**Concept**

**System Architecture**

The proposed concept is realized in form of a system with the following components:

* Context monitor
* Repository
* Safety monitors
* Deployment infrastructure

**Context Monitor**

Context monitor is responsible for observing the autonomous system’s environment and current status. In the given example, the robot’s 1. state of motion and 2. gripper status are constantly checked by the context monitor. The component monitors the environment with the help of sensors. On the other hand, it is subscribed to different components of the robot so that it stays well-informed about their current statuses. The context monitor instantly updates the observed changes to the repository so that the information is shared with other necessary components.

**Repository**

Repository is considered to be the central component in the proposed system. It acts as the knowledge base that contains data that were added during the design-time and also the information updated during the run-time. Data added during the design-time mostly include static(does not change during run-time) information like total number of existing hardware platforms in the system with their features, constraints to be satisfied for selecting a platform etc., On the other hand, information updated during the run-time are dynamic in nature. Memory availability in each platform is a good example for a property that changes during run-time. It is desirable to consider current working status of a sensor to be dynamic in nature. This is helpful in avoiding deployments in the platform with faulty sensors.

The Repository offers an interface to other components to insert new data to the repository and to query some specific data from it. It broadcasts the updates to other components with the help of the “notify” functionality. Broadcasting is triggered by events and changes that happen in the component.

**Safety Monitors**

Safety Monitors are responsible for constantly checking whether the task designated to the autonomous system is executed in a safe manner. In the given example where the robot is responsible for picking, transporting and placing various objects, it is important to keep an eye on how safely these tasks are executed without harming humans in a collaborative environment. One of the ways to ensure safety in this case is to constantly check for slippage in the gripper with help of suitable sensors such as force and tactile. Slip detectors correspond to software components of the sensors in our system. Whenever a slippage is detected, appropriate force has to be exerted on the object by the gripper in order to prevent its fall.

It turns out that the performance of slip detectors (force and tactile) in detecting slippage differs according to the context (robot’s environment). The tactile slip detector is suitable to detect slips from a firmly held object whereas the force slip detector is more accurate than the other in detecting a complete slip. There are cases in which the fusion between force and tactile slip detectors are effective.

**Deployment Infrastructure**

There are often more than a hardware platform available in a distributed autonomous systems. In order to achieve maximum utilization of available platforms, there arises the need to adaptively deploy software components on them based on the current context. At the same time, there is a need to firstly resolve suitable platforms/platform for the chosen software component based on some or all of the following requirement types as specified by OMG deployment specification [1]. The following table gives examples for each of them in the context of the object picking robot that is considered for our discussion.

|  |  |
| --- | --- |
| **Requirement type** | **Example** |
| Quantity | Minimum number of tactile sensors to select a platform |
| Capacity | Minimum memory availability required for a deployment |
| Minimum | Minimum system latency |
| Maximum | Maximum system latency |
| Attribute | Presence of force sensor in a platform |
| Selection | Suitable force sensor type |

**Working**

As soon as the context monitor reports a change in the robot’s context which expressed in terms of its current state of motion and gripper status, the repository broadcasts the information so that it enables both selection of the right safety monitor and platform for its deployment.

**Selection**

**I. Safety Monitors**

Selection of the right safety monitor according to the context in [2] is carried out with help of the Perception graph deployment module. However, its working is specific to RPSL. In our implementation, this achieved differently with help of the safety monitor selector as **described in ---.** Safety monitors are selected based on the current context.

**II. Platforms**

**a. Criteria for selection**

Once the slip detector according to the context is chosen, it has to deployed on an appropriate platform. In our implementation, platform selection is done with help of the platform selector module which In the object picking robot example, the suitability of platform is described is expressed in terms of the following requirements. The first three requirements are identical to our reference [2], whereas the other three are our contribution:

**R1:** The force slip detector should be deployed on a platform to which the force sensor is connected.

**R2:** The tactile slip detector should be deployed on a platform to which all tactile sensors are connected.

**R3:** The combined slip detector should be deployed on a platform with at least 250MB working memory

**R4:** The combined slip detector has to be deployed on a platform whose latency is at least 200 ms

**R5:** The combined slip detector has to be deployed on a platform whose latency is 5 ms at the maximum

**R6:** The force slip detector has to deployed on a platform that contains a simple pressure sensor.

**b. Possible scenarios**

Following are possible scenarios during the selection of platforms:

1. Selection of exactly one suitable platform: This is the ideal case where the selector returns exactly one platform meeting the requirement and is suitable for deployment.

2. Selection of multiple suitable platforms: This is the scenario in which at a given time more than a platform satisfies the given requirements. In this case, the selector returns all of them. However, there arises the need to rank and choose the most suitable platform.

3. Unavailability of a suitable platform: In this case, the platform selector returns None or any other equivalent to convey the absence of a suitable of platform meeting the specified requirements.