ASN.1, MSC, SDL and TTCN Today

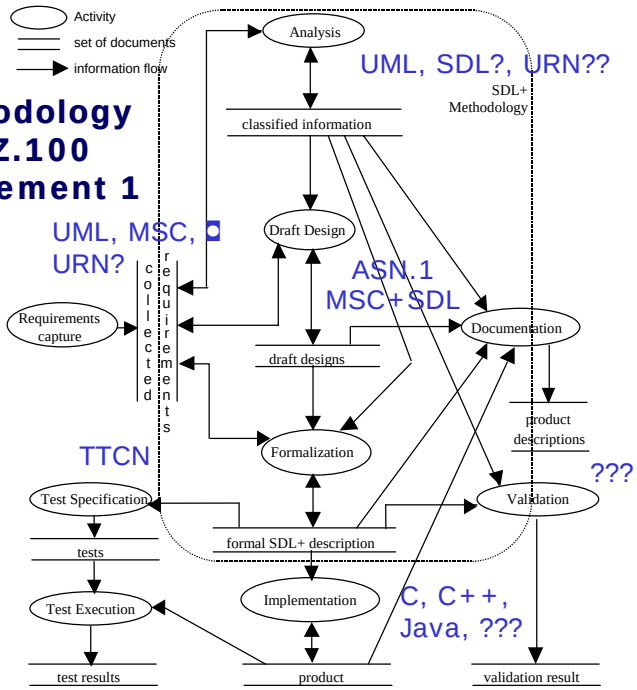
Jens Grabowski - Göttingen University
Rick Reed - TSE Ltd

Objectives of this tutorial

- Overview of ITU languages used today
  - **ASN.1, MSC, SDL and TTCN**

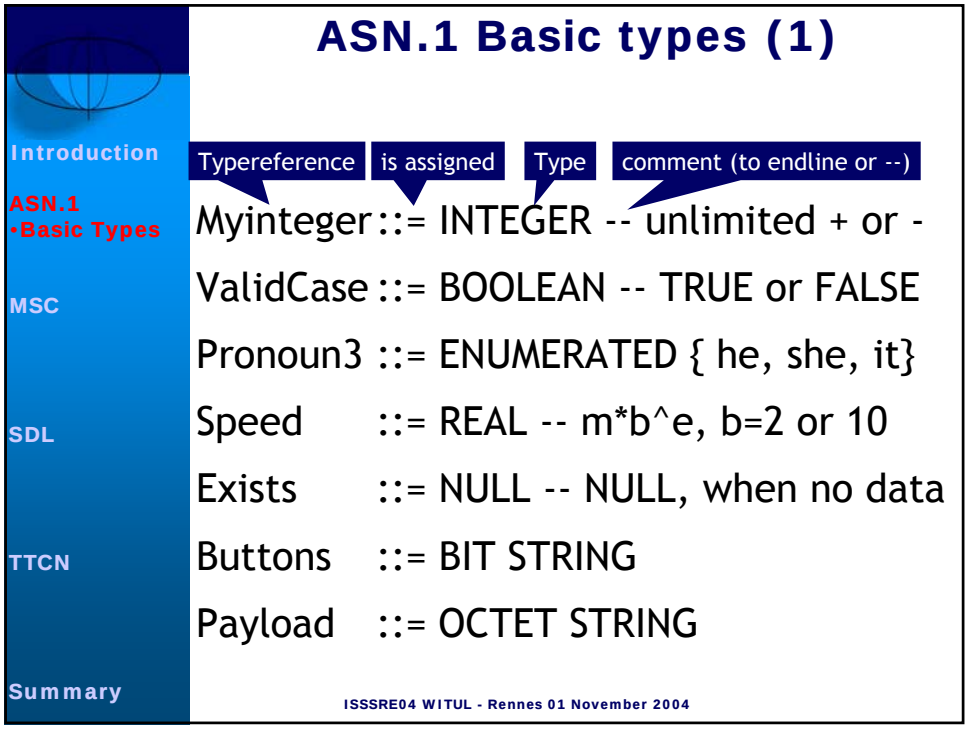
  NOT
  - GDMO, CHILL, eODL or URN
- Introduction for potential new users
- Review key features and current status
- Background for work shop discussion
  - Relevance with respect to UML
  - Current integration of languages

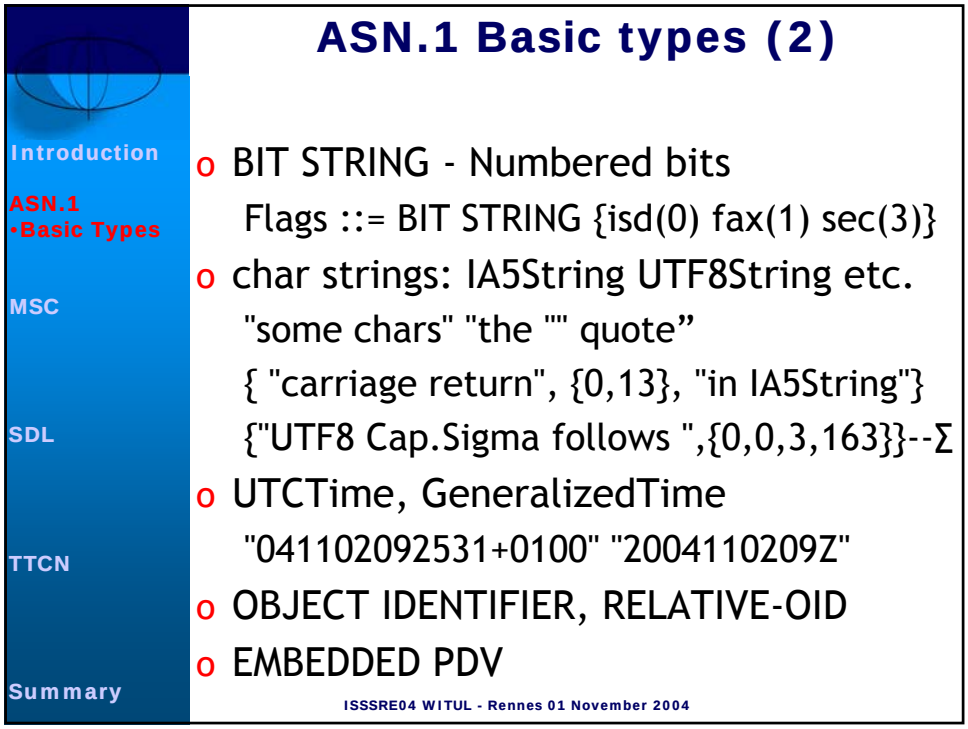
Methodology of Z.100 Supplement 1

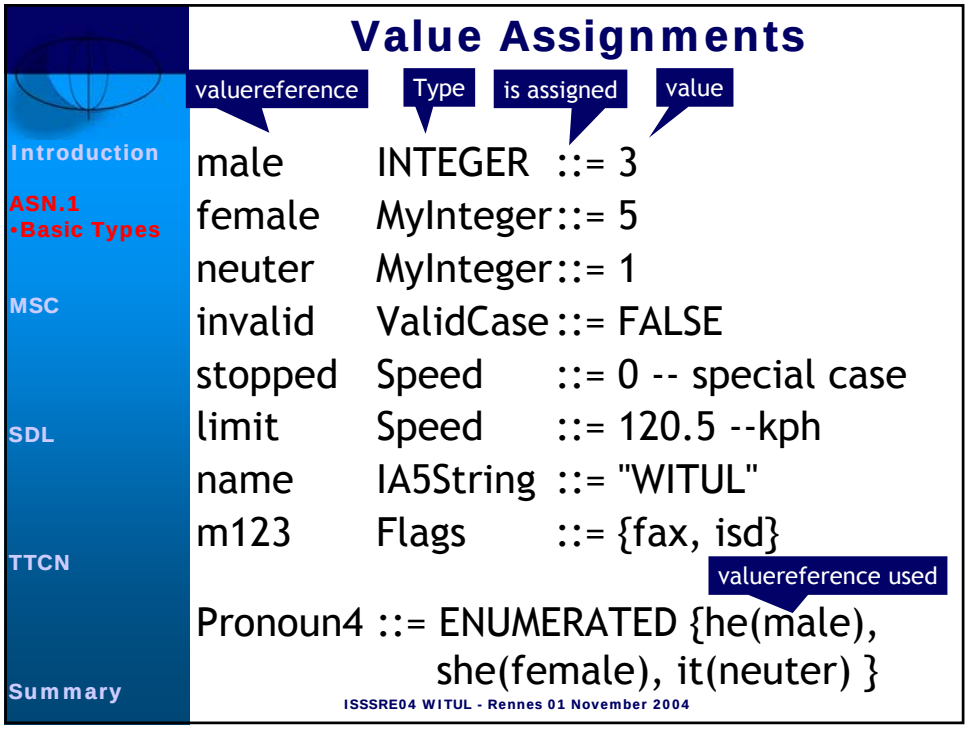


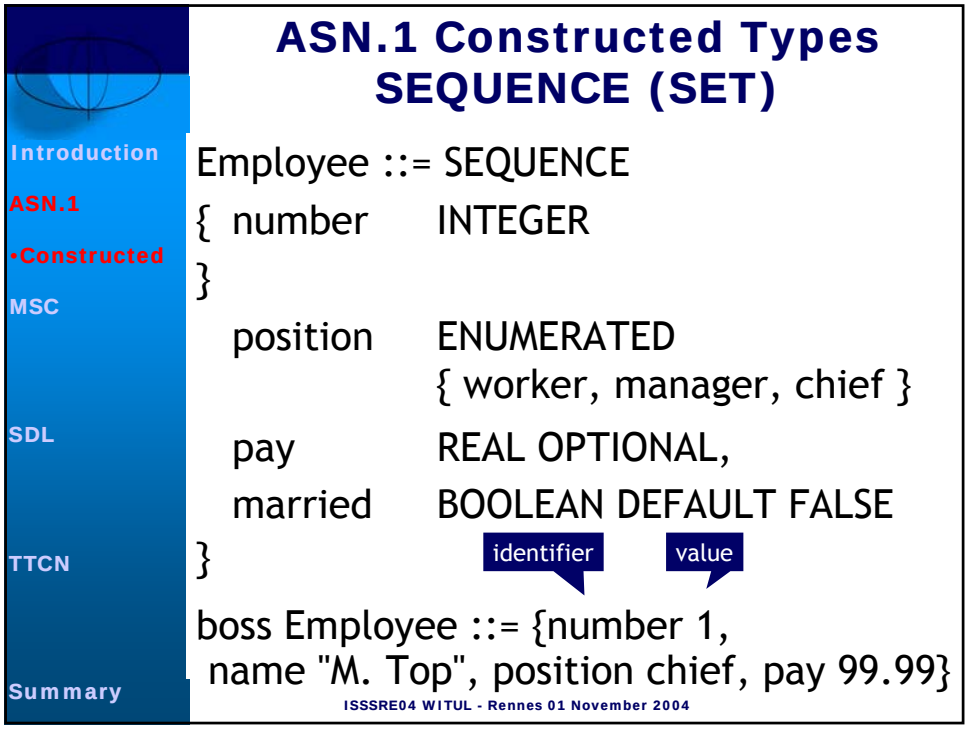
ASN.1
Abstract Syntax Notation number One

- Is a textual notation
- Separated from encoding rule standards
- Defines the “abstract syntax” of data
  That is
  - Named “types” - sets of data values
    - basic types (INTEGER, BOOLEAN etc.)
    - types constructed from other types
    - subtypes (constraining the values of a type)
  - Names (some) data values of the types
- Data extensible for version interworking
- Has parameters of types, + CLASS defs.
- No behaviour operations are defined


ASN.1 Basic types (1)
Introduction
ASN.1
•Basic Types
MSC
SDL
TTCN
Summary
Typereference
is assigned
Type
comment (to endline or --)
Myinteger ::= INTEGER -- unlimited + or -
ValidCase ::= BOOLEAN -- TRUE or FALSE
Pronoun3 ::= ENUMERATED { he, she, it}
Speed ::= REAL -- m*b^e, b=2 or 10
Exists ::= NULL -- NULL, when no data
Buttons ::= BIT STRING
Payload ::= OCTET STRING
ISSSRE04 WITUL - Rennes 01 November 2004


ASN.1 Basic types (2)
Introduction
ASN.1
•Basic Types
MSC
SDL
TTCN
Summary
o BIT STRING - Numbered bits
Flags ::= BIT STRING {isd(0) fax(1) sec(3)}
o char strings: IA5String UTF8String etc.
"some chars" "the "" quote"
{ "carriage return", {0,13}, "in IA5String"}
{"UTF8 Cap.Sigma follows ",{0,0,3,163}}--Σ
o UTCTime, GeneralizedTime
"041102092531+0100" "2004110209Z"
o OBJECT IDENTIFIER, RELATIVE-OID
o EMBEDDED PDV
ISSSRE04 WITUL - Rennes 01 November 2004


Value Assignments
valuereference
Type
is assigned
value
Introduction
ASN.1
•Basic Types
MSC
SDL
TTCN
Summary
male INTEGER ::= 3
female MyInteger ::= 5
neuter MyInteger ::= 1
invalid ValidCase ::= FALSE
stopped Speed ::= 0 -- special case
limit Speed ::= 120.5 --kph
name IA5String ::= "WITUL"
m123 Flags ::= {fax, isd}
valuereference used
Pronoun4 ::= ENUMERATED {he(male),
she(female), it(neuter) }
ISSSRE04 WITUL - Rennes 01 November 2004


ASN.1 Constructed Types
SEQUENCE (SET)
Introduction
ASN.1
•Constructed
MSC
SDL
TTCN
Summary
Employee ::= SEQUENCE
{ number INTEGER
}
position ENUMERATED
{ worker, manager, chief }
pay REAL OPTIONAL,
married BOOLEAN DEFAULT FALSE
}
identifier
value
boss Employee ::= {number 1,
name "M. Top", position chief, pay 99.99}
ISSSRE04 WITUL - Rennes 01 November 2004

ASN.1 Constructed Types SEQUENCE OF (SET OF)

```
AutoTach ::= SEQUENCE OF Speed
   truck1 AutoTach ::=
     { 0, 45.5, 30, 98.7, 50.3, 0 }

NameSet ::= SET OF IA5String
   group NameSet ::=
     {"Jim", "Jane", "Frida", "Gilbert"}
```


ASN.1 Constructed Types CHOICE

```
PeerPeer2 ::= CHOICE
{    rr    ReceiveReady,
     rej   Reject,
     disc Disconnect }
     -- where
     ReceiveReady      ::= NULL
     Reject            ::= INTEGER (0.255)
     Disconnect        ::= NULL
```

identifier value

```
rr2 PeerPeer2      ::= rr: NULL
rj1 PeerPeer2      ::= rej: 1
```

ASN.1 Subtypes

SinglePrimes ::= INTEGER (1,2,3,5,7)
DuoBitChar ::= IA5String ("0","1","2","3")
MyByte ::= INTEGER (0..255)
ParityByte ::= BIT STRING (SIZE (9))
Len4to8Oct ::= OCTET STRING (SIZE(4..8))
VT100text ::= SEQUENCE (SIZE(24)) OF
PrintableString (SIZE(80))
Zdta ::= OCTET STRING (ENCODED BY zid)

- FROM, Regular Expression
- Component(s) - subset, size, present

ASN.1 Extensibility


```
Simple ::= SEQUENCE
{  head Header,
   data Payload,
   ...
} -- version 1
```

```
Simple SEQUENCE
{  head Header,
   data Payload,
   ...,
   crc CRCcode
}  version 2
```

ASN.1 Relationship to UML

Introduction
ASN.1
•Comparison
MSC
SDL
TTCN
Summary

- ASN.1 type = passive class (data type)
- Well-defined subtypes (=subsets)
- Well-defined value notation
- Designed with encoding in mind
- Supports XML

BUT

- No Generalization (inheritance)
- No Dependency
- No Operations/Behaviour
- No "Structure"

