BRICS

BEST PRACTICE IN ROBOTICS

Separation of Concerns in Component-based Robotics

Davide Brugali

Università degli Studi di Bergamo, Italy

CAR 2010, Douai, 19-05-2010

Robot Control Architectures

- Typical functions implemented in software
  - Acquiring and interpreting input from sensors
  - Controlling the motion of all moving parts
  - Representing the robot’s and environment’s state
  - Managing all kinds of resources (power, CPU, memory, devices)
  - Planning future activities
  - Reacting to unpredictable events

- Typical control paradigms
  - Sense → Plan → Act
  - Behavior-based control
  - Layered control

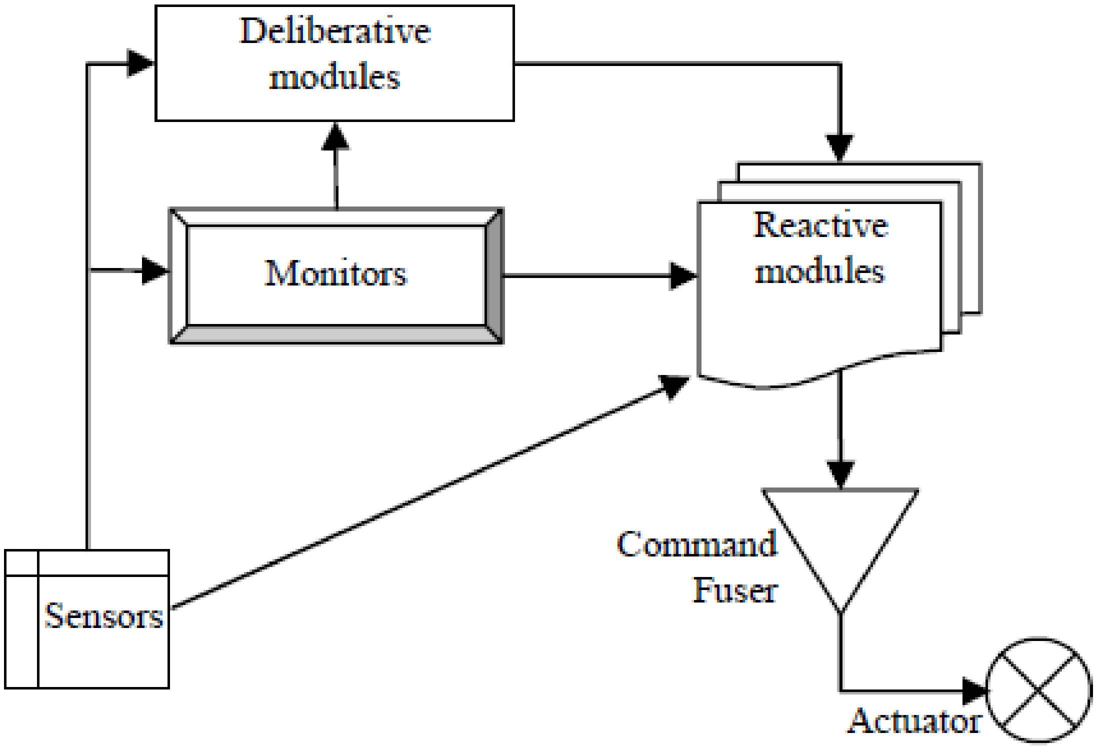
Robot Software Architectures

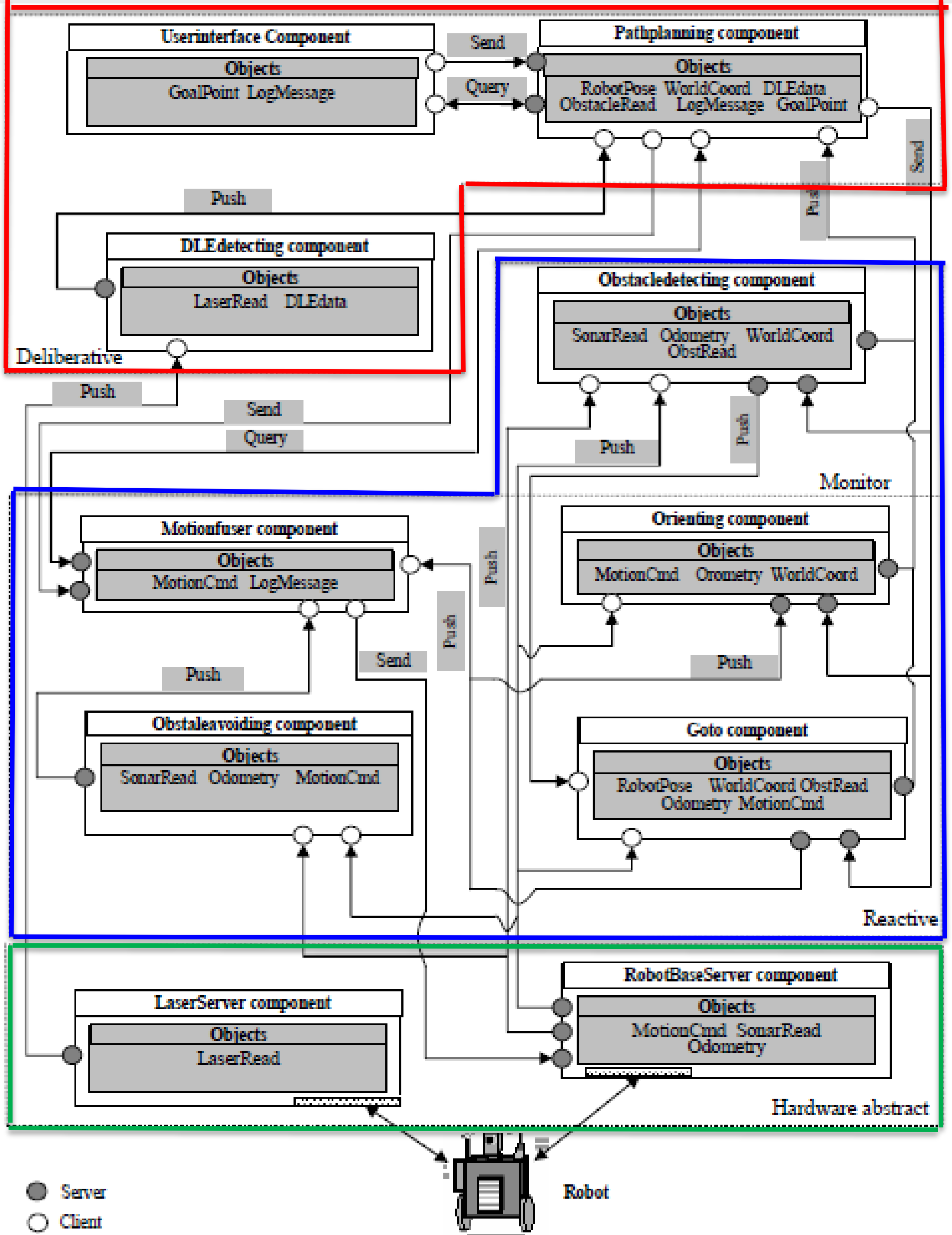
- Typical aspects of software architectures
  - decomposition of the robot control system into a collection of software components
  - encapsulation of functionality and control activities into components
  - definition of data flow and control flow among components

- Typical software quality factors
  - Reusability
  - Portability
  - Interoperability
  - Maintainability

