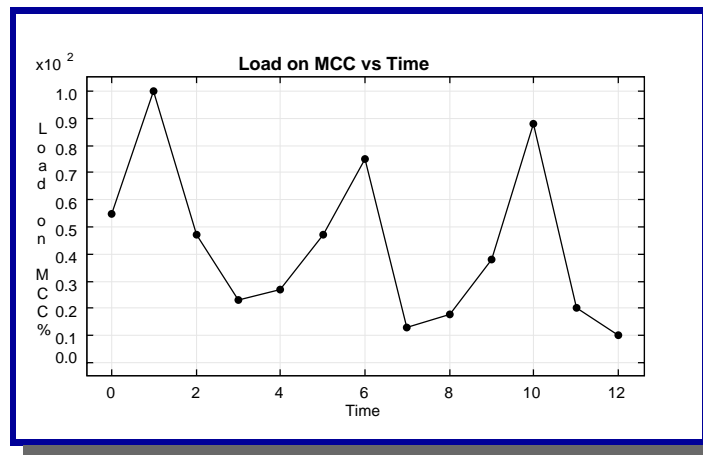


# Using Design/CPN to Design a Visualisation Extension for Design/CPN

CPN'2002



Mathew Elliot, Jonathan Billington, and Lars M. Kristensen  
Computer Systems Engineering Centre  
School of Electrical and Information Engineering  
University of South Australia  
AUSTRALIA

## Background

---

- ❑ Research project on CPN modelling and analysis of [avionics mission systems](#) with the Australian Defence Science and Technology Organisation (DSTO).
- ❑ A timed CPN model of a representative avionics mission system was constructed (18 pages, 22 transitions, 87 places, 28 colour sets).
- ❑ DSTO was interested in investigating the possibility for providing high-level and domain specific visualisations of the constructed CPN model:
  - Useful for presenting results of investigations for senior management and the Australian Defence Force (ADF).
  - Support analysis of avionics mission system based on CPN models by non CPN and Design/CPN experts.
- ❑ Design/CPN already provides support for visualisation of the execution of CPN models:
  - Token game of the Design/CPN simulator.
  - Mimic/CPN and Message Sequence Chart (MSC) Libraries.
- ☞ It was decided to develop an [external visualisation package](#) to meet DSTO requirements.

## Background

---

- ❑ The visualisation package was developed by a group of two students for their final year honours project in computer systems engineering.
- ❑ Approximately 5 person months were used on the honours project.
- ❑ The students followed a full system engineering approach:
  - Fortnightly meetings with supervisors chaired by the students.
  - Documentation: project plan, requirements specification, design description, test plan, test specification, test report, user and system manual.
  - Research proposal and paper to fulfill research requirements for honours students.
- ❑ The students followed a rapid prototyping paradigm to obtain feedback from DSTO at the earliest opportunity.
- ❑ One research paper considered the CPN modelling and functional analysis of the [visualisation protocol](#).
- ❑ The visualisation protocol facilitates the co-ordination between the Design/CPN simulator and the visualisation package.