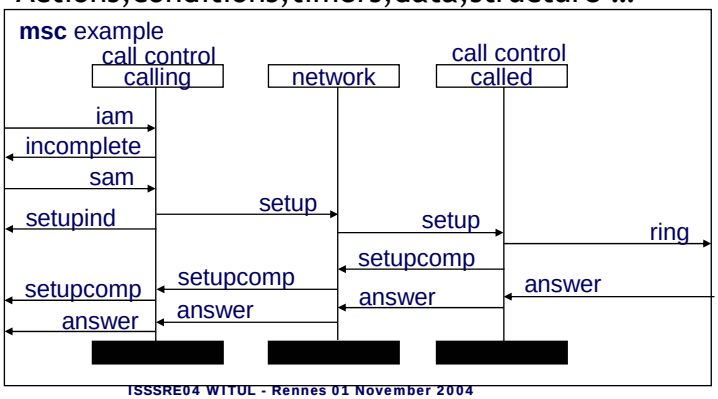
MSC

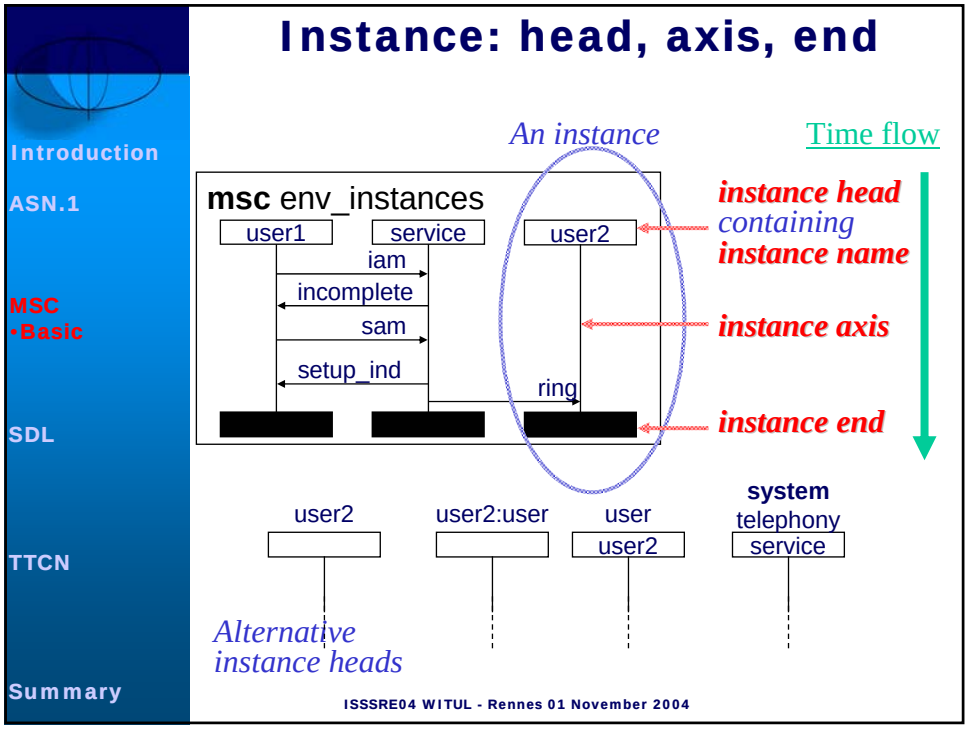
Message Sequence Chart

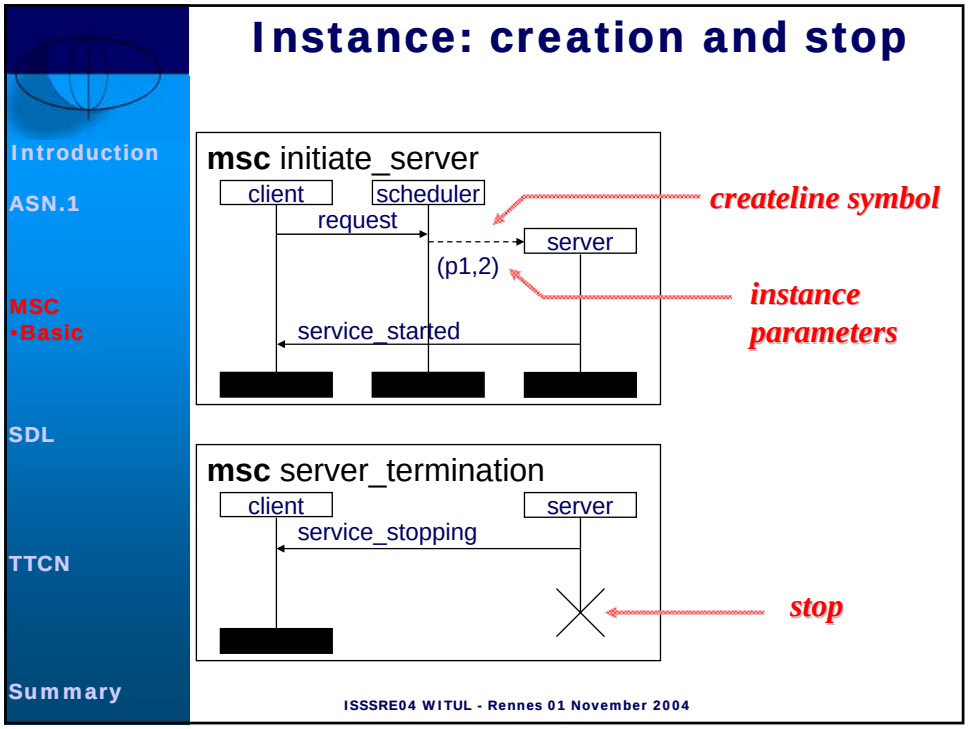
Introduction
ASN.1
MSC
SDL
TTCN
Summary

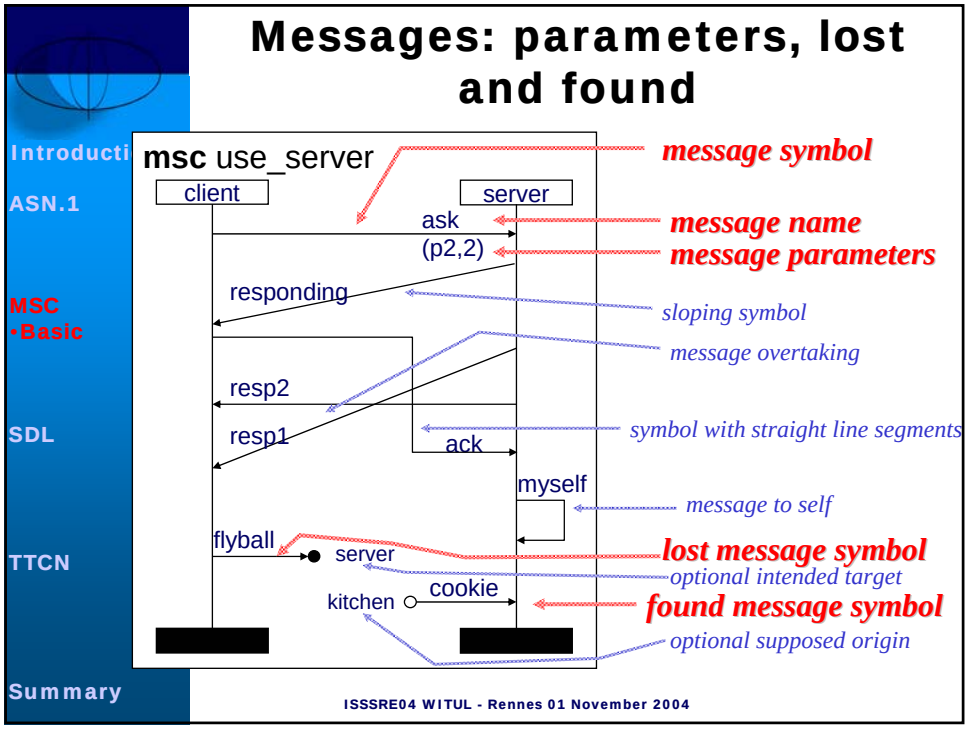
Scenarios or tracing

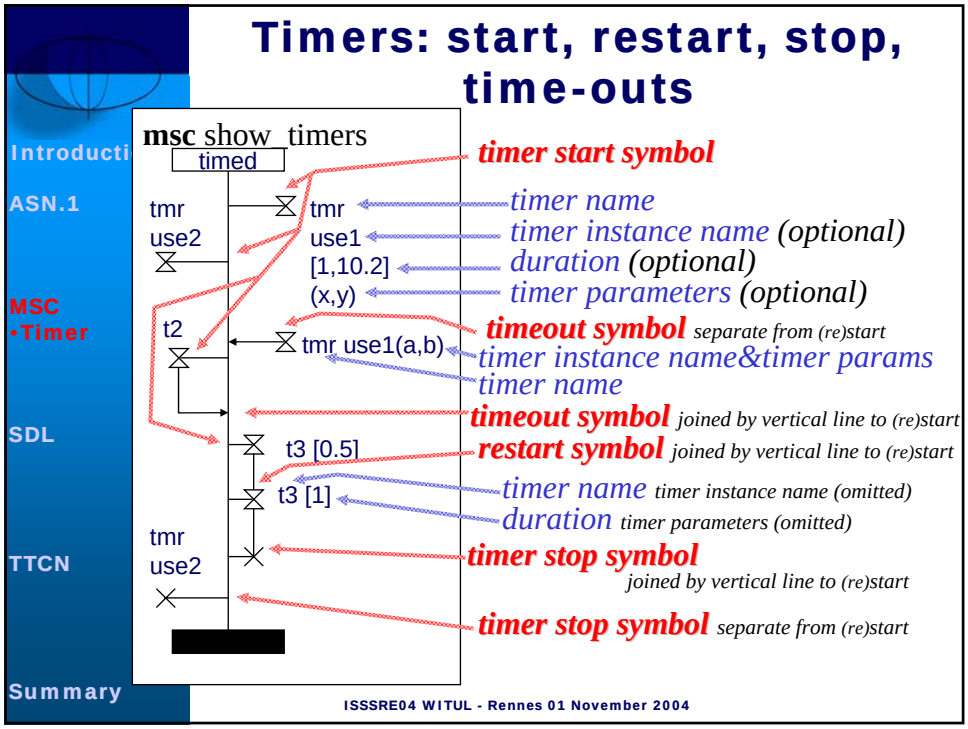
- Instance axes - Instance, time order
- Messages - class (type)name [params]
  - output event, input event
  - no message instance name (usually)
- Actions,conditions,timers,data,structure …

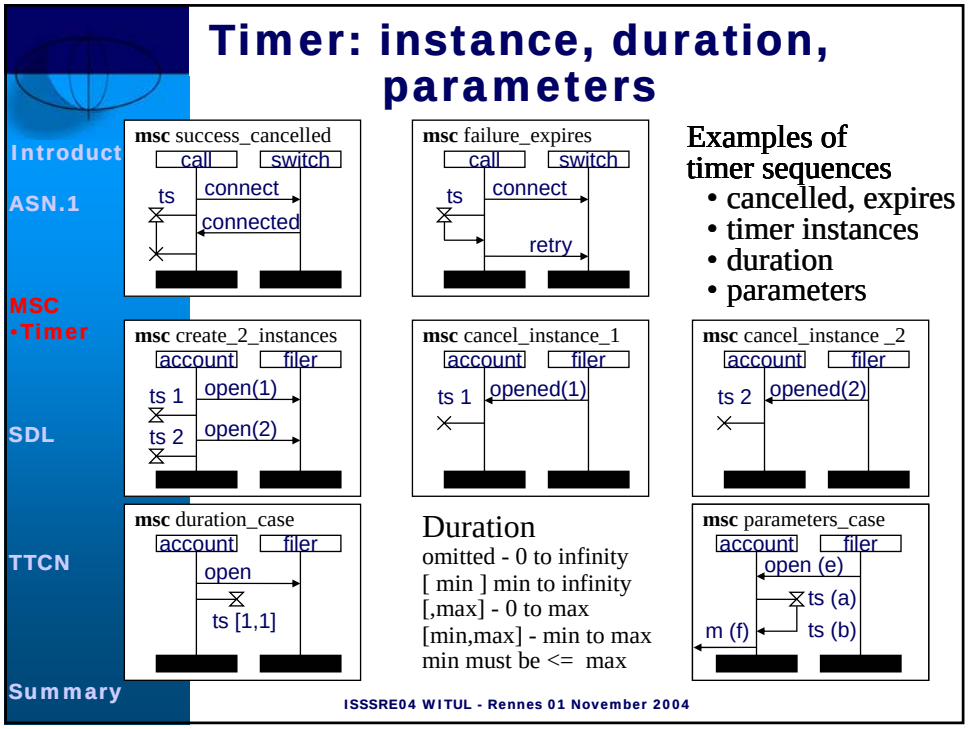


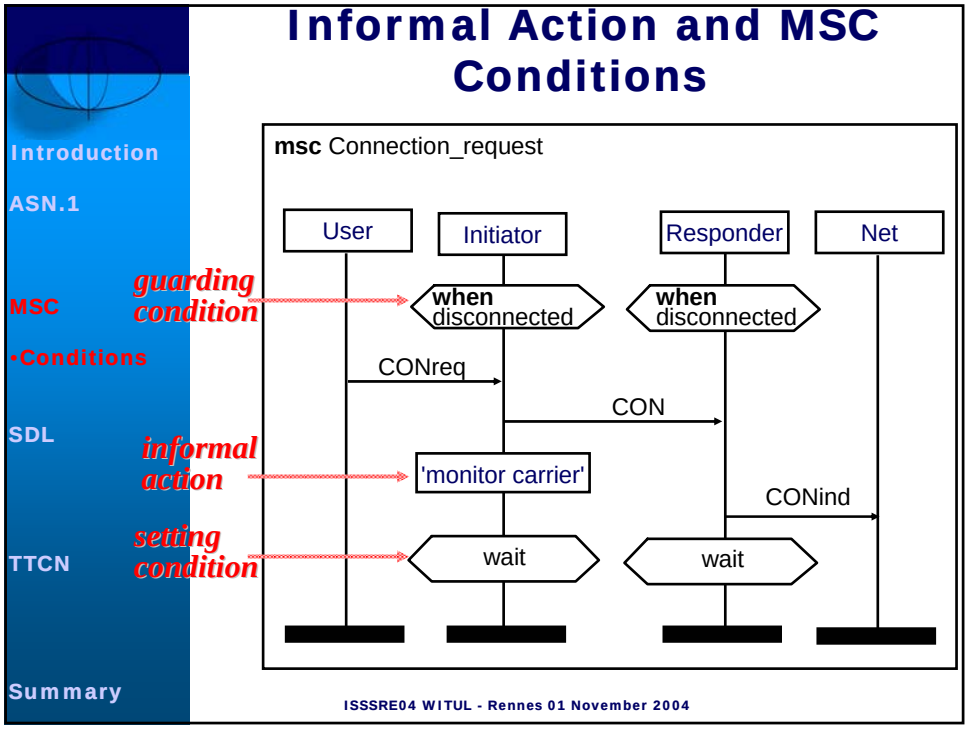

Instance: head, axis, end
Introduction
ASN.1
MSC
•Basic
SDL
TTCN
Summary
An instance
Time flow
msc env_instances
user1
service
user2
iam
incomplete
sam
setup_ind
ring
instance head
containing
instance name
instance axis
instance end
user2
user2:user
user
user2
system
telephony
service
Alternative
instance heads
ISSSRE04 WITUL - Rennes 01 November 2004