Control vs Software Architecture

W. Li, H. I. Christensen, A. Orebäck, D. Chen,
"An Architecture for Indoor Navigation", ICRA 2004

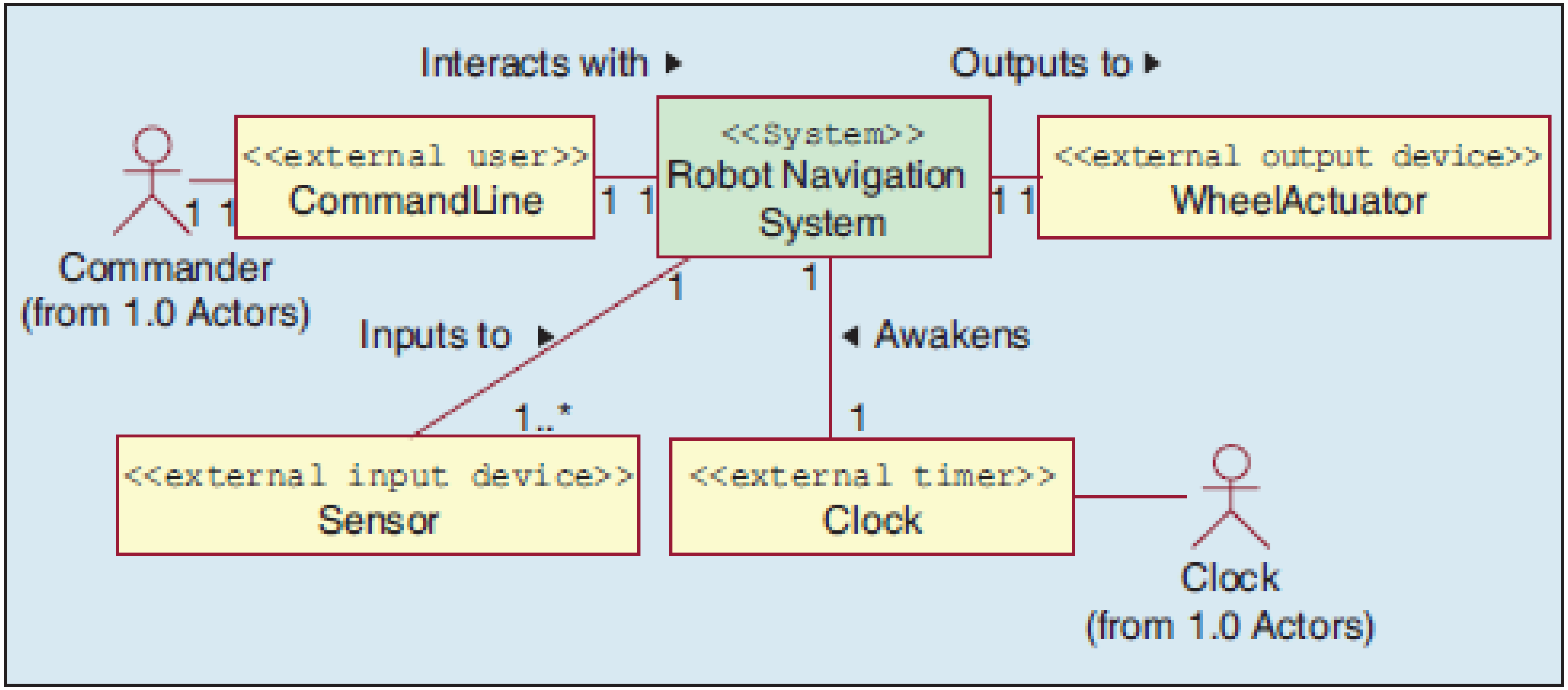




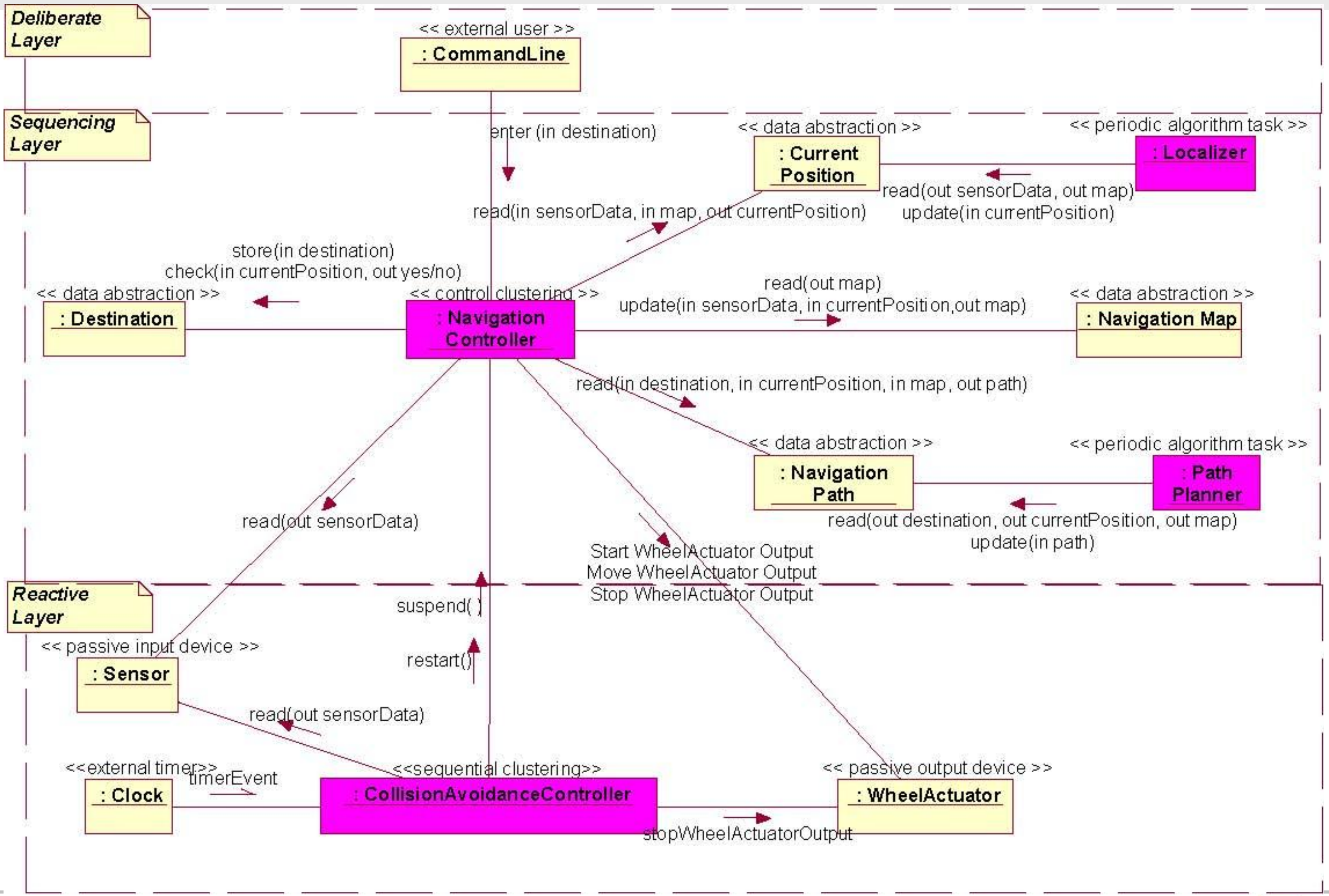
UNIVERSITÀ DEGL

Control vs Software Architecture

M. Kim, S. Kim, S. Park, M. Choi, M. Kim and H. Gomaa "Service Robot Software Development with the COMET/UML Method", IEEE Robotics and Automation Magazine, March 2009

