Technische Umsetzung des Anwendungsszenarios „Gewerbliche Reinigung“ im Projekt AutoPnP

Autoren: Dr.-Ing. Birgit Graf (IPA)
Ulrich Reiser (IPA)
Richard Bormann (IPA)
Christopher Parlitz (Schunk)
Frank Schermer (Dussmann)

Inhalt

1. Anwendungsszenario und benötigte Plug&Play-Fähigkeiten

Das Gesamtszenario „gewerbliche Reinigung“ befasst sich mit der autonomen, bedarfsgesteuerten Fußbodenreinigung von Büroräumen sowie der Papierkorbentleerung mit Hilfe eines mobilen Roboters. Das Szenario gleicht der sog. ergebnisorientierten Reinigung, welche eine hohe Anforderung an Erkennung und Umsetzung fordert, da nur zielgerichtet gearbeitet wird. Dadurch lassen sich „Leerreinigungen“ vermeiden und Arbeitsabläufe optimieren. Neben der Gesamtintegration sind beide Teilaufgaben auch gesondert voneinander implementiert, um die in AutoPnP entwickelte Plug&Play Architektur bestmöglich evaluieren und demonstrieren zu können.

Die Applikation zur autonomen **Fußbodenreinigung** lässt den Roboter Büroräume nach eventuellen Verschmutzungen absuchen, die dann mit Hilfe eines Reinigungsgerätes entfernt werden. Fähigkeiten des Roboters, die dazu genutzt werden, sind u.a. das autonome Anfahren offen stehender Büros (Komponente/Fähigkeit: Navigation), die Inspektion der Fußbodenfläche eines Raumes (Inspektion), die automatische Erkennung und Kartierung von Verschmutzungen (Schmutzdetektion) sowie die Reinigung der Verschmutzungen (Reinigung).

Zur autonomen **Papierkorbleerung** gehören Funktionen wie das autonome Anfahren offen stehender Büros (Navigation), die Inspektion eines Raumes (Inspektion), die automatische Erkennung von Papierkörben (Papierkorbdetektion) sowie die Entleerung des Papierkorbes in ein zentrales Sammelbehältnis (Papierkorbleerung).

Abbildung 1: Anwendungsszenario „gewerbliche Reinigung“ mit reinigendem Serviceroboter.

Im Folgenden werden die Plug&Play-Anforderungen des Gesamtszenarios beschrieben, und zwar sowohl bezüglich Hardware als auch Software Komponenten.

1.1. Hardware-Plug&Play

Unter Hardware-Plug&Play soll die automatische Erkennung von angeschlossener Hardware und deren Einbindung ins System (Laden des korrekten Treibers, Anbindung an die Middleware, Registrierung der Fähigkeiten in der Komponentenverwaltung) verstanden werden.

Dies wird im Teilszenario Fußbodenreinigung dadurch demonstriert, dass die Reinigungsanwendung nicht gestartet kann, so lange am Roboterarm nicht das erforderliche Hardware-Modul verbaut ist. Die Komponentenanalyse muss feststellen, dass die für die Funktionalität *Reinigung* notwendige Hardware (z.B. ein Staubsauger) nicht am System angeschlossen ist, und informiert den Nutzer darüber. Nach dem Wechsel der Hand gegen das Reinigungsgerät erkennt das System die neue Hardware und ermöglicht das Starten der Reinigungsapplikation. Ebenso kann die Anwendung nicht gestartet werden, solange der für die Reinigung essentielle RGB-D-Sensor nicht am Robotersystem angeschlossen ist.

Im Anwendungsfall Papierkorbleerung wird auf ähnliche Weise gezeigt, dass die Anwendung nur dann gestartet werden kann, wenn am Roboterarm eine Hand angeschlossen ist. Ein angebautes Reinigungsgerät hat wiederum eine Aufforderung an den Nutzer zum Wechsel des Arbeitsgerätes zur Folge.

1.2. Automatische Komponentenkonfiguration

Im Falle einer nicht angeschlossenen RGB-D Kamera soll die Anwendung Papierkorbleeren trotzdem lauffähig sein, da sie nur Farbbilddaten benötigt. Die Middleware muss deshalb erkennen, dass die von der Anwendung ursprünglich genutzte RGB-D-Kameraressource auch durch eine der anderen Farbkameras des Roboters ersetzt werden kann, da beide Daten desselben Typs bereitstellen mit vergleichbaren Attributen. Das Programm soll dementsprechend auf den anderen Datenerzeuger umkonfiguriert werden.

1.3. Software-Plug&Play

Die Austauschbarkeit von Softwarekomponenten durch den Einsatz generischer Software-Schnittstellen wird durch die Wahl des Navigationsalgorithmus demonstriert. Üblicherweise nutzt ein omnidirektionaler Roboter für die Navigation Laserscanner, die alle Winkelbereiche (vorne, hinten, seitlich) um den Roboter abdecken und damit eine kollisionsfreie omnidirektionale Navigation ermöglichen. Programme, die von der Intelligenten Steuerung erzeugt werden, nutzen in diesem Falle die Vorteile der omnidirektionalen Navigation. Entfernt man nun zwei der Laserscanner, so muss die Intelligente Steuerung Programme mit einer differentiellen Navigation erzeugen. Zwar ist die Navigation damit u.U. weniger effizient, kann aber weiterhin alle Bewegungsvorgänge des Roboters durchführen. Je nach Hardwareausstattung kann also eine vorhandene Softwarekomponente durch eine andere, funktional ähnliche und mit gleichen Schnittstellen versehene ersetzt werden.

2. Umsetzung der anwendungsspezifischen Systemarchitektur auf Basis des generischen Plug&Play-Konzepts

2.1. Überblick

Generell entspricht die Systemarchitektur im Szenario „Gewerbliche Reinigung“ der Referenzarchitektur, die in allen drei AutoPnP-Szenarien zum Einsatz kommt und im Dokument „Technische Umsetzung des Plug&Play-Konzepts auf Basis von CHROMOSOME im Projekt AutoPnP“ beschrieben ist. Sie erfüllt damit die im selben Dokument formulierten Voraussetzungen für die Unterstützung von Plug&Play (anforderungsszentrische Ausrichtung, Modularität, einheitliche Syntax und Semantik, ressourcenschonende und deterministische Implementierung).

Das in diesem Szenario eingesetzte Automatisierungssystem ist der Serviceroboter Care-O-bot 3. Durch die Verwendung einer modularen Plug&Play Architektur ist es jedoch mit geringem Aufwand mittelfristig möglich, die erarbeitete Lösung auf eine speziell für dieses Einsatzfeld angepasste und damit entsprechend kostengünstige Roboterplattform zu übertragen. In Zusammenarbeit mit dem DAI-Labor wird das entsprechende Hardware-Plug&Play und die automatische Komponentenkonfiguration für die Fußbodenreinigung sogar durch einen kompletten Wechsel der Roboterplattform demonstriert, nämlich indem die Fußbodenreinigungsapplikation auf der Festo F5 Roboterplattform des DAI-Labors von der Intelligenten Steuerung automatisch installiert und ausgeführt wird, ohne dass weitere Softwareanpassungen notwendig sind. Hierbei müssen sich beispielsweise Softwarekomponenten wie die Navigation und die Schmutzdetektion auf veränderte Sensorkonfiguration und –typen einstellen.