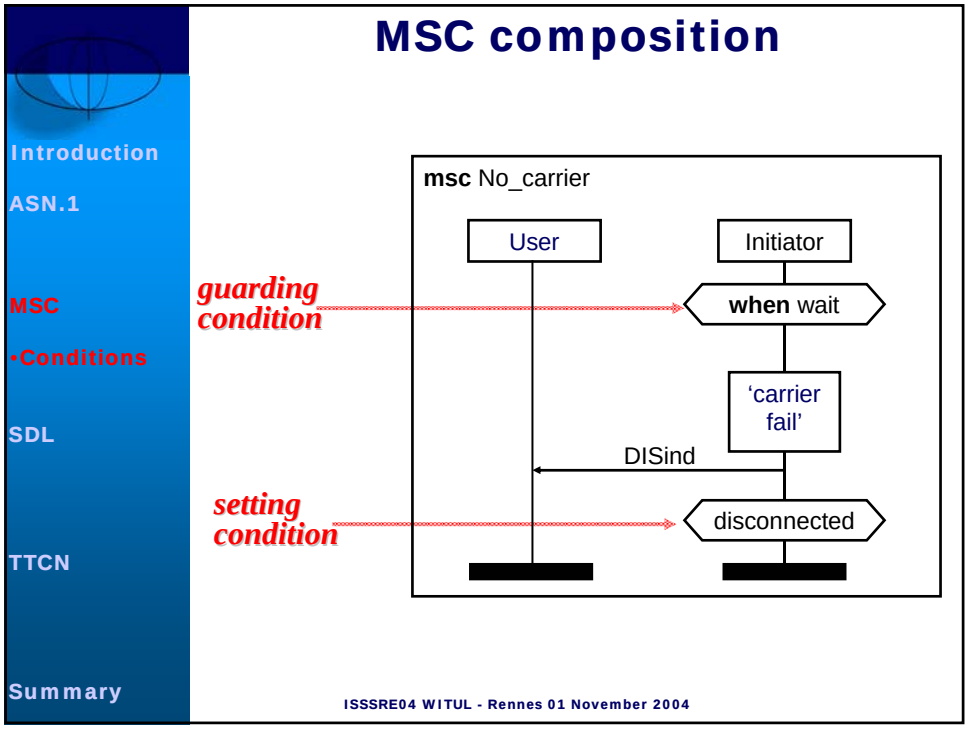

Instance: creation and stop
Introduction
ASN.1
MSC
•Basic
SDL
TTCN
Summary
msc initiate_server
client
scheduler
request
server
(p1,2)
service_started
createline symbol
instance
parameters
msc server_termination
client
server
service_stopping
stop
ISSSRE04 WITUL - Rennes 01 November 2004

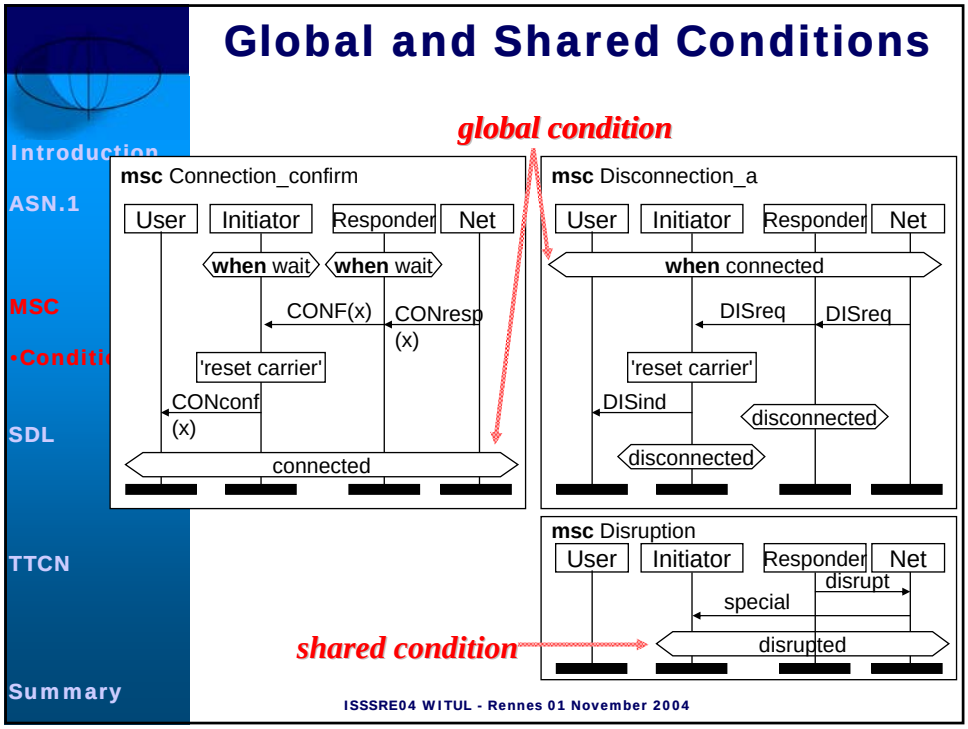

Messages: parameters, lost and found
Introducti
ASN.1
MSC
•Basic
SDL
TTCN
Summary
msc use_server
client
server
ask
(p2,2)
responding
resp2
resp1
ack
myself
flyball
server
cookie
kitchen
message symbol
message name
message parameters
sloping symbol
message overtaking
symbol with straight line segments
message to self
lost message symbol
optional intended target
found message symbol
optional supposed origin
ISSSRE04 WITUL - Rennes 01 November 2004

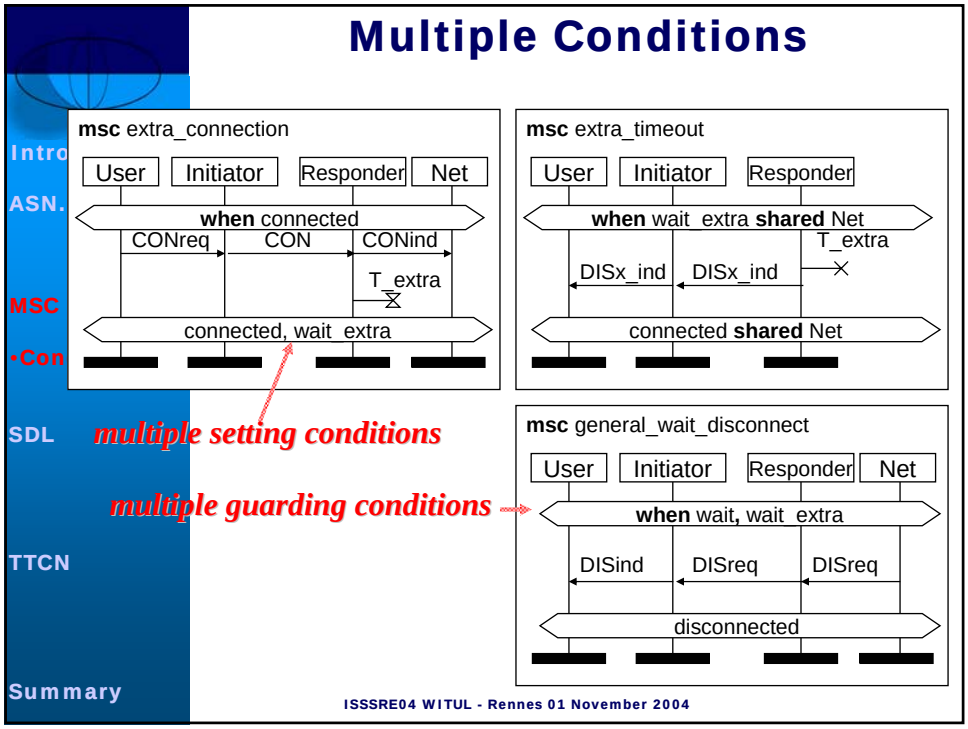

Timers: start, restart, stop, time-outs
Introducti
ASN.1
MSC
•Timer
SDL
TTCN
Summary
msc show_timers
timed
tmr
use2
tmr
use1
[1,10.2]
(x,y)
t2
tmr use1(a,b)
t3 [0.5]
t3 [1]
tmr
use2
timer start symbol
timer name
timer instance name (optional)
duration (optional)
timer parameters (optional)
timeout symbol separate from (re)start
timer instance name&timer params
timer name
timeout symbol joined by vertical line to (re)start
restart symbol joined by vertical line to (re)start
timer name timer instance name (omitted)
duration timer parameters (omitted)
timer stop symbol
joined by vertical line to (re)start
timer stop symbol separate from (re)start
ISSSRE04 WITUL - Rennes 01 November 2004

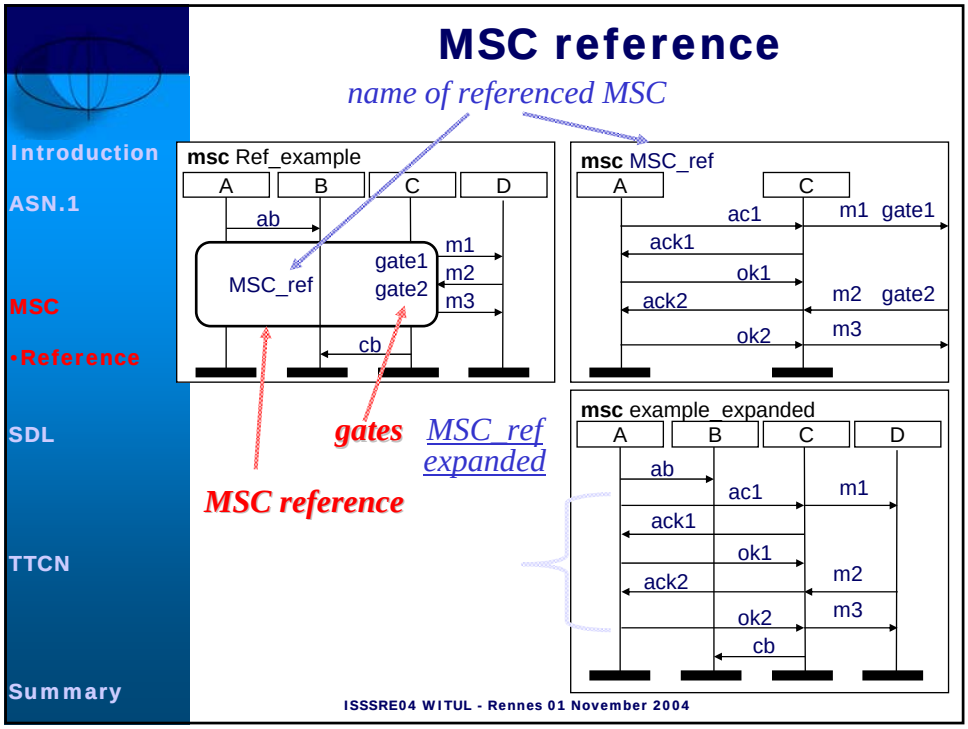

Timer: instance, duration, parameters
Introduct
ASN.1
MSC
•Timer
SDL
TTCN
Summary
msc success_cancelled
call
switch
ts
connect
connected
msc failure_expires
call
switch
ts
connect
retry
Examples of timer sequences
• cancelled, expires
• timer instances
• duration
• parameters
msc create_2_instances
account
filer
ts 1
open(1)
ts 2
open(2)
msc cancel_instance_1
account
filer
ts 1
opened(1)
msc cancel_instance _2
account
filer
ts 2
opened(2)
msc duration_case
account
filer
open
ts [1,1]
Duration
omitted - 0 to infinity
[min] min to infinity
[,max] - 0 to max
[min,max] - min to max
min must be <= max
msc parameters_case
account
filer
open (e)
ts (a)
m (f)
ts (b)
ISSSRE04 WITUL - Rennes 01 November 2004


Informal Action and MSC Conditions
Introduction
ASN.1
MSC
•Conditions
SDL
TTCN
Summary
guarding condition
informal action
setting condition
msc Connection_request
User
Initiator
Responder
Net
when disconnected
when disconnected
CONreq
CON
'monitor carrier'
CONind
wait
wait
ISSSRE04 WITUL - Rennes 01 November 2004

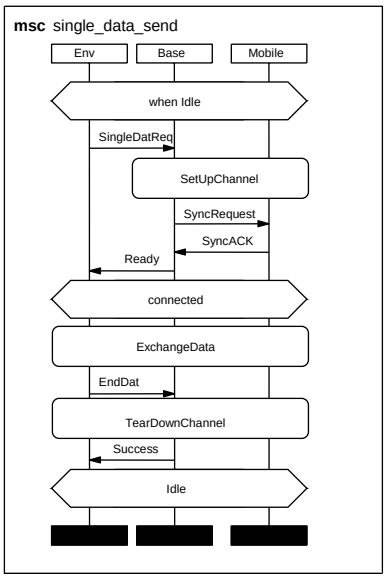

MSC composition
Introduction
ASN.1
MSC
•Conditions
SDL
TTCN
Summary
msc No_carrier
User
Initiator
guarding condition
when wait
'carrier fail'
DISind
setting condition
disconnected
ISSSRE04 WITUL - Rennes 01 November 2004

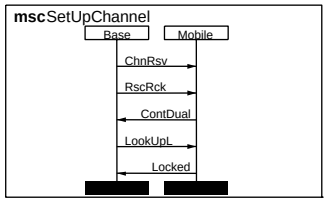

Global and Shared Conditions
Introduction
ASN.1
MSC
SDL
TTCN
Summary
global condition
msc Connection_confirm
User
Initiator
Responder
Net
when wait
when wait
CONF(x)
CONresp (x)
'reset carrier'
CONconf (x)
connected
msc Disconnection_a
User
Initiator
Responder
Net
when connected
DISreq
DISreq
'reset carrier'
DISind
disconnected
disconnected
msc Disruption
User
Initiator
Responder
Net
disrupt
special
shared condition
disrupted
ISSSRE04 WITUL - Rennes 01 November 2004

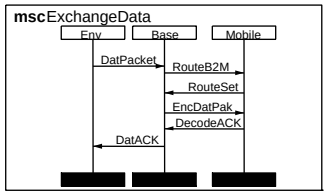

Multiple Conditions
Intro
ASN.
MSC
•Con
SDL
TTCN
Summary
msc extra_connection
User
Initiator
Responder
Net
when connected
CONreq
CON
CONind
T_extra
connected, wait_extra
msc extra_timeout
User
Initiator
Responder
when wait_extra shared Net
T_extra
DISx_ind
DISx_ind
connected shared Net
multiple setting conditions
multiple guarding conditions
msc general_wait_disconnect
User
Initiator
Responder
Net
when wait, wait_extra
DISind
DISreq
DISreq
disconnected
ISSSRE04 WITUL - Rennes 01 November 2004

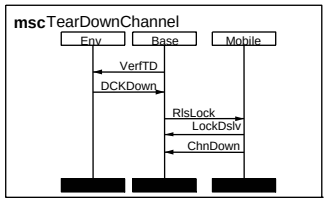

MSC reference
name of referenced MSC
Introduction
ASN.1
MSC
•Reference
SDL
TTCN
Summary
msc Ref_example
A
B
C
D
ab
MSC_ref
gate1
gate2
m1
m2
m3
cb
msc MSC_ref
A
C
ac1
m1
gate1
ack1
ok1
ack2
m2
gate2
ok2
m3
msc example_expanded
A
B
C
D
ab
ac1
m1
ack1
ok1
ack2
m2
ok2
m3
cb
gates
MSC_ref expanded
MSC reference
ISSSRE04 WITUL - Rennes 01 November 2004