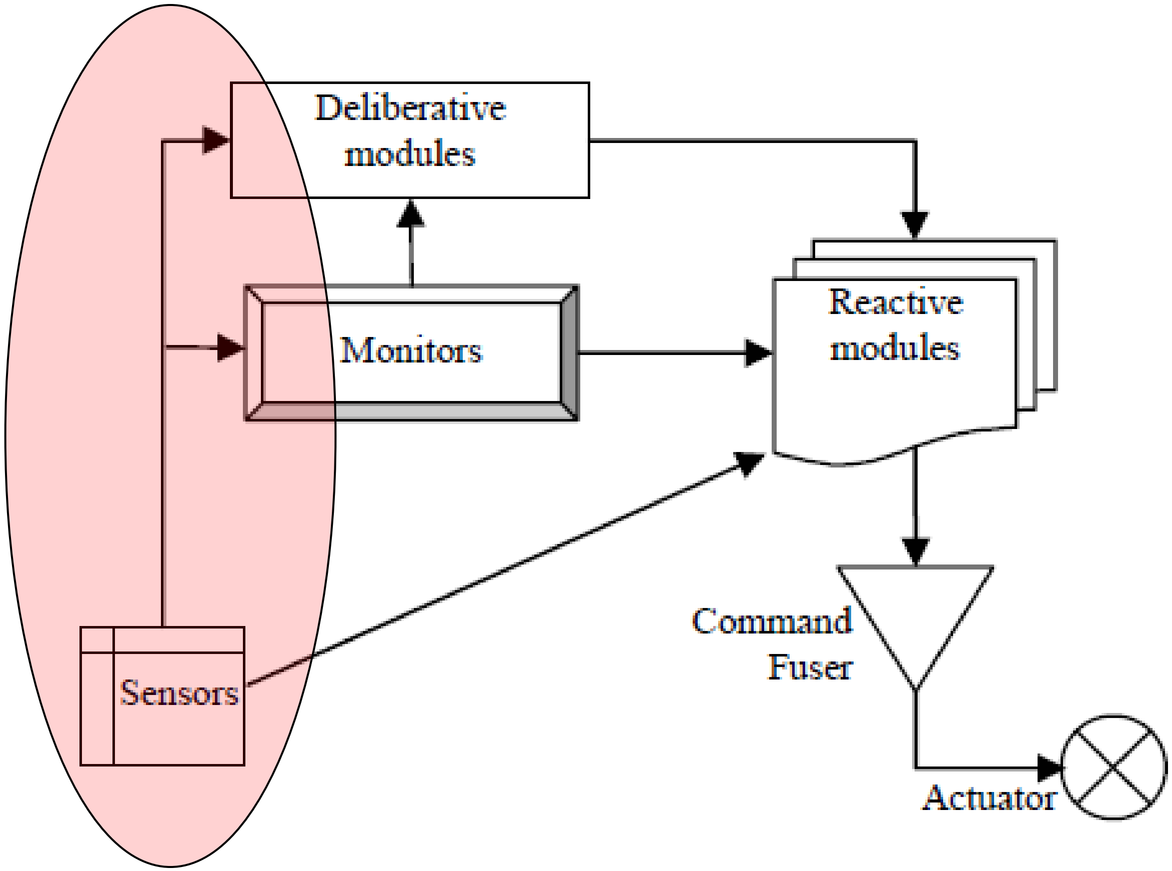


Control vs Software Architecture



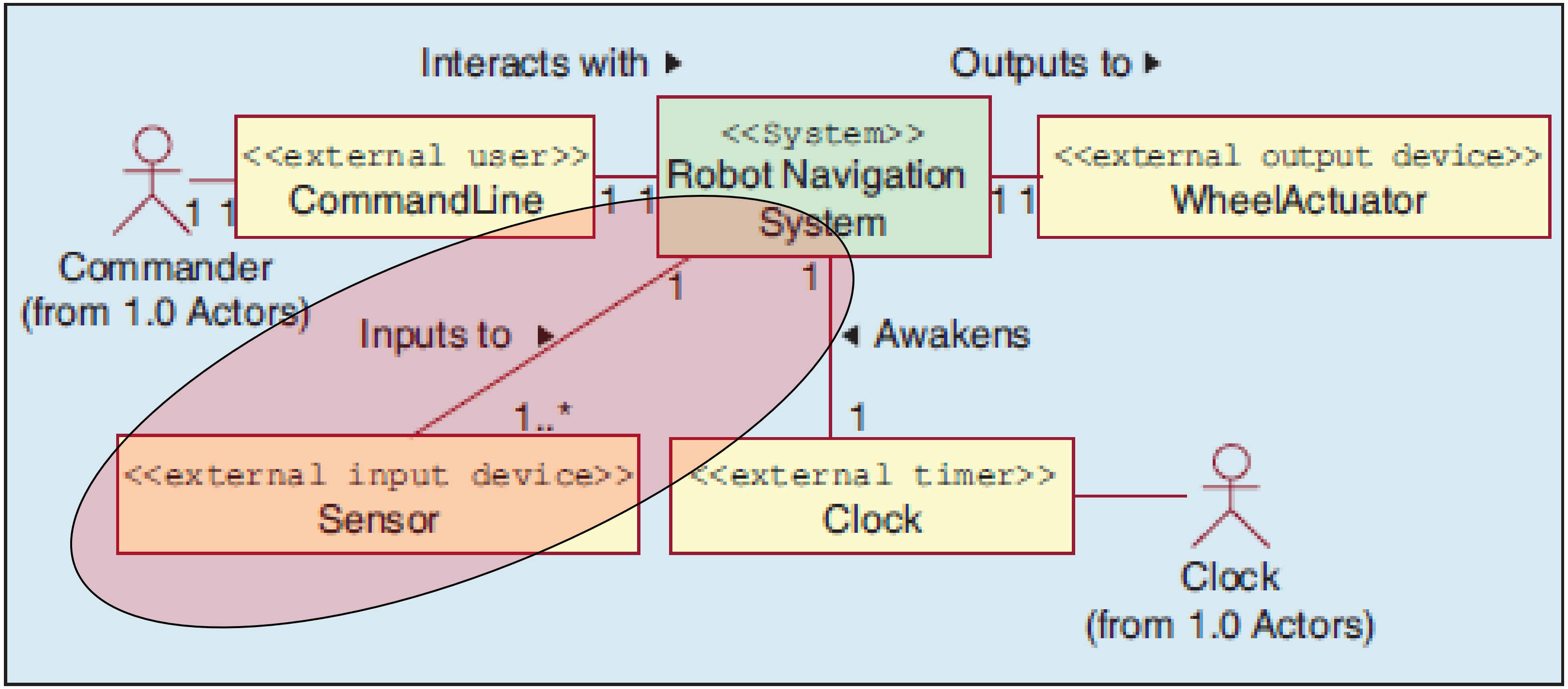
Control vs Software Architecture

W. Li, H. I. Christensen, A. Orebäck, D. Chen,
"An Architecture for Indoor Navigation", ICRA 2004

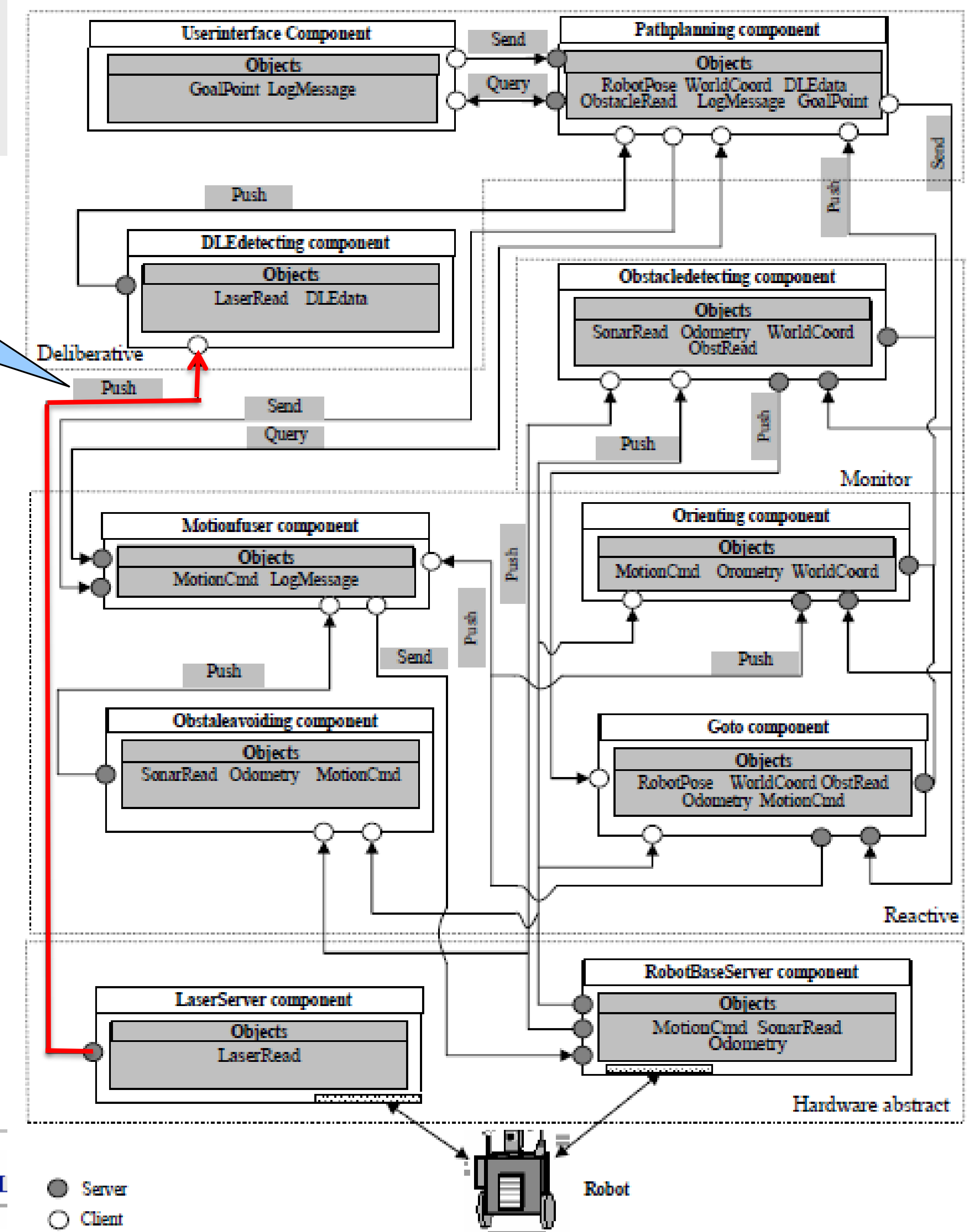


Control vs Software Architecture

M. Kim, S. Kim, S. Park, M. Choi, M. Kim and H. Gomaa "Service Robot Software Development with the COMET/UML Method", IEEE Robotics and Automation Magazine, March 2009

