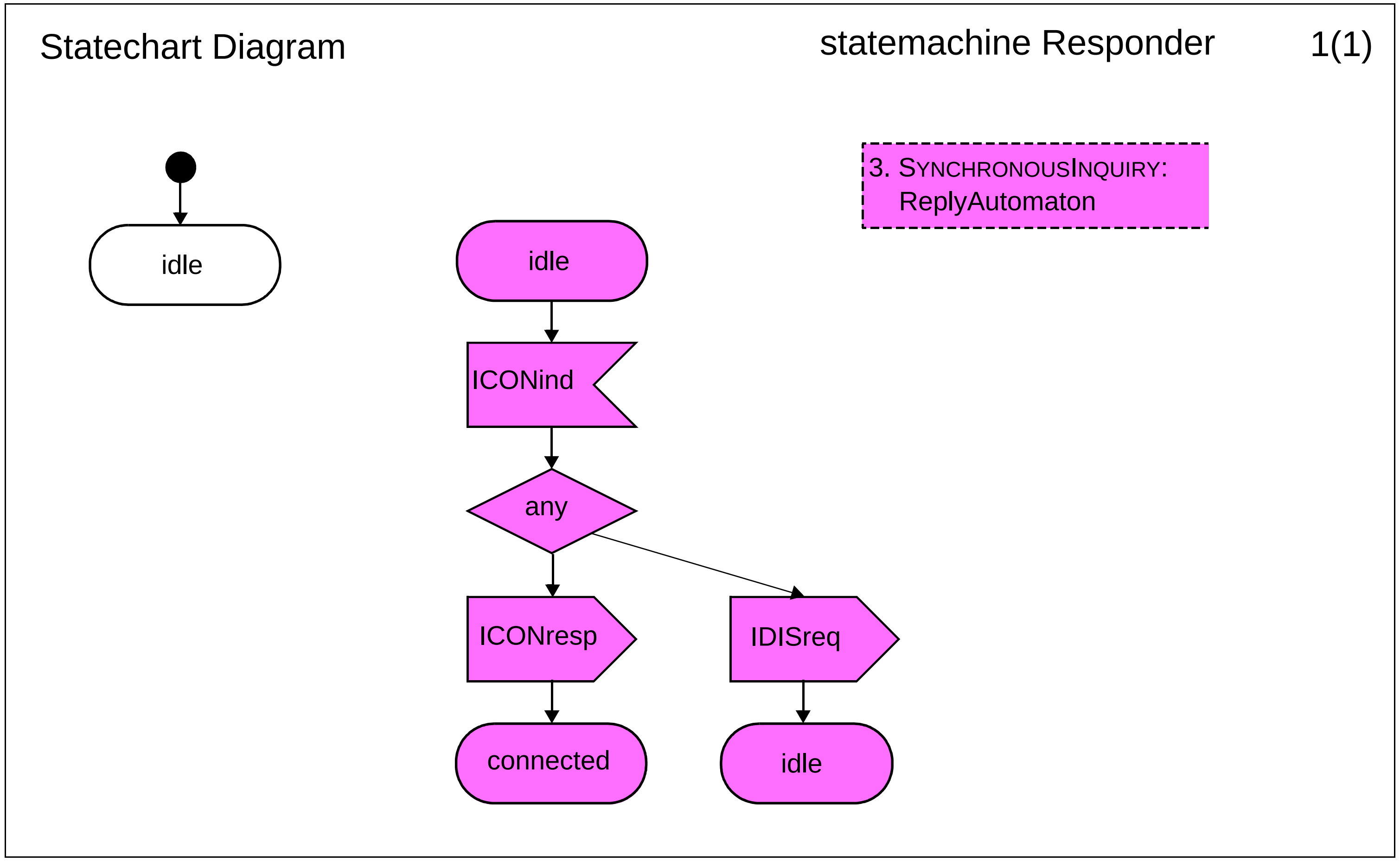
Pattern based Development with SPT

- InRes System - Step II: Design Specification (3) -



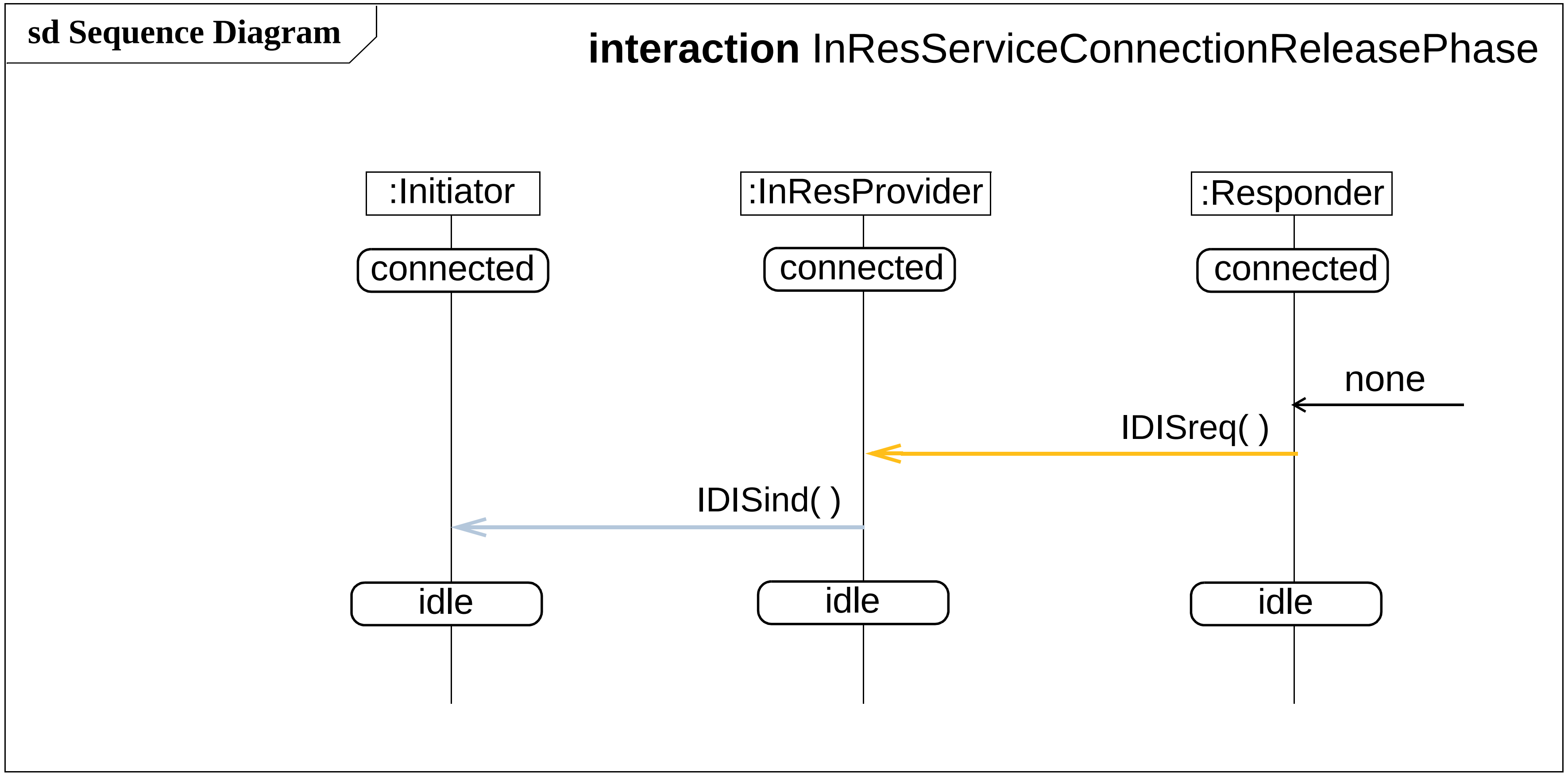
Pattern based Development with SPT