Example of MSC reference use

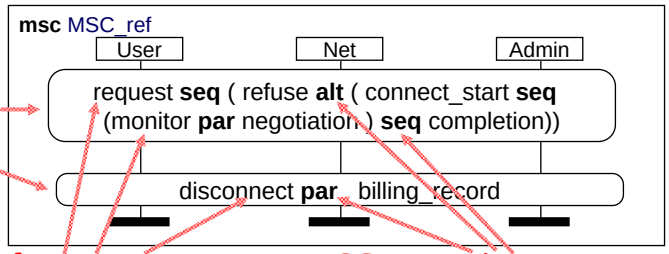








Reference expressions



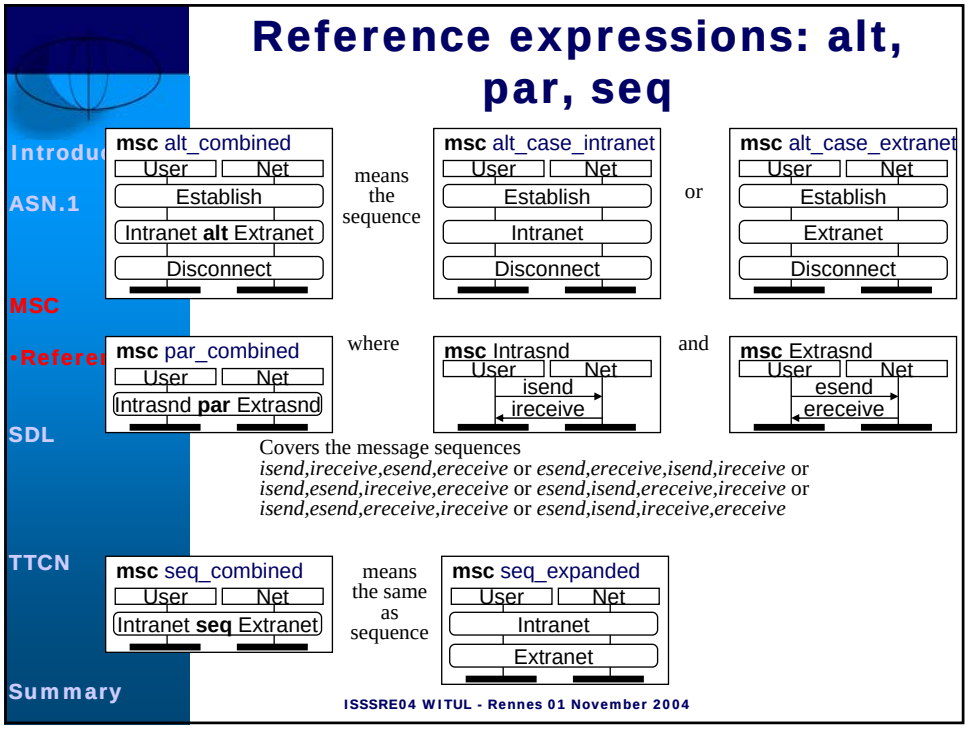
MSC reference expressions

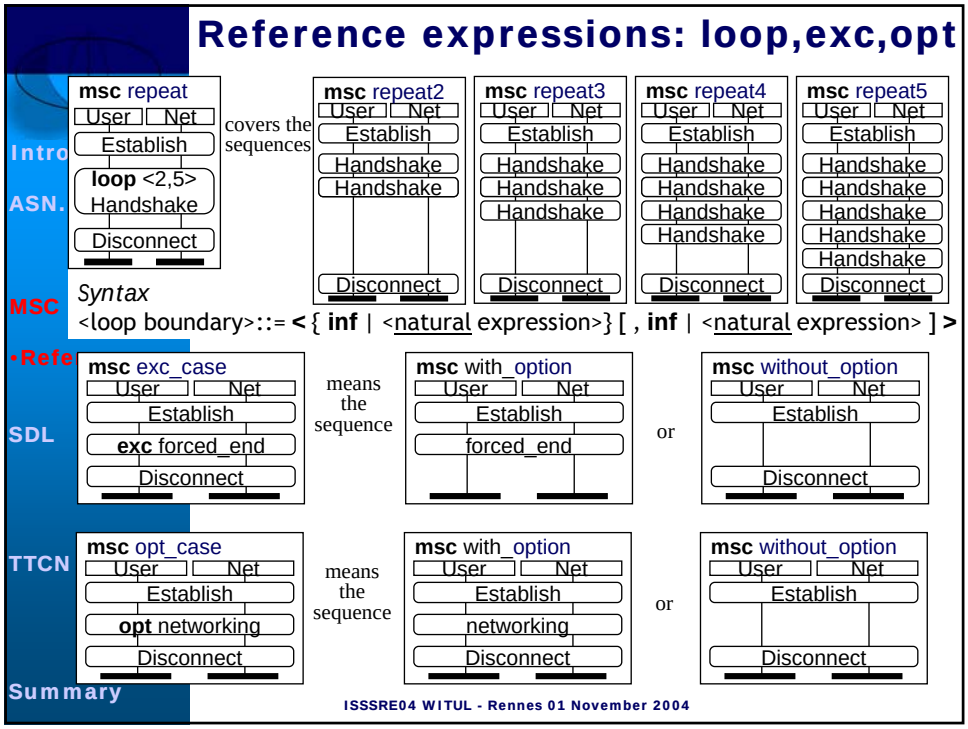
MSC reference names
(request, monitor, disconnect)

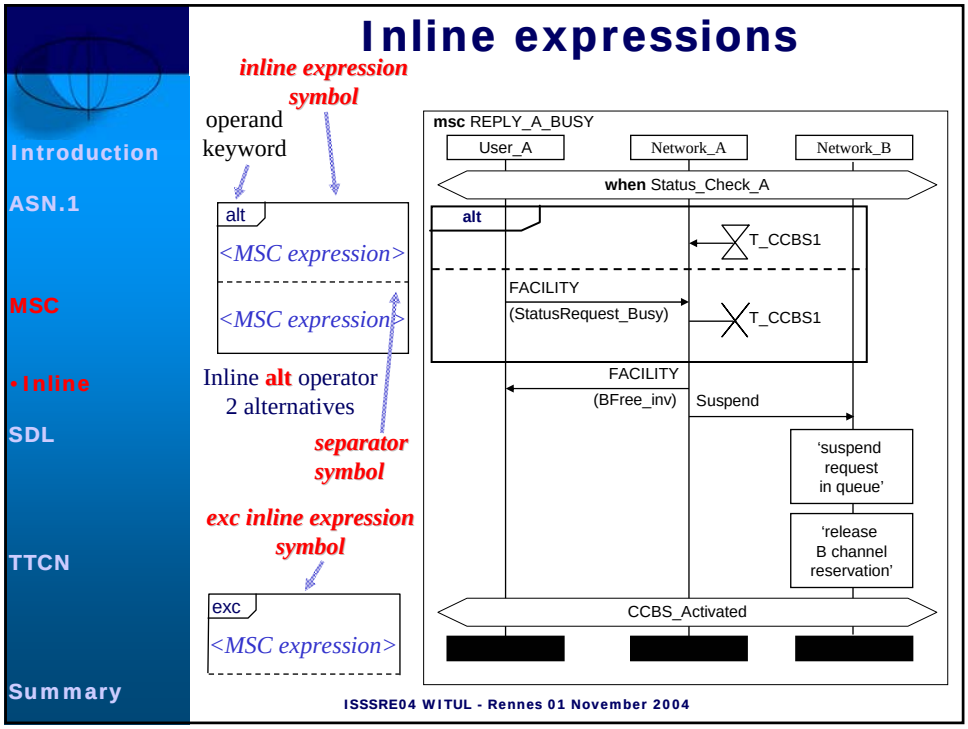
MSC expression operators
(**alt**, **par**, **seq**)

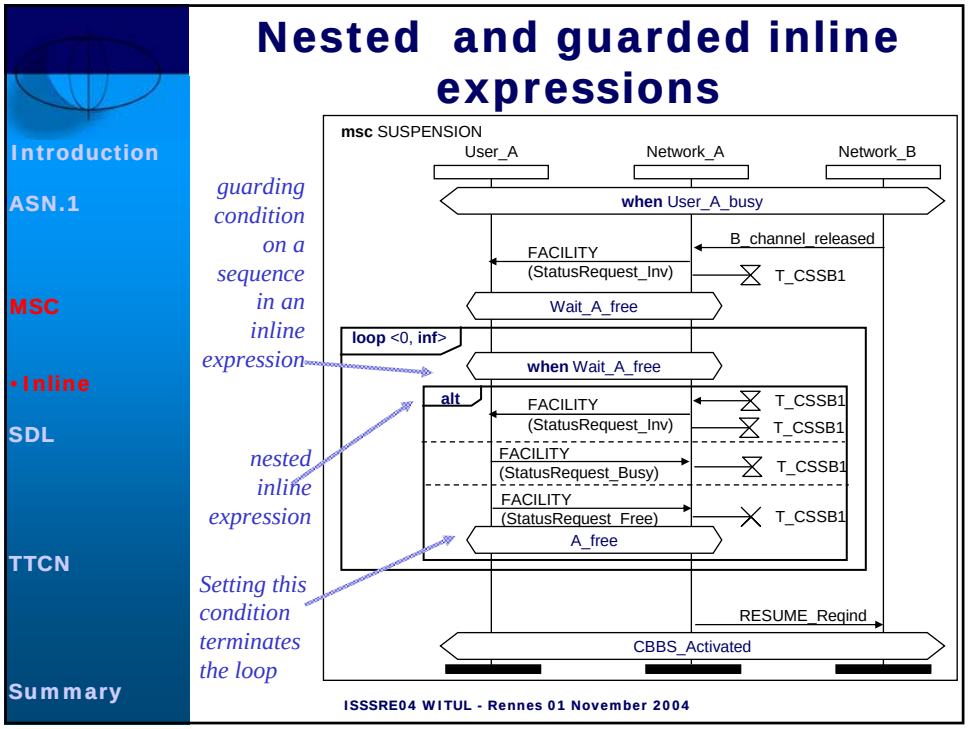
Syntax for msc reference (slightly simplified):

<msc reference area> ::= <msc reference symbol> *contains* <msc ref expr>
<msc ref expr> ::= <msc ref par expr> { **alt** <msc ref par expr> }*
<msc ref par expr> ::= <msc ref seq expr> { **par** <msc ref seq expr> }*
<msc ref seq expr> ::= <msc ref ident expr> { **seq** <msc ref ident expr> }*
<msc ref ident expr> ::= **loop** [<loop boundary>] <msc ref ident exp>
| **exc** <msc ref ident expr>
| **opt** <msc ref ident expr>
| <msc name> | **empty** | (<msc ref expr>)

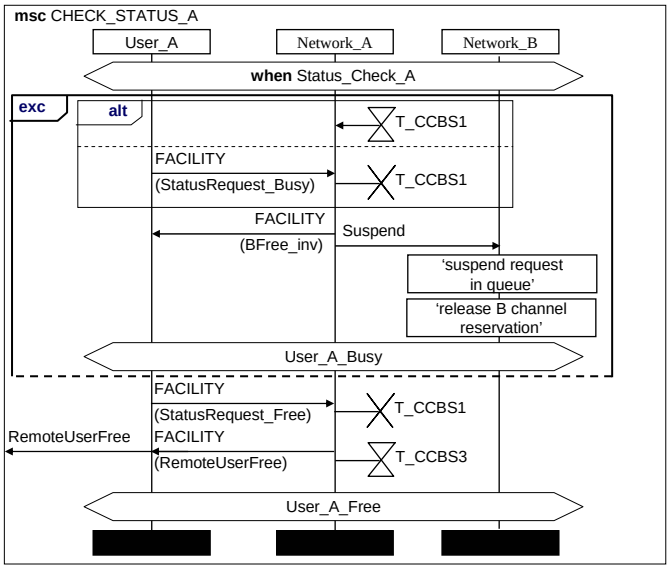

Reference expressions: alt, par, seq
Introdu
ASN.1
MSC
•Referer
SDL
TTCN
Summary
msc alt_combined
User
Net
Establish
Intranet alt Extranet
Disconnect
means the sequence
msc alt_case_intranet
User
Net
Establish
Intranet
Disconnect
or
msc alt_case_extranet
User
Net
Establish
Extranet
Disconnect
msc par_combined
User
Net
Intrasnd par Extrasnd
where
msc Intrasnd
User
Net
isend
ireceive
and
msc Extrasnd
User
Net
esend
ereceive
Covers the message sequences
isend,ireceive,esend,ereceive or esend,ereceive,isend,ireceive or
isend,esend,ireceive,ereceive or esend,isend,ereceive,ireceive or
isend,esend,ereceive,ireceive or esend,isend,ireceive,ereceive
msc seq_combined
User
Net
Intranet seq Extranet
means the same as sequence
msc seq_expanded
User
Net
Intranet
Extranet
ISSSRE04 WITUL - Rennes 01 November 2004


Reference expressions: loop,exc,opt
Intro
ASN.
MSC
•Refe
SDL
TTCN
Summary
msc repeat
User
Net
Establish
loop <2,5>
Handshake
Disconnect
covers the sequences
msc repeat2
User
Net
Establish
Handshake
Handshake
Disconnect
msc repeat3
User
Net
Establish
Handshake
Handshake
Handshake
Disconnect
msc repeat4
User
Net
Establish
Handshake
Handshake
Handshake
Handshake
Disconnect
msc repeat5
User
Net
Establish
Handshake
Handshake
Handshake
Handshake
Handshake
Disconnect
Syntax
<loop boundary>::= < { inf | <natural expression>} [, inf | <natural expression>] >
msc exc_case
User
Net
Establish
exc forced_end
Disconnect
means the sequence
msc with_option
User
Net
Establish
forced_end
or
msc without_option
User
Net
Establish
Disconnect
msc opt_case
User
Net
Establish
opt networking
Disconnect
means the sequence
msc with_option
User
Net
Establish
networking
Disconnect
or
msc without_option
User
Net
Establish
Disconnect
ISSSRE04 WITUL - Rennes 01 November 2004

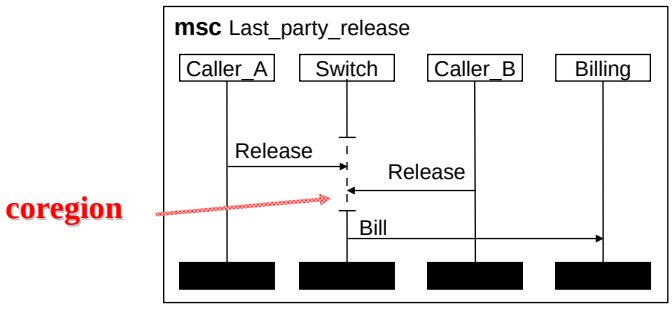

Introduction
ASN.1
MSC
•Inline
SDL
TTCN
Summary
Inline expressions
inline expression symbol
operand keyword
alt
<MSC expression>
<MSC expression>
Inline alt operator
2 alternatives
separator symbol
exc inline expression symbol
exc
<MSC expression>
msc REPLY_A_BUSY
User_A
Network_A
Network_B
when Status_Check_A
alt
T_CCBS1
FACILITY
(StatusRequest_Busy)
T_CCBS1
FACILITY
(BFree_inv)
Suspend
'suspend request in queue'
'release B channel reservation'
CCBS_Activated
ISSSRE04 WITUL - Rennes 01 November 2004


Introduction
ASN.1
MSC
•Inline
SDL
TTCN
Summary
Nested and guarded inline expressions
guarding condition on a sequence in an inline expression
nested inline expression
Setting this condition terminates the loop
msc SUSPENSION
User_A
Network_A
Network_B
when User_A_busy
B_channel_released
FACILITY
(StatusRequest_Inv)
T_CSSB1
Wait_A_free
loop <0, inf>
when Wait_A_free
alt
FACILITY
(StatusRequest_Inv)
T_CSSB1
T_CSSB1
FACILITY
(StatusRequest_Busy)
T_CSSB1
FACILITY
(StatusRequest_Free)
T_CSSB1
A_free
RESUME_Reqind
CBBS_Activated
ISSSRE04 WITUL - Rennes 01 November 2004