Care-O-bot 3 ist das Produkt langjähriger Entwicklungen und wird auf unterer Ebene vom Robot Operation System (ROS) verwaltet. Für eine Anbindung des Roboters an die CHROMOSOME-Architektur, zur Nutzung der Vorteile von semantischem Plug&Play wie unten beschrieben, wurde – ähnlich wie im Szenario Industrieautomatisierung im Falle der SPS auf den Bearbeitungsstationen – anstelle einer vollständigen Portierung aller Algorithmen, Treiber, Komponenten und Werkzeuge des Robotersystems die Entscheidung getroffen, wichtige Funktionalitäten zu kapseln und an CHROMOSOME anzubinden. Durch eine vollständige Überführung aller relevanten ROS-basierten Komponenten in CHROMOSOME-Implementierungen würde man sich der Möglichkeit verschließen, an funktionalen Weiterentwicklungen der ROS-Komponenten zu partizipieren, abgesehen vom jeweils recht hohen Portierungsaufwand, der Entwickler von der Portierung gar gänzlich abhalten könnte. Zur Anbindung von Fähigkeiten, Softwarekomponenten oder Hardwaremodulen an CHROMOSOME ohne Verlust der Plug&Play-Fähigkeiten werden deswegen Proxies eingesetzt, welche die angebotenen Funktionalitäten und Daten der ROS-Komponenten nahtlos in der CHROMOSOME-Architektur zur Verfügung stellen. Damit erschließt sich CHROMOSOME einen großen Softwarepool, dessen Funktionalitäten ohne größeren Aufwand für den Anwender per Proxy gekapselt werden können.

Demnach sind Komponentenschnittstellen also sowohl CHROMOSOME über Proxies zugänglich, als auch direkt zwischen zwei ROS-Komponenten verschaltbar. Damit kann sowohl die Kommunikation zu ROS-fremden Komponenten oder der Anwendung (Central Control) über CHROMOSOME gewährleistet werden als auch die aus Performanzgründen günstigere Direktverschaltung zweier ROS-Komponenten eingerichtet werden, welche die zweifache Passage der Nachricht durch CHROMOSOME-ROS-Proxies vermeidet.

Die Anbindung von mit der Middleware ROS betriebenen Servicerobotern (Care-O-bot 3: IPA, Festo F5: DAI-Labor) an CHROMOSOME und der Einsatz von CHROMOSOME als übergeordnete Middleware bringt dabei folgende Vorteile:

- Gemeinsame Anbindung von Automatisierungssystemen verschiedener Domänen (Serviceroboter – ROS, Industrieautomatisierung – SPS, Haushaltsnetzwerke – CL-Bus).
- Umsetzung einer einheitlichen Komponentenverwaltung mit Komponentenbeschreibungsfunktionen, welche die Nutzung einer Intelligenten Steuerung zur automatischen Generierung von Applikationen ermöglicht.
- Realisierung einer domänenübergreifend einheitlichen Behandlung von Plug-Ereignissen mit Aktualisierung der Komponentenverwaltung.
- Nutzung der Intelligenten Steuerung und des Komponentenrepositories auf gleiche Art und Weise in verschiedenen Domänen, Möglichkeit zur Generierung gemischter Programme (z.B. Serviceroboter übernimmt Transportaufgaben im Rahmen der Fabrikautomatisierung).
- Relativ einfacher Zugriff auf große Anzahl vorhandener Softwarekomponenten aus anderen Domänen (z.B. ROS oder SPS-gesteuerten Bearbeitungsstationen) über Proxies und Wrapper, die zwischen den Kommunikationsmechanismen der verschiedenen Systeme übersetzen.

Im Folgenden bespricht dieses Dokument die Architektur unter Berücksichtigung proprietärer ROS-Komponenten, beschreibt die Play-Sicht generell und am Beispiel und geht danach auf die Plug-Sicht im Allgemeinen ein und erläutert mehrere Plug-Ereignisse technisch detailliert an Beispielen.

2.2. Systemarchitektur und Play-Sicht bei Verwendung proprietärer Komponenten

Die Anbindung von proprietären ROS-Komponenten ändert zwar grundlegend nichts am Architekturbild der Middleware CHROMOSOME (siehe Dokument „Technische Umsetzung des Plug&Play-Konzepts auf Basis von CHROMOSOME im Projekt AutoPnP“, Abb. 20 und 21), soll aber im Folgenden kurz für den Anwendungsfall der gewerblichen Reinigung präzisiert werden. Abbildung 2 gibt hierzu einen Überblick über die Systemarchitektur aus der Play-Sicht, also während der Ausführung von Anwendungen. Als zentrale Schnittstelle zum Benutzer ist die als CHROMOSOME-Komponente implementierte cleaning_gui genannte Bedienoberfläche zu sehen. Diese bietet die Möglichkeit zur Ausführung von Anwendungen wie der bedarfsgerechten Fußbodenreinigung oder der Papierkorbleerung. Solche Anwen-

dungen können vom Benutzer programmiert oder von der Intelligenten Steuerung erzeugt worden sein (siehe Abschnitt 2.3) und können beispielsweise als Zustandsautomaten implementiert sein, welche nacheinander verschiedene Funktionen des Roboters aktivieren und damit eine sinnvolle Aufgabe erfüllen. Angenommen, der Nutzer hat über die GUI die Anwendung Fußbodenreinigung gestartet, die als CHROMOSOME-Komponente vorliegt und im Kommunikationsnetzwerk von CHROMOSOME integriert ist, damit sie ggf. auch domänenübergreifend Automatisierungssysteme ansteuern kann (z.B. Bearbeitungsstationen, Heimnetzwerk-Bussysteme). Diese Anwendung beginnt im Zustand *detect dirt visually*. In diesem Zustand soll der Roboter eine Schmutzerkennungssoftware ausführen und dabei den Raum durchstreifen und nach Verschmutzungen am Boden suchen. Da diese Funktionen innerhalb der ROS-Umgebung roboternah implementiert wurden, wird CHROMOSOME eine Funktionalität detect_dirt_visually via Proxy bereitgestellt. Für CHROMOSOME ist damit egal, ob der Proxy selbst alle Algorithmen ausführt oder das Ausführungskommando nur an eine ROS-Komponente oder ein Netzwerk solcher weiterleitet. Tatsächlich wandelt der Proxy den Funktionsaufruf in einen ROS Service Call um, welcher anschließend über das ROS Netzwerk an die entsprechende ROS-Komponente geschickt und dort von dieser behandelt wird. Der Service Call kann Argumente enthalten, also eventuell vorhandene Argumente des CHROMOSOME-Funktionsaufrufes in die Service Nachricht verpacken. In CHROMOSOME gibt es u.a. eine Netzwerktopologie und einen Login Manager, einen PnP Manager sowie einen Execution Manager; ROS besitzt eine vergleichbare Netzwerktopologie und Mechanismen zur direkten Verschaltung von ROS-Komponenten. In Abbildung 2 werden beide Managementschichten als großer blauer (CHROMOSOME) bzw. roter (ROS) Kasten dargestellt.

