2012 IEEE International Conference on
Multisensor Fusion and Integration for Intelligent Systems (MFI)
September 13-15, 2012. Hamburg, Germany

A Multi-Robot Platform for Mobile Robots – A Novel Evaluation and Development Approach with Multi-Agent Technology

Sebastian Rockel, Denis Klimentjew, Jianwei Zhang
{rockel, klimentjew, zhang}@informatik.uni-hamburg.de

Abstract— **In the field of robotics, typical, single-robot systems encounter limits when executing complex tasks. Todays systems often lack flexibility and inter-operability, especially when interaction between participants is necessary. Nevertheless, well developed systems for robotics and for the cognitive and distributive domain are available. What is missing is the common link between these two domains.**
This work deals with the foundations and methods of a middle layer that joins a multi-agent system with a multi-robot system in a generic way. A prototype system consisting of a multi-agent system, a multi-robot system and a middle layer will be presented and evaluated. Its purpose is to combine high-level cognitive models and information distribution with robot-focused abilities, such as navigation and reactive behavior based artificial intelligence. This enables the assignment of various scenarios to a team of mobile robots.

Index Terms— **mutli-agent system, mutli-robot system, mixed reality, distributed architectures, sensor fusion, intelligent systems, middle layer .**

I. Introduction

Multiple robots collectively committed to a task play a more important role in today's robot research than in the past. In earlier research one, probably quite sophisticated, robot was adapted to solve a task very efficiently. There has been some scientific research into the collaborative effort of multiple robots assigned to a task. Robot teams have been applied in various research fields, with the navigation of multiple robots in a synchronized manner being one of the most active. In [1] a team of two hexapod robots collaborate in navigating unknown territory. Other works propose a dedicated architecture for navigation and tracking using multiple robots in unknown environments. In [15] a team of robots cooperates to navigate and to organize dynamic formations. The latter work overlaps another quite active field in robotics, namely research into *formations* of multiple robots. Research has also been done in a grid-based formation and the synchronization and configuration of agents. Other studies discuss fault-tolerant formations of mobile robots, while in [14] the focus is on stable and spontaneous self-assembly of an MRS. Another frequently investigated topic is that of a team of robots *building a map*. [6] explores a ground-mobile robot and a quadrocopter cooperating to build a three dimensional map. Traditional two dimensional SLAM with multiple robots in unknown territory, using Pioneer-3DX mobile robots and barcode markers, is described in [5]. Theoretical algorithms for cooperative, multi-robot, area exploration are described in [16]. In contrast to conventional two or three dimensional grid maps, [4] solves the problem of an MRS collaboratively creating a topological map. Another study, focusing on three dimensional, laser-based modeling with a heterogeneous team of mobile robots combining ICP, SLAM and GPS in an outdoor scenario can be found in [7]. Other areas of research concentrate on the problems of *task allocation and sub-division,* which arise when a task is to be split into sub-tasks for each robot. [11] uses a box pushing *scenario* in a heterogeneous MRS, while a framework for multi-robot coordination and task allocation is proposed in [12]. A traditional Artificial Intelligence (AI) topic, reinforcement learning in cooperative MRS, is described in [13].

II. System Architecture

The goal is to connect existing Multi-Robot Systems (MRS) and Multi-Agent Systems (MAS) using a new middle layer, the *Robot System Abstraction Layer* (RSAL), to create a three-layer architecture.

The introduction of the RSAL alleviates some mutual constraints imposed by the MAS and MRS. Robot and device actions take significant time to execute (moving an arm or actuator for example), whereas agent methods must return immediately or are at least designed for operations separable in small chunks of work. The middle layer has to manage the transition between a synchronous interface to the robot hardware and an asynchronous interface provided to the MAS. Therefore it encapsulates service oriented facilities, such as subscribing to notifications of lower level device events. Moreover, while MRS APIs differ, it is desirable that the MAS be independent of the MRS. In summary, an additional middle layer decouples the MAS and MRS.

This work combines a high-level MAS with an MRS. Current technologies in MAS and mobile robotics are used to achieve a high degree of task flexibility. Some specialized algorithms are implemented, such as for robot control, but in general, out-of-the box drivers and interfaces are used. An MAS is assumed to provide certain features:

- that a sample task can be *described* by the MAS tools and that the required *definitions* (such as of agents) can be created conveniently within the framework;
- that complex tasks can be *divided* into sub-tasks and *distributed* to multiple agents;
- and that distributed agents can use the MAS network *communication* layer to exchange data.

The MRS also has to fulfill some minimal requirements. The most important feature is path planning, although there might be use-cases where this is not needed. Path planning therefore involves several sub-tasks, especially the ability

978-1-4673-2511-0/12/$31.00 ©2012 IEEE

to *localize* on a map using sensors, such as laser or sonar rangers. Knowing its position, the robot should be able to plan a valid trajectory through unoccupied space (floors and rooms) and not hit anything, including both static objects (those on the map) and dynamic obstacles (those not on the map, such as people, furniture or other robots). Last but not least, the MRS has to provide the drivers for all hardware, such as the robot, sensors and effectors.