exc inline expression

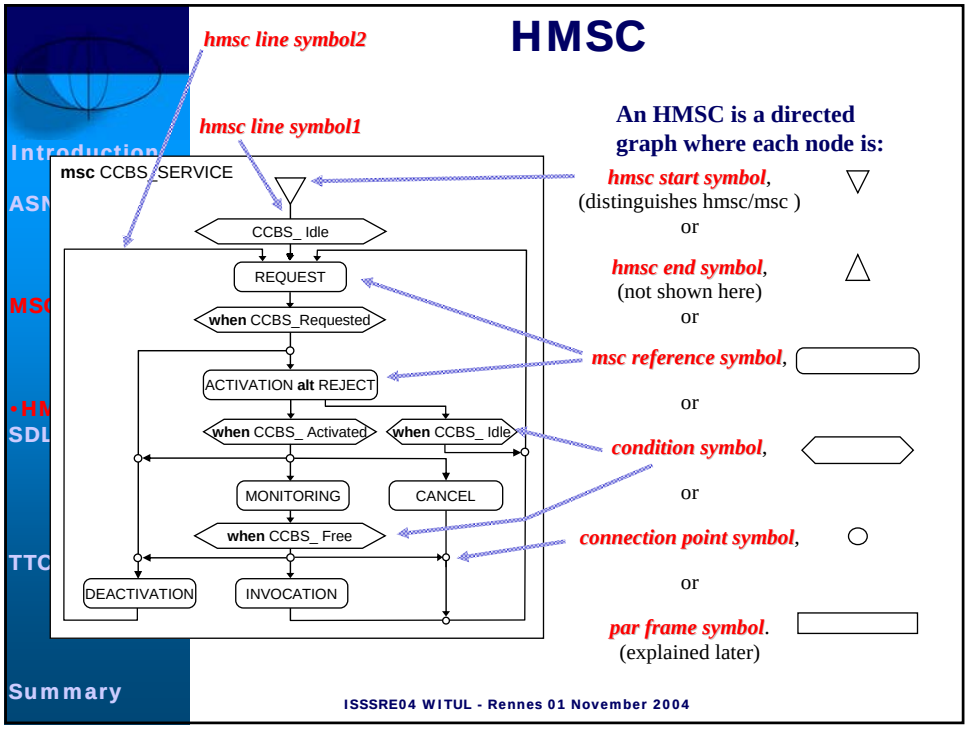


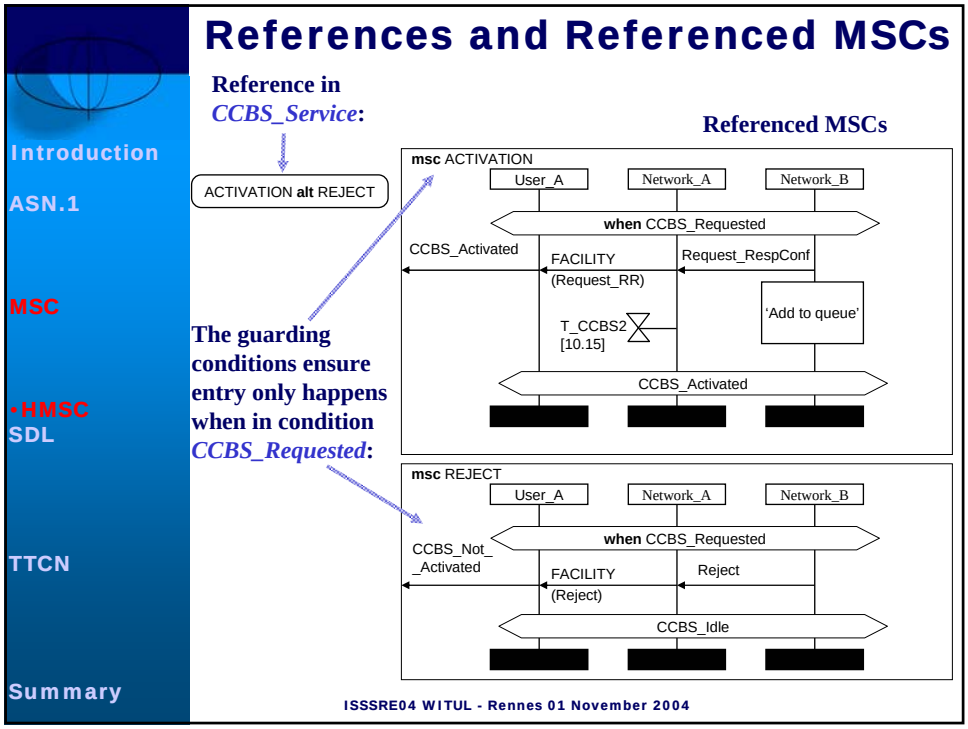
Simple coregion

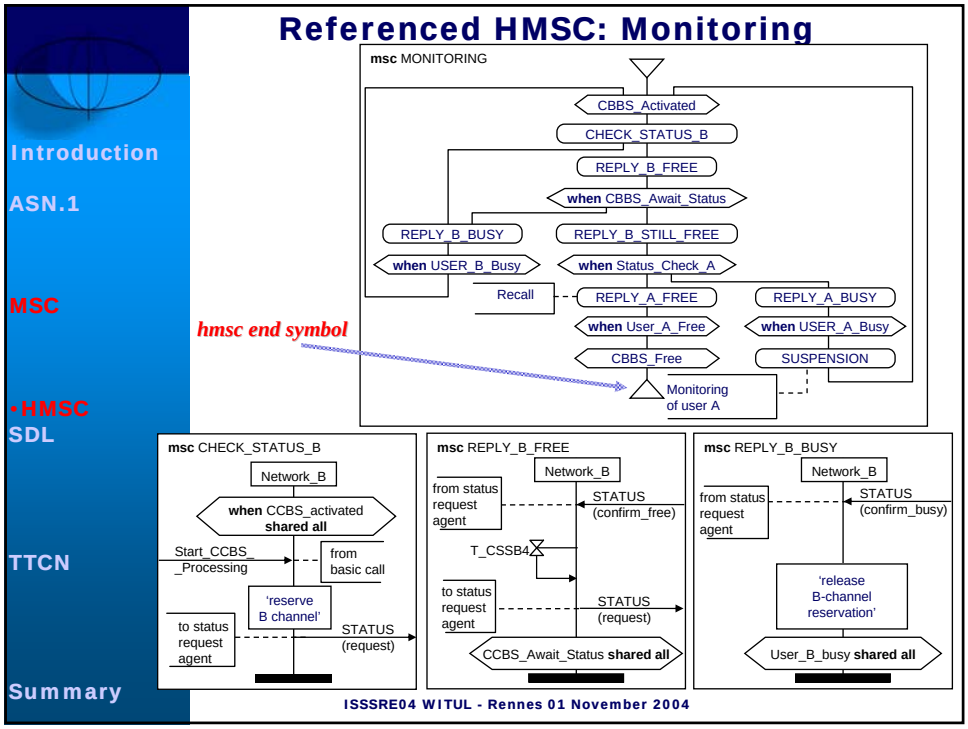


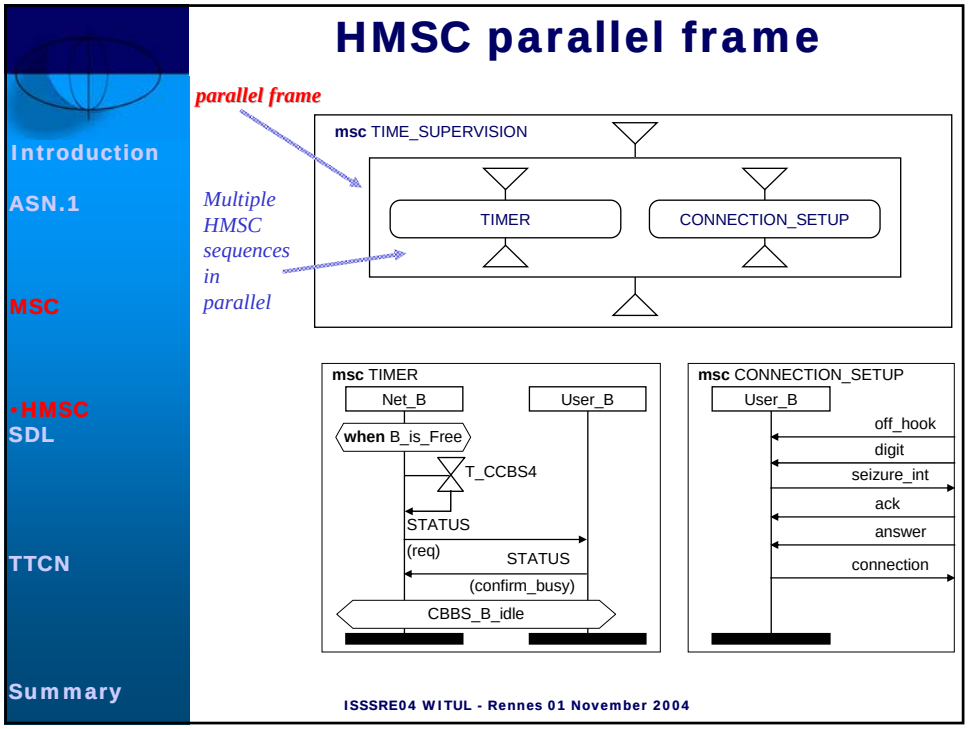
coregion

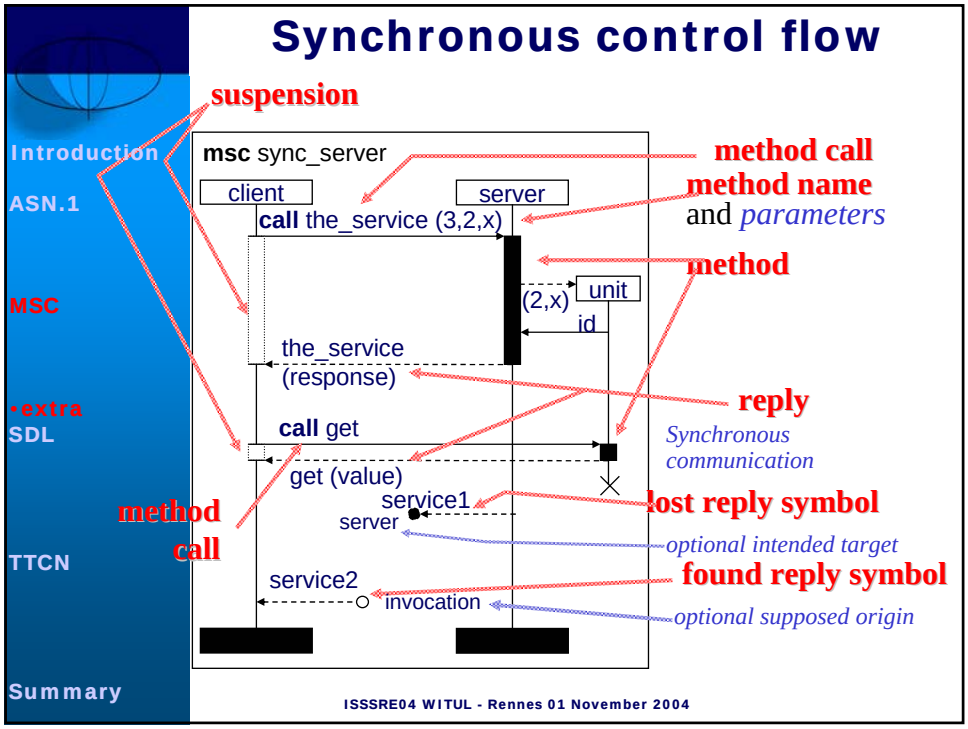
- A coregion on an instance axis indicates that any event order is allowed;
- All events in the coregion take place before any event after the coregion;
- In a simple coregion there is no causal relationship between events.

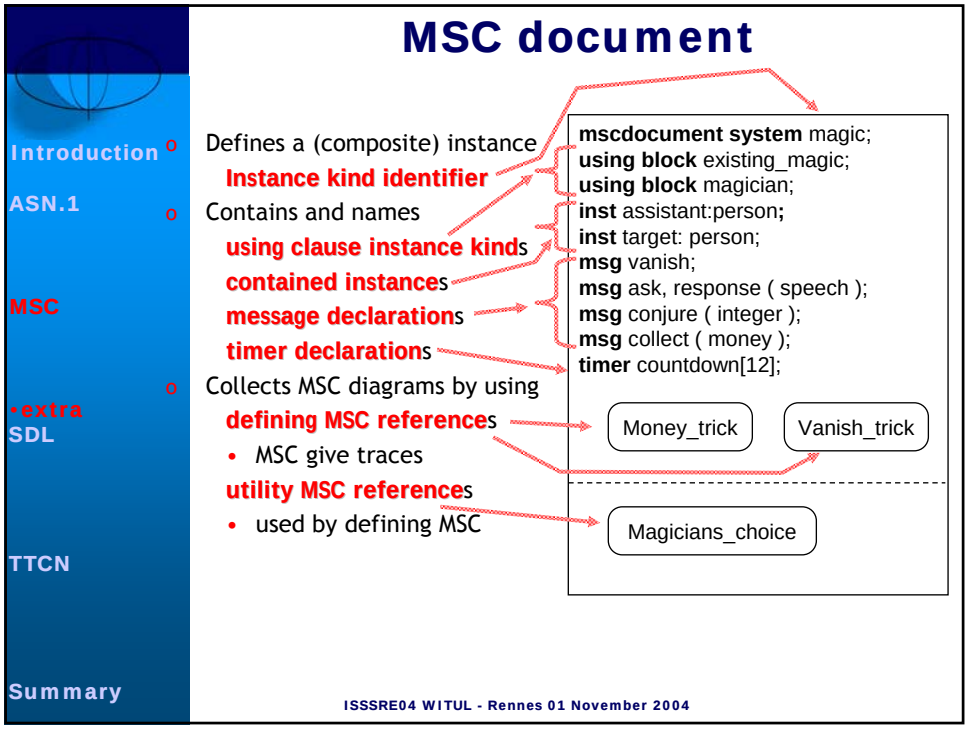

HMSC
hmsc line symbol2
hmsc line symbol1
msc CCBS_SERVICE
CCBS_ Idle
REQUEST
when CCBS_Requested
ACTIVATION alt REJECT
when CCBS_ Activated
when CCBS_ Idle
MONITORING
CANCEL
when CCBS_ Free
DEACTIVATION
INVOCATION
An HMSC is a directed graph where each node is:
hmsc start symbol, (distinguishes hmsc/msc)
or
hmsc end symbol, (not shown here)
or
msc reference symbol,
or
condition symbol,
or
connection point symbol,
or
par frame symbol. (explained later)
Introduction
ASN
MSC
•HM
SDL
TTC
Summary
ISSSRE04 WITUL - Rennes 01 November 2004


References and Referenced MSCs
Reference in CCBS_Service:
ACTIVATION alt REJECT
The guarding conditions ensure entry only happens when in condition CCBS_Requested:
Referenced MSCs
msc ACTIVATION
User_A
Network_A
Network_B
when CCBS_Requested
CCBS_Activated
FACILITY
(Request_RR)
Request_RespConf
'Add to queue'
T_CCBS2
[10.15]
CCBS_Activated
msc REJECT
User_A
Network_A
Network_B
when CCBS_Requested
CCBS_Not_ _Activated
FACILITY
(Reject)
Reject
CCBS_Idle
Introduction
ASN.1
MSC
•HMSC
SDL
TTCN
Summary
ISSSRE04 WITUL - Rennes 01 November 2004


Referenced HMSC: Monitoring
Introduction
ASN.1
MSC
HMSC
SDL
TTCN
Summary
hmsc end symbol
msc MONITORING
CBBS_Activated
CHECK_STATUS_B
REPLY_B_FREE
when CBBS_Await_Status
REPLY_B_BUSY
REPLY_B_STILL_FREE
when USER_B_Busy
when Status_Check_A
Recall
REPLY_A_FREE
REPLY_A_BUSY
when User_A_Free
when USER_A_Busy
CBBS_Free
SUSPENSION
Monitoring of user A
msc CHECK_STATUS_B
Network_B
when CCBS_activated shared all
Start_CCBS_Processing
from basic call
'reserve B channel'
to status request agent
STATUS (request)
msc REPLY_B_FREE
Network_B
from status request agent
STATUS (confirm_free)
T_CSSB4
to status request agent
STATUS (request)
CCBS_Await_Status shared all
msc REPLY_B_BUSY
Network_B
from status request agent
STATUS (confirm_busy)
'release B-channel reservation'
User_B_busy shared all
ISSSRE04 WITUL - Rennes 01 November 2004


HMSC parallel frame
Introduction
ASN.1
MSC
HMSC
SDL
TTCN
Summary
parallel frame
Multiple HMSC sequences in parallel
msc TIME_SUPERVISION
TIMER
CONNECTION_SETUP
msc TIMER
Net_B
User_B
when B_is_Free
T_CCBS4
STATUS (req)
STATUS (confirm_busy)
CBBS_B_idle
msc CONNECTION_SETUP
User_B
off_hook
digit
seizure_int
ack
answer
connection
ISSSRE04 WITUL - Rennes 01 November 2004


Synchronous control flow
Introduction
ASN.1
MSC
+extra
SDL
TTCN
Summary
suspension
msc sync_server
client
server
call the_service (3,2,x)
method call
method name
and parameters
method
unit
(2,x)
id
the_service
(response)
reply
call get
Synchronous
communication
get (value)
service1
server
lost reply symbol
optional intended target
method
call
service2
invocation
found reply symbol
optional supposed origin
ISSSRE04 WITUL - Rennes 01 November 2004


MSC document
Introduction
ASN.1
MSC
+extra
SDL
TTCN
Summary
Defines a (composite) instance
Instance kind identifier
Contains and names
using clause instance kinds
contained instances
message declarations
timer declarations
Collects MSC diagrams by using
defining MSC references
• MSC give traces
utility MSC references
• used by defining MSC
mscdocument system magic;
using block existing_magic;
using block magician;
inst assistant:person;
inst target: person;
msg vanish;
msg ask, response (speech);
msg conjure (integer);
msg collect (money);
timer countdown[12];
Money_trick
Vanish_trick
Magicians_choice
ISSSRE04 WITUL - Rennes 01 November 2004

Names and comments

The syntax for names is:

<name> ::= { <letter> | <decimal digit> | <underline> | <full stop> }+

but excluding keywords. Lowercase and uppercase are distinct.

