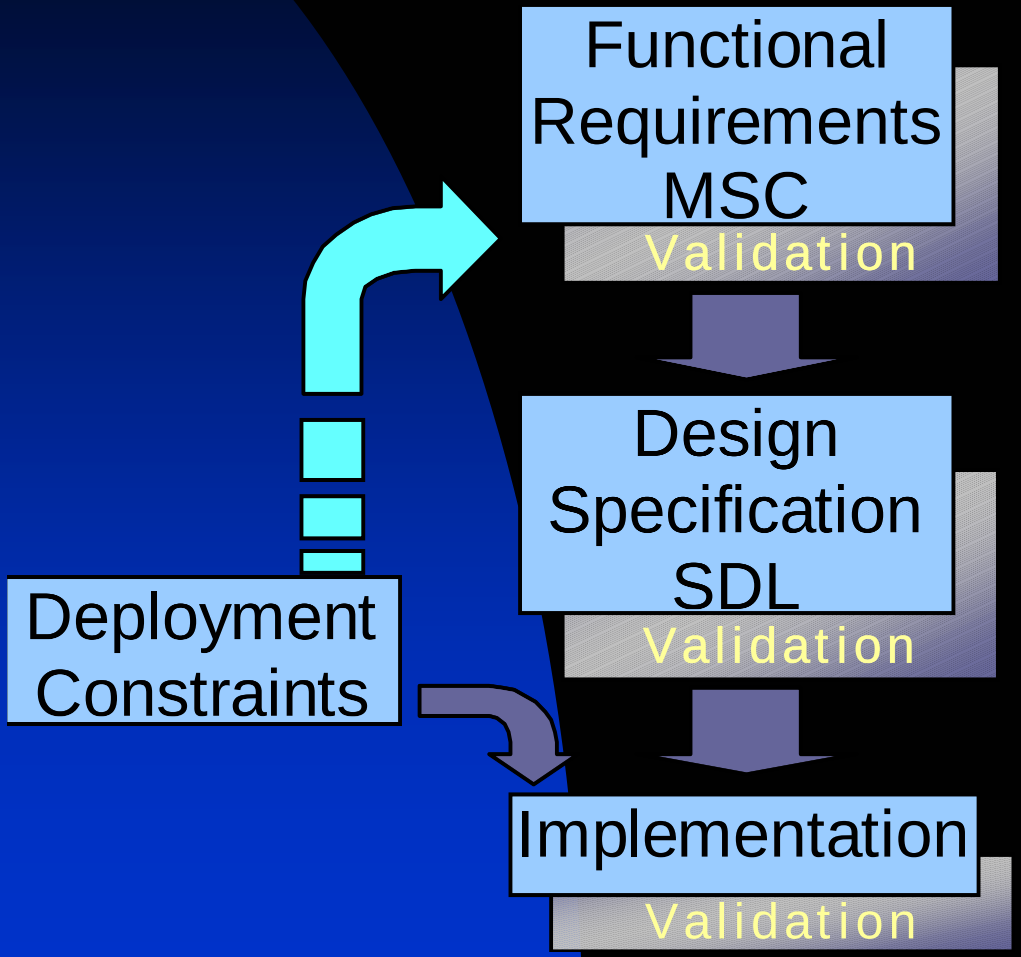
Early Validation of Deployment and Scheduling Constraints for MSC Specifications