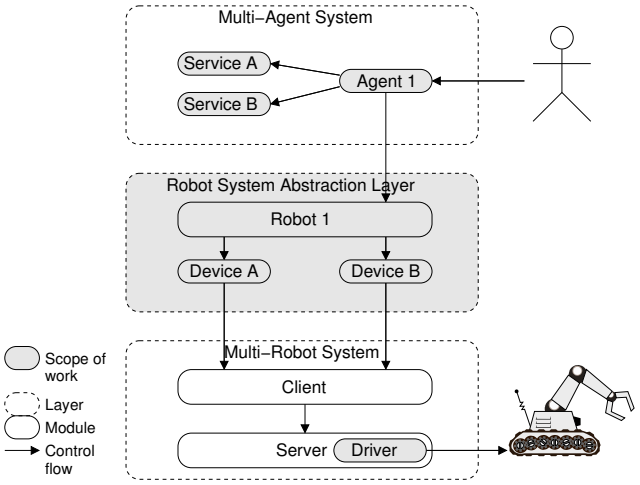


Fig. 1: The RSAL three-tier architecture: the MAS, RSAL and the MRS (here Player/Stage)

The meta-platform consists of three layers, as depicted in Figure 1. The user interacts with the top-most layer, the MAS. More specifically the GUI is provided by Jadex[1], the MAS in the current implementation. Jadex provides a clear and efficient interface for starting scenarios and individual agents and for passing the necessary arguments.

Note that although some MRS have a built-in robot navigation GUI (Playernav, RVIZ) that can observe current plans and waypoints, the typical user interface is the MAS. The MAS includes agents that will be started according to the scenario. Additional agent-related components, such as services, are also included in the MAS layer. Agents call robot facilities from the RSAL. A robot can be controlled using its external interface or using interfaces to attached devices, such as a planner device for navigation.

The interface between the MAS and RSAL layers provides callback facilities that allow different timing requirements to be accommodated and that decouple the MAS and MRS. The RSAL robots are device modules that include calls to the MRS. In the current Player/Stage[2] MRS these calls are passed to *PlayerClients* and are synchronous (as supported by the MRS). Thus the device implementation has to decouple these synchronous, blocking calls and its asynchronous interface. A PlayerClient is, in RSAL terminology, a *DeviceNode*. A DeviceNode is a unique access point within the network scope that provides interfaces to a group of hardware devices. Normally each device has a driver included in the MRS to access the hardware. For example, the p2os Player/Stage driver controls the Pioneer motors, whereas another driver (*hokuyo_aist* from the GearBox project[3]) controls the laser ranger hardware. An overview of the architecture is given in Figure 1 and depicts all three layers as well as internal modules.

Agents typically use services to perform tasks and for communication throughout the network (Figure 2b). These communication services are used to create information channels to which an agent can subscribe in order to read or publish interesting information (Figure 2a). Each agent can subscribe to a different set of services according to its activities and abilities.

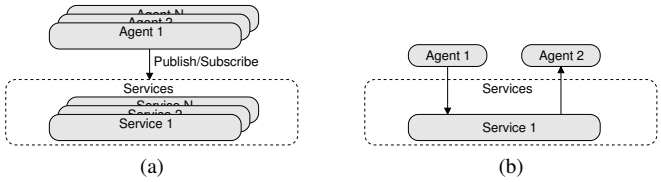


Fig. 2: Agents can subscribe to dedicated Services which represent communication channels (a). Multiple Agents participate in a conversation (b)

The MRS Player/Stage provides another method of remote access to robot hardware, shown in Figure 1 as a client/server system. The client is a local proxy for the device hardware from the caller's point of view. Any call is transparently transferred by the client through the network to a server. The server, which normally runs on the robot-attached host computer, parses messages and controls the hardware via its drivers.

III. RSAL Middle Layer

The RSAL layer embeds data, robot, device and behavior components. Furthermore it serves as the middle layer between MAS and MRS. The logical structure is organized in components.

The data component contains central data types used throughout other components. The fundamental type is *Position* which is depicted in Figure 3a. This class implements convenient methods related to (planar) distance and (robot) coordinate transformation. Thus a homogeneous matrix contains the rotational and translatory transformation as shown in Equation 1. In order to preserve object orientation in the world frame, normalization of the object's and robot's combined orientation is necessary, as shown in Equation 2. The angle normalization is done within the range $-\pi$ to π where π itself is excluded.