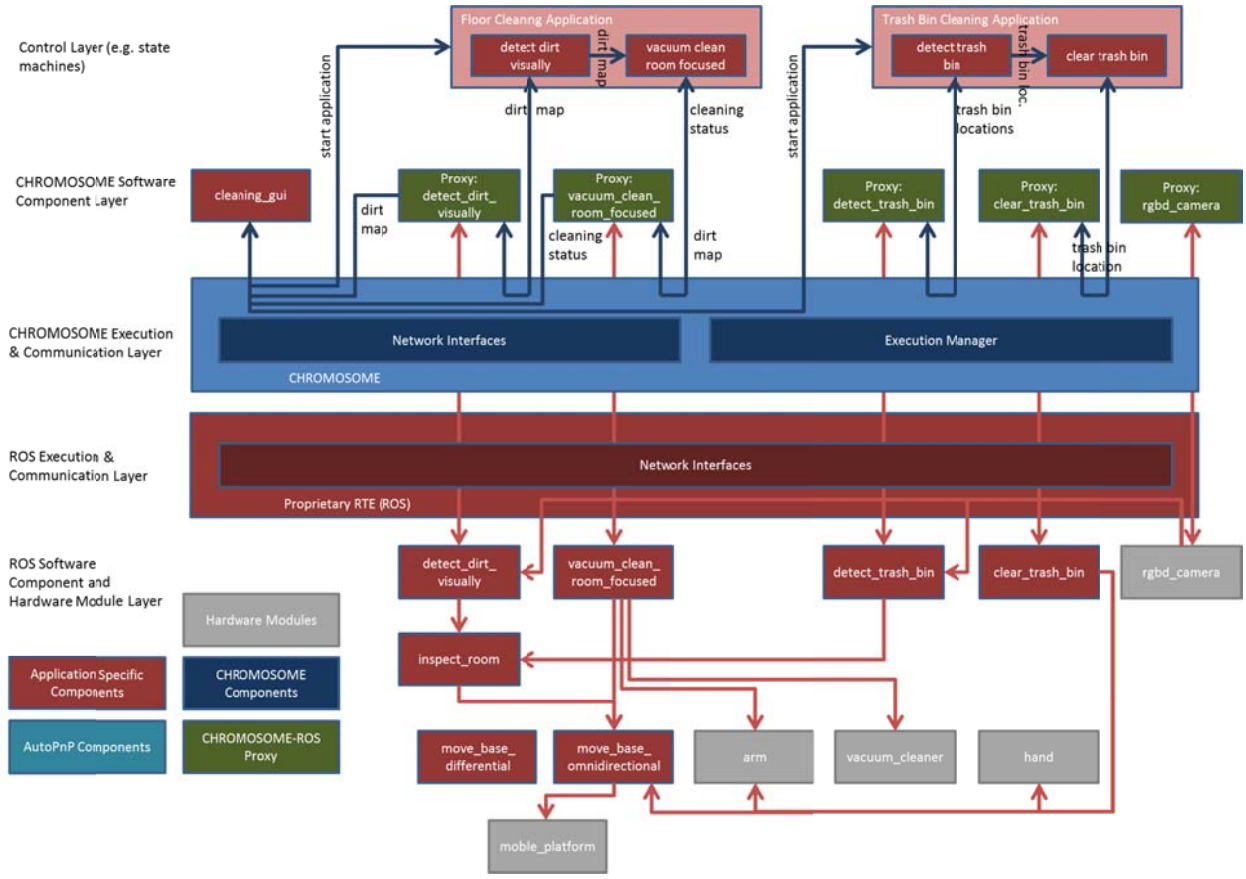

Abbildung 2: Architekturübersicht im Szenario „gewerbliche Reinigung“ aus der Play-Sicht

Die aufgerufene ROS-Komponente kann nun die eventuell vorhandenen Argumente des Service Calls auslesen und entsprechende Algorithmen ausführen. Unter Umständen ist die ROS-Komponente auf Daten weiterer ROS-Komponenten angewiesen. Diese wurden während des Deployments in CHROMOSOME mitgestartet und erhalten eigene CHROMOSOME-ROS-Proxies. Zur Vermeidung unnötiger Datenumwandlungen im Proxy erfolgt die direkte Verschaltung zweier ROS-Komponenten jedoch ausschließlich über die ROS-Topic-Mechanismen. Die ROS-Komponente kann u.U. sogar weitere Services zu anderen ROS-Komponenten aufrufen, hier gilt ähnliches. Im Beispiel würde etwa die Schmutzerkennung nebenher noch die inspect_room Funktionalität ausführen lassen. Nach Beendigung der Funktion der ROS-Komponente schickt diese eine Service Call Return Nachricht an den Aufrufer, also den Proxy. In dieser Nachricht können ebenfalls Daten enthalten sein; im aktuellen Beispiel ist es eine Karte, die Verschmutzungen in der geprüften Umgebung markiert. Mit der Rückmeldung von der ROS-Komponente terminiert der Proxy den Funktionsaufruf der Anwendung. Eventuell vorhandene Rückgabedaten werden vorher noch in die entsprechenden Variablen geschrieben. Im Beispiel wird die versandte Schmutzkarte in ein CHROMOSOME-kompatibles Format umgewandelt und der Anwendung zurückgegeben. Damit terminiert auch der erste Zustand der Anwendung.

Diese geht dann in den Zustand *vacuum clean room focused* über und übergibt die Schmutzkarte an diesen Zustand. Dieser Zustand ruft mit der Schmutzkarte die Funktion vacuum_clean_dirt_focused auf, die ebenfalls als Proxy-Komponente vorliegt. Wie vorher

auch, schickt der Proxy einen ROS Service Call an die entsprechende ROS-Komponente, die den Roboter für jede Verschmutzung auf der übergebenen Karte positioniert, die Reinigung durchführt und das Ergebnis prüft. Die Rückmeldung der ROS-Komponente an den Proxy enthält dann alle Koordinaten von erfolglos gereinigten Stellen. Nach Reinigung der verschmutzten Stellen terminiert die Anwendung. Währenddessen hat die Benutzeroberfläche über die Topics der Schmutzsuche- und Reinigungs-Proxies Informationen über gefundenen, gereinigten sowie nicht entfernbaren Schmutz empfangen. Diese Informationen werden in einer Karte dargestellt.

Die Entleerung von Papierkörben erfolgt analog, wie im Schaubild durch die Pfeile auch dargestellt. Der Übersicht halber wurden in Abbildung 2 nicht alle Proxies eingezeichnet. Zwar muss nicht jede ROS-Komponente einen entsprechenden Proxy besitzen, da die Funktionalität mehrerer ROS-Komponenten gekapselt sein kann und damit durch das Deployment einer Funktionalität alle abhängigen ROS-Komponenten mitgestartet werden (hierzu genügt der Aufruf eines Startskriptes, welche alle relevanten ROS-Komponenten enthält); jedoch haben insbesondere Hardwaremodule, für die Plug-Ereignisse behandelt werden sollen, einen Proxy. Beispielhaft ist dies für die rgbd_kamera dargestellt, die per USB und passiver Detect Funktion entdeckt und eingebunden werden kann. Dieser Sensor erhält während des Einbindevorgangs einen Proxy in CHROMOSOME, u.a. weil dadurch der Komponentengraph für die Berechnungen der Intelligenten Steuerungen komplett ist und auch Hardwareabhängigkeiten proprietärer System abgebildet werden können. CHROMOSOME-Komponenten könnten nun über den Proxy sogar Bilder der Kamera auslesen, was im vorliegenden Anwendungsfall aber nicht geschieht. Stattdessen werden Bilder der Kamera direkt innerhalb der ROS-Schicht an Komponenten wie detect_dirt_visually weitergeleitet.

Der Care-O-bot 3 besitzt 3 interne Rechner. Zur Umsetzung der Beschriebenen Systemarchitektur wird auf jedem dieser Rechner ein CHROMOSOME-Knoten gestartet und ein ROS Master auf dem ersten PC. Der Einsatz zweier Laufzeitumgebungen auf demselben Rechner hat zwar den Nachteil, dass von Seiten der CHROMOSOME-Middleware keine Echtzeitgarantien mehr möglich sind, dies kann aufgrund der üblichen Nutzungscharakteristika jedoch als unproblematisch für die Servicerobotersteuerung angesehen werden.

Um nicht alle Softwarekomponenten und Hardwaremodule über Plug-Ereignisse beim Start der CHROMOSOME-Middleware einbinden zu müssen, kann natürlich eine gewisse statische Grundkonfiguration entsprechend angeschlossener Hardware und Fähigkeiten vorgegeben werden. Dazu wird beim Start ein Basispaket an Treibern und Komponenten für den Serviceroboter Care-O-bot 3 geladen. Hierfür genügt die einmalige Einrichtung des ROS Masterknotens sowie die Ausführung eines Ladeskriptes für die Grundausstattung an Low-level Komponenten und Treibern.