Ferhat Khendek, Christophe Lohr, Li Xin Wang,
Xiao Jun Zhang, Tong Zheng

Concordia University

Motivation

Development Process

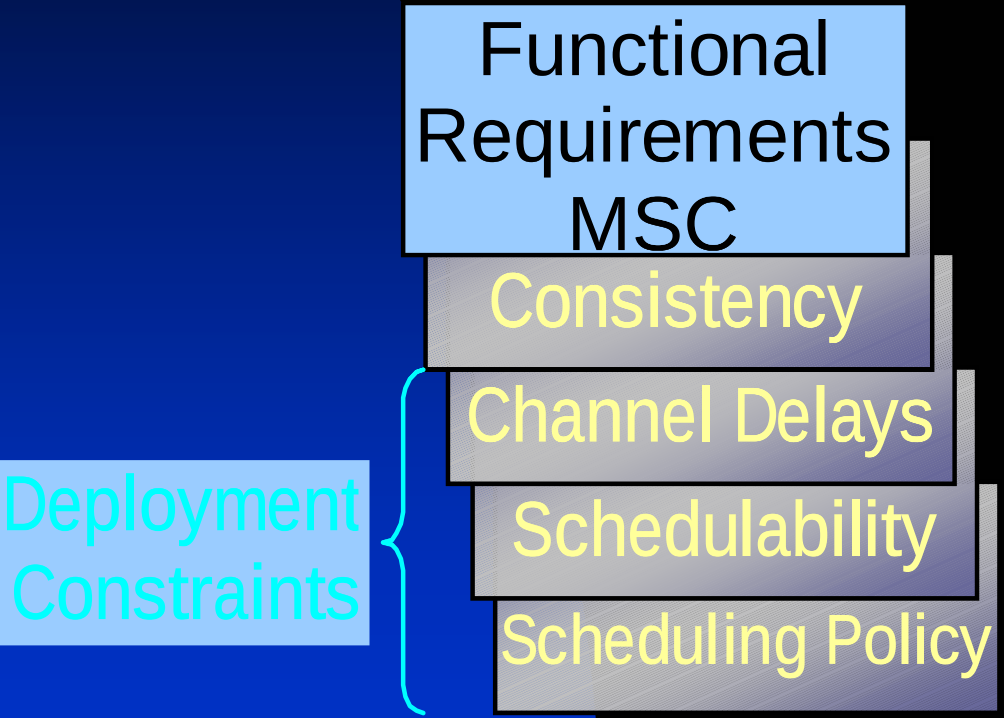


- MSC
  - Logical and time constraints
  - Functional requirements
  - Consistency validation
- SDL
  - Design
  - Validation
- Implementation
  - Add deployment constraints
  - Test cases (MSC)