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}_O^W = \begin{pmatrix} cos(\theta_R) & -sin(\theta_R) & x_R \\ sin(\theta_R) & cos(\theta_R) & y_R \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}_O^R \quad (1)$$

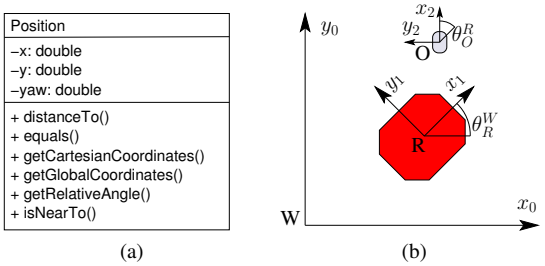
$$\theta_O^W = norm(\theta_R^W + \theta_O^R) \quad (2)$$

[1]http://jadex-agents.informatik.uni-hamburg.de/xwiki (May 29, 2012)

[2]http://playerstage.sourceforge.net

[3]http://gearbox.sourceforge.net

Big letters indicate frames (coordinate systems), such as for the world (W), the robot (R) and the object (O). A vector can have superscript and subscript frame letters. If a vector is given with a superscript frame letter it means this vector contains coordinates relative to that frame origin. In other words, the coordinates are local to the frame origin. A subscript frame letter means that the vector contains coordinates of the given frame (origin) relative to another frame (origin). The position class and frame conversion are depicted in Figure 3.



(a) (b)

Fig. 3: The Position class (a) and the robot coordinate frames *W, R, O* (b)

Goals within a scenario can be specified by the *Goal* class. They are typically defined by a planar position within the world frame. A Goal is used for example by the *BoardObject* of the blackboard implementation. Another central type is that of a *Board*, which implements a blackboard pattern. With this class, memory is provided to store various chunks of data that would represent notes on a bulletin board in the real world. This blackboard has several advanced features compared to its real world analogue. Notes or board objects can be ordered into topics and can have an associated timeout value. The *Robot* component contains a generic Robot class from which currently implemented specialized robots inherit. The Robot base class represents a mobile robot moving on the ground. The specialized Pioneer robot adds a behavioral model (Figure 4) and states that can be monitored. Specialized robot classes are derived from it. The device class provides the abstract class for all real robot hardware devices, and for virtual devices such as those of the *Blobfinder* or *Simulation* type. Devices can inherit from each other for specialization, which is the case for the laser ranger and sonar ranger. These devices both inherit from the type *Ranger*. A device is typically event-based and concurrent (Figure 5).

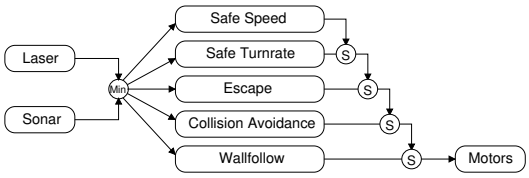


Fig. 4: The (RSAL) Pioneer robot sensor fusion model

As some devices have other devices attached (logically or physically), objects can be linked within an ordered hierarchy of devices. For example a robot is represented as a device and typically has other devices, such as ranger sensors, mounted. This introduces recursion into the class model (Figure 6). The dynamic handling of devices and the searching for it in the device tree at runtime is provided.

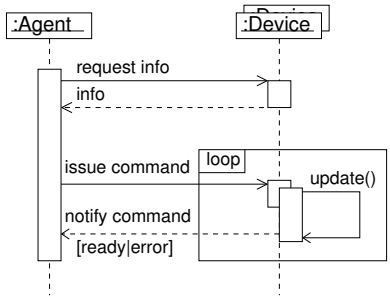


Fig. 5: The Agent Device access

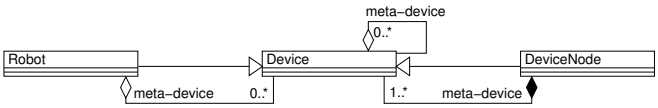


Fig. 6: The Device class hierarchy

The *Localize* Device class implements features that locate the robot within a world frame. It uses the underlying MRS localization driver, currently AMCL, encapsulates any dependencies and provides a consistent interface. As most localization algorithms keep track of the current position hypotheses with a (symmetric) covariance matrix. The covariance matrix implemented is depicted in Equation 3. Each matrix element (a covariance of two variables) is calculated as shown in Equation 4, where a_i is a raw coordinate value, $\overline{a}$ is the mean of all values and N is the number of all values.

$$COV^{3x3} = \begin{pmatrix} cov(x,x) & \dots & \dots \\ cov(y,x) & cov(y,y) & \dots \\ cov(\theta,x) & cov(\theta,y) & cov(\theta,\theta) \end{pmatrix} \quad (3)$$

$$cov(a,b) = \Sigma(a_i - \overline{a})(b_i - \overline{b})/N \quad (4)$$

The *Planner* class provides access to the underlying path planning driver. It encapsulates different drivers and specifies a generic interface. The *Simulation* class provides access to a simulation environment such as that provided by Stage. Interaction between software clients and the virtual world is provided so that the position of dynamic objects, such as robots or furniture, can be changed dynamically. This is currently used to test software units with repeatable environmental configurations. Moreover this interface is integrated to allow a mixed reality approach (Definition 1). This means that the positions and orientation of real robots can be used to place virtual proxies of those robots into a simulation environment. Such a scenario allows interaction between

real and virtual robots. Further robot related devices are available (gripper, motors etc.). An overview of all currently implemented devices and their inheritance is depicted in Figure 7.