- InRes System - Step II: Design Specification (4) -



Pattern based Development with SPT

- InRes System - Step III: Analysis Model -

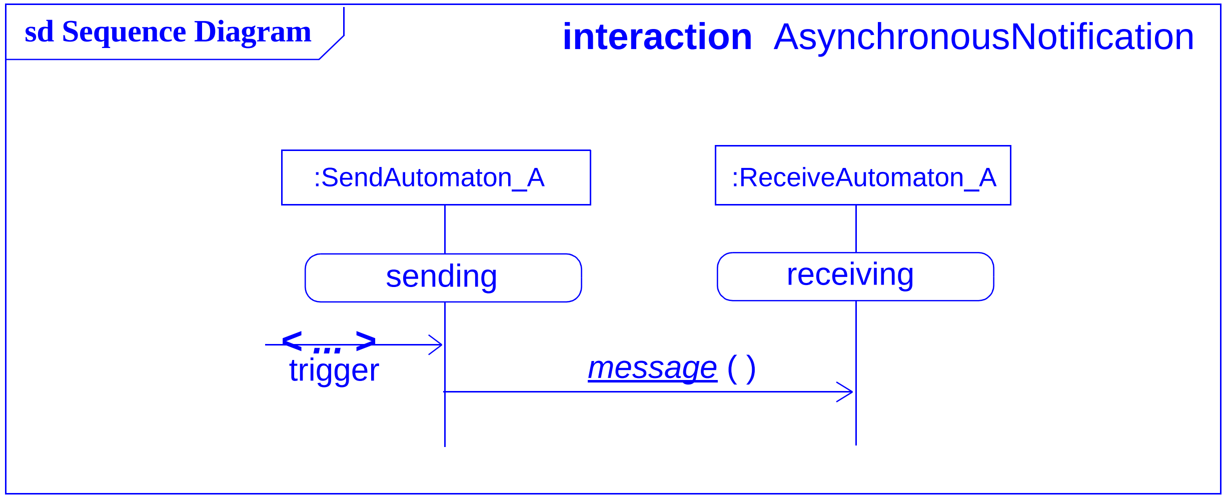


Pattern based Development with SPT

- SDL Pattern AsynchronousNotification (1) -