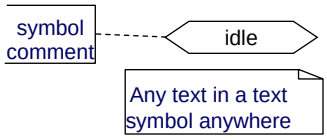
A name must be unique within its entity class
(MSC document, MSC, instance, condition, timer, message)

Together with meaningful names, four kinds of comment can be used to provide explanation:

- In text before a semi-colon

  <end> ::= [**comment** <character string>] ;

- Attached to a graphical symbol
- In a **text symbol** anywhere on a diagram
- In text as a /* **note** */



Other MSC features and relationship with UML & SDL

Other features

- Method Calls, Data, Formal Actions
- Time Intervals, Multiple Conditions
- MSC Document, Instance Decomposition

Relationships

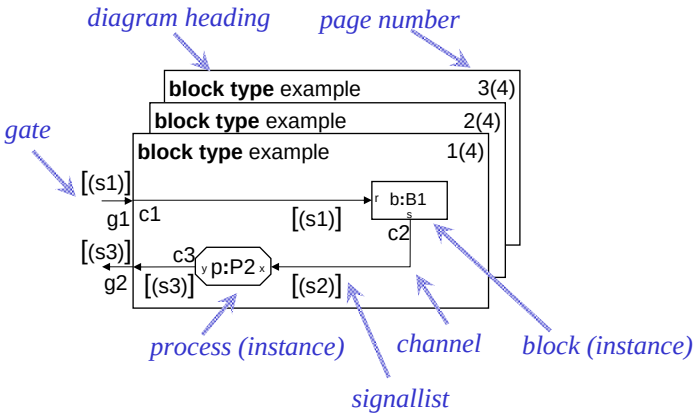
- Almost = UML Sequence Diagrams
- Data binding with SDL (but not ASN.1)
- Messages = SDL signals, Events = SDL?
- SDL timers weaker? No intervals in SDL
- Weak on multiple instances

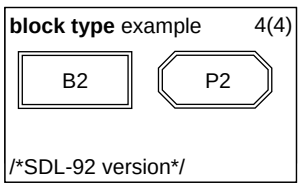
SDL

Specification and Description Language

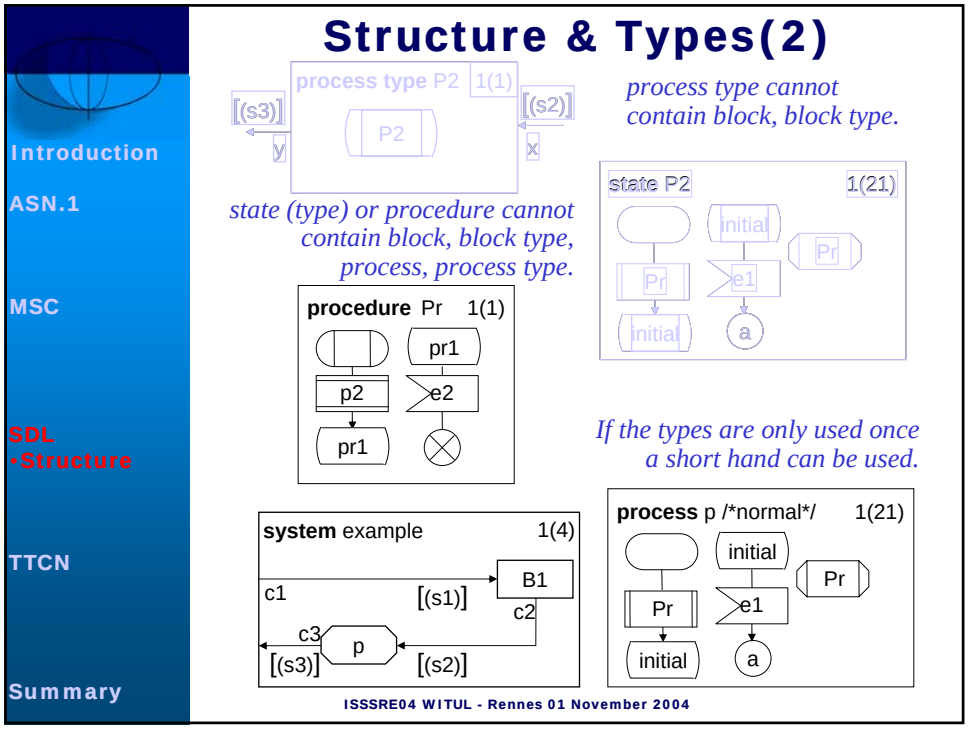
- Structure and types
  - for abstraction and information hiding, or as a requirement
  - support reuse of designs
- Behaviour
  - stimulus/response
  - sequence
  - timing
- Data
  - information structuring
  - meaning
- Interfaces
  - environment
  - communication paths
  - signals (messages)

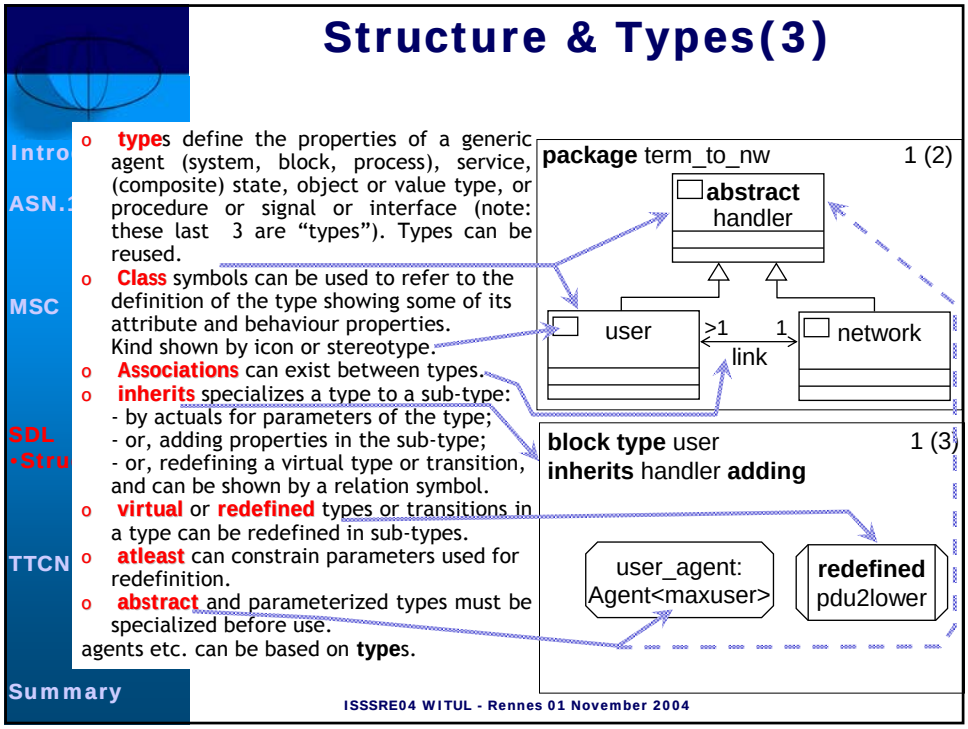
Structure & Types(1)





block type can contain block, block type, process, process type, other items, types …


Structure & Types(2)
Introduction
ASN.1
MSC
SDL
•Structure
TTCN
Summary
process type P2 1(1)
[(s3)]
[(s2)]
y
x
P2
process type cannot contain block, block type.
state P2 1(21)
initial
Pr
e1
a
state (type) or procedure cannot contain block, block type, process, process type.
procedure Pr 1(1)
pr1
p2
e2
pr1
If the types are only used once a short hand can be used.
system example 1(4)
B1
c1
[(s1)]
c2
c3
p
[(s3)]
[(s2)]
process p /*normal*/ 1(21)
initial
Pr
e1
a
ISSSRE04 WITUL - Rennes 01 November 2004

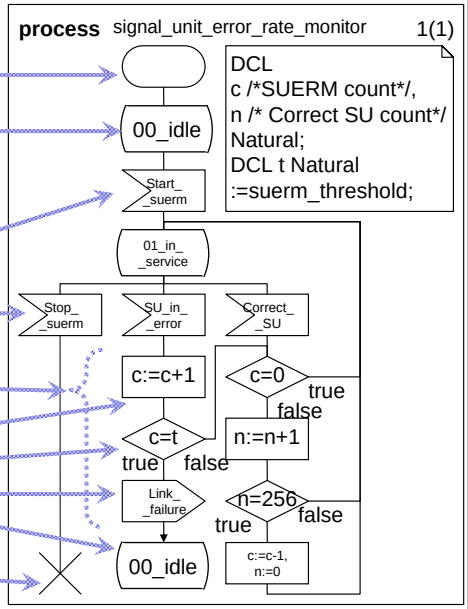

Structure & Types(3)
Intro
ASN.1
MSC
SDL
•Stru
TTCN
Summary
types define the properties of a generic agent (system, block, process), service, (composite) state, object or value type, or procedure or signal or interface (note: these last 3 are "types"). Types can be reused.
Class symbols can be used to refer to the definition of the type showing some of its attribute and behaviour properties. Kind shown by icon or stereotype.
Associations can exist between types.
inherits specializes a type to a sub-type:
- by actuals for parameters of the type;
- or, adding properties in the sub-type;
- or, redefining a virtual type or transition,
and can be shown by a relation symbol.
virtual or redefined types or transitions in a type can be redefined in sub-types.
atleast can constrain parameters used for redefinition.
abstract and parameterized types must be specialized before use.
agents etc. can be based on types.
package term_to_nw 1 (2)
abstract handler
user
>1
1
link
network
block type user 1 (3)
inherits handler adding
user_agent: Agent<maxuser>
redefined pdu2lower
ISSSRE04 WITUL - Rennes 01 November 2004

Key feature: Behaviour

Introd
ASN.1
MSC
SDL
•Beha
TTCN
Summary

Extended finite state machine:EFSM

- **Start** (symbol) followed by initialisation going to
- **State** (symbol) where the machine waits until an
- **Input** stimulus (a **Signal**) of the state as defined by the attached **Input Symbol**s is available in the input queue.
- the **Transition**, to the next State consumes the first such signal and interprets its actions such as each **Task** (symbol) or **Decision** (symbol) or **Output** (symbol) sending a signal leading to the **NextState** or a
- **Stop** (symbol) terminating the process



Key SDL-2000 feature: Data

Introduc
ASN.1
MSC
SDL
•Data
TTCN
Summary

Used in
- Variables (owned by processes)
- Parameters (for example in signals)

Built-in with defined operations
- Boolean, Character, Charstring, Integer, Natural, Real, Duration, Time, Bit, Bitstring, Octet, Octetstring, Pid (agent references)

Build-in parameterised with operations
- Strings (lists) of any type (not just characters) indexed by Naturals
- Arrays of any type indexed by any type
- Structures (records) with optional (and default) fields
- Choice Structure with all fields optional
- Powerset, Bag (maths. set and bag)

User defined sorts of data with operations
- Object type - elements are references
- Value type - elements are values
- Syntype - check on range of values

```
object type Linkedlist
    <type Elementsort>
    struct
        prev, next this Linkedlist;
        data Elementsort;
    operators
    "in" (Elementsort, Linkedlist)
        ->Boolean;
    methods
    delete (Elementsort);
    operator "in" referenced;
    method delete referenced;
endobject type Linkedlist;
object type Natlist
    inherits Linkedlist <Natural>
endobject type Natlist;
dcl primes Natlist
    := (. Null, Null, 1 .);
```

Example: Bit-stuffing System

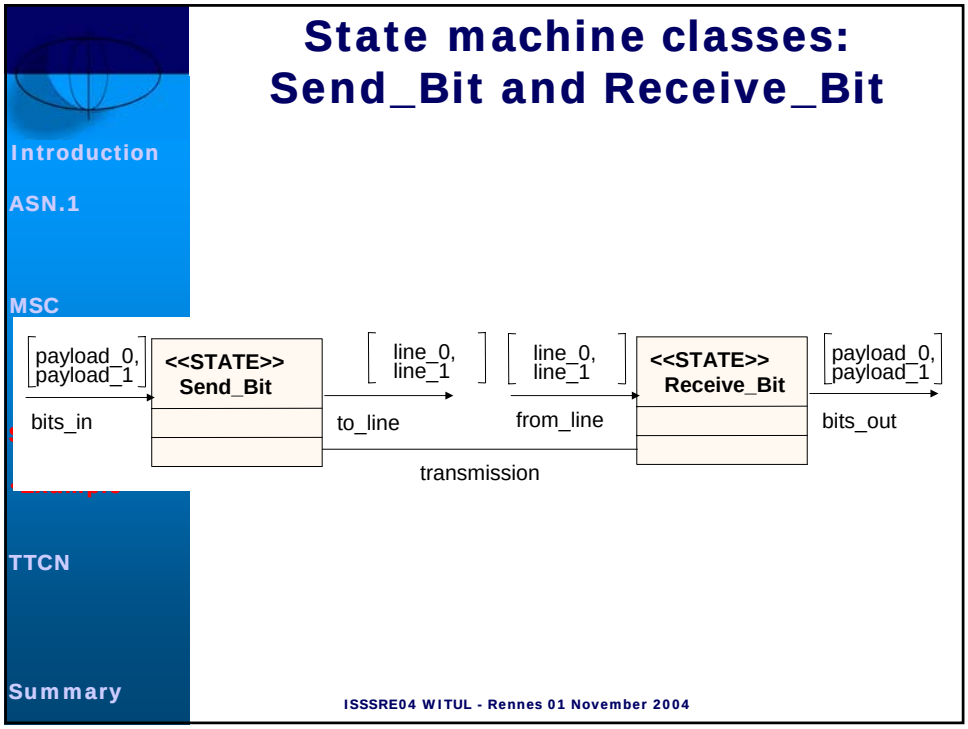
Bits are received one at a time for link transmission over a 100% reliable medium and are delivered at the other end of the link one at a time. When the system is ready to send a bit, but there is no bit ready to send, 5 identical bits are sent as a filler. The fillers alternate between ones and zeros. If the last payload bit sent was a one, a zeros filler is sent, if it was a zero - a ones filler. If no bit has ever been sent either filler can be sent.

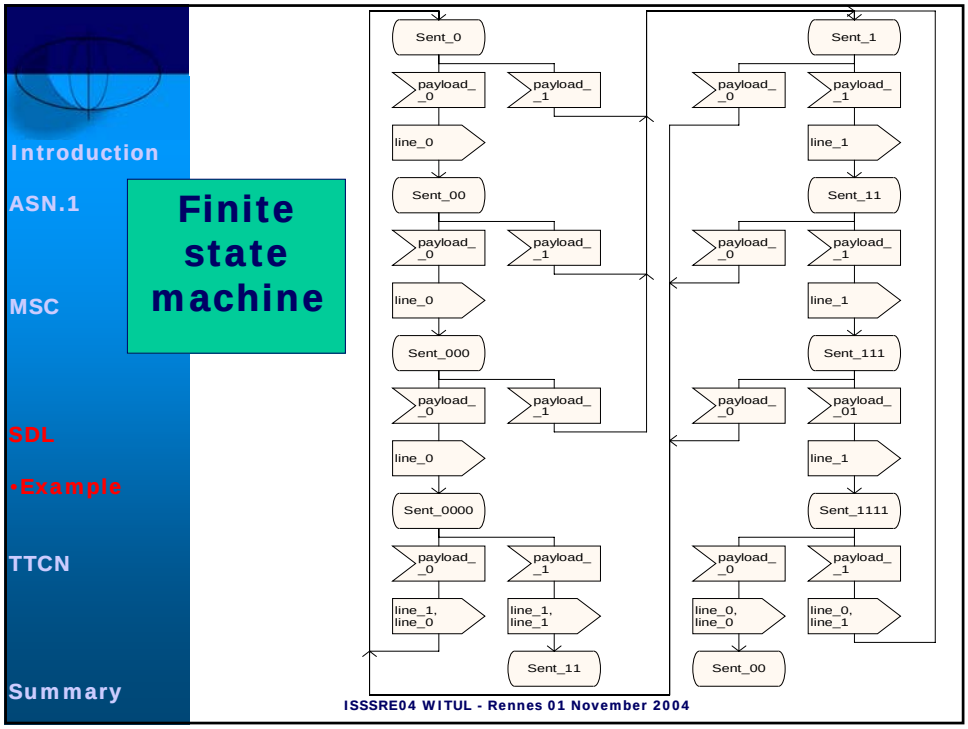
If there are 4 consecutive identical payload bits, a extra bit is inserted before the next bit to avoid the payload being interpreted as a filler.