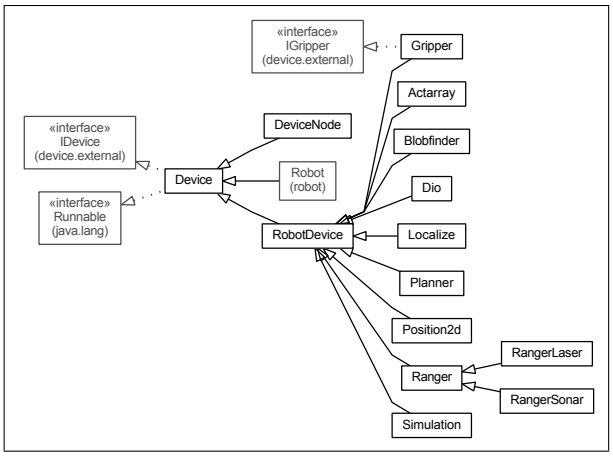


Fig. 7: The Device class and its specialized classes

The Behavior component implements a basic set of behaviors for a mobile robot. The behaviors are combined in a hierarchical subsumption architecture [3] to provide robust obstacle avoidance and escape strategies in case the robot becomes stuck. Sensory input is provided by laser and sonar rangers. These sensor inputs are combined to allow accurate detection of environmental obstacles and to avoid weaknesses of standalone sensors: highly reflective or glass walls confuse laser rangers and soft surfaces confuse sonar rangers. The behaviors are optimized for the Pioneer robot and work out of the box. Nevertheless they are easily adaptable to other robots and sensors and can be combined with behaviors at a higher abstraction level than wall-following (implemented here).

IV. Evaluation and Mixed Reality

The technologies used include recent developments in the areas of multi-agent systems (Jadex [9], [10]) and multi-robot systems (Player/Stage). Jadex is a proven, reliable and efficient system for research and industrial purposes[4]. The Player/Stage project has been used by many problem-solving activities in robotics world-wide[5]. Based upon these major systems, the implemented abstraction layer (RSAL) benefits from stability and flexibility. RSAL is implemented in Java, which supports state-of-the-art object-oriented software development. In particular, the RSAL implementation is heavily threaded: each robot and device runs in its own thread, handles synchronization and provides an asynchronous interface. This approach dynamically decouples platform components and also benefits from current and future multi-core computers.

In addition to the major components described above, other implementation aspects are noteworthy:

- Current classes for real-world robots, devices and agents facilitate software re-use within extended or alternative scenarios;
- The callback mechanism in use provides for easy, event-driven, access to RSAL services by any client;
- The blackboard design pattern allows easy object storage and retrieval and provides basic features for a robot-learning architecture. The lightweight implementation does not degrade performance, allowing components, such as agents, to have their own "memory" in addition to memory distributed across a network of participants.

The use of different robot types, or even of similar robots with different device configurations, often introduces a high platform configuration effort. The presented implementation allows for automatic robot device detection. This feature focuses on a tree-node approach where devices are connected to a root node, the *DeviceNode*. Dynamic retrieval of currently active devices is provided and includes the recognition of devices that malfunction and can no longer be used. This approach adds a layer of abstraction above the robot hardware and allows agent design to focus on cognitive features.

The mixed reality configuration presented has been implemented using pre-existing components of the platform. These include a device handling a simulation interface and an agent that listens on active services to retrieve the information needed to update the simulation. This feature benefits from software reuse and provides a completely new facility. The mixed reality approach is currently an active field of robot research. It provides facilities for a graphical user interface, enabling it to display virtual and real robots simultaneously within a simulation environment. The approach can be extended.

Some of the scenarios presented in this work run in a mixed reality environment. The difference between this and purely real environments is as follows. In a purely real environment robots can sense only physical objects present in their environment. In contrast, in an augmented real environment physical robots interact with virtual objects, such as simulated robots.