- **Name**: **AsynchronousNotification**

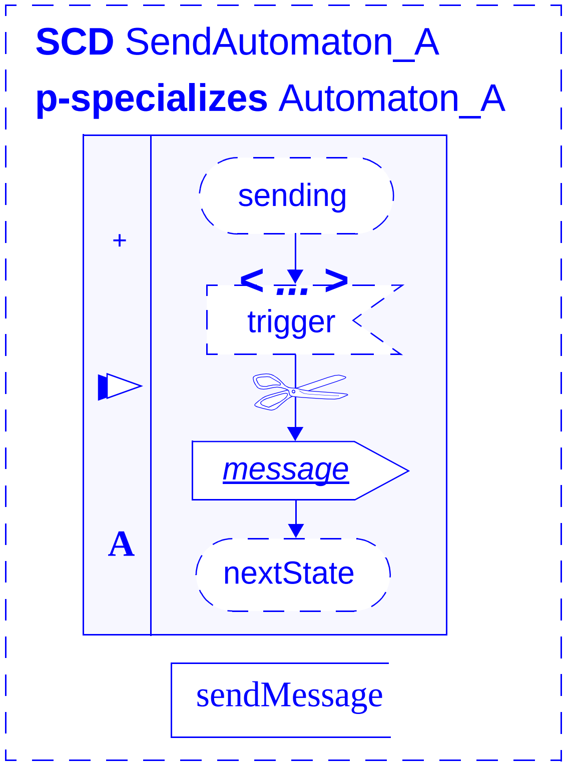
- **Message Scenario**: The following shows a typical generic usage scenario between two communicating peers following the AsynchronousNotification pattern.