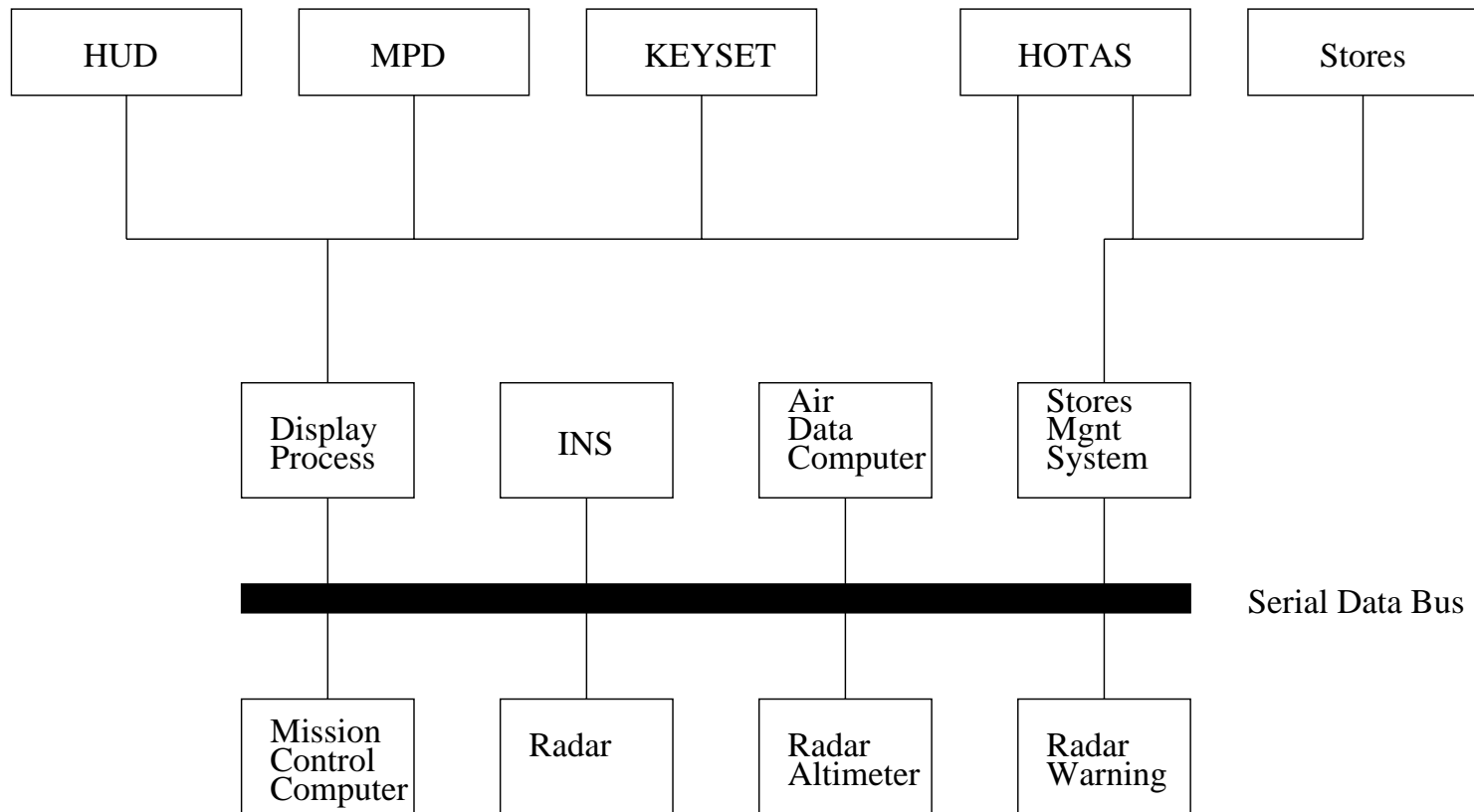
# Outline

---

- ❑ Avionics mission systems.
- ❑ Architecture of the visualisation package.
- ❑ Coloured Petri Net modelling of the visualisation protocol.
- ❑ State space analysis of the visualisation protocol.
- ❑ Conclusions.

# Avionics Mission Systems

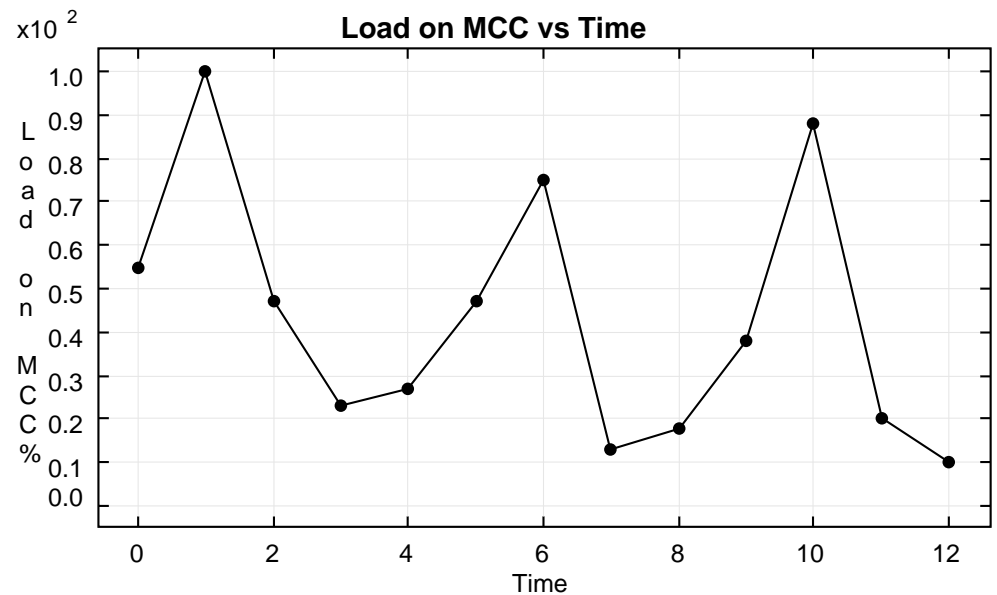
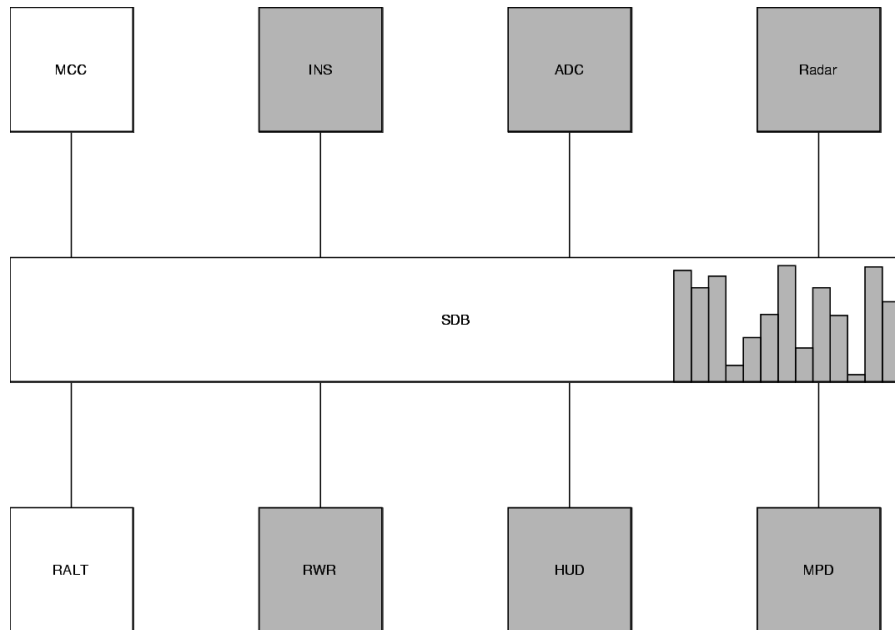
- An Avionics Mission System (AMS) can be viewed as complex real-time system consisting of interactive components and subsystems.