Die Umsetzung von Plug-Ereignissen und die Anbindung neuer Hardwaremodule oder Softwarekomponenten erfolgt im Allgemeinen wie im Dokument „Technische Umsetzung des Plug&Play-Konzepts auf Basis von CHROMOSOME im Projekt AutoPnP“ beschrieben und in Abbildung 3 dargestellt. Im Zusammenhang mit der Verwendung proprietärer Komponenten und Module auf ROS-Basis sind jedoch noch einige Besonderheiten für einen optimalen Betrieb zu beachten, die in den folgenden Abschnitten näher erläutert werden. Abschließend

gibt dieses Kapitel noch eine detaillierte Übersicht über die Anbindung der wichtigsten Komponenten aus Abbildung 2.

2.3. Generierung eines Anwendungsprogramms durch die Intelligente Steuerung

Zur Erinnerung an die standardisierten Vorgänge in CHROMOSOME während eines Plug-Ereignisses oder der Neugenerierung einer Anwendung sind diese in Abbildung 3 nochmals zusammengefasst. Diese Darstellung ist identisch zu Abbildung 20 im Dokument [PnP-CHROMOSOME].

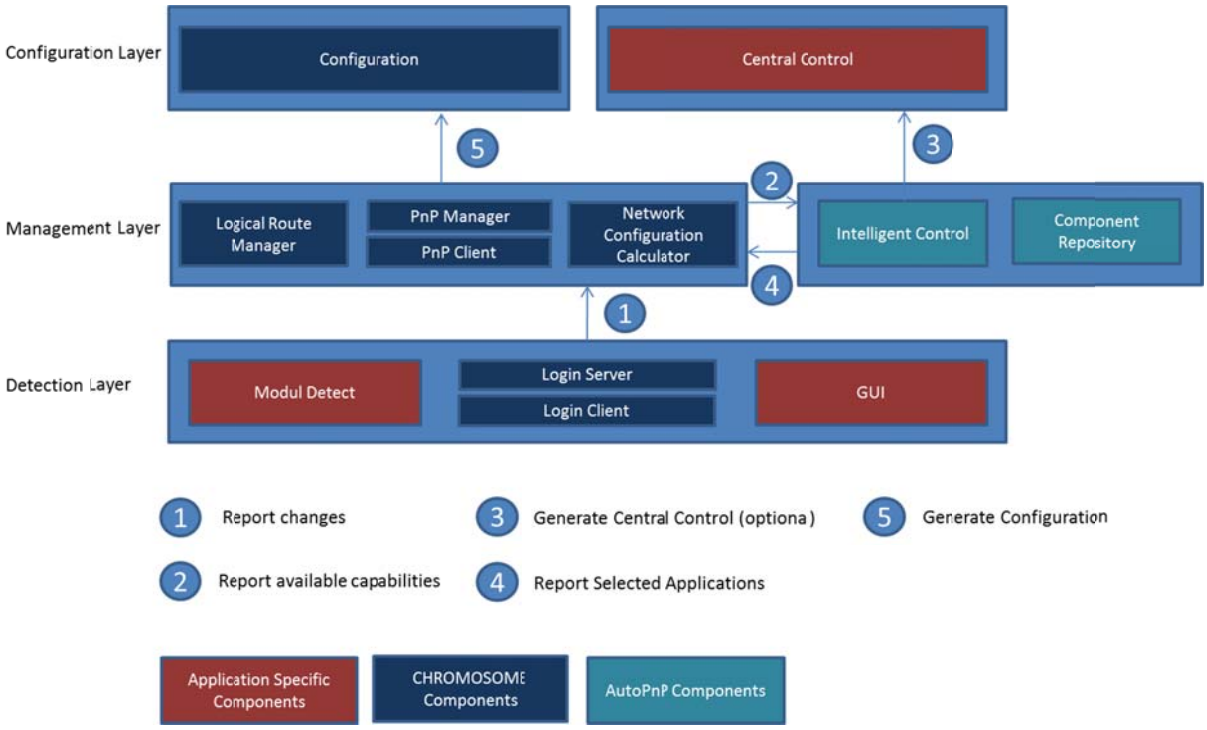

Abbildung 3: Architekturübersicht im Szenario „gewerbliche Reinigung“ aus der Plug-Sicht

Die Programmgenerierung durch die Intelligente Steuerung unter Einbeziehung von proprietären ROS-Komponenten und ROS-Hardwaremodulen gestaltet sich im Allgemeinen sehr ähnlich zu den Abläufen der Embed-Phase (siehe Dokument [PnP- CHROMOSOME], Abschnitt 5.2):

1. Der Nutzer fordert bei der Intelligenten Steuerung die Generierung einer bestimmten Anwendung an.
2. Die Intelligente Steuerung fordert beim PnP Manager über die Funktion generateGraph den Komponenten- und Kommunikationsgraphen an, welcher wiederum über die Funktion getLogicalRoutes vom Logical Route Manager generiert

und an den PnP Manager weitergeben wird. Der Graph enthält alle möglichen logischen Verbindungen zwischen momentan auf dem System verfügbaren Komponenten.

3. Die Intelligente Steuerung versucht nun mit den im System bzw. im Komponentenrepository (z.B. online Komponentendatenbank) vorhandenen Komponenten ein Programm zu generieren, welches das gewünschte Ziel erreichen kann.
    a. Dabei ist das Ziel in Form eines gewünschten Endzustandes oder in Form einer auszuführenden Fähigkeit zu formulieren.
    b. Jede Komponente, die unter CHROMOSOME verfügbar ist, also auch solche, die auf dem System nicht installiert sind, aber im Komponentenrepository vorliegen, besitzt eine Beschreibung ihrer Fähigkeiten, Abhängigkeiten zu anderen Komponenten sowie den Ergebnissen ihrer Ausführung. Im System vorhandene Komponenten werden in CHROMOSOME über ihr Typmanifest mit einem globalen Identifizierer für bestimmte Fähigkeitsklassen versehen. Dieser Identifizierer kann beispielsweise geteilt sein in eine Oberklasse Navigationsalgorithmen und eine Unterklasse omnidirektionale Navigation. Dieser Identifizierer wird der Intelligenten Steuerung für die im Komponentengraphen vorhandenen Komponenten mitgegeben. Damit kann die Intelligente Steuerung in seiner Fähigkeitendatenbank die vorhandenen Komponenten auf semantischen Informationen wie Fähigkeiten, Abhängigkeiten und Ausführungsergebnisse abbilden.
    c. Die Intelligente Steuerung sucht nun im aktuellen Komponenten- und Kommunikationsgraphen des Zielsystems nach Komponenten, welche den vom Benutzer gewünschten Zielzustand erreichen bzw. die gewählte Fähigkeit umsetzen. Findet sich unter den aktiven Komponenten keine passende, so kann die Intelligente Steuerung im Komponentenrepository eine geeignete Komponente aussuchen.
    d. Die Anforderungen der gewählten Komponente werden nun von der Intelligenten Steuerung mit dem vorhandenen Komponentengraphen verglichen und nach Möglichkeit Komponentenverschaltungen bzw. Komponentenaufrufe in die Anwendung aufgenommen, die bereits im System vorhanden sind. Nur wenn sich keine passende Komponente findet, kann das Komponentenrepository mit weiteren Fähigkeiten aushelfen. Iterativ wird dieser Vorgang solange wiederholt, bis alle Abhängigkeiten aller Komponenten erfüllt sind oder alle möglichen Verschaltungen auf Basis der vorhandenen Komponenten und des Komponentenrepositories untersucht wurden.
    e. Sind alle Komponenten und Module der nun generierten Anwendung bereits im System instanziiert, so ist die Arbeit der Intelligenten Steuerung abgeschlossen.
    f. Wurden jedoch in der generierten Anwendung Softwarekomponenten eingesetzt, die momentan noch nicht auf dem System installiert und gestartet sind, so lädt die Intelligente Steuerung diese Komponenten aus dem

Komponentenrepository nach und veranlasst ggf. deren Start. Anschließend geht es mit Punkt 2 weiter, also der Anforderung des aktualisierten Komponenten- und Kommunikationsgraphen über die Funktion generateGraph des PnP Managers.