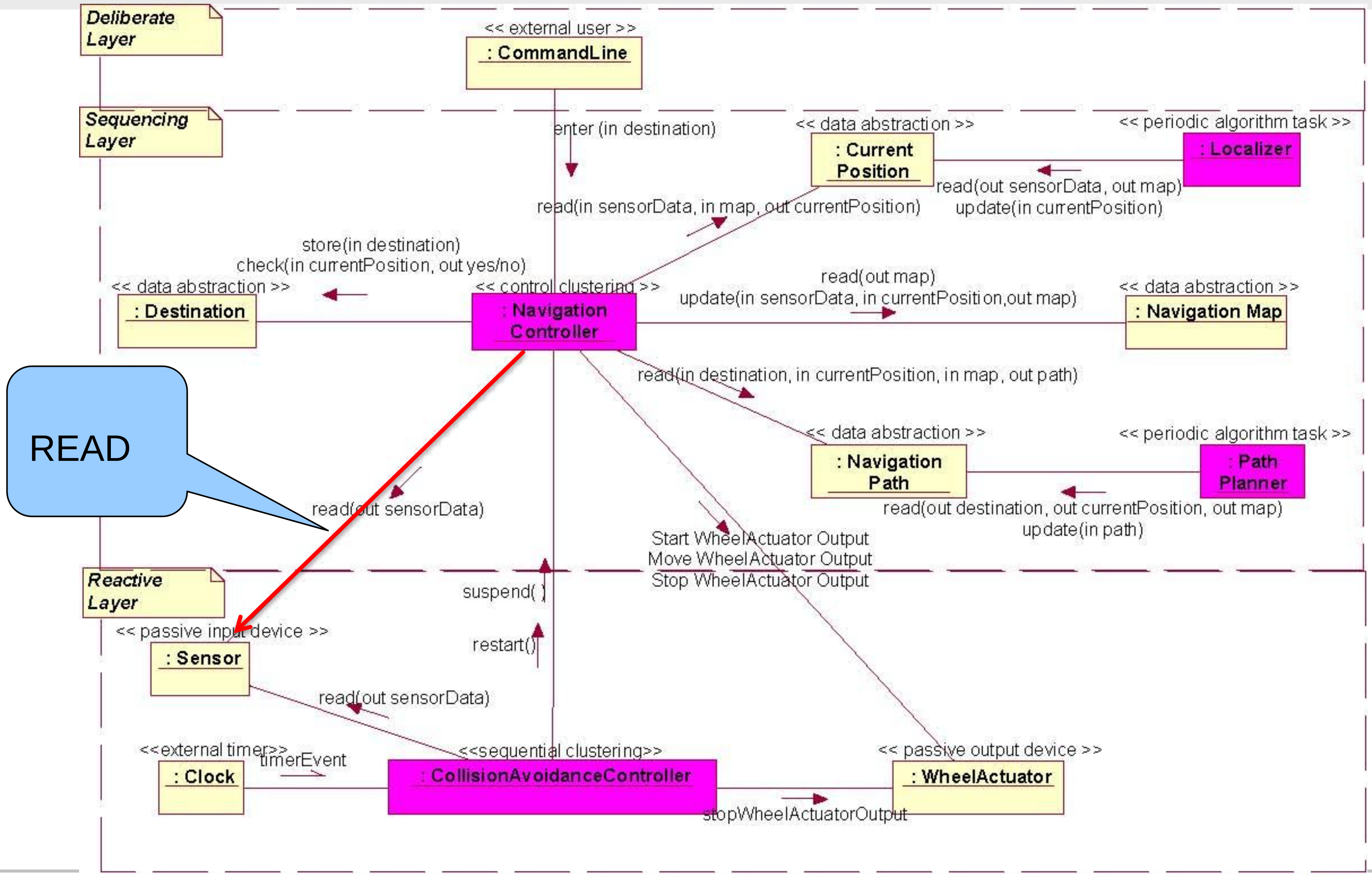


PUSH



UNIVERSITÀ DEGI

Control vs Software Architecture

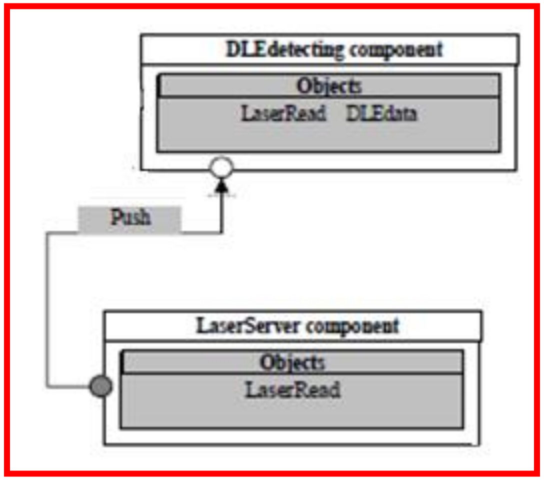


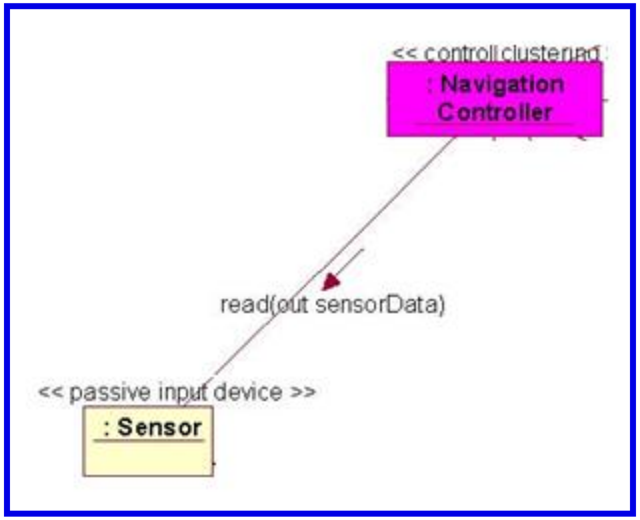
Control vs Software Architecture

- The same sensor component is not reusable in two different robotic systems
- Where is the problem?
- Not in the control architecture
  - Both architectures indicate that the sensors supply data to the control modules
- Not in the software architecture
  - The interaction pattern (push / pull) depends on specific requirements of each application

Separation of concerns

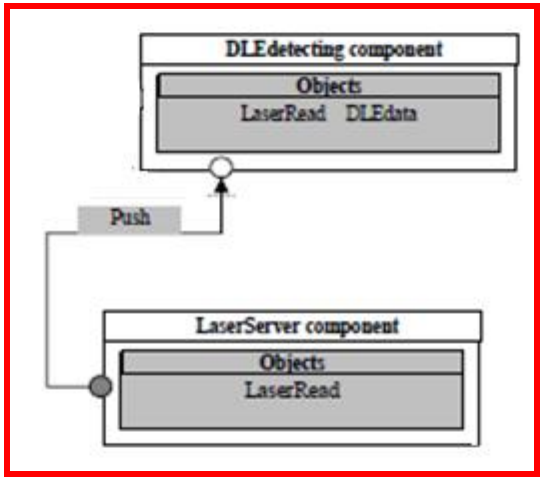
- The implementation of the sensor components mixes two different aspects:
  - The component functionality
  - The interaction mechanism

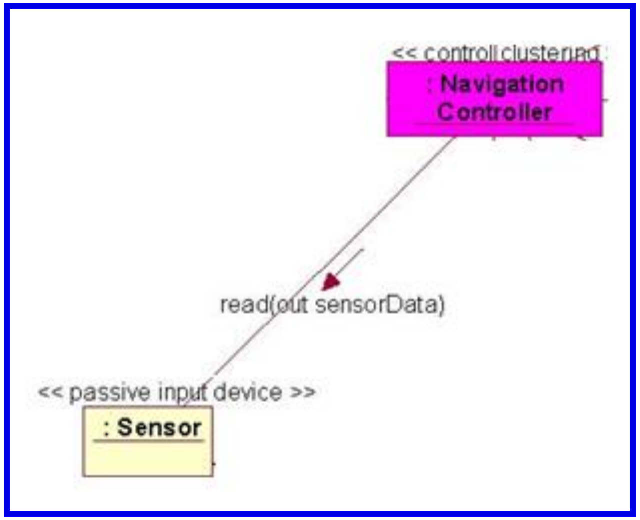




Separation of concerns

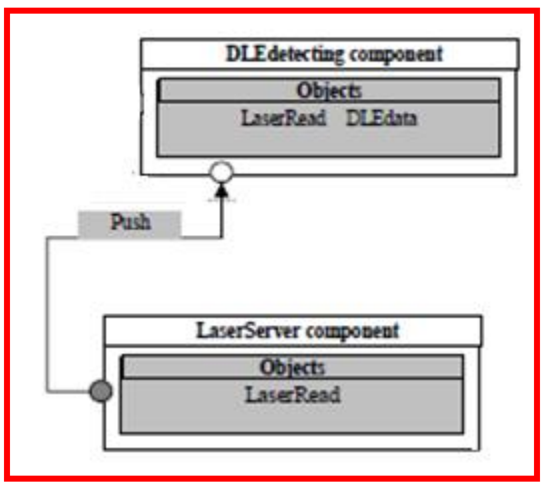
- The implementation of the sensor components mixes two different aspects:
  - The component functionality **is stable → should be reusable**
  - The interaction mechanism

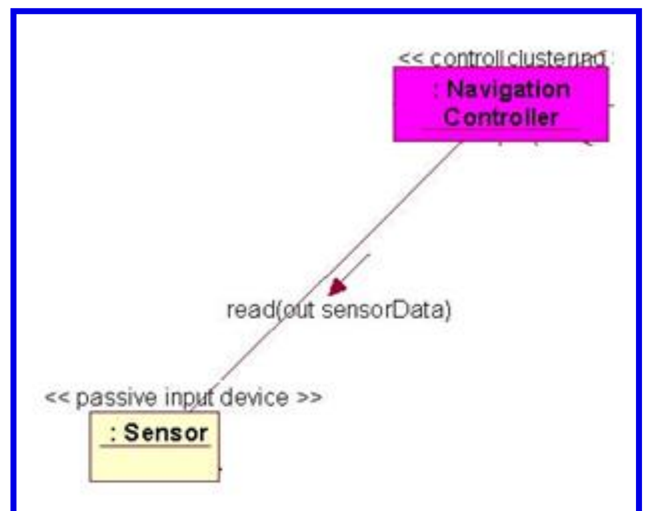




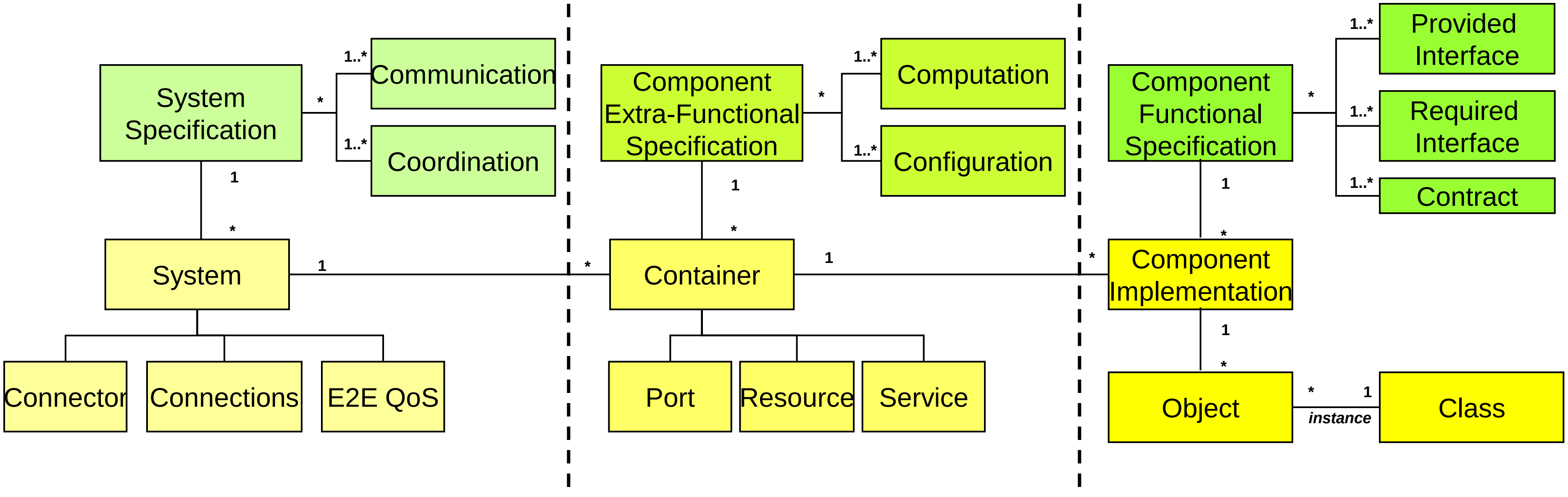
Separation of concerns

- The implementation of the sensor components mixes two different aspects:
  - The component functionality **is stable → should be reusable**
  - The interaction mechanism **is variable → should be flexible**