- The AMS is responsible for the communication between subsystems in the course of the mission (e.g., sea search and rescue mission).

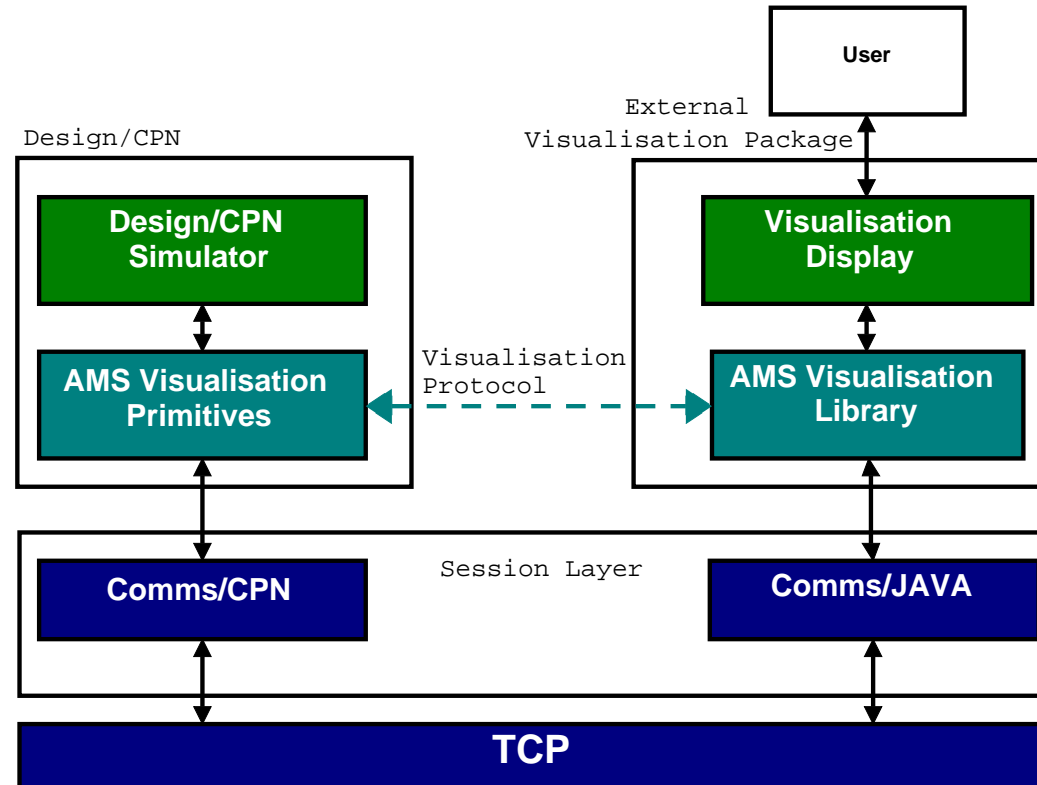
# Visualisation of AMS

- The visualisation package was required to support several visualisations:
  - Block diagrams of the AMS and its components.
  - MSCs showing detail of communication between subcomponents.
  - A chart showing the progress of the mission.
  - Utilisation on the Mission Control Computer and the Serial Data Bus.



# Architecture of Visualisation Extension

- Client/Server architecture with the Design/CPN simulator acting as client and the external visualisation package acting as server:



- The visualisation package implemented in JAVA to exploit different visualisation libraries available.

# Visualisation Protocol

- **Initialisation instructions** are used to initialise the visualisation package:

Instruction	Purpose
<i>useLibrary</i>	Specifies the visualisation library to be used.
<i>createComponent</i>	Creates an instance of a component to be used.
<i>initComplete</i>	Specifies that the initialisation is complete.