g. Wurden in der generierten Anwendung Module eingesetzt, die momentan nicht auf dem System installiert sind, so informiert die Intelligente Steuerung den Anwender, dass dieser entsprechende Hardware an das System anschließen muss. Der Anschluss entsprechender Module wird dann durch die jeweilige Detect-Methode dem entsprechenden PnP Client gemeldet, welcher die neue Hardware vom PnP Manager im System anmelden lässt (siehe Dokument „Technische Umsetzung des Plug&Play-Konzepts auf Basis von CHROMOSOME im Projekt AutoPnP“, Abschnitt 5.1.2, sowie Abschnitt 2.4 in diesem Dokument). Dadurch wird wiederum ein neuer Komponenten- und Kommunikationsgraph berechnet und an in Intelligente Steuerung weitergegeben. Dann geht der Ablauf wieder mit dem Beginn von Punkt 3 weiter.

h. Kann in Ermangelung geeigneter Komponenten oder Module kein zielführender Ablauf bzw. keine passende Konfiguration für eine Anwendung erstellt werden, so bricht die Intelligente Steuerung die Applikationsgenerierung ab und informiert den Nutzer über die Gründe für den Fehlschlag. Dieser hat damit die Möglichkeit, fehlende Funktionalitäten selbst zu programmieren und die Anwendungsgenerierung erneut anzustoßen.

4. Hat die Intelligente Steuerung eine zielführende Anwendung erstellen können, so wird diese zusammen mit dem anwendungsspezifischen Komponenten- und Kommunikationsgraphen zurück an den PnP Manager gemeldet.
5. Der PnP Manager veranlasst beim Network Configuration Calculator die Prüfung der neuen Konfiguration. Dabei werden die physikalischen Verbindungen sowie Marshaller und Demarshaller eingerichtet. Darüber hinaus prüft der PnP Manager mit Hilfe der Ressource Manager der einbezogenen Knoten, ob ausreichend Verarbeitungsressourcen zur Verfügung stehen.
6. Bei erfolgreicher Prüfung wird der Komponentengraph in entsprechende Teilgraphen unterteilt, die an die jeweiligen PnP Clients der Knoten gesendet werden. Bei Bedarf werden von diesen noch nicht ausgeführte Komponenten gestartet. Dabei können aufgerufene Proxy-Komponenten die jeweiligen Komponenten im proprietären Laufzeitsystem auf dem entsprechenden Knoten laden (und natürlich auch beenden).
7. Nach erfolgreicher Konfiguration schicken die PnP Clients eine Bestätigung an den PnP Manager.
8. Der PnP Manager aktiviert schließlich die Anwendung, welche sowohl dezentral ausgeführt werden kann, indem der Datenaustausch zwischen Komponenten automatisch den gewünschten Ablauf erzeugt, als auch zentral über eine Anwendung, die direkt in einer bestimmten Abfolge Befehle an Komponenten kommandiert.

2.4. Anschluss neuer Hardwaremodule mit proprietären Treibern

Einige Hardwaremodule, wie beispielsweise Anbaugeräte für den Roboterarm oder Sensoren, die direkt auf der Serviceroboterplattform angeschlossen werden, besitzen bereits auf dieser Plattform entsprechende Treiber. Diese können sowohl aktive Implementierungen der Detect Funktion besitzen (z.B. Anbaugeräte für den Arm), die periodisch das Vorhandensein entsprechender Komponenten abfragen, als auch passive Detect Funktionen (z.B. RGB-D Kameras, Laserscanner über USB), welche durch Anschluss des Gerätes über das Betriebssystem initiiert werden. Um diese Sensorik auch CHROMOSOME zugänglich machen zu können, werden diese Treiber folgendermaßen erweitert.

Im Falle der **aktiven Detect Funktion** muss der aktiv abfragende Treiber grundsätzlich in der statischen Anwendungskonfiguration vorgesehen und geladen werden. Dies geschieht, indem CHROMOSOME zu jeder Anwendung, die z.B. den Serviceroboter Care-O-bot 3 benutzt, eine spezielle Gerätekomponente mit einem Basispaket an Treibern und Komponenten für diesen Roboter startet. Dazu genügen die Einrichtung eines ROS Masterknotens sowie die Ausführung eines Ladeskriptes für die Grundausstattung an Low-level Komponenten und Treibern. Unter diesen Treibern ist beispielsweise der Treiber für Anbaugeräte am Arm, die über eine standardisierte Hardwareschnittstelle angeschlossen werden (siehe Abbildung 4). Elektrisch sind diese Anbaugeräte über einen CAN-Bus verbunden, der u.a. die Übermittlung von Steuerkommandos sowie einer eindeutigen Geräte-ID zulässt. Der Treiber fragt nun zyklisch den CAN-Bus nach vorhandenen Geräten ab und wertet im Erfolgsfall die gesendeten IDs aus. Entsprechend einer empfangenen ID wird nun der spezielle proprietäre Gerätetreiber geladen, der ein abstraktes Interface auf Geräte dieser Klasse nach außen zur Verfügung stellt (Hardware Abstraction). Der Treiber wird bereits zu diesem Zeitpunkt auf dem entsprechenden Knoten geladen und nicht erst während des Deployments durch die CHROMOSOME PnP Clients ganz am Ende, weil nur dort ein Start sinnvoll ist, wo auch die reale Hardware angeschlossen ist. Da die Kommunikationsschnittstellen noch im proprietären Format (hier: ROS) vorliegen, wird nun der PnP Client des lokalen CHROMOSOME Knotens über die Funktion reportModule aufgerufen, um das neue Gerät in CHROMOSOME bekannt zu machen. Wie im Abschnitt 5.1.2 des Dokuments „Technische Umsetzung des Plug&Play-Konzepts auf Basis von CHROMOSOME im Projekt AutoPnP“ beschrieben, werden dabei Informationen wie Modultyp und Informationen über die Anbindung des neuen Moduls übertragen. Nach Vergabe der Komponentenkennung werden Modultyp und Manifest der Komponente an den PnP Manager weitergegeben, welcher schlussendlich in Zusammenspiel mit der Intelligenten Steuerung bei Bedarf das Deployment umsetzt. Unter Deployment ist in diesem Fall die Einrichtung und Verschaltung eines Proxies zu verstehen, der die vom proprietären Treiber bereitgestellten Schnittstellen auf der einen Seite ansprechen kann und diese in das Kommunikationsformat von CHROMOSOME übersetzt. Durch die Verwendung des Topic-Konzeptes in ROS und in CHROMOSOME ist eine Übersetzung der Kommunikation relativ einfach zu ermöglichen.

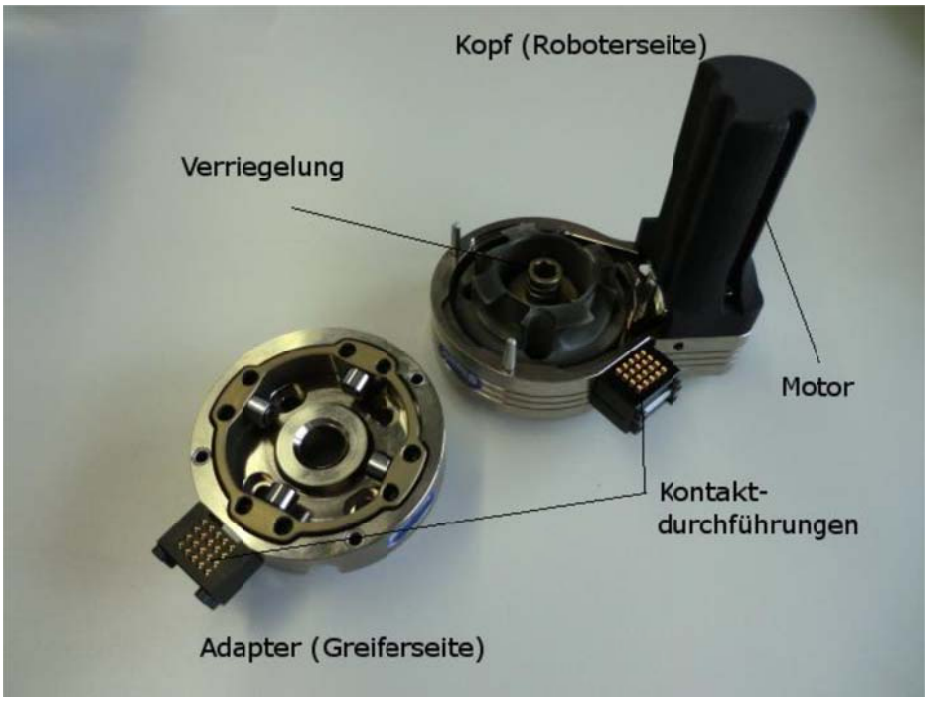

Abbildung 4: Werkzeugwechsler als einheitliche Schnittstelle für Anbaugeräte am Roboterarm