→ Early validation of Deployment Constraints

Functional Requirements Validation

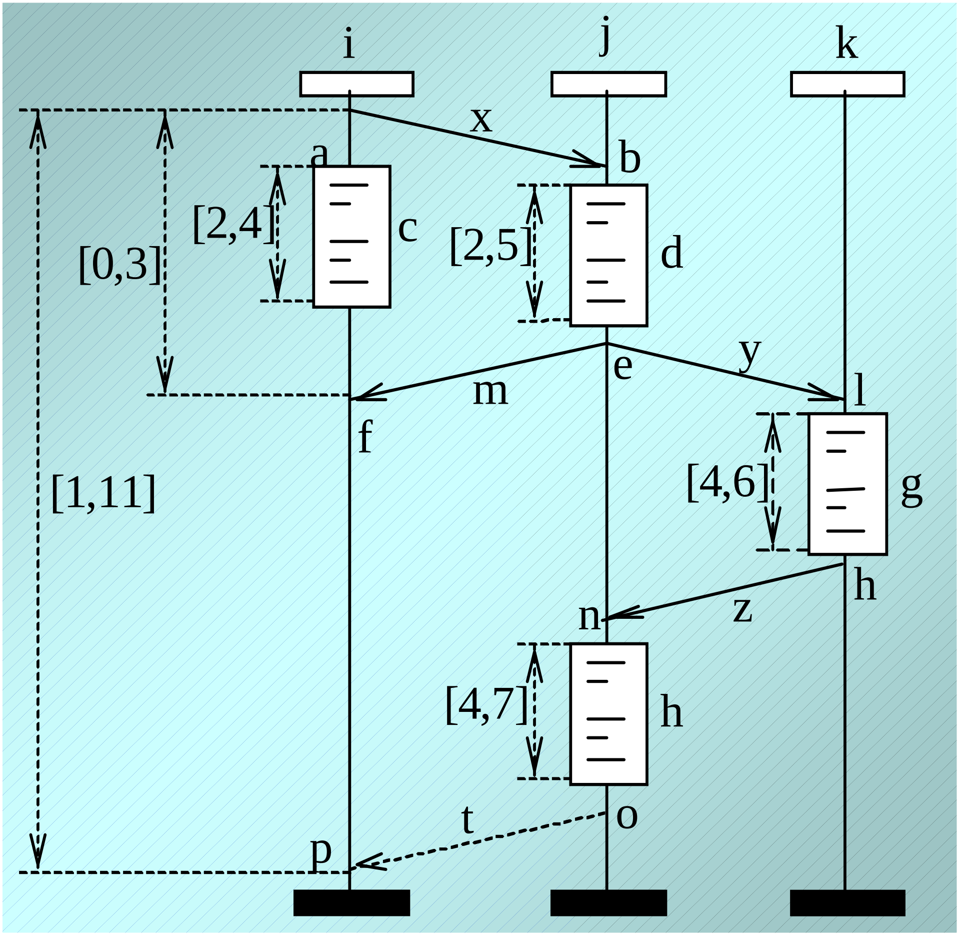
Stepwise Validation of MSCs



- Consistency
  - Intrinsic requirements consistency (time & logical)
  - Lposets semantics & validation
- Channel Delays
  - Are message channels fast enough to meet requirements?
- Processes Distribution
  - Are processes schedulable (can they meet their constraints) if they share a processor?
- Scheduling Policy
  - Are scheduled processes able to follow a given scheduling policy and meet functional requirements?

Example

Functional Requirements & Deployment Constraints



- i an j assigned to CPU1
- k assigned to CPU2
- Maximum channel delay between CPU1 and CPU2: 3
- Maximum channel delay inside CPU1&2: 1

➔ *Not deployable*

- Action boxes c & d in sequence
- Needs more than 4 units of time
- Violates the constraint [0,3]

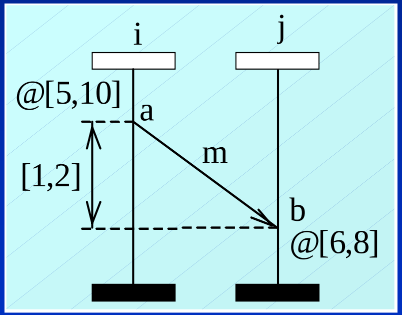
Presentation Overview

1. MSC Consistency
2. Channel Delay
3. Processes Distribution
4. Scheduling policy
5. Conclusion

1. Consistency of Timed MSCs

- ***Previous work as a basis of current work*** -

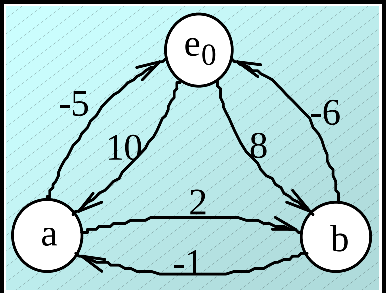
- Timed MSCs semantics based on lposets
- Consistency = all time and causal order are respected
- Validation to avoid semantic errors (timing & order conflicts)
- Validation technique:



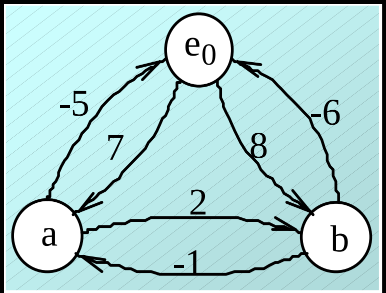
MSC

↱	a	b
a		T
b	F	

Event Order Table



Distance Graph



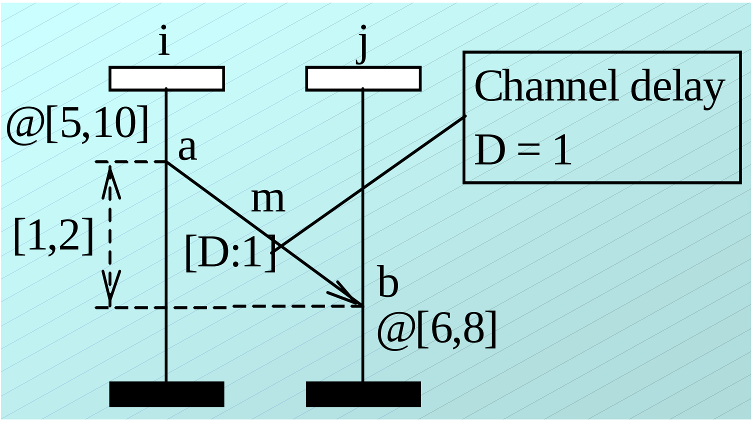
Floyd-Warshall Algorithm

a@[5,7] b@[6,8]

Reduced absolute time constraints

2. Communication Channel Delay

- Ensure that physical communication channels are fast enough to meet the functional timing requirements
- E.g. channels inside CPU or between CPUs