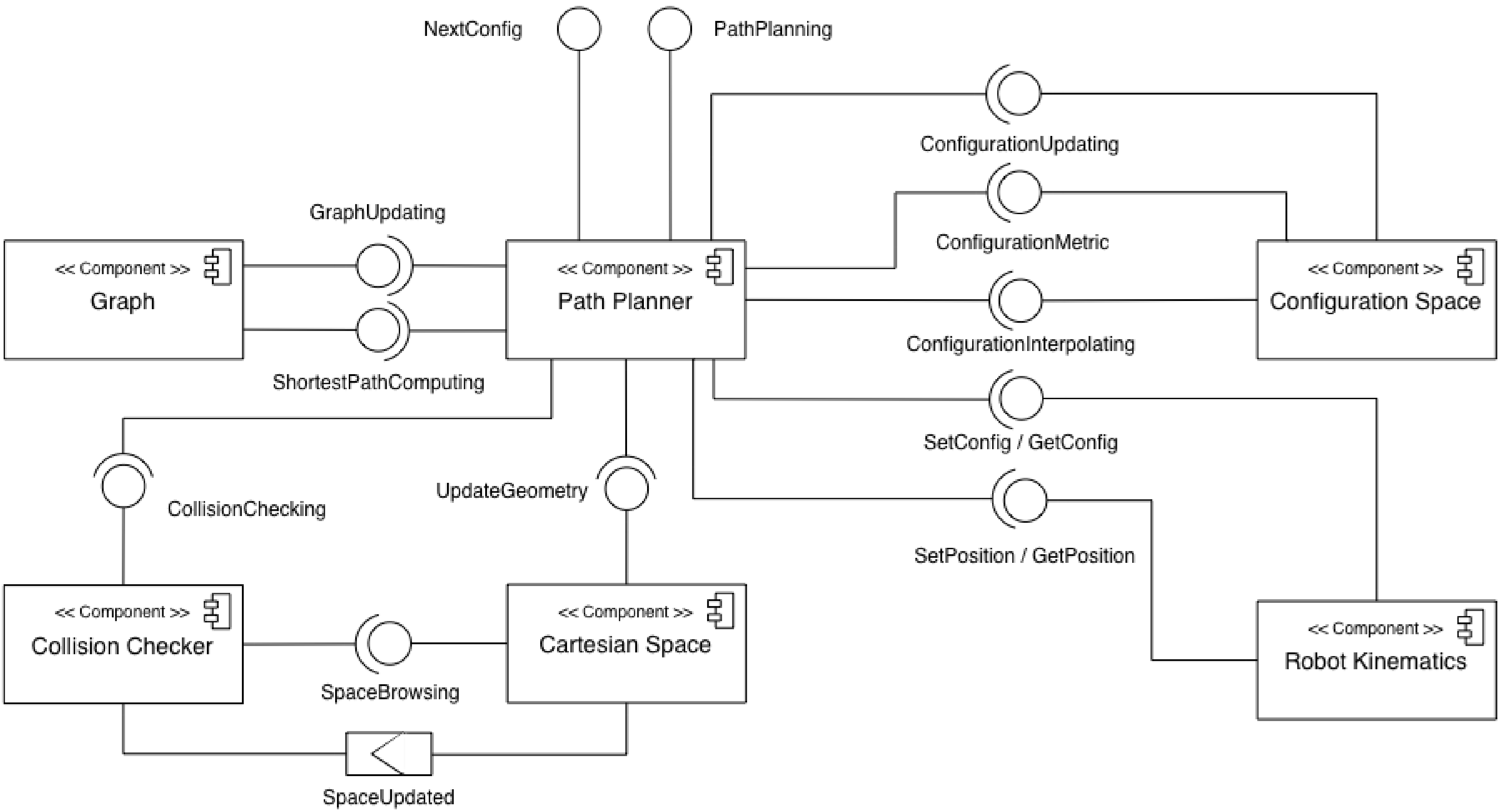


Separation of Concerns

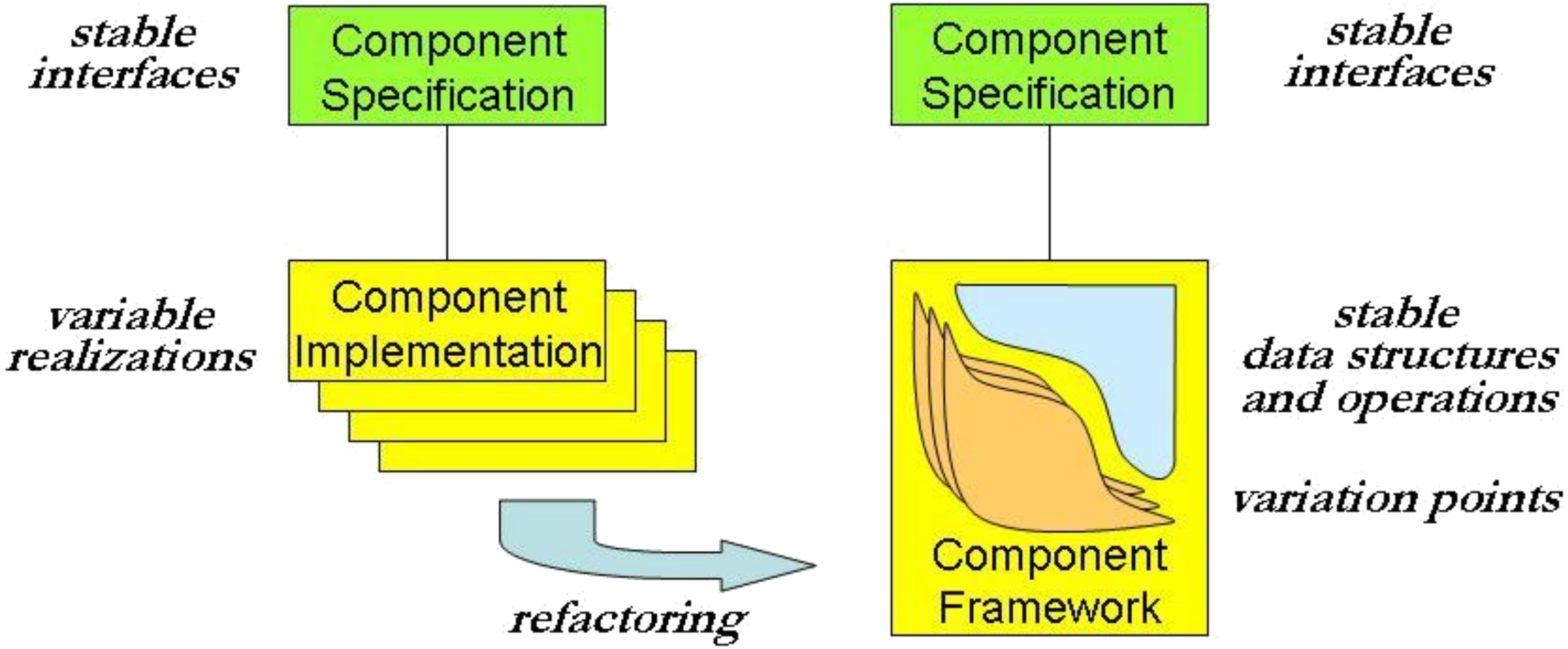


- IEEE RAM Tutorial on Component Based Robotic Engineering
  - Part I : December 2009
  - Part II : March 2010

Component Interface & Implementation



Component Interface & Implementation



- Refactoring aims at restructuring a set of existing software libraries without affecting their external behavior in order to harmonize their architecture, data structures, and APIs.

Refactoring Mobile Manipulation libraries

CONFIGURATION

MSL	MSLVector: double array, includes size
MPK_{Kernel}	Configuration: (self-defined) vector of doubles, includes call to OpenGL
CoPP	Config: typedef for vector<double>
MPK_{Kit}	mpkConfig inherits from vector<double>, includes various functions
OpenRAVE	TPOINT includes vector<dReal>, additionally velocities and time
OOPSMP	State_t: typedef for double* (needs external storage of size)
OMLP	State: double array (needs external storage of size) and flags

C-SPACE

MSL	Model (and Problem) have upper/lower limits, and various more things covering control inputs and system simulation
MPK_{Kernel}	upper/lower limits are derived from joints
CoPP	upper/lower limits stored explicitly where needed
MPK_{Kit}	limits implicitly in planner
OpenRave	ConfigurationState includes limits and number of DoF
OOPSMP	StateSpace includes bounding box and various other functions. Many concrete implementations
OMLP	SpaceInformation includes: start and goal config, dimension, StateDistanceEvaluator, StateValidityChecker

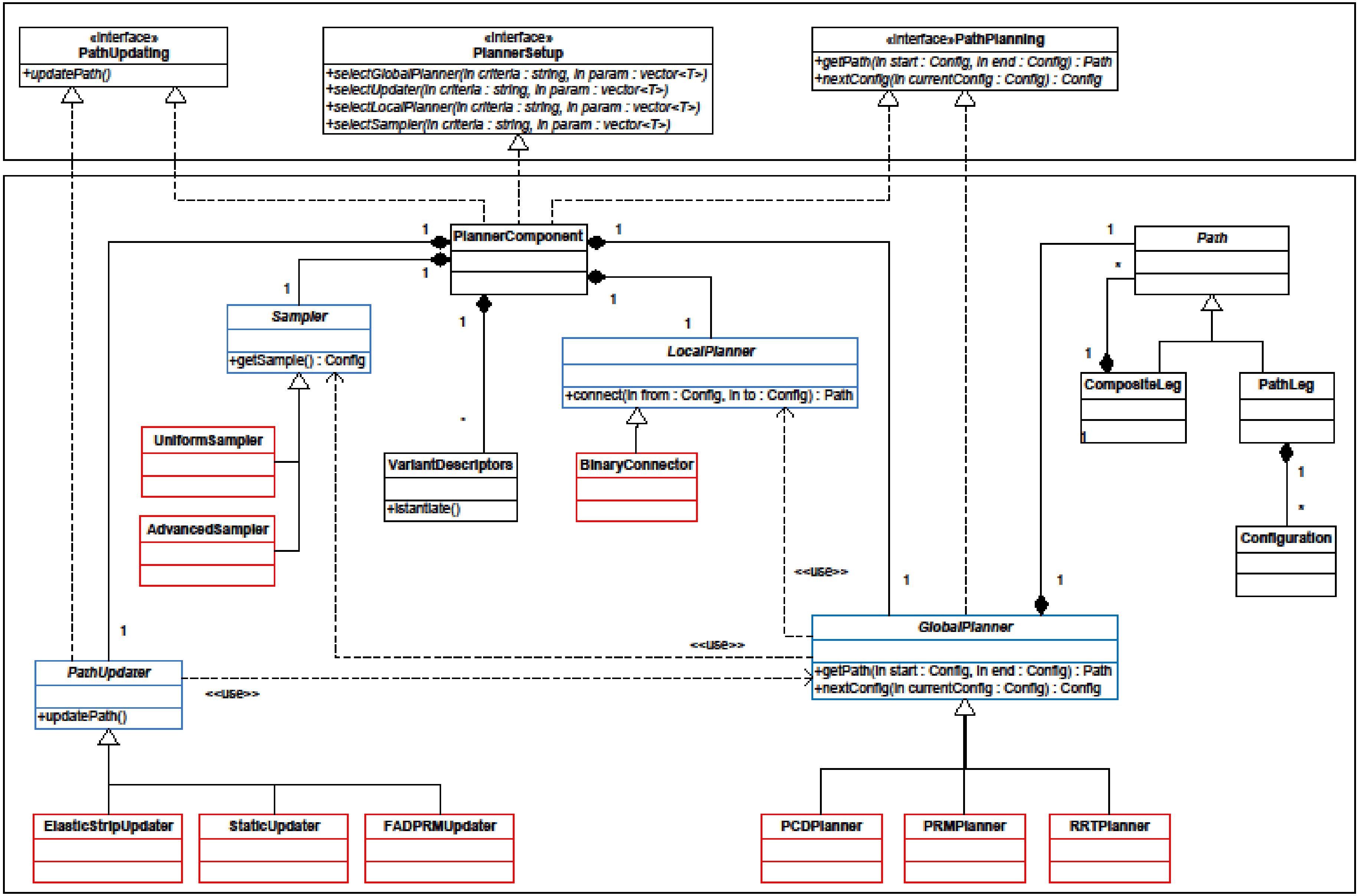
INTERFACES FOR COLLISION DETECTION

MSL	Geom with derived class for PQP
MPK_{Kernel}	CollisionDetectorBase. Universe has an array of Mesh which can model various objects.
CoPP	ObjectSet. Base class Geom stores a position, with inherited classes for triangles and convex objects. Geometric objects are attached to kinematic structures by pointing to their transformation.
MPK_{Kit}	mpkCollDistAlgo uses PQP or own collision detector
OpenRAVE	CollisionCheckerBase. KinBody includes TRIMESH and GEOMPROPERTIES for modelling triangle meshes
OOPSMP	CollisionDetector. Workspace holds list of Part, support of polygons
OMLP	Based on ROS with interfaces of CollisionSpace and various geometry messages