Grundsätzlich verläuft die Anbindung von Geräten mit **passiver Detect Funktion** sehr ähnlich. Allerdings muss anfänglich kein Treiber gestartet werden, da der Anschluss des Gerätes im Betriebssystem ein Ereignis auslöst, was im Falle von Linux z.B. durch eine udev-Regel dazu genutzt wird, den jeweiligen proprietären Treiber zu starten. Der Treiber wird wiederum auf demselben Knoten direkt gestartet, an dem auch das Gerät angeschlossen ist. Ebenso stellt der proprietäre Treiber wieder eine abstrakte Schnittstelle für Geräte dieser Klasse im proprietären Kommunikationsformat bereit. Bei Starten des Treibers wird deswegen wieder die Funktion reportModule des lokalen PnP Clients aufgerufen, um das neue Modul in CHROMOSOME anzumelden. Die dann folgenden Schritte sind identisch zu den obigen Beschreibungen für Treiber mit aktiver Detect Funktion und die üblichen Vorgängen in CHROMOSOME folgen.

Durch diese Vorgehensweise der Anbindung proprietärer Gerätetreiber sind die Schnittstellen sowohl für Komponenten im proprietären Format ansprechbar, als auch durch native CHROMOSOME-Komponenten. Dieser Vorteil zieht natürlich den Nachteil mit sich, dass CHROMOSOME nicht notwendigerweise einen garantierten Exklusivzugriff auf die Module besitzt und dadurch gewisse Echtzeitgarantien nicht mehr zuverlässig vergeben kann. Dieser Nachteil ist im Anwendungsbereich der Servicerobotik jedoch größtenteils unproblematisch, da harte Echtzeitanforderungen nur partiell in abgeschlossenen low-level Subsystemen (z.B. Robotersteuerung) benötigt werden. Des Weiteren ist eine Lösung des Problems auch durch die Bereitstellung exklusiver Zugriffsrechte auf den Treiber sowie der Verwendung eines Echtzeitbetriebssystems denkbar.

2.5. Nachladen von proprietären Softwarekomponenten

Die Intelligente Steuerung bietet die Möglichkeit, fehlende Komponenten bei Bedarf aus einem Komponentenrepository zu beziehen und auf dem System zu installieren. Dieser Vorgang funktioniert sowohl für native CHROMOSOME Komponenten, als auch für proprietäre Komponenten, die in CHROMOSOME über einen Proxy angebunden werden. Im letzteren Falle muss die Intelligente Steuerung den CHROMOSOME-Proxy installieren und die proprietäre Komponente. Die Installation von ROS-Komponenten im Release Status mit allen benötigten Abhängigkeiten ist über das Softwaremanagement des Betriebssystems einfach möglich.

Während des Deployments müssen dann üblicherweise neu installierte Komponenten gestartet werden, sofern sie in der generierten Anwendung Verwendung finden. Die Intelligente Steuerung übermittelt den anwendungsspezifischen Komponenten- und Kommunikationsgraphen an den PnP Manager. Ob und auf welchem Knoten eine Komponente gestartet werden muss, entscheidet entweder die Intelligente Steuerung oder der PnP Manager. Nach Aufteilung des Graphen auf die vorhandenen Knoten, werden die jeweiligen PnP Clients über die zu instanziierenden Teilgraphen informiert. Noch nicht gestartete Komponenten lädt nun der jeweilige PnP Client nach. Zuerst instanziiert er den komponentenspezifischen Proxy. Dieser wiederum kennt dann den passenden Aufruf zum Start der proprietären Komponente. In ROS besteht der Start dieser Komponente im Ausführen eines Startskriptes.

2.6. Ablauf der szenario-spezifischen Plug&Play-Ereignisse

Im Folgenden sollen die eingangs vorgestellten Demonstrationsteilszenarien hinsichtlich der internen Plug&Play-Abläufe detailliert werden. Die technische Beschreibung der Plug&Play-Vorgänge orientiert sich dabei eng an die für den allgemeinen Fall besprochenen Vorgänge in Abschnitt 2.3. Um bei einer öffentlichen Demonstration Interessierten auch die hier beschriebenen internen Vorgänge (fehlende, angeschlossene Geräte, verwendete Softwarekomponenten und deren Verschaltung) verdeutlichen zu können, wird ein entsprechendes Visualisierungswerkzeug implementiert.

2.6.1. Starten des Fußbodenreinigungsszenarios ohne angeschlossenes Reinigungsgerät

In diesem Beispiel wird die Umsetzung von Hardware-Plug&Play demonstriert. Die Nummerierung der nachfolgend beschriebenen technischen Vorgänge entspricht dabei jener aus Abschnitt 2.3. In Abbildung 5 sind die darstellbaren Vorgänge während der Programmzusammenstellung durch die Intelligente Steuerung (Schritt 3) sowie die nachfolgende Umsetzung in CHROMOSOME (Schritt 5-8) visualisiert.

1. Der Nutzer fordert bei der Intelligenten Steuerung die Generierung einer Anwendung zur Fußbodenreinigung an.

2. Die Intelligente Steuerung fordert beim PnP Manager über die Funktion generateGraph den Komponenten- und Kommunikationsgraphen an. Dieser enthält im momentanen Systemzustand die notwendigen Funktionalitäten detect_dirt_visually und vacuum_clean_room_focused, da diese Komponenten bereits früher installiert wurden.
3. Die Intelligente Steuerung versucht nun mit den im System bzw. im Komponentenrepository (z.B. online Komponentendatenbank) vorhandenen Komponenten ein Programm zu generieren, welches das gewünschte Ziel erreichen kann.
    a. Der Reinigungsvorgang wird durch den Ausführungswunsch der Komponente vacuum_clean_room_focused vom Benutzer spezifiziert.
    b. Über den Fähigkeitenidentifizierer aus dem Typmanifest der Komponente vacuum_clean_room_focused liest die Intelligente Steuerung die Anforderungen und Abhängigkeiten aus der Fähigkeitendatenbank.
    c. Die zur Zielerreichung geeignete Komponente vacuum_clean_room_focused ist bereits auf dem System installiert.
    d. Unter den Anforderungen und Abhängigkeiten dieser Komponente findet sich die Anforderung, dass mit einem Erzeuger einer Karte der Verschmutzungen verschaltet werden muss. Im aktuellen Komponentengraphen wird dafür bereits die Komponente detect_dirt_visually vorgeschlagen. Weiterhin stellt die Intelligente Steuerung fest, dass eine Hardwareabhängigkeit zu einem Reinigungsgerät (Staubsauger) besteht. Ein entsprechendes Hardwaremodul ist jedoch im aktuellen Komponentengraphen nicht zu finden. Die Prüfung der weiteren Abhängigkeiten aller relevanten Komponenten wirft keine weiteren Unzulänglichkeiten auf.
    e. Es sind nicht alle notwendigen Module bereits im System vorhanden, daher ist die Arbeit der Intelligenten Steuerung noch nicht beendet.
    f. Es müssen keine weiteren Softwarekomponenten aus dem Komponentenrepository installiert werden.
    g. Es wurde ein fehlendes Hardwaremodul (Reinigungsgerät) festgestellt, ohne welches die Anwendung nicht lauffähig ist. Der Benutzer wird über den Mangel informiert und die Optionen zur Behebung des Mangels bzw. zum Abbruch der Planung unterbreitet. Der Benutzer schließt daraufhin ein Reinigungsgerät an den Roboterarm an. Nun findet intern der im Abschnitt 2.4 beschriebene Plug-Vorgang mit aktiver Detect Funktion statt. Im Detail sind das:
        i. Der periodisch abfragende Treiber (ROS-Implementierung) erhält über den CAN-Bus vom Reinigungsgerät eine Geräte-ID.
        ii. Der Treiber assoziiert die Geräte-ID mit dem korrekten proprietären ROS-Treiber für das Reinigungsgerät und startet diesen.
        iii. Der PnP Client des lokalen CHROMOSOME Knotens wird über die Funktion reportModule über die neue Hardware informiert.
        iv. Der PnP Client vergibt eine Komponentenkennung für das neue Gerät.