- Require delivering m within [1,2]
- Channel capability: 1

➔ Deployable

Algorithm:

❶ Read computed distance graph

❷ Compare (send-receive) relative time constraints to channel delay capability

❸ If greater, then abort:

"system not deployable"

3. Processes Distribution

- Ensure that processes distributed on a same CPU can share it and still meet their functional time requirements

Serializing events impacts the functional requirements…

- ◆ Try all possible serializations / schedules of events on each CPU
- ◆ Revalidate consistency for each one
- ◆ If one is consistent, processes are schedulable / deployable on this CPU

Main issue: Serialization

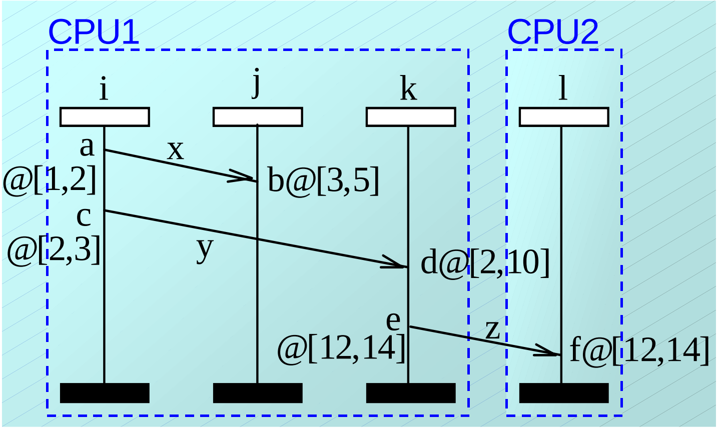
- Totally orders events in CPUs
- Add new orders compatible with existing ones

3. Processes Distribution

Serialization Algorithm

Event Order Table

↱	a	b	c	d	e	f
a		T	T	T	T	T
b	F		?	?	?	?
c	F	?		T	T	T
d	F	?	F		F	T
e	F	?	F	T		T
f	F	?	F	F	F	



Algorithm:

❶ Replace '?' by 'T' or 'F'

❷ Compute transitive closure

❸ Run F-W algo. if totally ordered, else continue

Output: list of consistent serializations

A serialization = new reduced absolute time constraints

Example (after 4 iterations, 2 serializations):

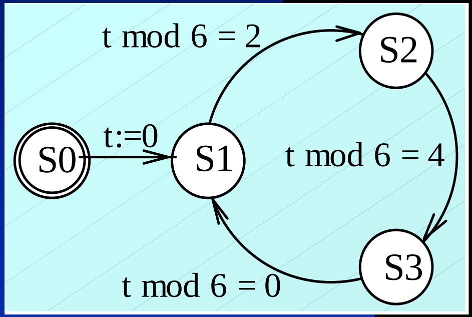
a@[1,2] c@[2,3] b@[3,5] d@[4,10] e@[12,14]
a@[1,2] c@[2,3] d@[3,4] b@[4,5] e@[12,14]

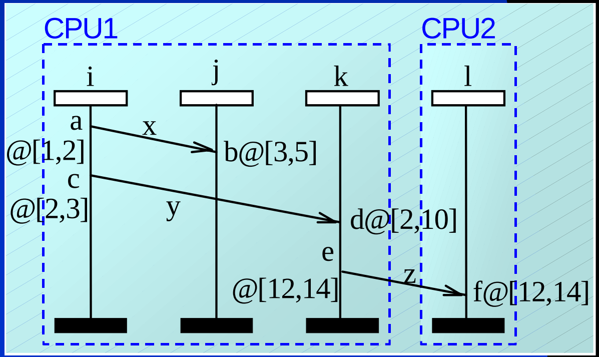
4. Scheduling Policy

- Ensure that processes distributed on a same CPU can follow a predefined scheduling policy and still meet functional requirements