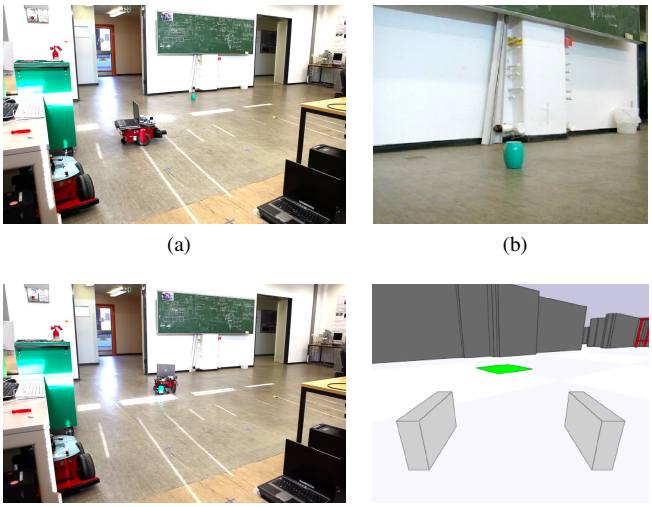
Definition 1. *Mixed Reality in Cognitive Robotics*

Mixed Reality in the context of this work refers to a hybrid environment mixing real and virtual objects as well as robots. Where as in augmented reality the physical world is extended with virtual visualizations, in a mixed reality environment a virtual or simulated environment is extended by real world information or models and in return updates the real world with information from the virtual environment at the same time.

The Hunt and Prey scenario can be run in either real-only, virtual-only or mixed reality configuration. For the MAS and RSAL layer components this distinction is not present as device access is encapsulated transparently within the MRS. One exception might be that a robot can be aware of a simulation environment when passing an appropriate simulation device to it (section III). Thus it is possible that

[4]jadex-agents.informatik.uni-hamburg.de/xwiki/bin/view/Usages/Projects
[5]http://playerstage.sourceforge.net/wiki/PlayerUsers (July 11, 2011)

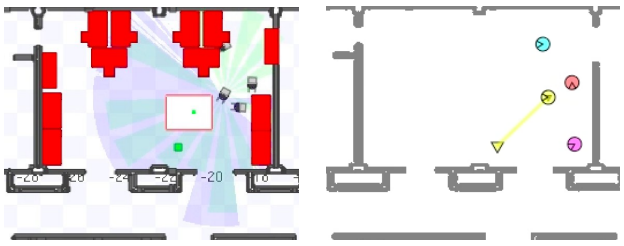
real robots hunt a virtual robot, virtual hunters follow a real prey or a team consists of both types. In either case there is interaction between virtual and real agents/robots. In the Find and Collect scenario interaction in the mixed reality configuration has been demonstrated (Figure 8, 9).



(a) (b) (c) (d)

Fig. 8: Several snapshots during the Find and Collect scenario. Four perspectives of situations during the scenario are shown. The (start) configuration of the mixed reality scenario is as follows. A virtual Pioneer robot carrying a blobfinder device (a virtual camera) searching for green blobs (objects) in the (virtual) environment. The presented wall-following behavior serves as the exploration algorithm. In order to allow real-world interaction, a green bin is placed at the exact blob position in the TAMS laboratory. A real robot, the Pioneer-2DX with a gripper attached, waits for blob position targets in order to approach, grasp the real bin and bring it back to its start position. The simulation environment is a necessary part of the scenario. (a) The active collector robot (center) waits for new targets. (b) The green bin target seen from the perspective of the approaching collector. (c) The collector has grasped the target object and returns home. (d) The same perspective as in *(b)* but in the virtual environment.

A big advantage in such a configuration is the possibility of simulating (robot) hardware that is not present in reality, for example where the required number of robots or the required device type is not available. Another advantage is the abstraction from hardware problems. Often an algorithm has been designed and needs to be tested. Typically a simulation environment is chosen for initial testing. Later on, the algorithm is tested on the real hardware. In the latter step, unexpected difficulties often arise as hardware introduces other possible errors. In preliminary work a wall-following algorithm performed well in the simulation and performed poor in reality (in early development state). In order to focus the research effort on the main goal such problems can be avoided by running a stable mixed reality environment having only components in reality that are mature enough.



(a) (b)

Fig. 9: Another event within the same run of the Find and Collect scenario depicted in Figure 8. (a) Virtual environment view. The event of the searching robot detecting a green blob within its field of view (white square in center) is depicted. Communication regarding the newly found target is triggered. (b) The collector robot listens for such targets and has planned a path to the target, shown here in the planner view.

A step-wise approach to migrating each component to reality is transparently possible. Research can focus on the global problem instead of dealing with low-level (hardware) issues.

Different scenarios provide an overview of basic platform features. All scenarios runnable with real robots such as Hunt and Prey, or Find and Collect, demonstrate successful integration of the robot hardware in use. Navigation, as implemented, allowed accurate localization within the TAMS laboratory and the whole floor. The planner worked well with the configuration and hardware. Additional devices such as a gripper and robot-mounted laser ranger finder were integrated successfully.

The integration of the MAS demonstrated the seamless coexistence of MAS and MRS platforms. All features of the agent system worked successfully.

The Hunt and Prey scenario shows inter-agent communication as well as basic cooperation. Interaction happens in a dynamic environment applying multiple, state-of-the-art, robotic features such as wall-following, sensor fusion and a subsumption behavior architecture. This software performs well, is robust in the case of hardware errors (such as erroneous laser readings and hardware defects), and successfully handles different hardware configurations.