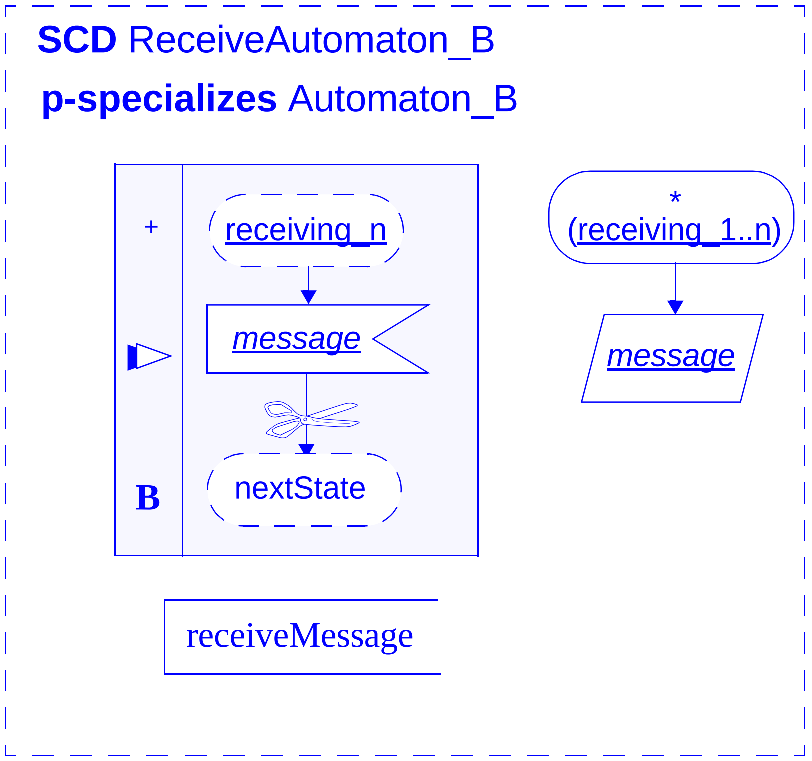


Pattern based Development with SPT

- SDL Pattern ASYNCHRONOUSNOTIFICATION (2) -

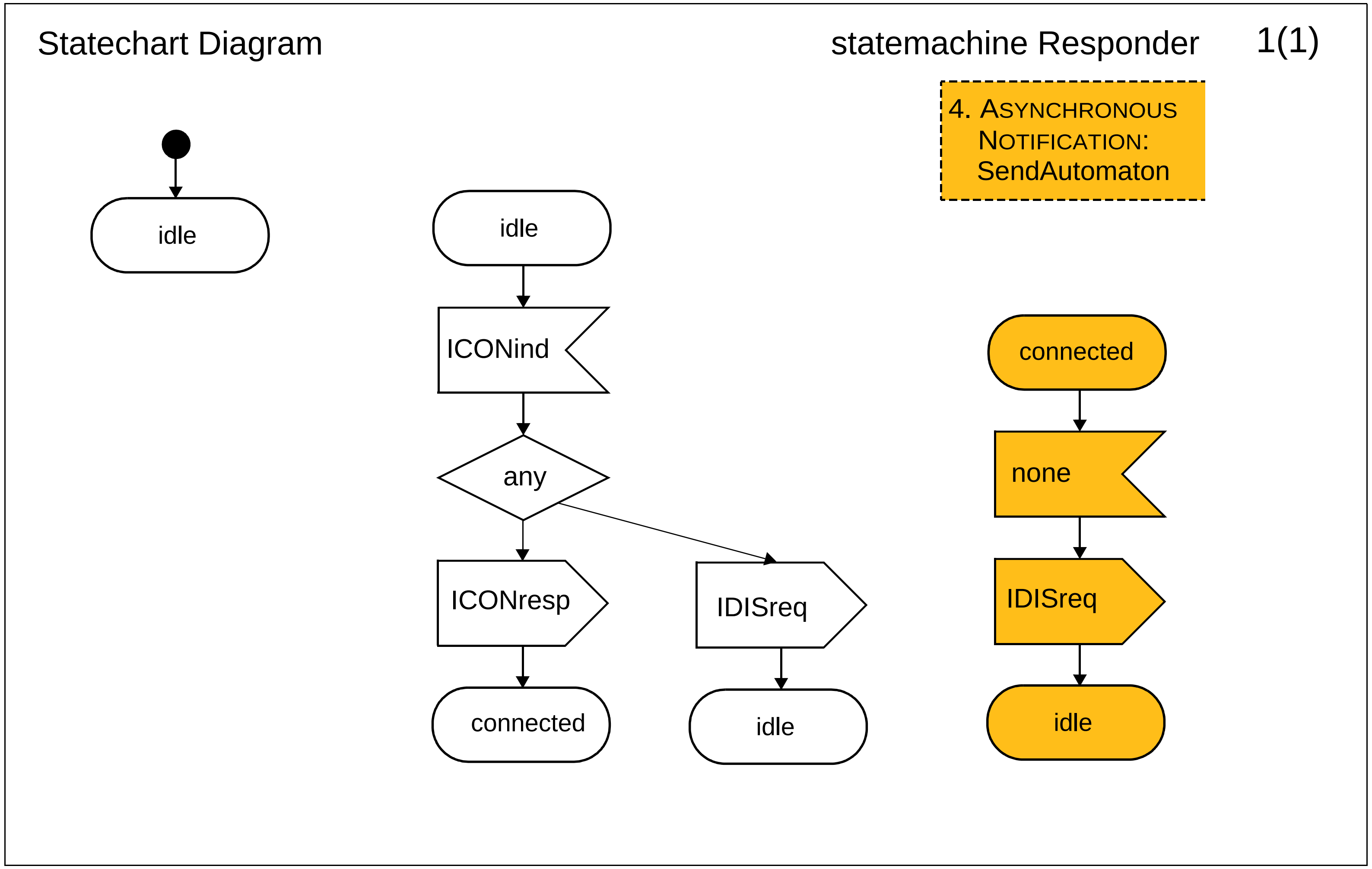
- **Design Fragment**: After being triggered, *SendAutomaton_A* sends a notification and proceeds. If the original transition of *Automaton_A* was branching, sending the message must be performed exactly once per branch. For *ReceiveAutomaton_B*, there must be at least one state where the message is explicitly consumed.