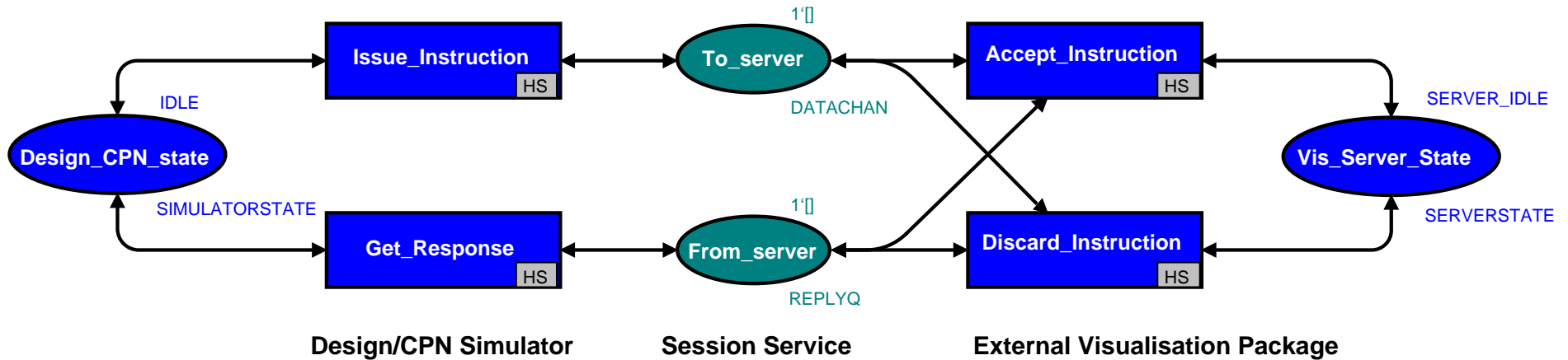
- **Visualisation instructions** are used to update the diagrams during simulation:

Instruction	Purpose
<i>isCommunicating</i>	displays that a specified component is communicating.
<i>stoppedCommunicating</i>	displays that a specified component is not communicating.
<i>createLink</i>	creates a visible link between two components used within a visualisation.
<i>MCCLoadIndication</i>	displays the load on the MCC component.
<i>SDBLoadIndication</i>	displays the load on an SDB component.
<i>getInstruction</i>	used to continue (CONT) or terminate (STOP) visualisation.

- The basic visualisation protocol used in the implemented prototype was modelled and analysed using Coloured Petri Nets.
- An improved version of the visualisation protocol was then developed, specified, and analysed using Coloured Petri Nets.

# CPN Modelling of the Visualisation Protocol

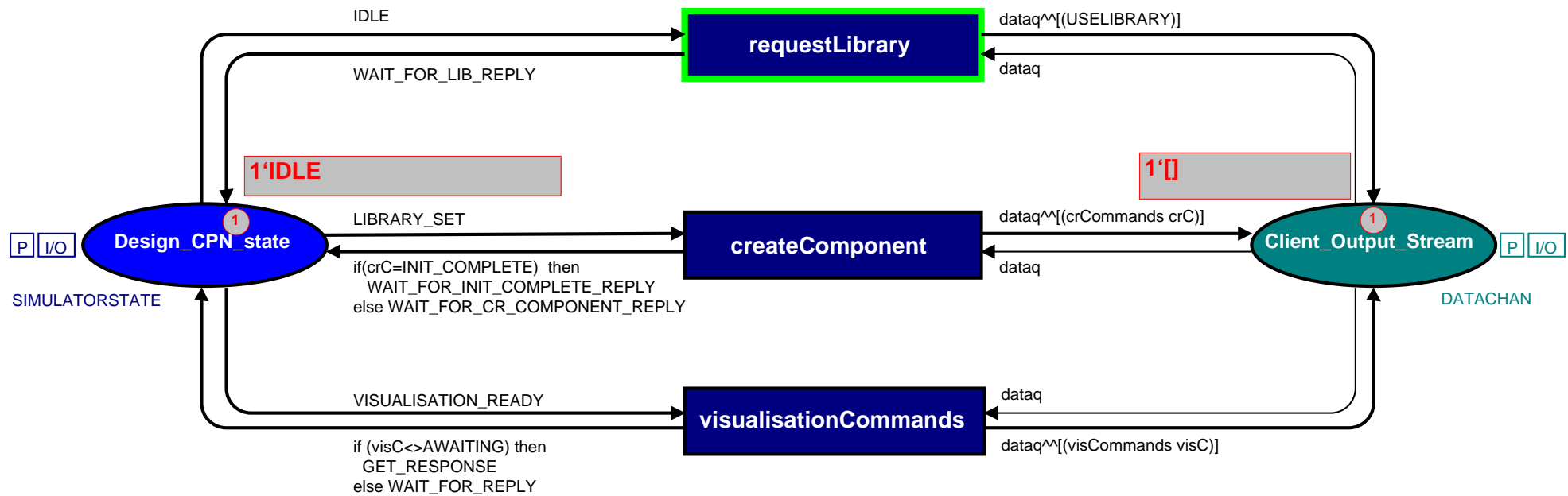
□ Prime (top-level) page of the CPN model:



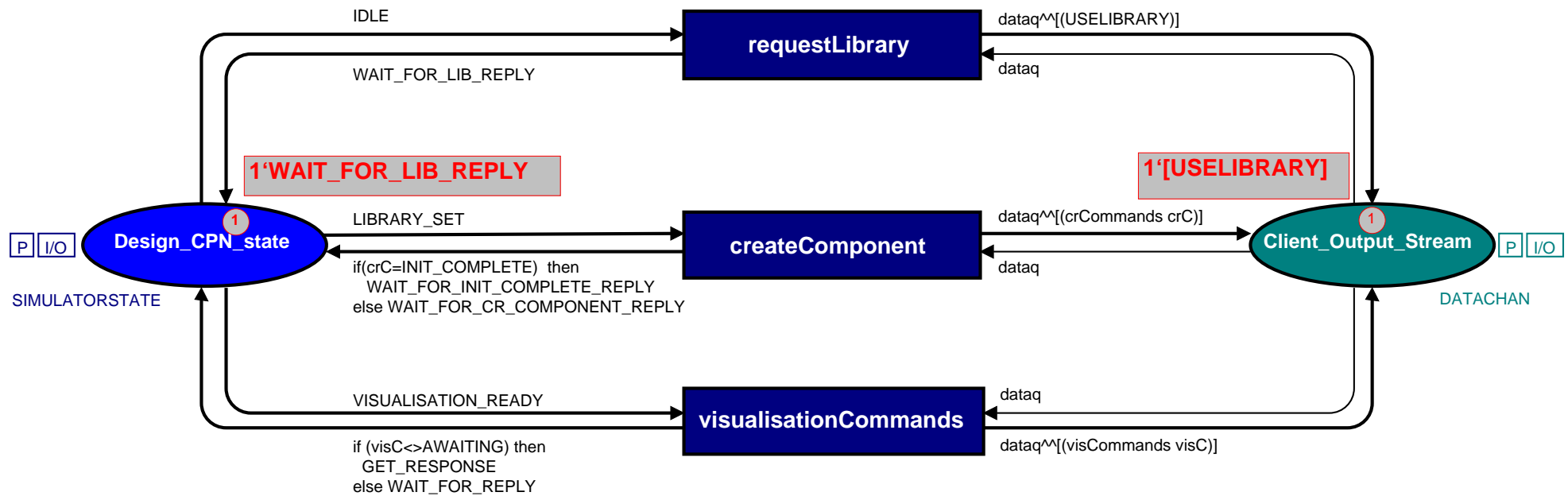
□ Assumptions and abstractions:

- The parameters (e.g., coordinates of objects) of the visualisation instructions are not important for the operation of the protocol.
- ☞ All visualisation instructions (except *getInstruction*) are abstracted into one generic message: *UPDATEGUI*.
- Only one instruction is processed at a time at the server.
- A session has already been established.
- The session service is free of errors and preserves order of messages.

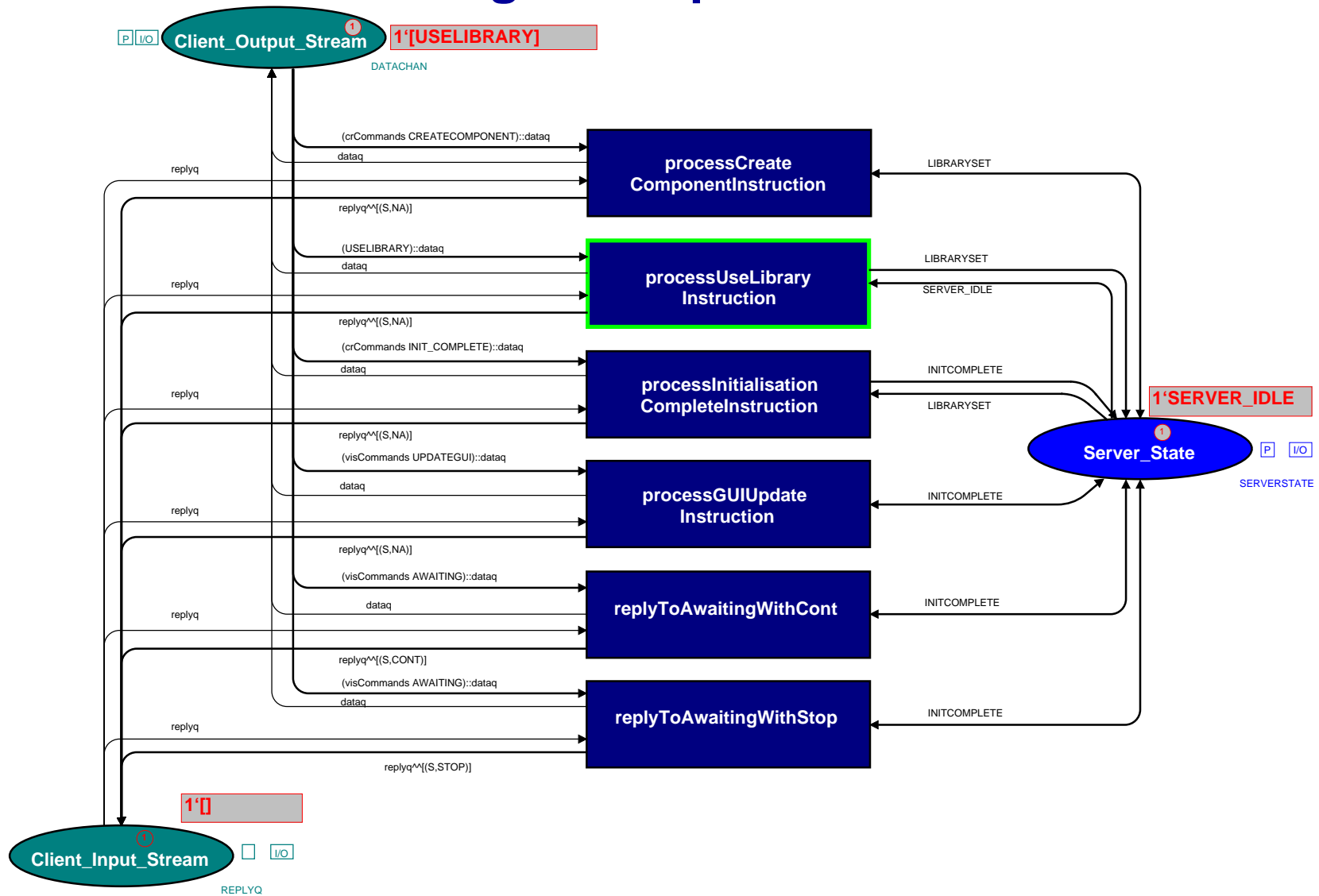
# Client Side Modelling - Issue Instruction