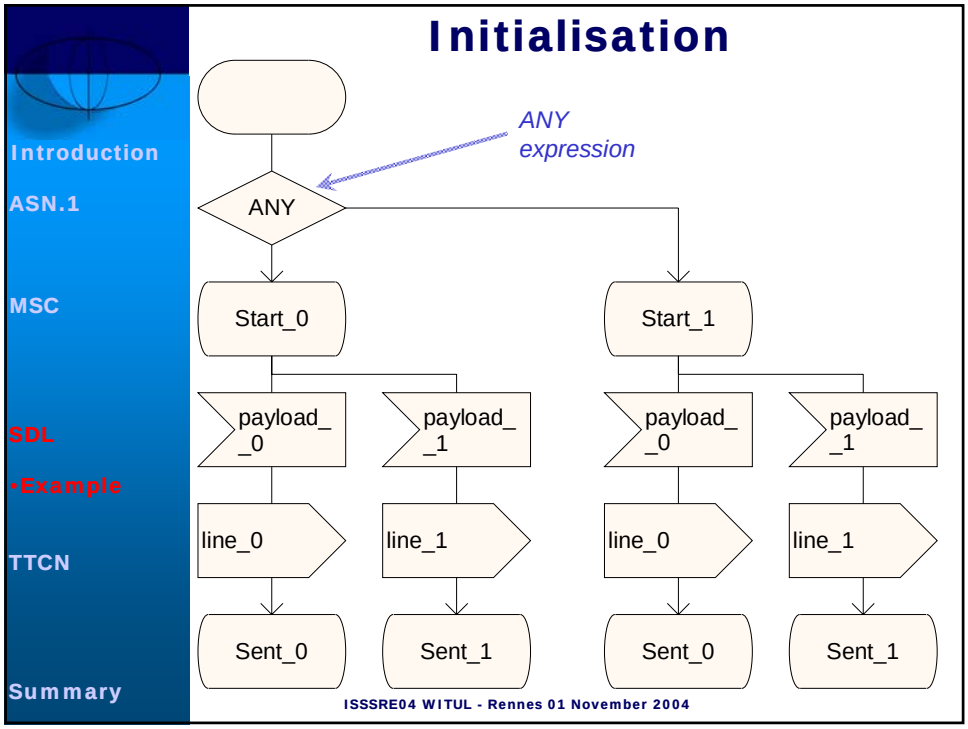
Signals

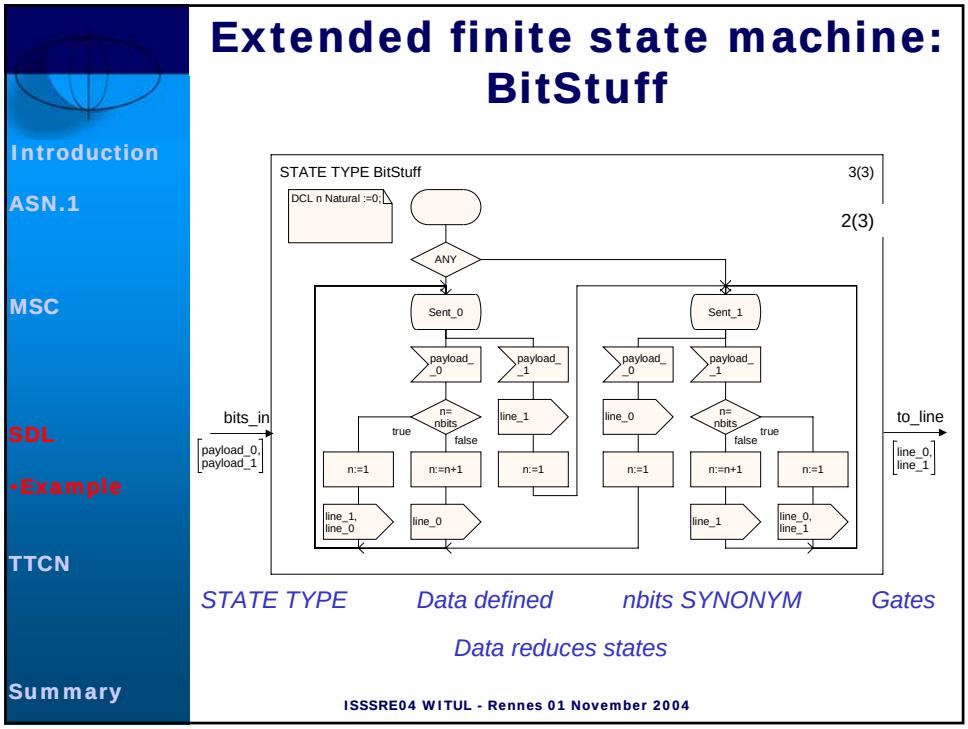
```
SIGNAL    payload_0,
          payload_1;
SIGNAL    line_0,
          line_1;

/* The payload_0 signal
represents a zero Bit
to be transmitted, and the
payload_1 signal represents a
one Bit.
Similarly, the line_0 signal
represents a zero Bit sent
the line, and line_1 signal
represents a one Bit. */
```

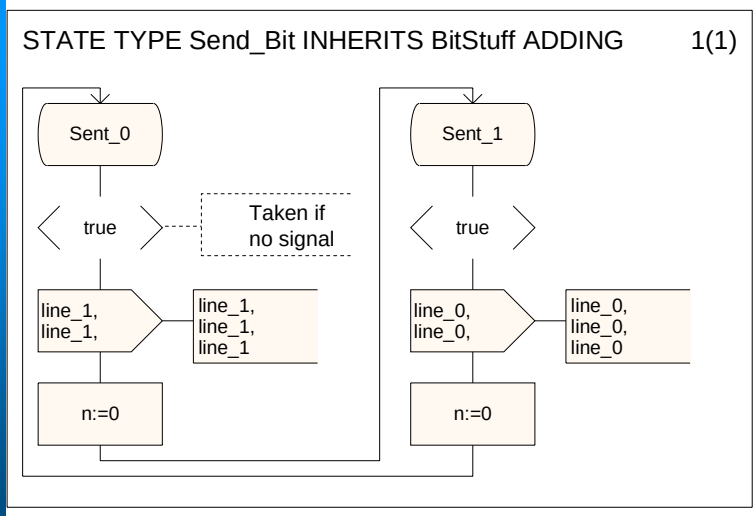

State machine classes:
Send_Bit and Receive_Bit
Introduction
ASN.1
MSC
TTCN
Summary
payload_0,
payload_1
<<STATE>>
Send_Bit
bits_in
line_0,
line_1
to_line
line_0,
line_1
from_line
<<STATE>>
Receive_Bit
payload_0,
payload_1
bits_out
transmission
ISSSRE04 WITUL - Rennes 01 November 2004


Introduction
ASN.1
MSC
SDL
•Example
TTCN
Summary
Finite state machine
Sent_0
payload_0
payload_1
line_0
Sent_00
Sent_000
Sent_0000
line_1, line_0
line_1, line_1
Sent_11
Sent_1
line_1
Sent_11
Sent_111
payload_01
Sent_1111
line_0, line_0
line_0, line_1
Sent_00
ISSSRE04 WITUL - Rennes 01 November 2004


Initialisation
Introduction
ASN.1
MSC
SDL
•Example
TTCN
Summary
ANY
expression
ANY
Start_0
Start_1
payload_ _0
payload_ _1
payload_ _0
payload_ _1
line_0
line_1
line_0
line_1
Sent_0
Sent_1
Sent_0
Sent_1
ISSSRE04 WITUL - Rennes 01 November 2004

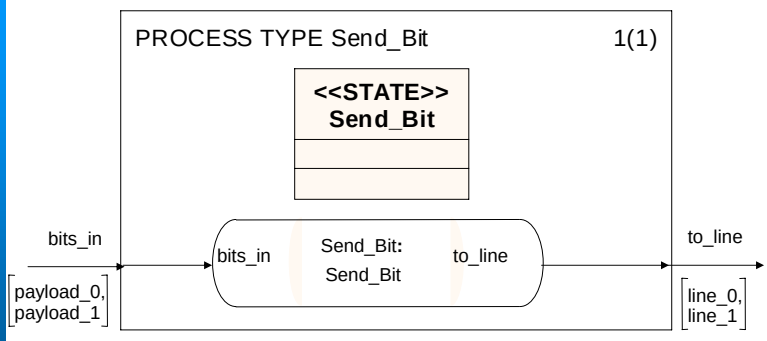

Extended finite state machine: BitStuff
Introduction
ASN.1
MSC
SDL
•Example
TTCN
Summary
STATE TYPE BitStuff
3(3)
2(3)
DCL n Natural :=0;
ANY
Sent_0
Sent_1
payload_ _0
payload_ _1
n:=1
n:=n+1
line_1
line_0
bits_in
payload_0, payload_1
to_line
line_0, line_1
STATE TYPE
Data defined
nbits SYNONYM
Gates
Data reduces states
ISSSRE04 WITUL - Rennes 01 November 2004

Type inheritance: Send_Bit



INHERITANCE *Continuous Signal* *Comment* *Text extension*

Process Type with type-based State

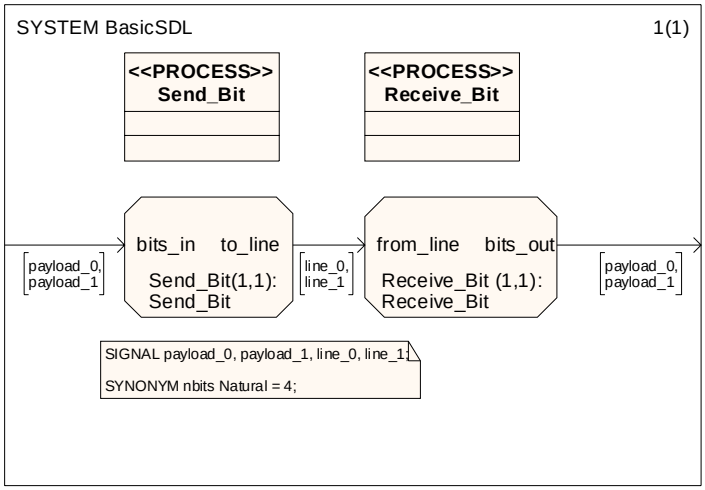


Reference to state type

Channels without names or signals

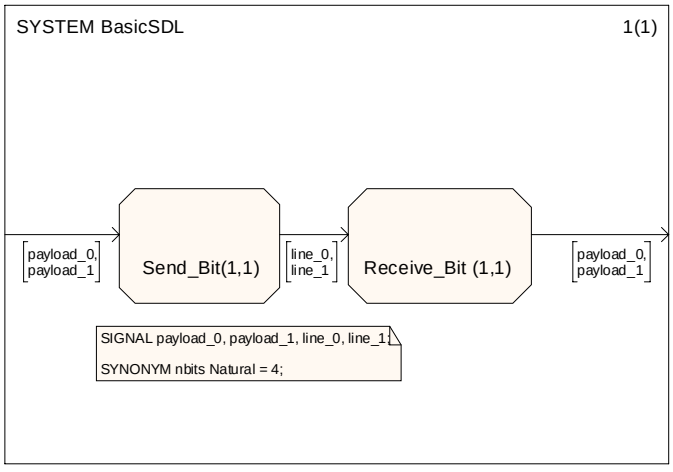
Use of gates on type-based state machine of process

Agent instances based on process types



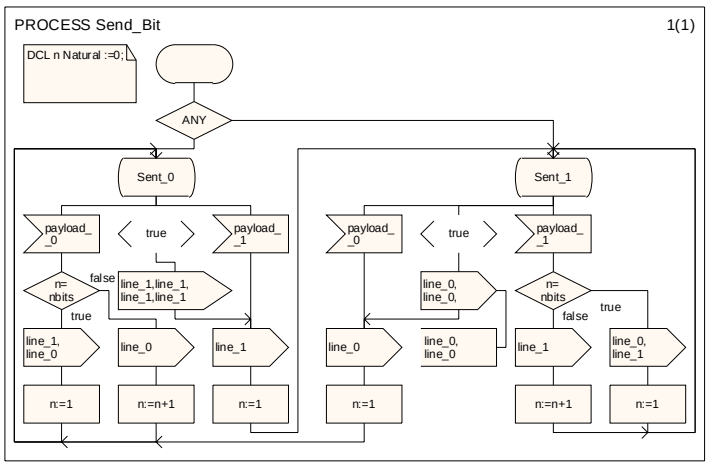
Process in system *signal definitions* *Synonym*

Removing process types



A process that is not type based has an implied process type of the same name.

Removing the composite state



Language relationships

- Specify - Implement - Test?
- Supports ASN.1 - adds operations
- Support for data in tools? Unified data?
- Adds action semantics to UML
- Some possible variations/additions e.g.
  - Message guarding, VIA Gate
  - Remote procedure as basic stimulus
- Too complex? Too weak?
- Time intervals - performance issues

Introduction
ASN.1
MSC
SDL
TTCN
Summary

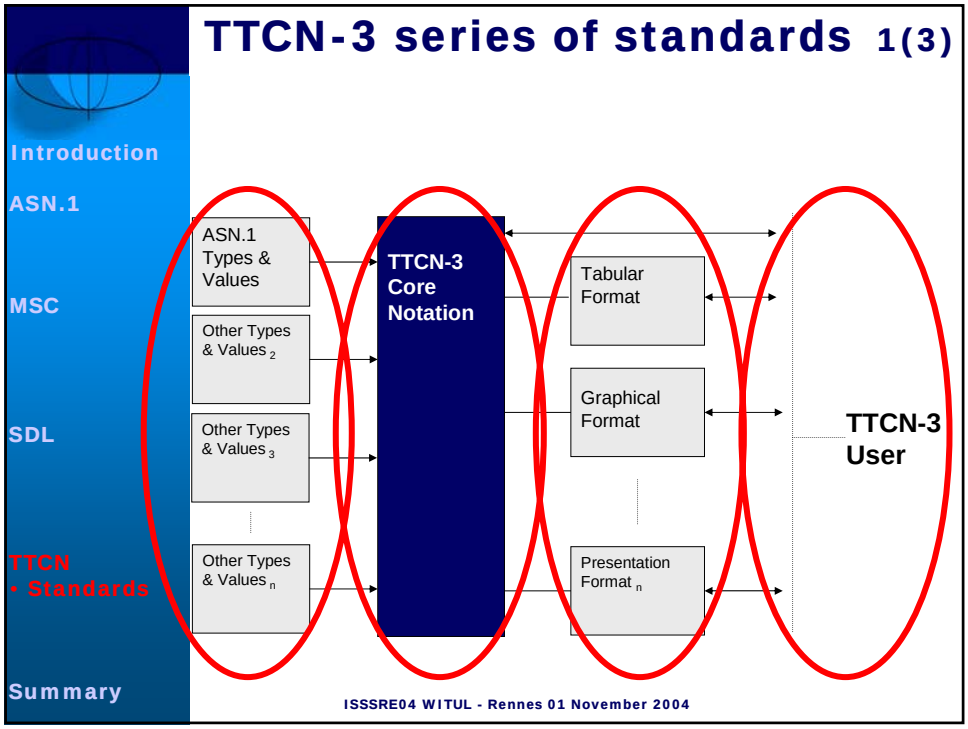
TTCN-3
Testing and Test Control Notation

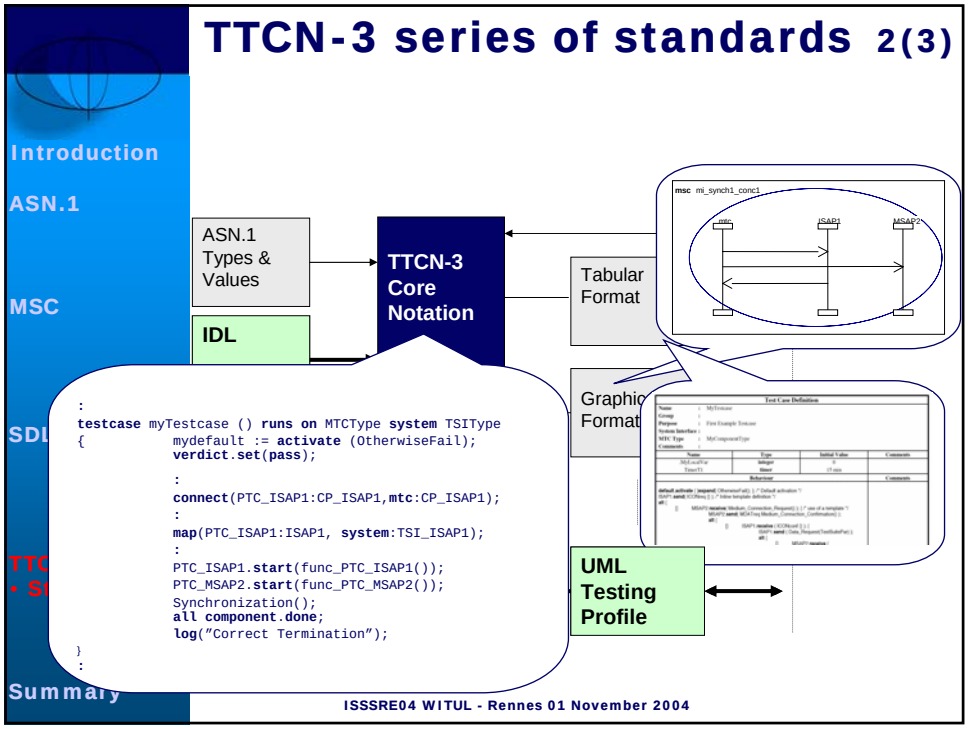
- Contents
  - What is TTCN-3 ?
  - The TTCN-3 series of standards
  - Concepts
  - Structure of TTCN-3 specs
  - TTCN-3 extensions under discussion
  - Conclusions and tool providers

Introduction
ASN.1
MSC
SDL
TTCN
• Standards
Summary

What is TTCN-3 ?