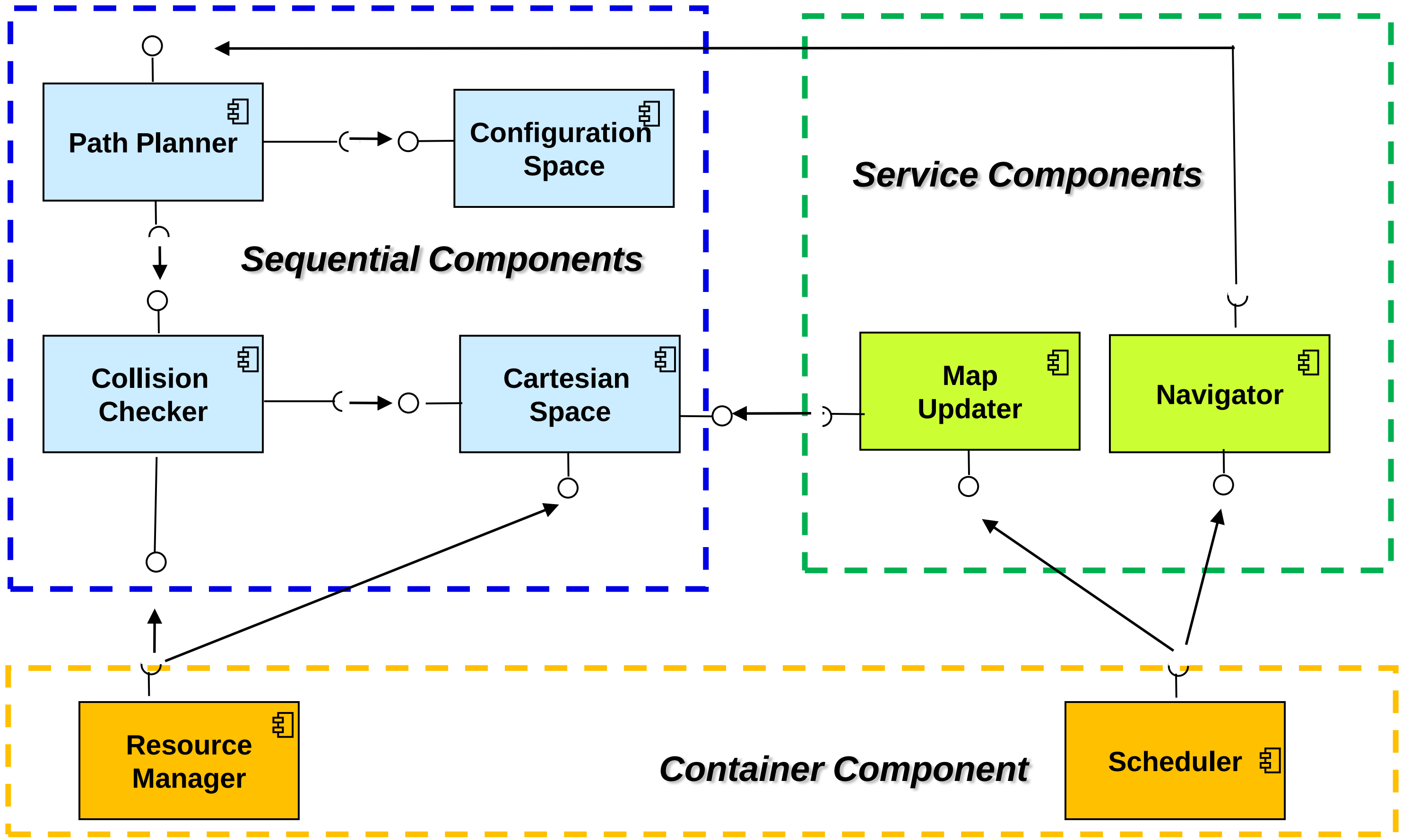
PATH

MSL	list<MSLVector>: list of vectors
MPK_{Kernel}	PathBase: abstract base class, and PA_Points with vector<Configuration>
CoPP	Path: own data type encapsulates list of configs and time
MPK_{Kit}	vector<mpkConfig>: vector (or list) of configurations
OpenRAVE	Trajectory includes vector of points and segments, in addition elements for dynamic motion control
OOPSMP	Path includes interfaces for times, splitting and more. Base class with various implementations
OMLP	Path: points to a SpaceInformation, derived classes include array of State