Whereas the Hunt and Prey scenario focuses on a simple behavioral model of participating agents, in which the behavior of each agent/robot does not change during the scenario, the Find and Collect scenario implements a two-step interaction. The collector agents are triggered only upon receipt, via the agent services, of information describing an "interesting" object. Both finders and collectors use local memory that implements the blackboard model, thus allowing them to represent an autonomous model of the real or simulated world. The Find and Collect scenario was inspired by the Jadex Cleaner World example and the trash collecting robots in [8].

The Swarm scenario demonstrates the scalability and stability of the platform on a single host. Each robot consists

of an agent component that communicates via services, a robot model that uses its own devices and finally the localization and navigation component of the MRS. This single-robot model is replicated 100 times. In addition, this scenario provides a starting point from which to explore and implement dedicated robot swarm behaviors.

The user interface provides a multi-window and component focused surface. The Jadex Control Center provides a well arranged user interface to start/stop agents and scenarios. Runtime parameters can be changed and passed to agents and sophisticated agent debugging features can be applied. An optional Java console supports information display and error localization. This is the main interface, as the user typically interacts at agent/scenario level. Each MAS typically provides its own, optimized interface.

For a more robot-centric view of a scenario, an MRS typically provides a variety of tools. For watching current robot localizations and planned trajectories, Player/Stage provides the *Playernav* utility. This provides an easy overview of all participating robots and their goals. Other MRS, such as ROS, provide their own tools for visualization, such as the *RVIZ* utility.

When working with simulated environments instead of real ones or when applying a mixed world scenario, the Stage simulator provides a convenient, almost three-dimensional, interface and allows robot data to be visualized. ROS also integrates the Stage simulation.

Overall performance depends upon the applied scenario. In general, total processing delays are the sum of delays in the MAS, RSAL and MRS layers. The threaded design of the RSAL and the flat, optimized, call hierarchy are apparent in the low overall impact of the RSAL on total delays. MAS and MRS delays vary according to the current state of a scenario. The MAS can produce significant startup delays. However, by creating many services when starting, it can reduce its impact on the overall delays within a running scenario. The major impact comes from localization components: calculation of initial pose estimates at startup takes a long time, as no previous positions are known (unless positions are exactly predefined) and many particles have to be processed by the localization component. With the currently implemented and optimized configuration, this delay is minimized. Moreover the delay is not recognizable during the main scenario processing. The AMCL algorithm used needs very few resources once it has a position estimate, as further updates take the previous position into account. Nevertheless, the localization and navigation components limit overall system responsiveness when many robots operate simultaneously, such as in the swarm scenario.

When multiple agents cooperate to solve a task, several key issues have to be considered. The meta-platform provides the basic infrastructure for nearly any task-solving strategy and there are usually many possible configurations, each with advantages and disadvantages. A suitable system configuration is the key to the effective solution of a given problem. The key issues to be considered are highlighted in [2] and are discussed below.

When approaching a task, one must consider the jobs each robot has to handle. Can all sub-tasks be managed by the same robot type or are robots with differing capabilities required? The answers to this question constrain the scenario configuration. If all robots are the same, they can be seen transparently, as the physical representation of an agent that can manage any task. In this case, the selection of a robot for a task depends only upon robot location. In contrast, where robots of differing capabilities are used, the selection of a robot for a task depends upon both robot type and location. Heterogeneous robot teams require additional interaction and communication effort, which has to be considered in the scenario design. Homogeneous robot teams require coordination, but are more flexible in terms of the specific robot assigned a task. In the scenarios presented, this differentiation can be understood. The Hunt and Prey and the Swarm scenario are based upon homogeneous robots, whereas the Find and Collect scenario directs a heterogeneous group consisting of explorer and collector robots with differing hardware that notably changes their use and behavior. Whereas in the homogeneous cases the role of each participant can be chosen arbitrarily, this is not true in the heterogeneous case. Here the role of each robot type is distinct in terms of its basic usage. The explorer has a blob-detecting device and no effector to collect anything. Therefore it has to interact with a supporting collector robot that does not have an optical detection device but instead has a gripper, allowing it to grip, move and release objects. The two groups need each other to complete the task of searching and collecting objects in their environment. The implemented scenario handles this by interactively triggering events on the appropriate agent communication channels (services).