- The standardised (black-box) test specification and test implementation language.
- Developed
  - by the European Telecommunications Standards Institute (ETSI) from 1999 to 2001.
  - based on the experiences from previous TTCN versions.
- Applicable for all kinds of black-box testing for reactive and distributed systems, e.g.
  - Telecom systems (ISDN, ATM);
  - Mobile (telecom) systems (GSM, UMTS);
  - Internet (has been and is applied to IPv6);
  - CORBA based systems.

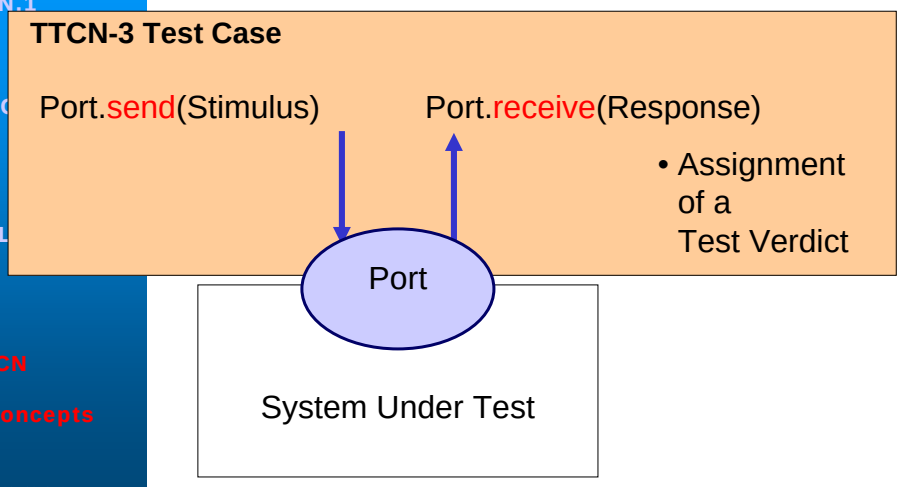

TTCN-3 series of standards 1(3)
Introduction
ASN.1
MSC
SDL
TTCN
• Standards
Summary
ASN.1 Types & Values
Other Types & Values 2
Other Types & Values 3
Other Types & Values n
TTCN-3 Core Notation
Tabular Format
Graphical Format
Presentation Format n
TTCN-3 User
ISSSRE04 WITUL - Rennes 01 November 2004


TTCN-3 series of standards 2(3)
Introduction
ASN.1
MSC
SDL
TTC
• St
Summary
ASN.1 Types & Values
IDL
TTCN-3 Core Notation
Tabular Format
Graphic Format
UML Testing Profile
:
testcase myTestcase () runs on MTCType system TSIType
{ mydefault := activate (OtherwiseFail);
verdict.set(pass);
:
connect(PTC_ISAP1:CP_ISAP1,mtc:CP_ISAP1);
:
map(PTC_ISAP1:ISAP1, system:TSI_ISAP1);
:
PTC_ISAP1.start(func_PTC_ISAP1());
PTC_MSAP2.start(func_PTC_MSAP2());
Synchronization();
all component.done;
log("Correct Termination");
}
:
ISSSRE04 WITUL - Rennes 01 November 2004

TTCN-3 series of standards 3(3)

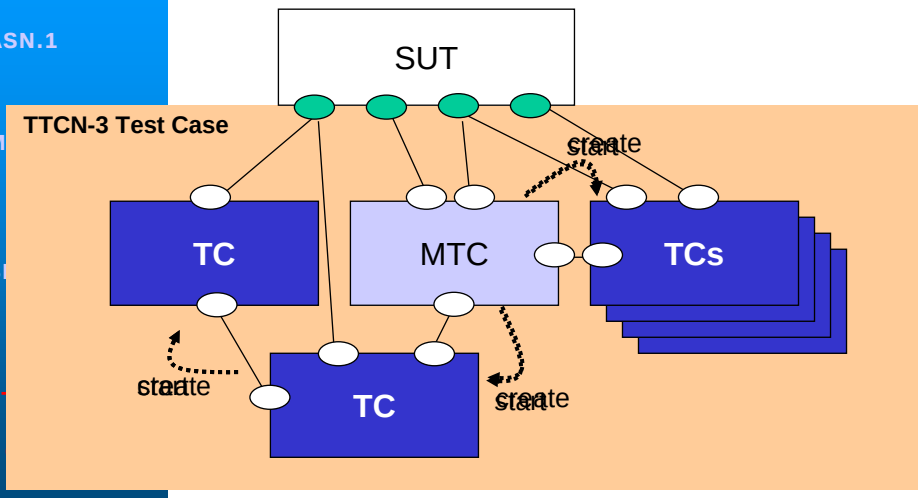
- European Standard (ES) in 10 parts
  - ES 201 873-1: TTCN-3 Core Language
  - ES 201 873-2: TTCN-3 Tabular Presentation Format (TFT)
  - ES 201 873-3: TTCN-3 Graphical Presentation Format (GFT)
  - ES 201 873-4: TTCN-3 Operational Semantics
  - ES 201 873-5: TTCN-3 Runtime Interface (TRI)
  - ES 201 873-6: TTCN-3 Control Interface (TCI)
  - ES 201 873-7: Using ASN.1 with TTCN-3
  - ES 201 873-8: Using IDL with TTCN-3
  - ES 201 873-9: Using XML with TTCN-3
  - ES 201 873-10: Using C/C++ with TTCN-3 (planned)

Black-box testing with TTCN-3



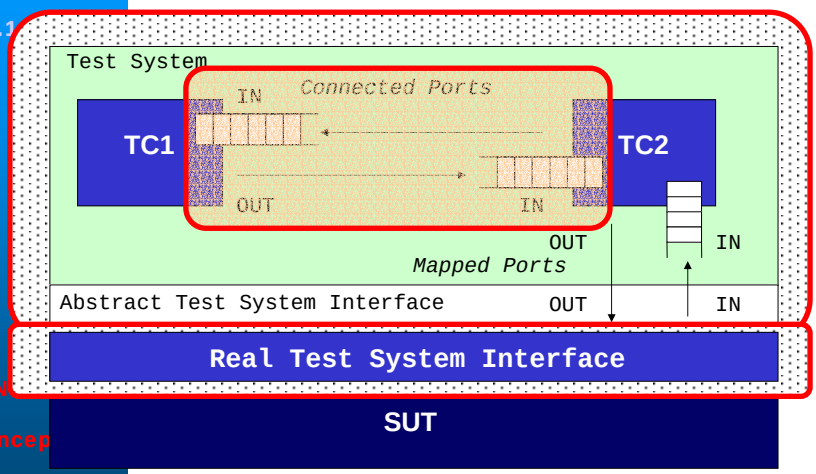
Test configuration 1(2)

Introduction
ASN.1
M
S
T
•
Summary



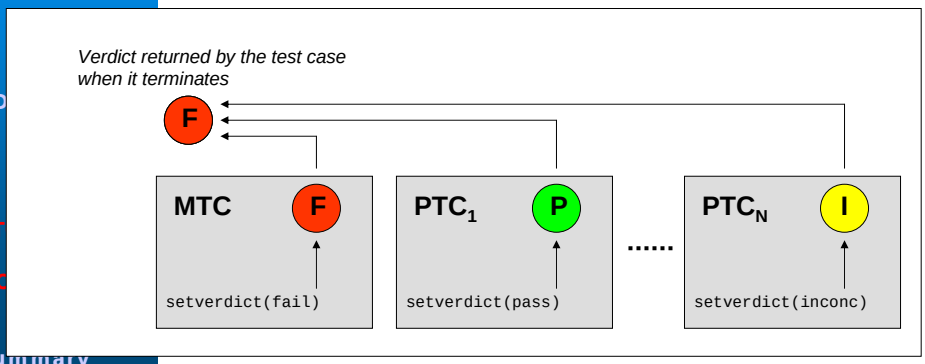
Test configuration 2(2)

Introduction
ASN.1
MSC
SDL
TTCN
• Concep
Summary

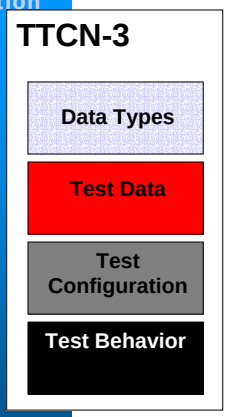


Test verdicts

- Test verdicts: none < pass < inconc < fail < error
- Each test component has its own local verdict, which can be set and read.
- A test case returns a global verdict

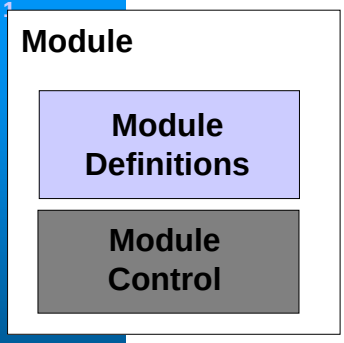


Ingredients of TTCN-3



- Built-in and user-defined generic data types (e.g., to define messages, service primitives, information elements, PDUs).
- Actual test data transmitted/received during testing.
- Definition of the components and communication ports that are used to build various testing configurations.
- Specification of the dynamic test system behavior.

Structure of TTCN-3 specs – TTCN-3 Modules 1(2)



- Modules are the building blocks of all TTCN-3 test specifications.
- A test suite is a module.
- A module has a definitions part and an (optional) control part.
- Modules can be parameterized.
- Modules can import definitions from other modules.

Structure of TTCN-3 specs – TTCN-3 Modules 2(2)

```
module Example {
  modulepar {
    integer Par_One, Par_Two;
    boolean Par_Three := true
  }
  import from AnotherModule {
    …
  }
  … // all definitions

control {
  … // execution of test cases
}
}
```

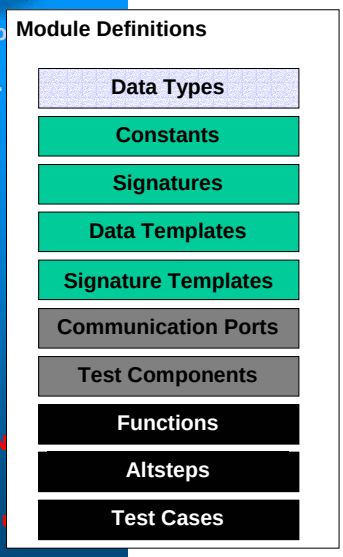
Module parameter definitions with and without default value

Import statement

Definitions part

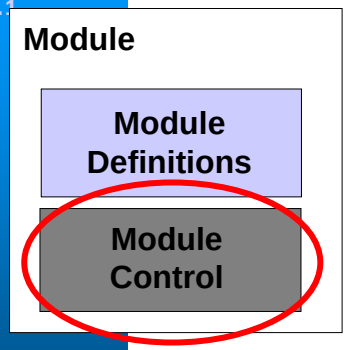
Control part

Structure of TTCN-3 specs – Module definitions part



- Module definitions are global to the entire module.
- Data Type definitions are based on TTCN-3 predefined and structured types.
- Templates define the test data.
- Ports and Components are used in Test Configurations.
- Functions, Altsteps and Test Cases define behavior.

Structure of TTCN-3 specs – Module control part 1(2)



- Module control is the 'dynamic' part of a TTCN-3 specification where the test cases are executed.
- Local declarations, such as variables and timers may be made in the control part.
- Basic programming statements may be used to select and control the execution of the test cases.

Structure of TTCN-3 specs – Module control part 2(2)


```
module … {
…
  control{
    var integer count;
    if(execute(SIP_UA_REC_V_001()) == pass) {
      // Execute test case 10 times
      count := 0;
      while( count <= 10) {
                  execute(SIP_UA_REC_V_002());
                  count := count + 1;
      } // end while
    } // end if
  } // end control
} // end module
```


TTCN-3 extensions
Work items in standardization

- Language extensions mechanisms
  - Packages and profiles
- Extended communication mechanisms
  - Broadcast / multicast (edition 3.0.0)
  - Synchronization / coordination (edition 3.0.0)
- Real-time extensions
  - Absolute time support
  - Time-constrained operations
- Better performance testing support
  - Implicit test configuration
  - Implicit communication
  - Performance measurement

New edition 3.0.0 of TTCN-3 in 2005

Introduction
ASN.1
MSC
SDL
TTCN
• Future
Summary

Conclusions

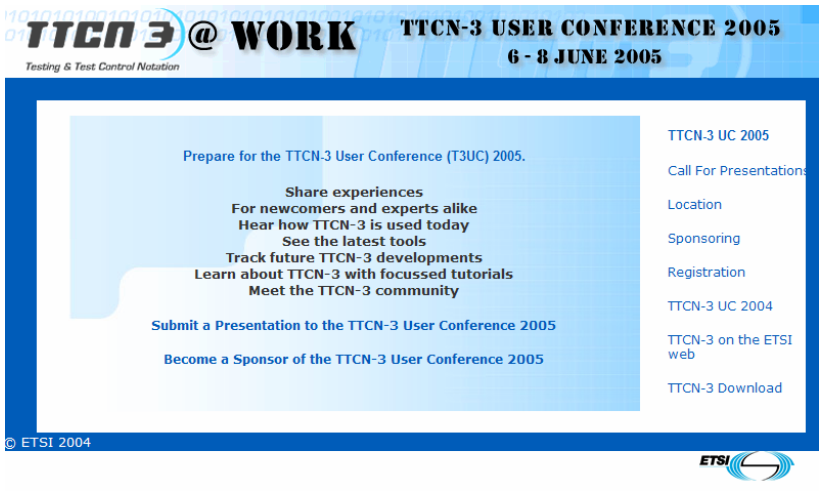
- TTCN-3 finds its way into practice
- Lots of interest from industry and academia
- Mature TTCN-3 standard
- Stimulates further research & development
  - New application domains, e.g. automotive
  - Real-time and performance testing
  - Test patterns
- Still possibilities to shape the future TTCN-3
  - TTCN-3 mailing list and change requests (see http://www.etsi.org/ptcc/ptccttcn3.htm)
  - TTCN-3 homepage: http://www.ttcn3.org

Introduction
ASN.1
MSC
SDL
TTCN
• Future
Summary

Current TTCN-3 tool providers

- Tool providers
  - Danet
  - DaVinci Communication
  - Open TTCN
  - Telelogic
  - Testing Technologies
  - Strategic Test Solutions
- Internal tools used by
  - Nokia
  - Ericsson
  - Motorola
- Existing test devices with TTCN-3 support (for telecommunication applications)
  - Alcatel A1100
  - Navtel InterWatch
  - Nethawk
  - Tektronix G20

TTCN-3 User Conference 2005
http://www.ttcn-3.org/



Work shop discussion

Although the ITU-T languages are used together there is scope for harmonisation

- Will the planned UML profiles help?
- What is the impact of meta-languages?
- ASN.1 for everything?
- ITU-T & OMG: partners or competitors?
- What methodology is appropriate?
- How much automation?
- How is reliability/security assessed?