v. Modultyp und Manifest der Komponente werden an den PnP Manager weitergegeben. Dieser berechnet einen aktualisierten Komponenten- und Kommunikationsgraphen und leitet diesen an die Intelligente Steuerung weiter.
vi. Die auf ein Anschließen des Reinigungsgerätes wartende Intelligente Steuerung nutzt den neuen Graphen um mit dem bisherigen Ziel (Reinigungsapplikation) den Ablaufpunkt 3 nochmals durchzuarbeiten. Die Beschreibung ist identisch zu obigen Ausführungen, nur mit dem Unterschied, dass nunmehr keinerlei fehlende Komponenten und Module entdeckt werden und die Intelligente Steuerung in Schritt e) terminiert.

4. Die Intelligente Steuerung hat damit eine zielführende Anwendung erstellen können. Diese wird zusammen mit dem anwendungsspezifischen Komponenten- und Kommunikationsgraphen zurück an den PnP Manager gemeldet.
5. Der PnP Manager veranlasst beim Network Configuration Calculator die Prüfung der neuen Konfiguration. Dabei werden die physikalischen Verbindungen sowie Marshaller und Demarshaller eingerichtet. Darüber hinaus prüft der PnP Manager mit Hilfe der Ressource Manager der einbezogenen Knoten, ob ausreichend Verarbeitungsressourcen zur Verfügung stehen. Da für die Servicerobotikanwendung keine harten Echtzeitanforderungen gestellt werden, genügt es, dass der PnP Manager die Komponenten gleichmäßig anhand ihrer geschätzten Berechnungskomplexität auf die 3 Knoten des Care-O-bot 3 verteilt.
6. Nach erfolgreicher Prüfung wird der Komponentengraph in entsprechende Teilgraphen unterteilt, die an die jeweiligen PnP Clients der Knoten gesendet werden. Hierbei wird noch der Proxy für das Reinigungsgerät gestartet. Der Treiber in der ROS-Ebene läuft bereits und verbindet sich zum Proxy. Ggf. werden außerdem die noch nicht aktiven, relevanten Softwarekomponenten gestartet.
7. Nach erfolgreicher Konfiguration schicken die PnP Clients eine Bestätigung an den PnP Manager.
8. Der PnP Manager aktiviert schließlich die Reinigungsanwendung, welche zentral als eine Anwendung modelliert ist, die direkt in einer bestimmten Abfolge Befehle an die eingesetzten Komponenten detect_dirt_visually und vacuum_clean_room_focused kommandiert.

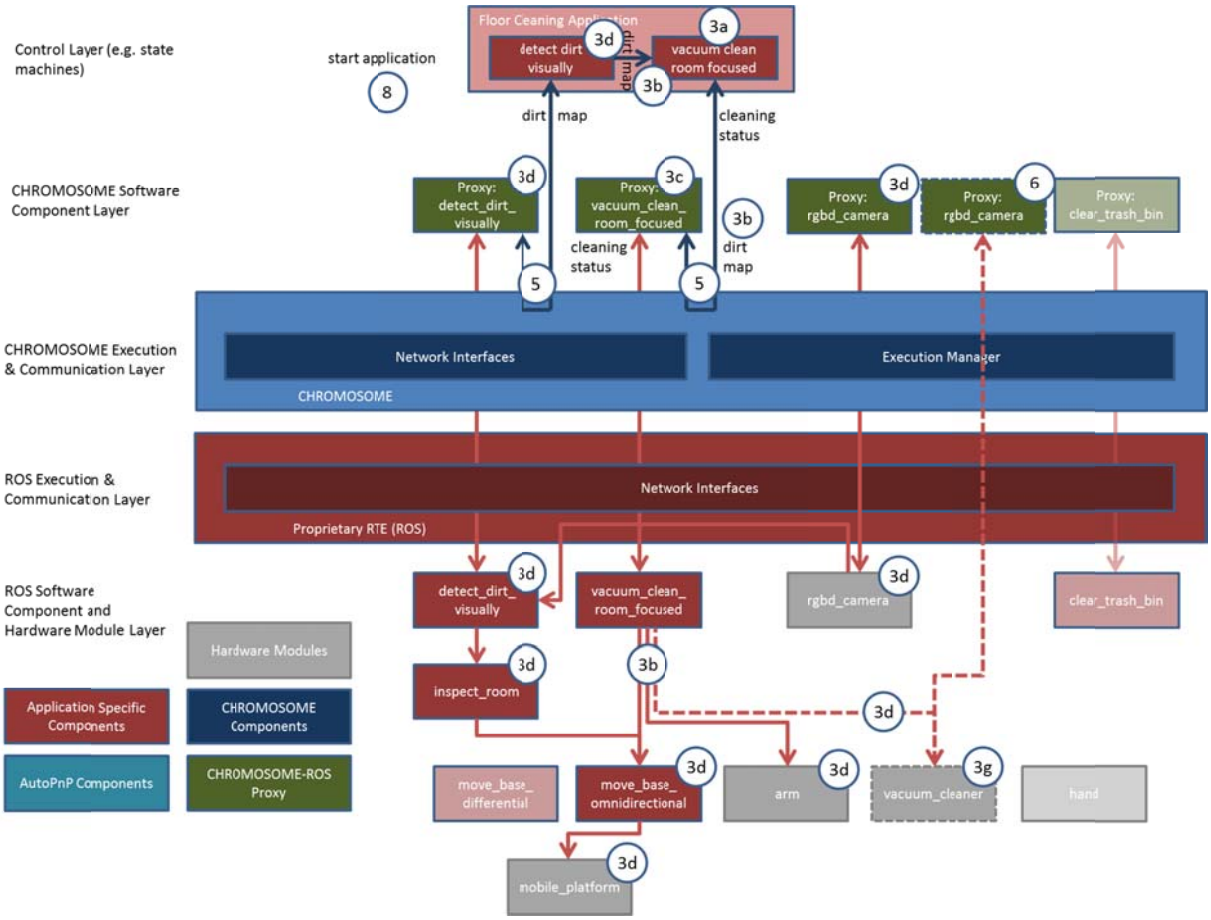

Abbildung 5: Programmgenerierung durch die Intelligente Steuerung und Umsetzung durch CHROMOSOME annotiert mit den Nummern der textuellen Beschreibung. Anmerkung: (i) nicht verwendete Komponenten sind ausgeblichen dargestellt, (ii) die Notierung des Schrittes 3d an ROS-Komponenten bezieht sich eigentlich auf deren hier nicht dargestellte CHROMOSOME-ROS-Proxies (die ROS-Komponenten selbst sind schließlich nicht für CHROMOSOME und die Intelligente Steuerung sichtbar).