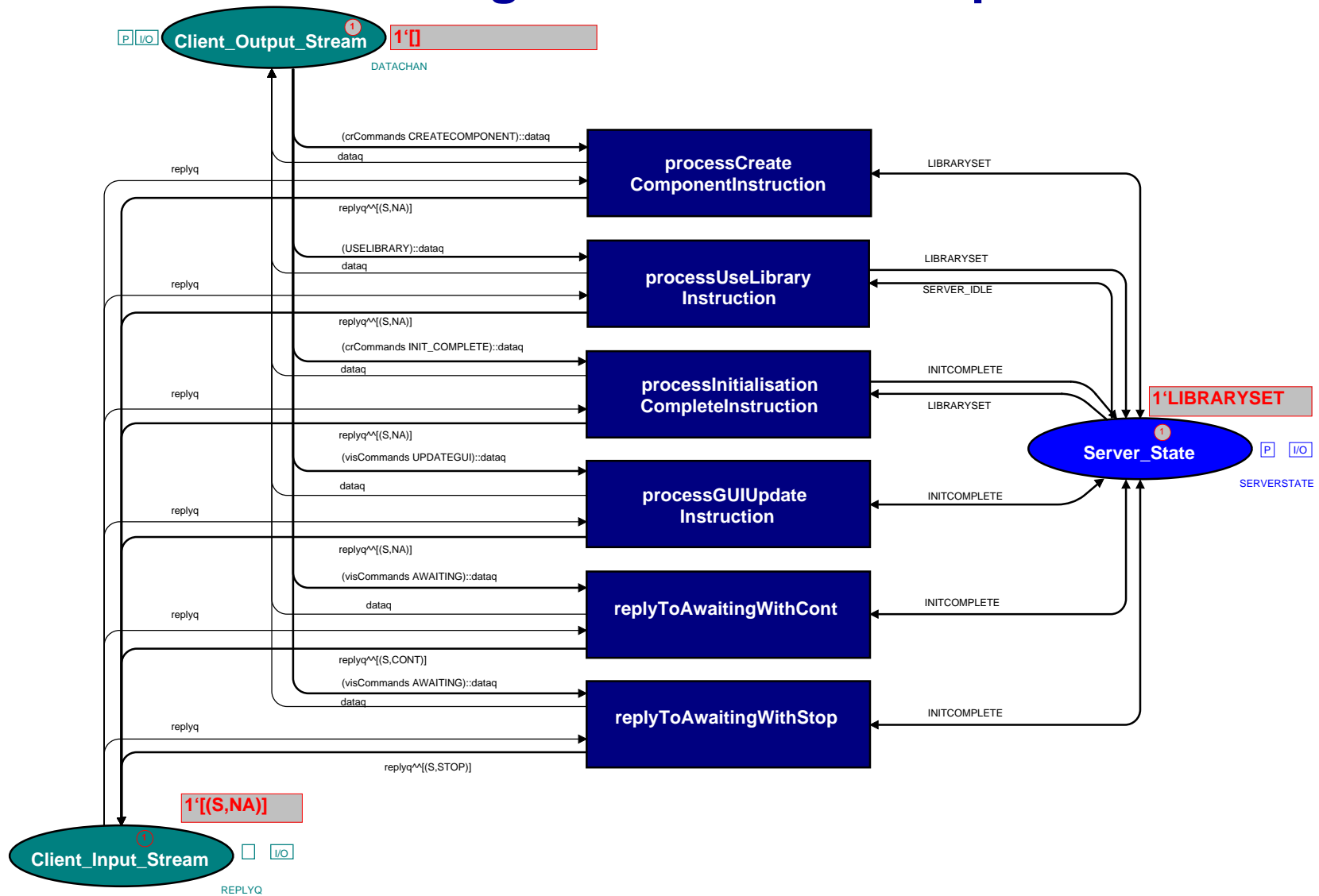
# Client Side Modelling - Instruction Issued