Another key aspect of cooperative task solving is the control of distributed robots. Whereas centralized control over multiple participants allows efficient and redundancy-free management, distributed or autonomous control of each robot introduces another level of robustness. The central control approach implements a dedicated planner that communicates with scenario participants in a peer-to-peer manner. Information retrieval in this case is simple for the planner, as it processes all data anyway. It can collect data and make decisions from experience in order to control other robots. This design can be exploited for complex scenarios where a lot of information from different sources must be processed in order to make reasonable and time-critical decisions. In contrast, a decentralized design introduces more responsibilities to individual robots. Each robot has its own goal, resources and plans. Coordination is achieved by each robot solving its individual task. This approach requires less complexity from each individual participant than a central planner, which reduces the risk of design or implementation failures. Nevertheless this comes at the cost of adjusting each component in order to integrate a team for cooperative task solving. The implementation introduces a central planner, such as the distribution agent, which triggers and receives robot data and controls their formation according to the number of robots and their positions. A practical scenario

would most likely benefit from a mixed control design.

When formations are required for a task, the design has to focus on the question of loosely or tight-coupled robot teams. Although in the presented scenarios almost no relative robot positions were important, in other scenarios, robots might operate as a group and therefore have to dynamically coordinate relative positions.

Environmental and task requirements can introduce constraints on communication links. Whereas in some scenarios it is possible to share all robot information, for example by broadcasting, in other scenarios such sharing is not allowed or is impossible because of the environment. Good communication should prefer information-hiding to polluting the network with unwanted or unimportant data. Nevertheless information should be available to all receivers that need it.

Another key aspect of such a meta-platform is the human-robot interface, or in other words, task assignment. How is a task assigned to a group of robots? This topic is related to control design: where a central planner is available, it can accept a task, process it and delegate sub-tasks to participating robots. If no such central process is available, the task has to be subdivided in advance in order to delegate sub-tasks directly.

Finally certain environmental constraints, such as indoor or outdoor territories, introduce special system designs. Furthermore, learning requires its own cognitive components and must be handled at a more abstract level of interaction, as for single robots.

The meta-platform presented here provides all services to support scenario development in the field of robotics. The following paragraphs introduce some suitable application areas.

In the research field of search and rescue robots, interaction becomes mandatory as typical scenarios are too complex to be accomplished by a single robot. A search operation benefits from the number of search participants. The deployment area can be covered in less time with an increased number of robots. In order to efficiently cover the environment without re-discovering the same area it is important to coordinate the robots' targets dynamically.

Swarm formation, a recent research topic, benefits from this work by applying dedicated formation algorithms to the system.

The scenarios presented are based on a pre-defined static map that is available to planner components. Another scenario would be to explore unknown territory using a team of autonomous robots. Present state-of-the-art SLAM algorithms can robustly map a territory using a single robot system. For a distributed SLAM, new algorithms can be designed and implemented in the meta-platform in order to exploit its communication and data memory facilities and to benefit from a multi-robot team.

Communication between a group of agents or robots is another field of current research, one in which efficiency, scalability and flexibility are important topics. The currently implemented service infrastructure can be enhanced in order to explore these issues.

The important service-robot use-case for a foraging population can also take advantage of such a highly interactive platform. Different kinds of household robot might share information on activities to be performed and on environmental changes, or could pass operator requests to the best suited robot. Nevertheless, such complex scenarios can be combined with distributed sensor networks to support data retrieval by small, fixed sensors (both dedicated or robot-attached) in the domestic environment, such as refrigerators and washing machines.

A practical environment for the Find and Collect scenario would be that of a production line. Unused or rejected parts, as well as trash, are created during manufacturing. Such parts could be spotted by mobile or fixed sensors, such as cameras. Mobile collectors would be given the spot positions and would autonomously find their way to the target, grasp it and bring it to a disposal position before returning to their idle task.

Often certain facilities have to be under permanent surveillance, such as museums, company buildings and military territories. Such a task can be solved by multiple cameras watching the important area. The field of view can be augmented by various mobile robots that periodically observe covered territory. In the case of an emergency event, appropriate actions can be triggered, such as sending mobile robots to the location of the problem. Robots following surveillance routes through indoor or outdoor environments could maintain their battery charge state and servicing autonomously, for example by heading to a recharging point when necessary.

V. Design and Implementation

The MAS layer contains components related to high-level services that include cognition and distribution. It is implemented in Java and XML. Java is used for agent definition including initialization, body and de-initialization. XML serves as a scenario container, in which any scenario participants are declared along with related initialization parameters. A scenario can consist of different configurations, such as varying numbers of agents from the start. These configurations (in Jadex: *Applications*) can be grouped within one scenario (file). The current implementation prefers the Jadex agent model of *MicroAgents* over the more complex BDI agents for reasons of simplicity and efficiency. Nonetheless, any agent model provided by the MAS is supported.

All services inherit from a basic service class and implement sending and receiving facilities to communicate with subscribers. A subscriber can filter received messages so that it receives only those of interest. Each service represents an information channel and follows a defined (simple) protocol.