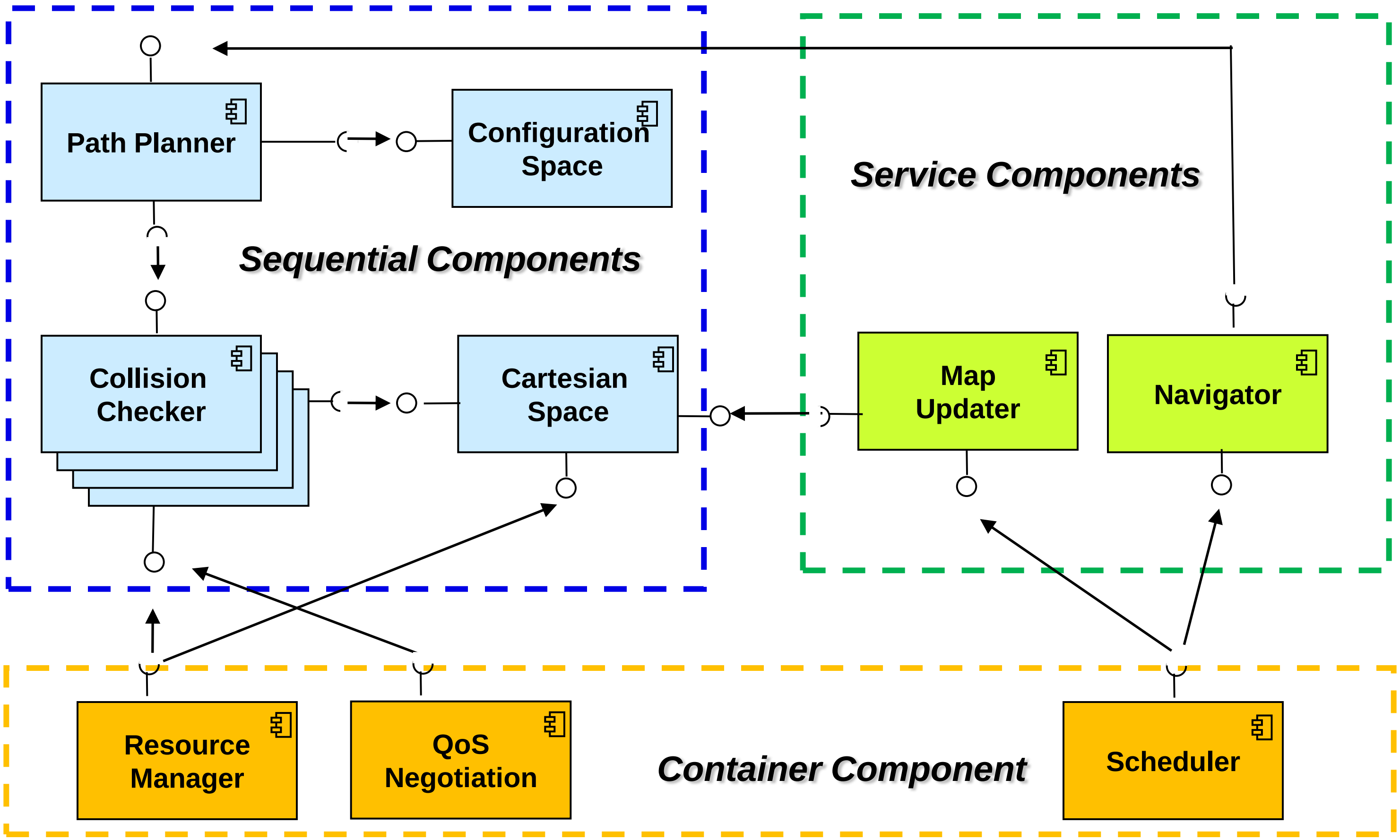
Component Interface & Implementation



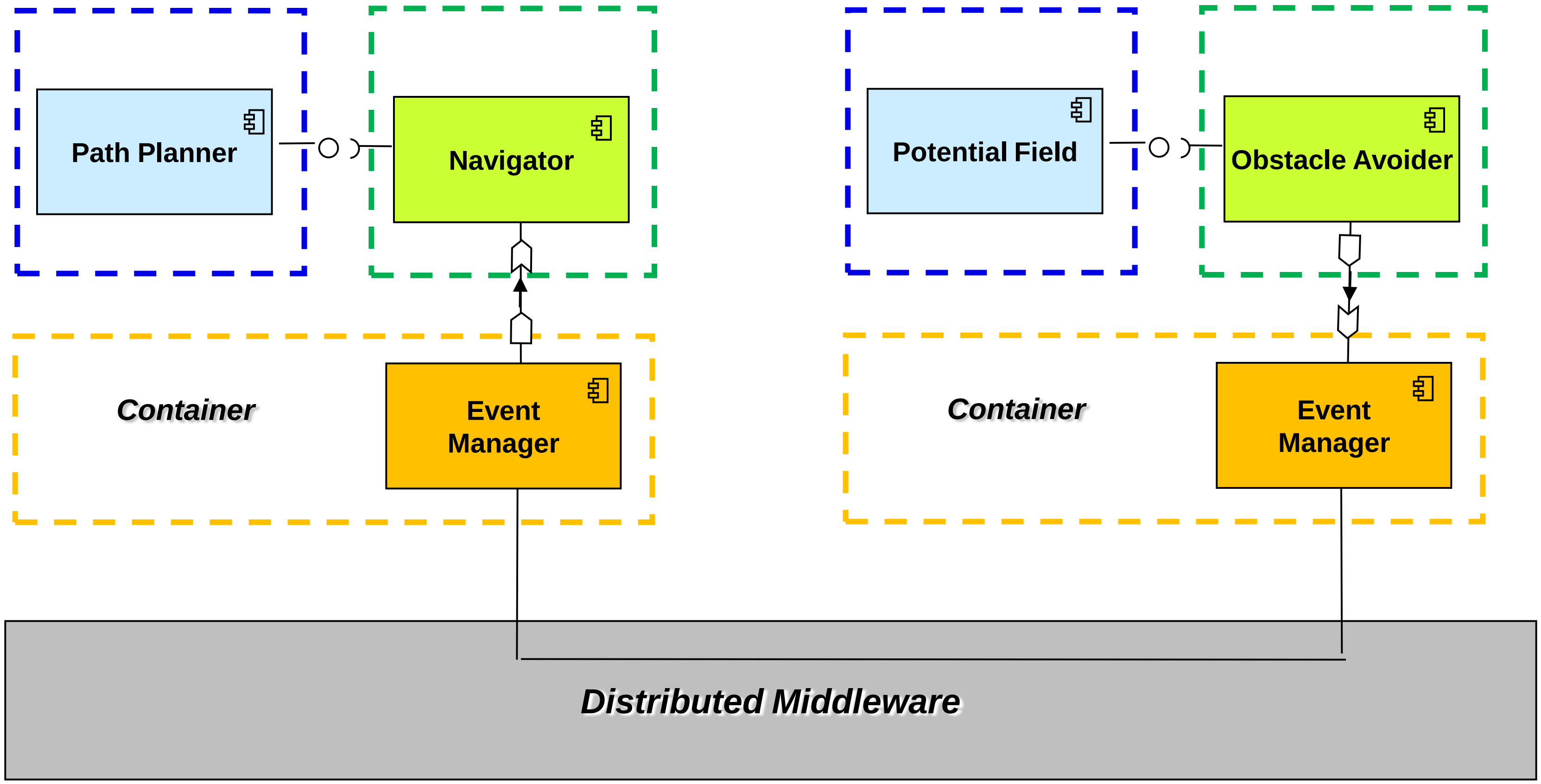
Computation



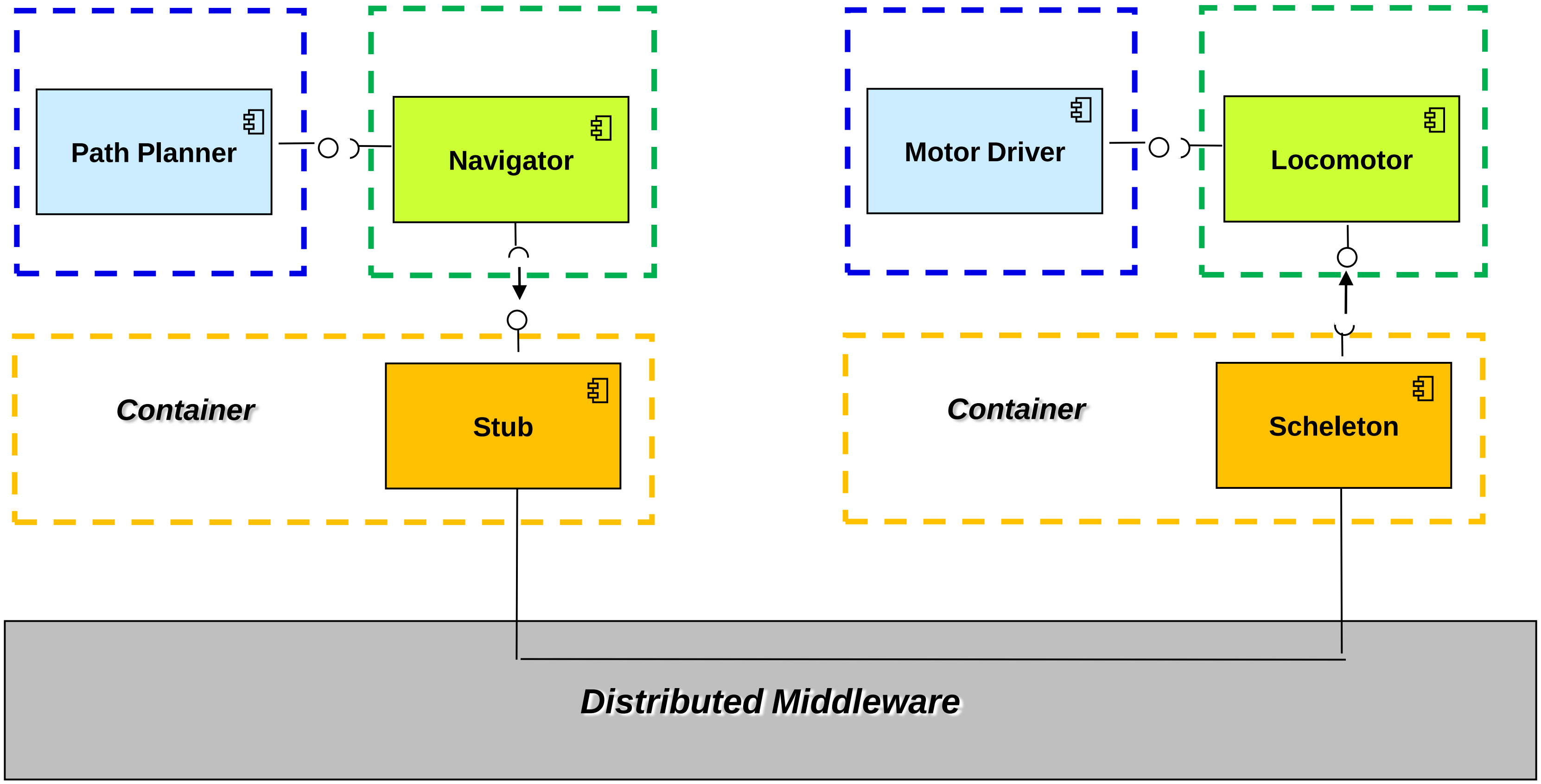
Configuration



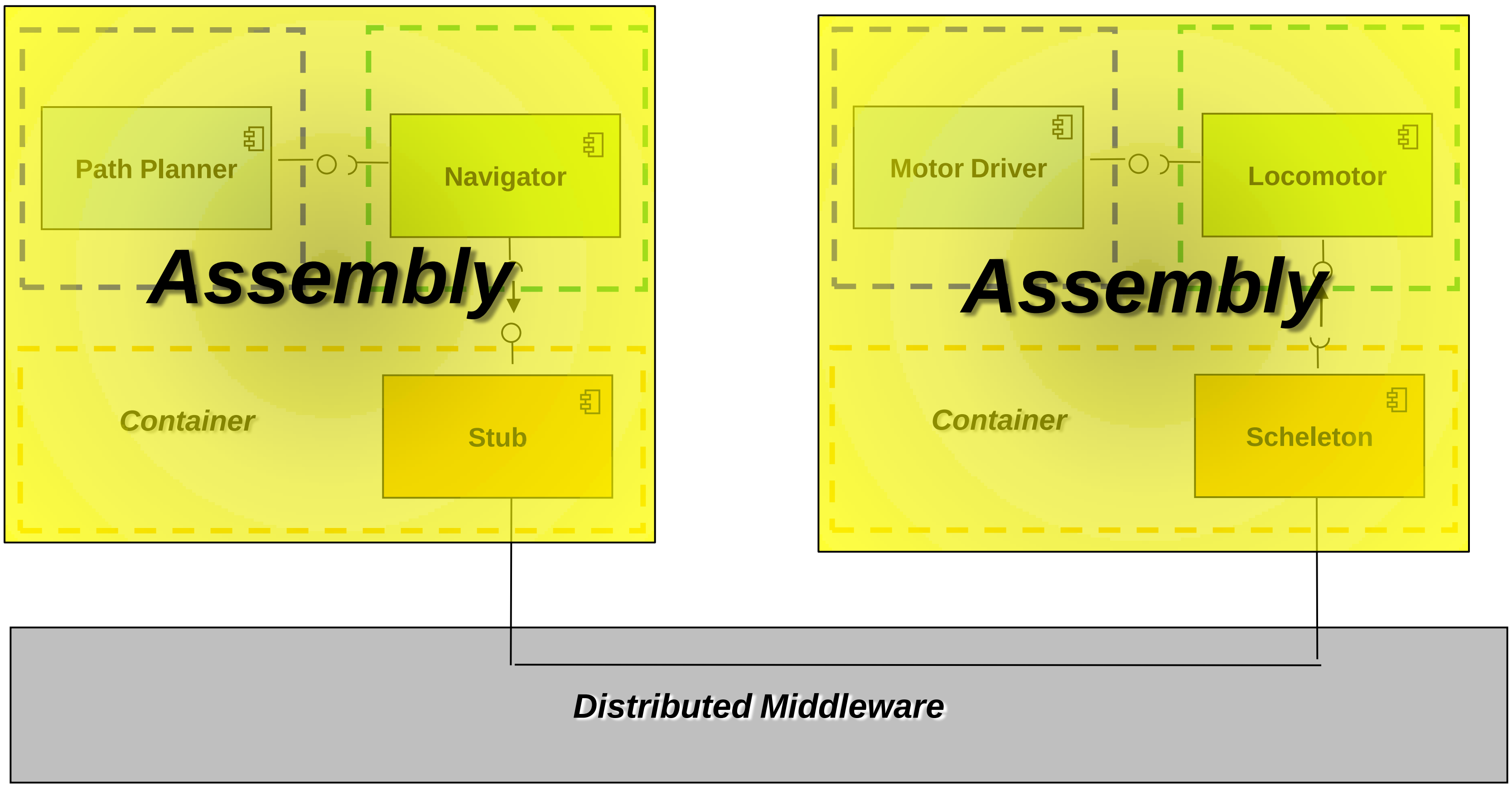
Communication



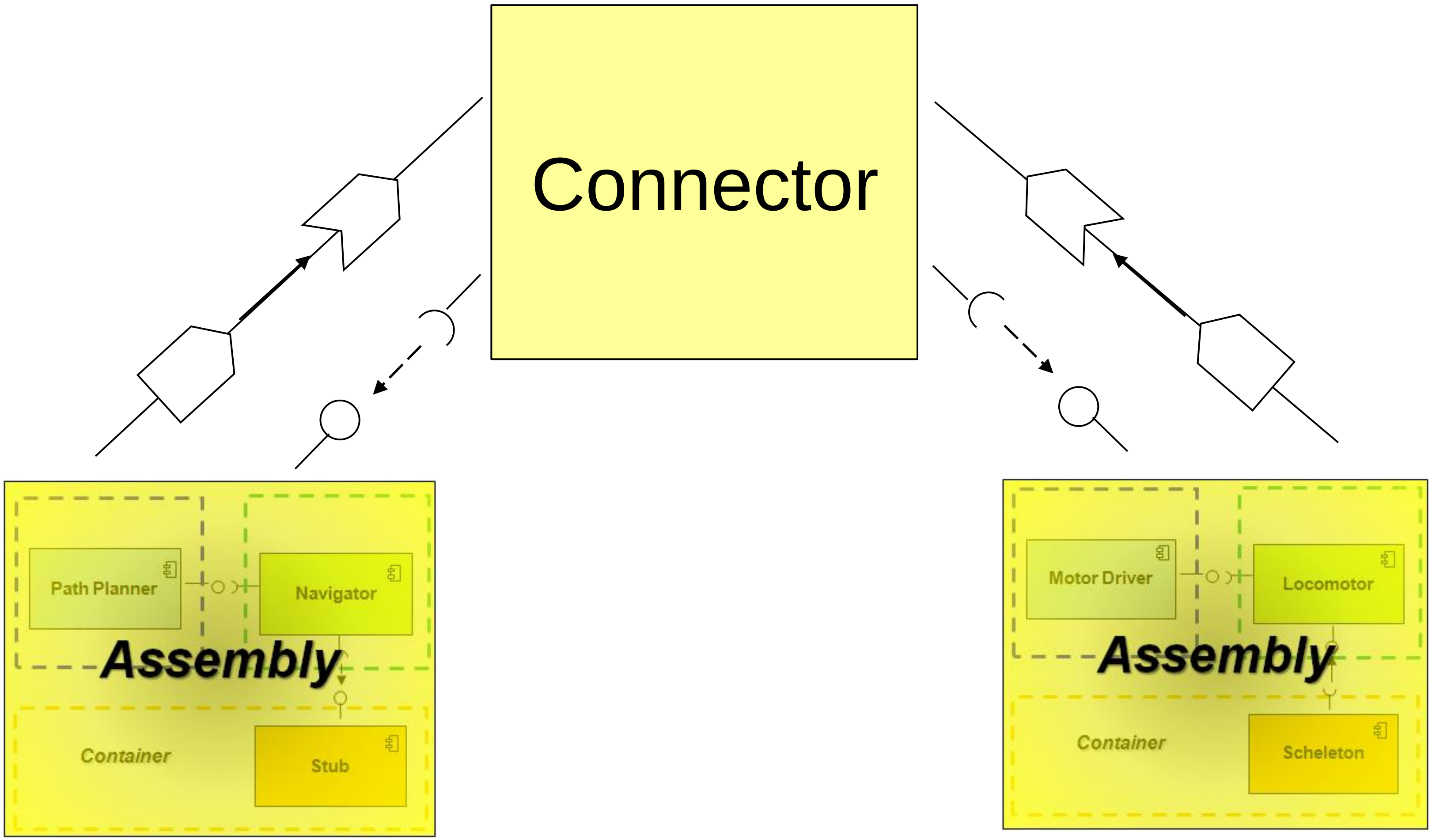
Communication



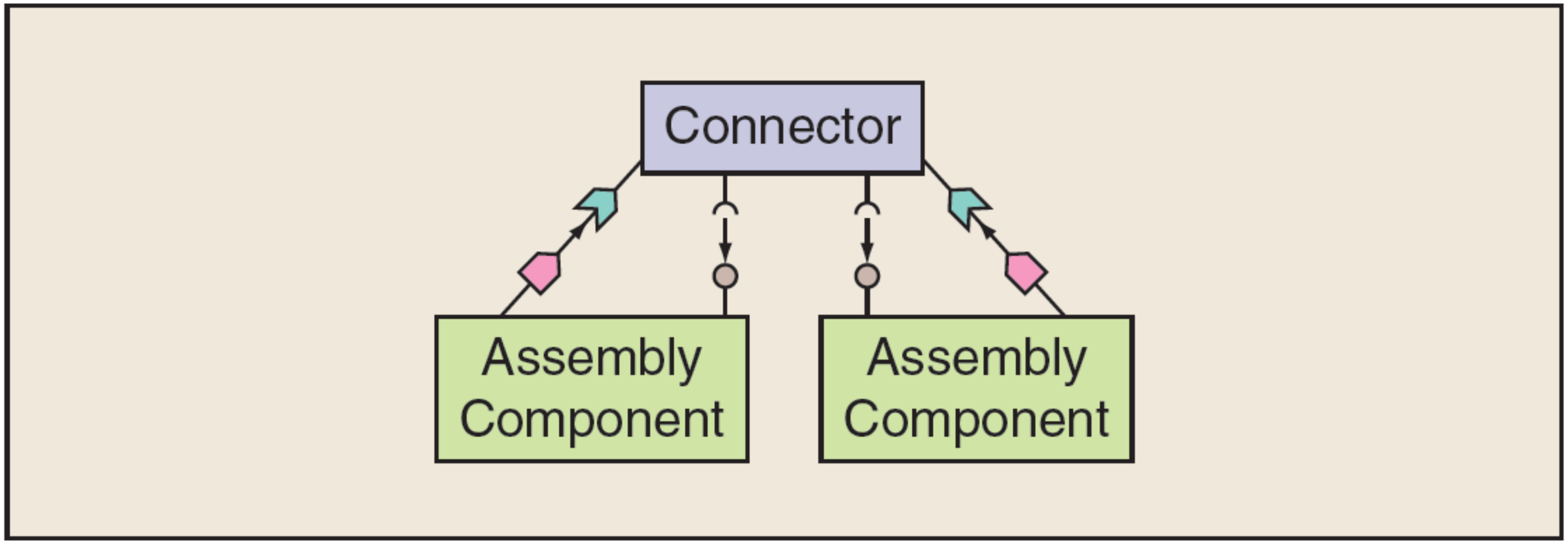
Communication

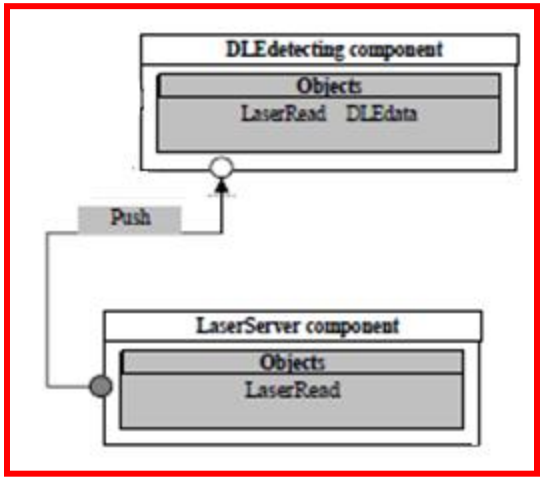


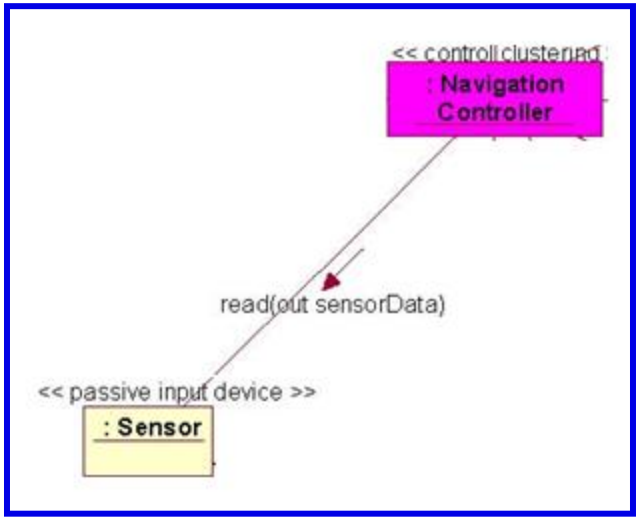
Coordination



Coordination







Malaga, Costa de Sol, Spain - October 24 - 29, 2010

- Invite best Ph.D. students AND PostDocs from all over the world
- We will provide
  - travel grants (1250 EUR for European students, 2000 EUR for international students)
  - the latest and coolest pieces of robot hardware in mobile manipulation
  - a DVD with best practice software for mobile manipulation
  - a fast Internet access
  - typical mobile manipulation tasks
- We expect in return
  - a competitive solution to the given tasks either using the provided or self-developed algorithms for mobile manipulation demonstrated in two competitions on the last day of the research camp
  - critical feedback and revisions of the provided hardware and software

Thank you for your attention!

Davide Brugali

Università degli Studi di Bergamo, Italy

CAR 2010, Douai, 19-05-2010