# Server Side Modelling - Accept Instruction



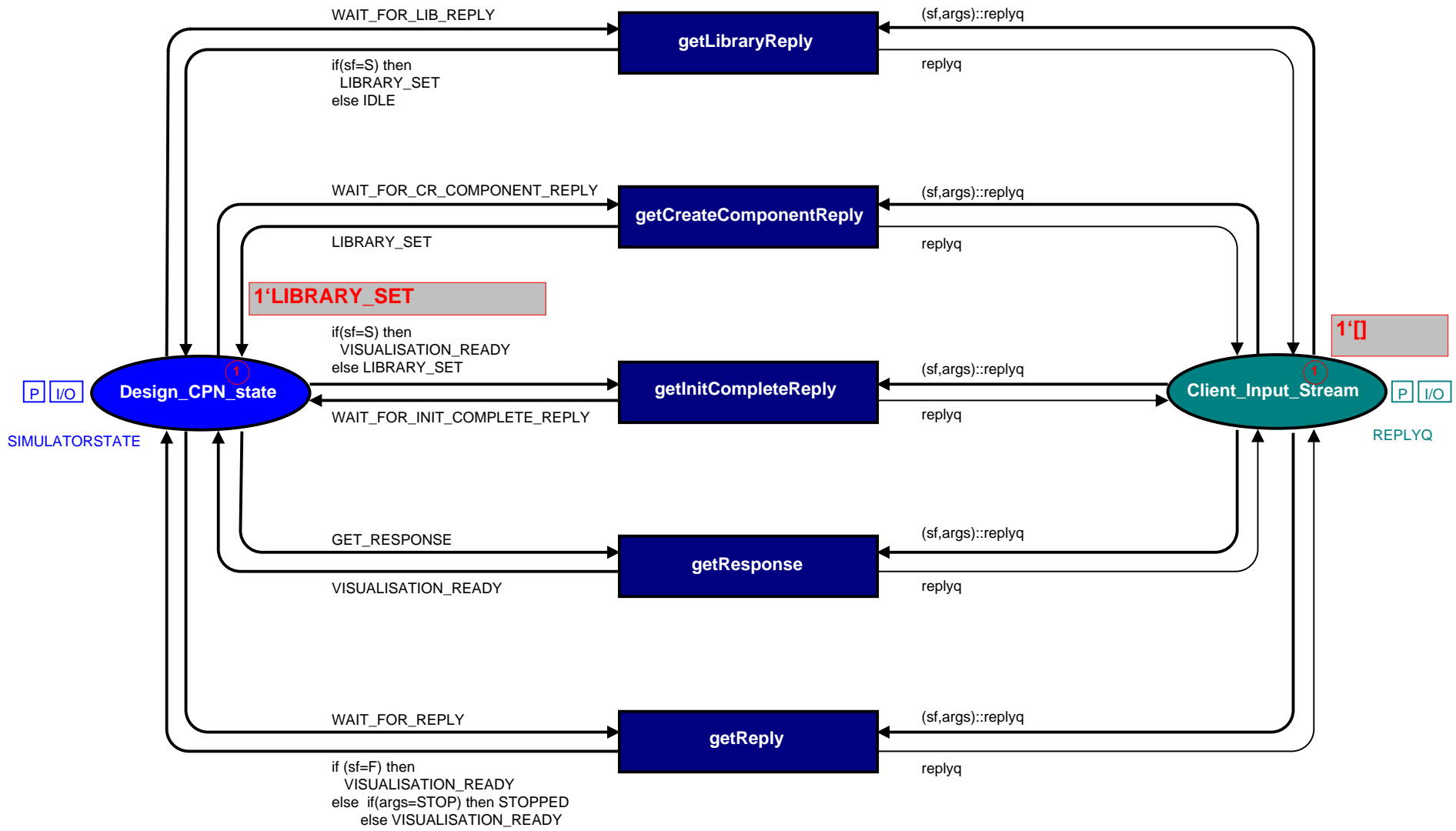
# Server Side Modelling - Instruction Accepted



# Client Side Modelling - Receive Reply



# Client Side Modelling - Reply Received



## State Space Analysis

---

- The CPN specification of the visualisation protocol was verified using state space/reachability analysis.
- Properties investigated:
  - Absence of **deadlocks** (no undesired terminal states).
  - Absence of **livelocks** (a desired terminal state can always be reached).
  - The sequences of instructions must conform to the formal language determined by the following regular expression:

InitPrimitives VisPrimitives

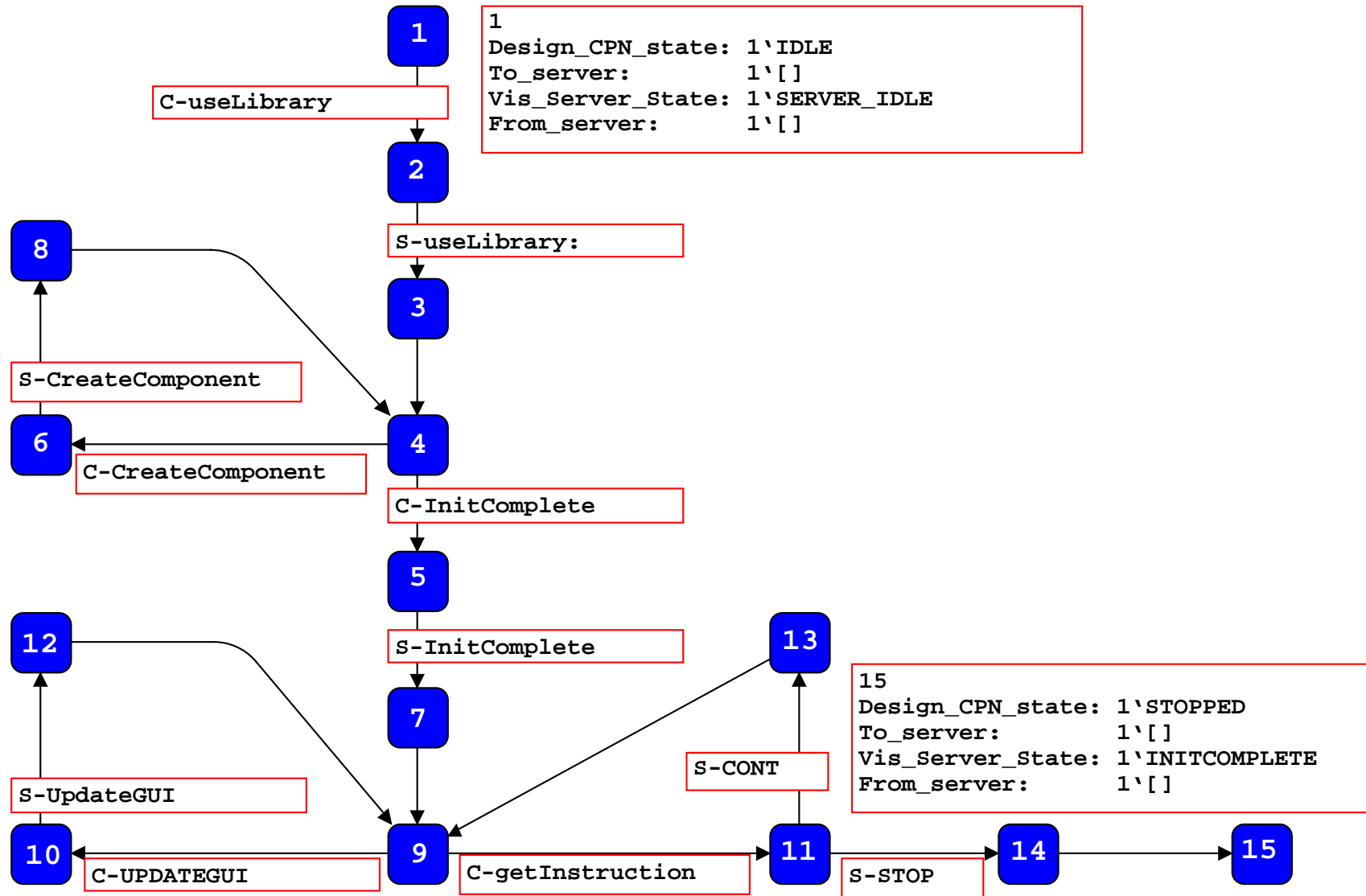
where:

**InitPrimitives**  $\equiv$  useLibrary createComponent\* initComplete

**VisPrimitives**  $\equiv$  [(getInstruction CONT)\* UPDATEGUI\*]\*getInstruction STOP

- The state space has 15 nodes and 17 arcs.
- ☞ The state space was small enough to check the properties by visual inspection of the state space.

# State Space for the Visualisation Protocol



## Conclusions

---

- ❑ A CPN specification of the visualisation protocol used in an external visualisation package for Design/CPN was constructed.
- ❑ The visualisation protocol was improved in the course of developing and analysing the CPN specification:
  - First-cut implementation: may deadlock and violate sequencing constraints.
  - Second version: no deadlocks, no livelocks, and satisfies sequencing constraint.
  - Third version (presented here): status reply from the server.
- ☞ The development of the CPN specification provided useful insight into the operation of the visualisation protocol.
- ☞ The development and analysis of the CPN specification led to a better and more robust visualisation protocol.
- ❑ The state space for this practical application of CP-nets was small enough to allow visual verification to be conducted.