A scheduling policy implies order on events…

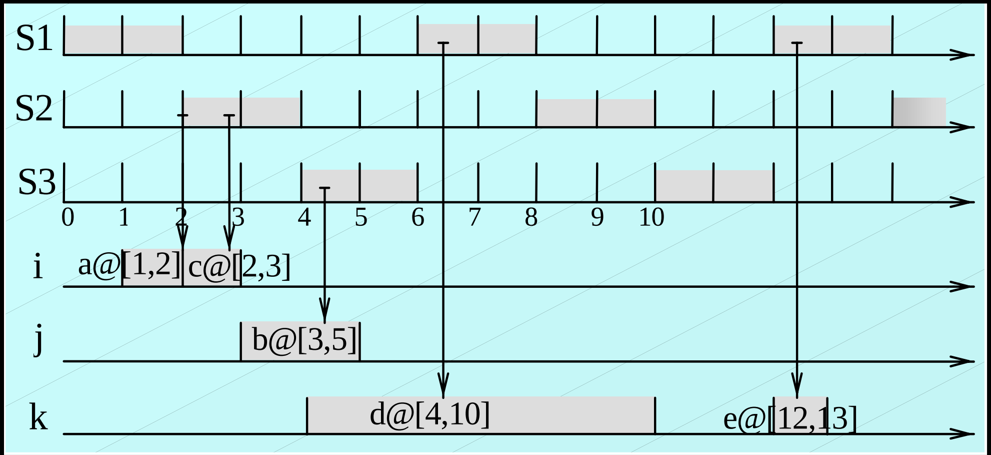
- Check if MSC is compatible with it





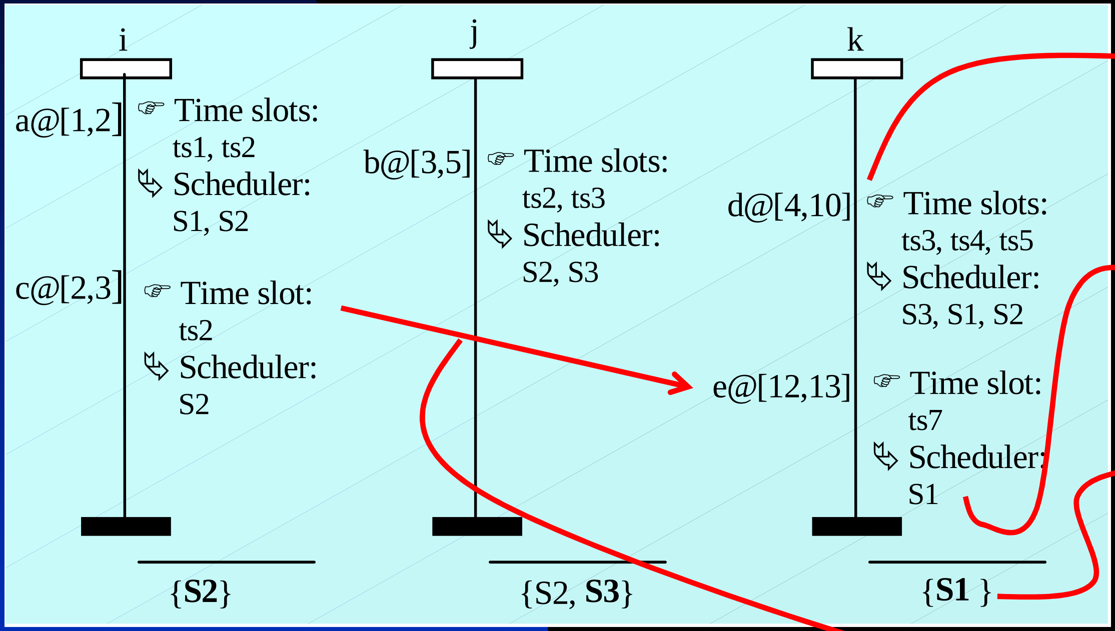
Main issue: Mapping

scheduler states ⇔ MSC instances



4. Scheduling Policy

Mapping Algorithm



Algorithm:

1. Lists time slots & scheduler states available for each event (compares date)
2. Intersects lists along each instance (it gives possible mappings for the instance)
3. Computes possible mappings for the MSC
4. Check precedence order (compare time slots & dates)

Output: list of mappings

Example: { (i,S2), (j,S3), (k,S1) }

Conclusion

- Handle certain deployment constraints at the specification stage
  - ➔ Are functional requirements still met and valid when deployment constraints are taken into account ? (channel delay, process distribution, scheduling policy)
  - ➔ Avoid backtracking from late stages of implementation and test

- Future works:
  - ◆ Consider further constraints and resources
  - ◆ Extend validation issues of process distribution and scheduling policy to HMSCs