VI. Summary

This work deals with a generic software platform integrating multi-agent technology and a multi-robot system. It describes the use of the platform for typical cooperative robotic scenarios. One Pioneer-3AT and two Pioneer-2DX

from MobileRobots Inc. serve as the hardware base for real-world applications. The robots have sonar and laser ranger sensors attached and some have grippers. With the ranger devices, the robots are able to locate their position on a static map in an indoor environment. The gripper is used for object manipulation. The multi-agent system, Jadex, is integrated for cognitive and task-distribution services. Furthermore, the multi-robot system Player/Stage is used to interact with the robot hardware and to provide a simulation environment.

The development of the software is described in Figure 1. The robot navigation stacks were configured to reach high localization and path planning accuracy within an indoor environment. For this purpose, a proportional and accurate grid-map of the territory was created by a particle-filter-based SLAM algorithm. The requirements for a generic system integrating MAS and MRS lead to the design of a generic middle layer. Implementation and testing use current software design patterns and focus on run-time efficiency. The middle layer consists of a flexible concept for robots, behaviors and devices independent of the specific MAS and MRS in use. Moreover agents, communication services and scenarios have been implemented and tested. A mixed reality graphical user interface allows simulated and augmented-reality scenarios. The efficiency of the overall system is scalable, allowing for control of a large number of robots simultaneously.

A multi-robot platform with multi-agent technology based on Player/Stage and Jadex has been introduced. Core components can be used with other systems of the same kind, such as other MAS and MRS. Two real-world scenarios and one virtual scenario served as the basis for the realization of interactive collaboration. A concluding discussion of current and future research into various topics related to the meta-platform has been presented.

Finally the use of a mixed reality environment provides certain advantages, in particular by allowing testing with more robots than are physically available. Moreover, it enables the use of unavailable or non-existent devices, allowing novel ideas to be simulated rather than being limited to present reality.

VII. Acknowledgments

This work has been conducted as part of RACE, funded under the European Community's Seventh Framework Programme FP7-ICT-2011-7 under grant agreement n° 287752 (http://www.project-race.eu/).

References

[1] M. Ahmed, M. Khan, M. Billah, and S. Farhana. A collaborative navigation algorithm for multi-agent robots in autonomous reconnaissance mission. In *Computer and Communication Engineering (ICCCE), 2010 International Conference on*, pages 1 –6, May 2010.

[2] G. A. Bekey. Systems of robots: from cooperation to swarm behavior. IEEE SMC 2005, Hawaii – Computer Science Department, University of Southern California, December 2005.

[3] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[4] C. Carletti, M. Di;Rocco, A. Gasparri, and G. Ulivi. A distributed transferable belief model for collaborative topological map-building in multi-robot systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2010*, 2010.

[5] L. Carlone, M. Ng, J. Du, B. Bona, and M. Indri. Rao-blackwellized particle filters multi robot slam with unknown initial correspondences and limited communication. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 243 –249, May 2010.

[6] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller. Multiple relative pose graphs for robust cooperative mapping. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3185 –3192, may 2010.

[7] R. Kurazume, Y. Noda, Y. Tobata, K. Lingemann, Y. Iwashita, and T. Hasegawa. Laser-based geometric modeling using cooperative multiple mobile robots. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3200 –3205, May 2009.

[8] D. C. Mackenzie, R. C. Arkin, and J. M. Cameron. Multiagent mission specification and execution, 1997.

[9] A. Pokahr and L. Braubach. From a research to an industrial-strength agent platform: Jadex v2. In H.-G. F. Hans Robert Hansen, Dimitris Karagiannis, editor, *Business Services: Konzepte, Technologien, Anwendungen - 9. Internationale Tagung Wirtschaftsinformatik (WI 2009)*, pages 769–778. Österreichische Computer Gesellschaft, 2 2009.

[10] A. Pokahr, L. Braubach, and K. Jander. Unifying agent and component concepts - jadex active components. In *In Proceedings of Seventh German conference on Multi-Agent System TEchnologieS (MATES-2010)*, 2010.

[11] C. Rossi, L. Aldama, and A. Barrientos. Simultaneous task subdivision and allocation for teams of heterogeneous robots. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 946 –951, may 2009.

[12] P. Shiroma and M. Campos. Comutar: A framework for multi-robot coordination and task allocation. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4817 –4824, 2009.

[13] X. Sun, T. Mao, J. Kralik, and L. Ray. Cooperative multi-robot reinforcement learning: A framework in hybrid state space. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1190 –1196, oct. 2009.

[14] K. Suzuki, T. Tsukidate, M. Shimizu, and A. Ishiguro. Stable and spontaneous self-assembly of a multi-robotic system by exploiting physical interaction between agents. In *Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, IROS'09, pages 4343–4348, Piscataway, NJ, USA, 2009. IEEE Press.

[15] P. Urcola and L. Montano. Cooperative robot team navigation strategies based on an environment model. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4577–4583, Oct. 2009.

[16] J. Yuan, Y. Huang, T. Tao, and F. Sun. A cooperative approach for multi-robot area exploration. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1390 –1395, 2010.