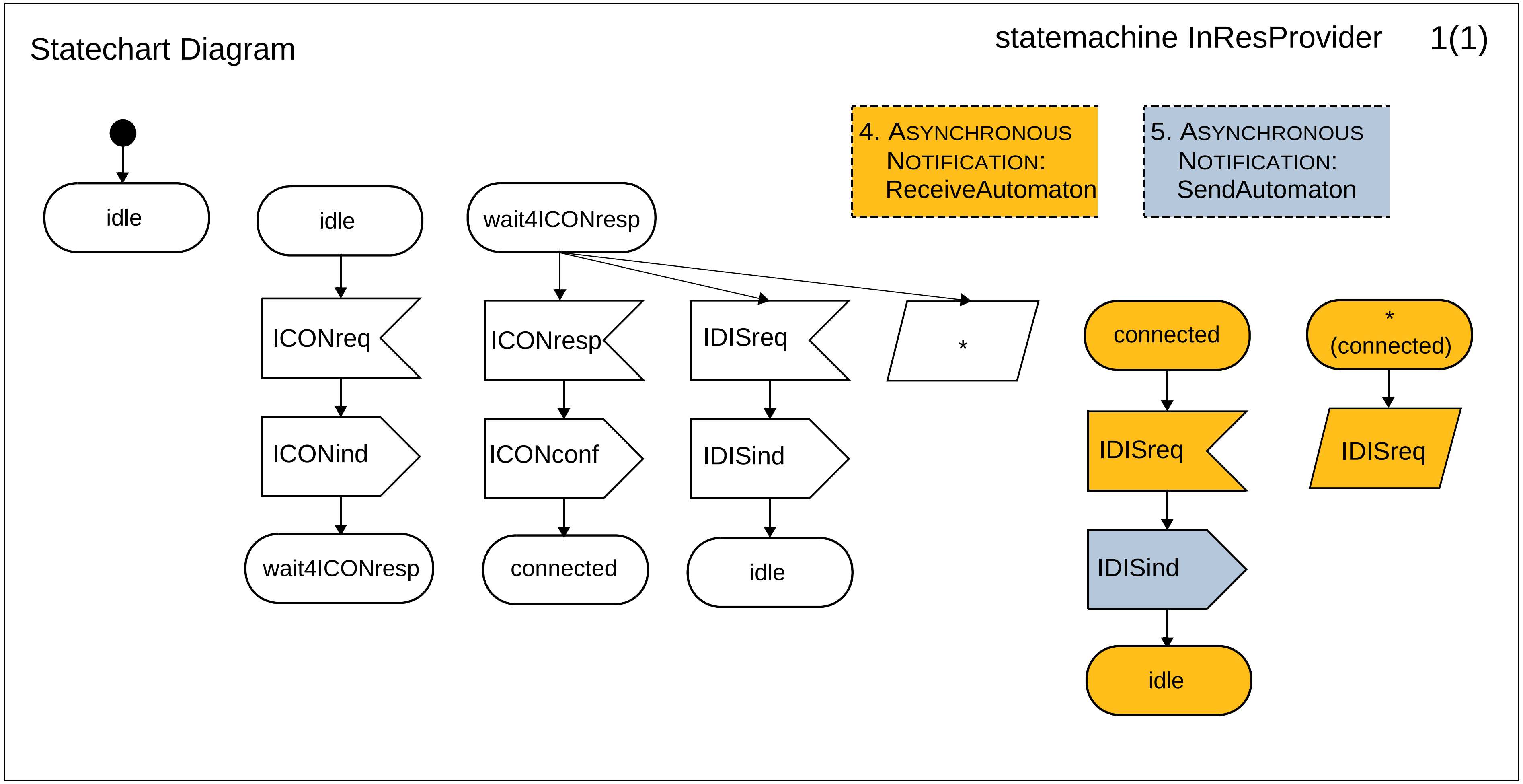
Pattern based Development with SPT

- InRes System - Step III: Design Specification (1) -



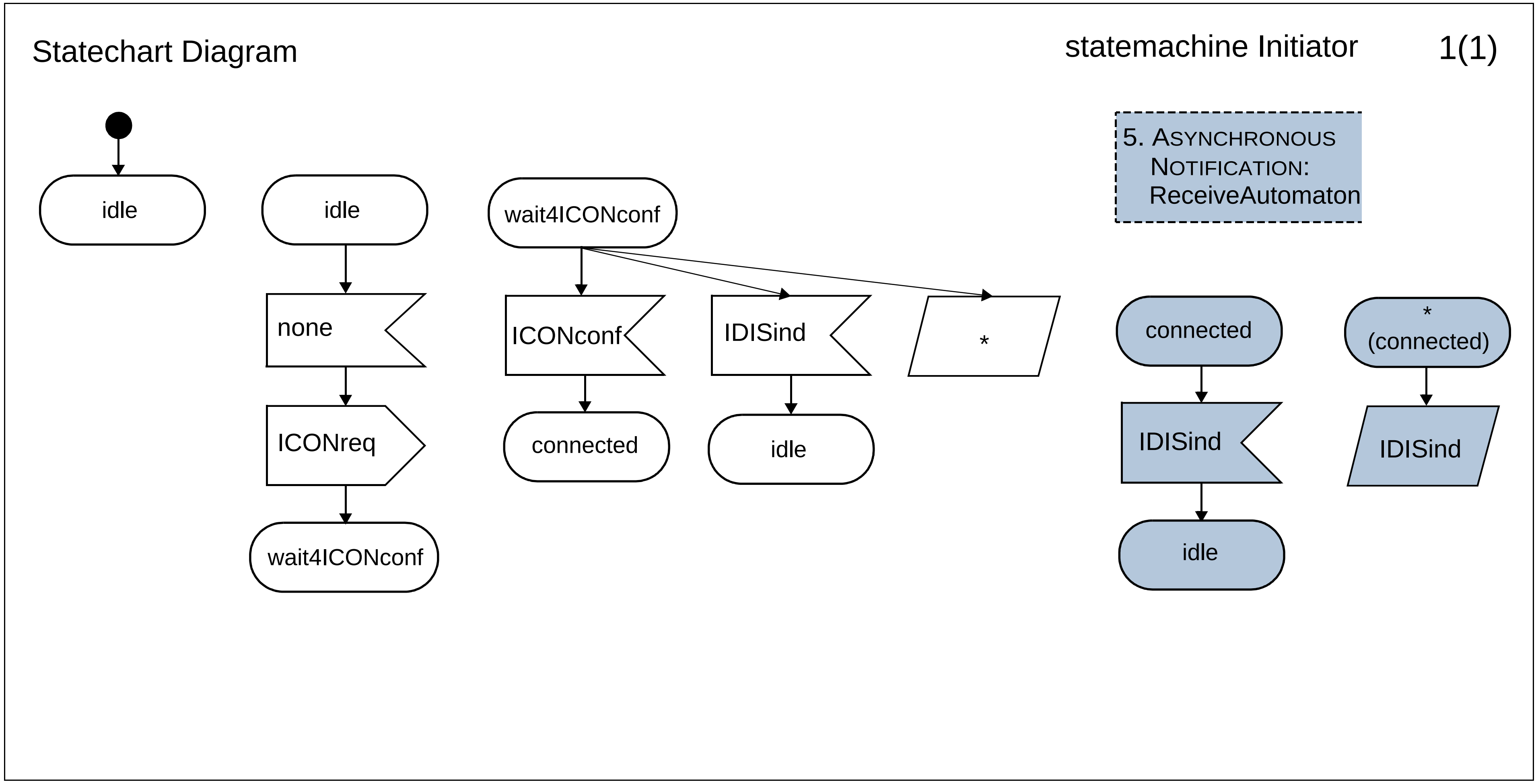
Pattern based Development with SPT

- InRes System - Step III: Design Specification (2) -



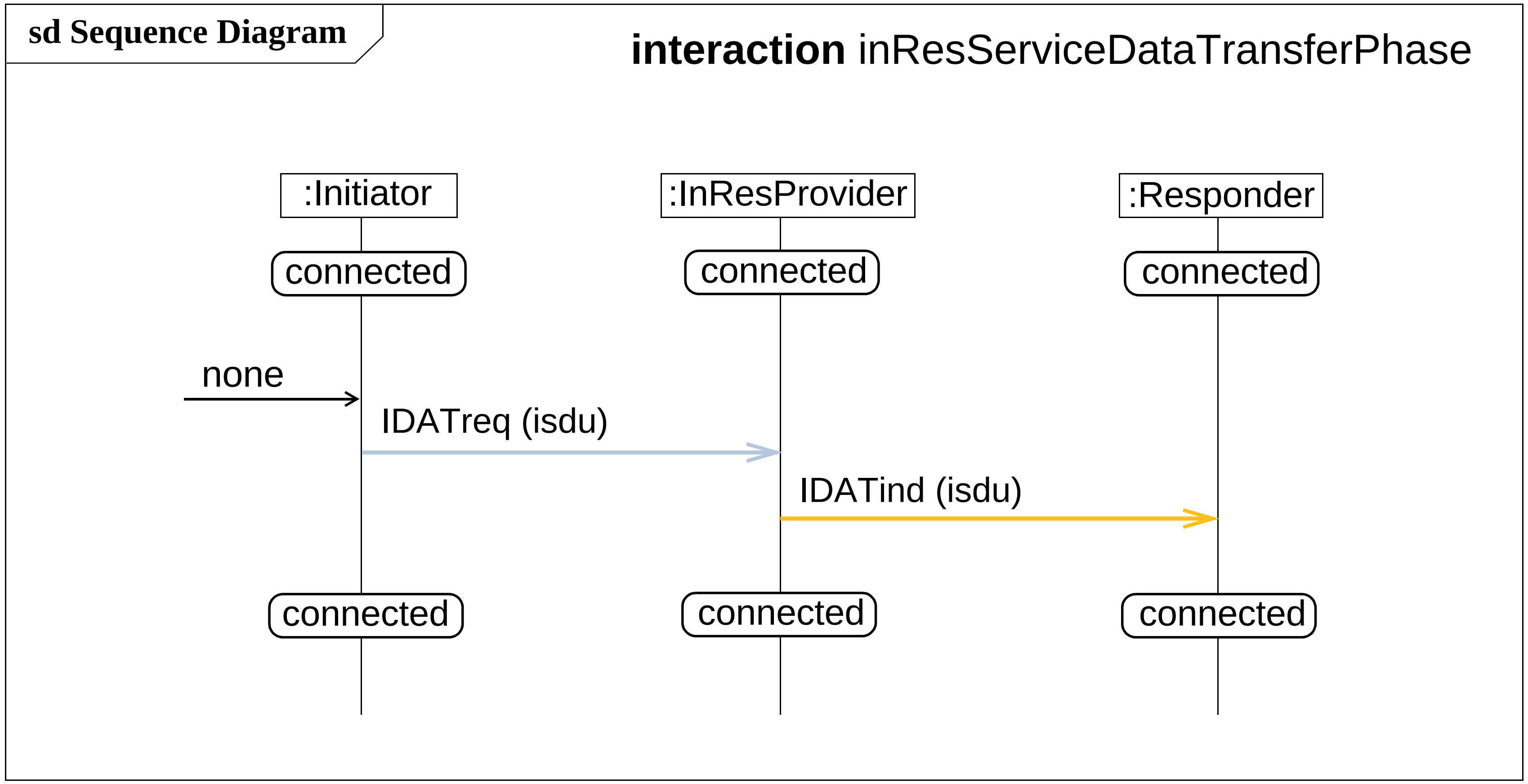
Pattern based Development with SPT

- InRes System - Step III: Design Specification (3) -



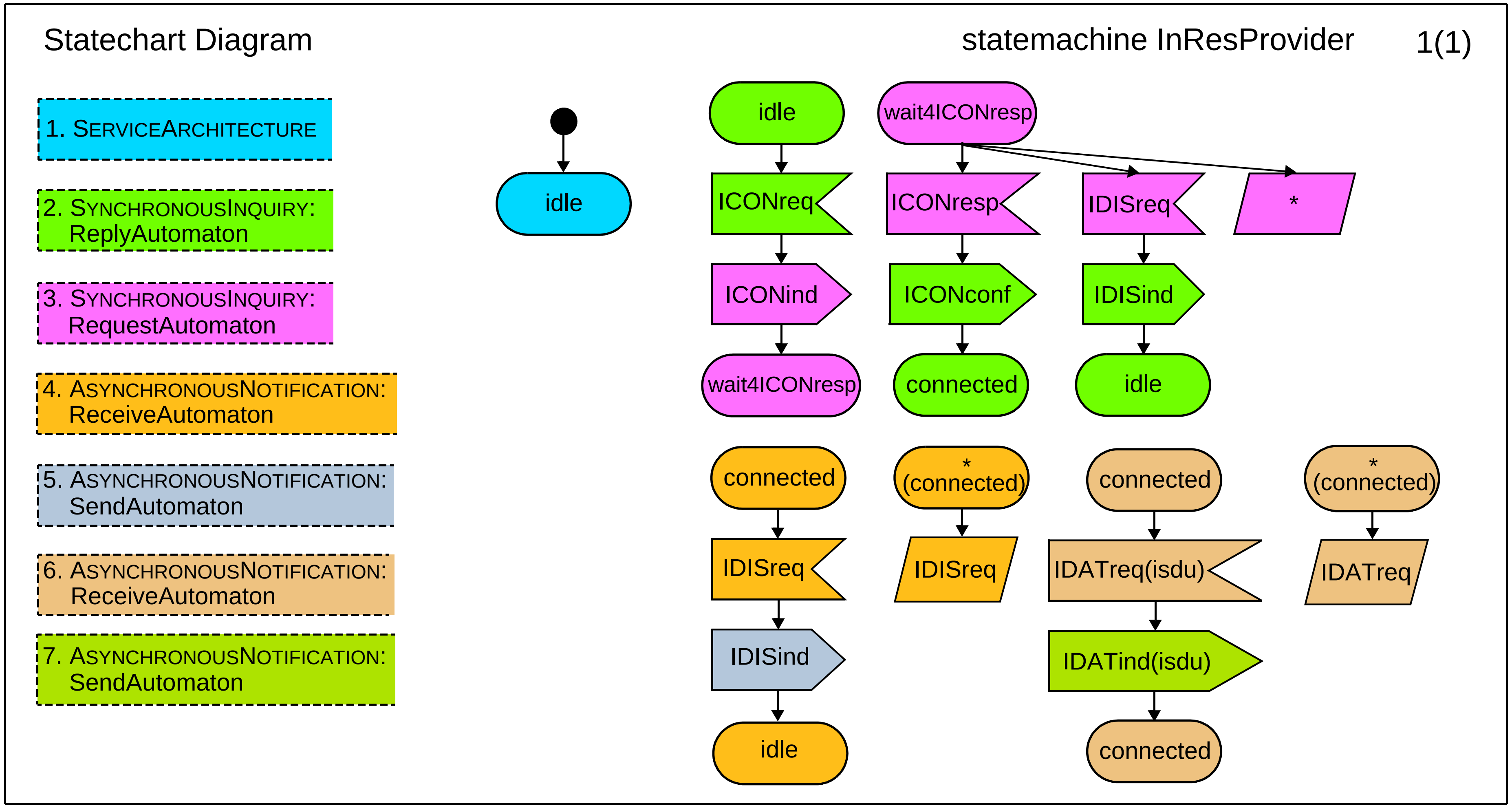
Pattern based Development with SPT

- InRes System - Step IV: Analysis Model -



Pattern based Development with SPT

- InRes System - Design Specification after Steps I-IV: InResProvider -


Statechart Diagram
statemachine InResProvider
1(1)
1. SERVICEARCHITECTURE
2. SYNCHRONOUSINQUIRY: ReplyAutomaton
3. SYNCHRONOUSINQUIRY: RequestAutomaton
4. ASYNCHRONOUSNOTIFICATION: ReceiveAutomaton
5. ASYNCHRONOUSNOTIFICATION: SendAutomaton
6. ASYNCHRONOUSNOTIFICATION: ReceiveAutomaton
7. ASYNCHRONOUSNOTIFICATION: SendAutomaton
idle
ICONreq
ICONind
wait4ICONresp
ICONresp
ICONconf
connected
IDISreq
IDISind
*
* (connected)
IDATreq(isdu)
IDATind(isdu)
IDATreq

Pattern based Development with SPT

- InRes System - Design Steps and Applied SDL Patterns -

Step	Description	Applied SDL Patterns
I	**communication service architecture**	**SERVICEARCHITECTURE**
II	**connection setup phase**	**2x SYNCHRONOUSINQUIRY**
III	connection release phase	2x ASYNCHRONOUSNOTIFICATION
IV	data transfer phase	2x ASYNCHRONOUSNOTIFICATION
V	service provider refinement (part 1)	SERVICEPROVIDERREFINEMENT
VI	service provider refinement (part 2)	2x CODEX
VII	flow control	SYNCHRONOUSINQUIRY
VIII	error control	2x LOSSCONTROL 2x DUPLICATEHANDLE

Conclusions and Perspectives

☞ Status of SPT

- prototype tool supporting selection, adaptation, and composition of 3 SDL patterns
- fully integrated with Telelogic Tau Developer TTD G2

☞ Improvements

- enhancement of TTD G2's basic API, e.g.:
  - selection of model elements during user dialogues
  - auto-layout and automatic refresh of the representation view
  - dynamic colouring of model elements
  - SDL pattern view
- provision of an SDL pattern command package, e.g.:
  - generic user dialogues for the selection of SDL patterns
  - generic high-level user dialogues for the application of SDL patterns
  - developer guidelines to conceive and implement tool support for SDL patterns