2.6.2. Starten des Fußbodenreinigungsszenarios ohne angeschlossene RGB-D Kamera

Auch dieses Beispiel widmet sich dem Hardware-Plug&Play. Der Unterschied zum vorherigen Beispiel besteht in der Implementierung der Detect Funktion, welche in diesem Falle passiver Natur ist. Da der größte Teil der vorherigen Beschreibung hier identisch aussieht, soll nur kurz auf die unterschiedlichen Punkte eingegangen werden.

3. Die Intelligente Steuerung versucht nun mit den im System bzw. im Komponentenrepository (z.B. online Komponentendatenbank) vorhandenen Komponenten ein Programm zu generieren, welches das gewünschte Ziel erreichen kann.
    d. Unter den Anforderungen und Abhängigkeiten der Komponente vacuum_clean_room_focused findet sich die Anforderung, dass mit einem

Erzeuger einer Karte der Verschmutzungen verschaltet werden muss. Im aktuellen Komponentengraphen wird dafür bereits die Komponente detect_dirt_visually vorgeschlagen. Weiterhin stellt die Intelligente Steuerung fest, dass bei dieser eine Hardwareabhängigkeit zu einer RGB-D Kamera besteht. Ein entsprechendes Hardwaremodul ist jedoch im aktuellen Komponentengraphen nicht zu finden. Auch im Komponentenrepository lassen sich keine alternativen Softwarekomponenten finden, die mit der aktuell installierten Hardware lauffähig wären. Die Prüfung der weiteren Abhängigkeiten aller relevanten Komponenten wirft keine weiteren Unzulänglichkeiten auf.

…

g. Es wurde ein fehlendes Hardwaremodul (RGB-D Kamera) festgestellt, ohne welches die Anwendung nicht lauffähig ist. Der Benutzer wird über den Mangel informiert und die Optionen zur Behebung des Mangels bzw. zum Abbruch der Planung unterbreitet. Der Benutzer schließt daraufhin eine RGB-D Kamera an einem freien USB-Port an einem der Steuerrechner an. Nun findet intern der im Abschnitt 2.4 beschriebene Plug-Vorgang mit passiver Detect Funktion statt. Im Detail sind das:
    i. Das Auslösen einer zur Hersteller- und Geräte-ID passenden udev Regel durch das Betriebssystem.
    ii. Die Regel startet automatisch den passenden proprietären ROS-Treiber zu den festgestellten Hersteller- und Geräte-IDs, in diesem Falle also einen RGBD-Kamera Treiber.
    iii. Der PnP Client des lokalen CHROMOSOME Knotens wird über die Funktion reportModule über die neue Hardware informiert.
    iv. Der PnP Client vergibt eine Komponentenkennung für das neue Gerät.
    v. Modultyp und Manifest der Komponente werden an den PnP Manager weitergegeben. Dieser berechnet einen aktualisierten Komponenten- und Kommunikationsgraphen und leitet diesen an die Intelligente Steuerung weiter.
    vi. Die auf ein Anschließen der RGB-D Kamera wartende Intelligente Steuerung nutzt den neuen Graphen um mit dem bisherigen Ziel (Reinigungsapplikation) den Ablaufpunkt 3 nochmals durchzuarbeiten. Die Beschreibung ist identisch zu obigen Ausführungen, nur mit dem Unterschied, dass nunmehr keinerlei fehlende Komponenten und Module entdeckt werden und die Intelligente Steuerung in Schritt e) terminiert.

2.6.3. Starten des Papierkorbleerungsszenarios ohne angeschlossene Hand

Das Hardware-Plug&Play ist in diesem Demonstrationsszenario absolut analog zu dem obibgen Fall in Abschnitt 2.6.1 (Starten des Fußbodenreinigungsszenarios ohne angeschlossenes Reinigungsgerät). Eine entsprechende Beschreibung der technischen Abläufe ergibt

sich somit direkt, wenn in der obigen Beschreibung die Komponenten detect_dirt_visually und vacuum_clean_room_focused durch detect_trash_bin und clear_trash_bin ersetzt werden, sowie die Zielsetzung der Intelligenten Steuerung durch Ausführung der Komponente clear_trash_bin spezifiziert wird und das Reinigungsgerät durch die Hand ersetzt wird. Auch in diesem Anwendungsfall wird die neue Hardware über eine aktive Detect Funktion erkannt.

2.6.4. Starten des Papierkorbleerungsszenarios ohne angeschlossene RGB-D Kamera

Diese Demonstration unterscheidet sich signifikant vom Fall aus Abschnitt 2.6.2 (Starten des Fußbodenreinigungsszenarios ohne angeschlossene RGB-D Kamera), indem nämlich nicht das Hardware-Plug&Play gezeigt wird, sondern eine automatische Umkonfiguration der Anwendung mit vorhandener Ersatzhardware stattfindet. Die technischen Abläufe gestalten sich daher folgendermaßen:

1. Der Nutzer fordert bei der Intelligenten Steuerung die Generierung einer Anwendung zur Papierkorbleerung an.
2. Die Intelligente Steuerung fordert beim PnP Manager über die Funktion generateGraph den Komponenten- und Kommunikationsgraphen an. Dieser enthält im momentanen Systemzustand die notwendigen Funktionalitäten detect_trash_bin und clear_trash_bin, da diese Komponenten bereits früher installiert wurden.
3. Die Intelligente Steuerung versucht nun mit den im System bzw. im Komponentenrepository (z.B. online Komponentendatenbank) vorhandenen Komponenten ein Programm zu generieren, welches das gewünschte Ziel erreichen kann.
    a. Der Leerungsvorgang wird durch den Ausführungswunsch der Komponente clear_trash_bin vom Benutzer spezifiziert.
    b. Über den Fähigkeitenidentifizierer aus dem Typmanifest der Komponente clear_trash_bin liest die Intelligente Steuerung die Anforderungen und Abhängigkeiten aus der Fähigkeitendatenbank.
    c. Die zur Zielerreichung geeignete Komponente clear_trash_bin ist bereits auf dem System installiert.
    d. Unter den Anforderungen und Abhängigkeiten der Komponente clear_trash_bin findet sich die Anforderung, dass mit einem Erzeuger von Koordinaten von Papierkörben verschaltet werden muss. Im aktuellen Komponentengraphen wird dafür bereits die Komponente detect_trash_bin vorgeschlagen. Weiterhin stellt die Intelligente Steuerung fest, dass eine Hardwareabhängigkeit zu einer Farbkamera besteht. Im Normalfall sind am System eine RGB-D Kamera sowie zwei Farbkameras angeschlossen. Dann wählt die Intelligente Steuerung die RGB-D Kamera, da ihr Typmanifest eine geringere Rechnerauslastung als bei den Farbkameras angibt. Wurde jedoch die RGB-D Kamera entfernt, so wählt die Intelligente Steuerung ersatzweise eine der vorhandenen Farbkameras aus. In beiden Hardwarekonfigurationen kann also die Intelligente Steuerung selbstständig die optimale Verschaltung

ohne Nutzereinwirkung bestimmen. Die Prüfung der weiteren Abhängigkeiten aller relevanten Module wirft keine Unzulänglichkeiten auf.

e. Es sind auch alle notwendigen Module bereits im System vorhanden, daher ist die Arbeit der Intelligenten Steuerung hiermit beendet.

4. Die Intelligente Steuerung hat damit eine zielführende Anwendung erstellen können. Diese wird zusammen mit dem anwendungsspezifischen Komponenten- und Kommunikationsgraphen zurück an den PnP Manager gemeldet.
5. Der PnP Manager veranlasst beim Network Configuration Calculator die Prüfung der neuen Konfiguration. Dabei werden die physikalischen Verbindungen sowie Marshaller und Demarshaller eingerichtet. Darüber hinaus prüft der PnP Manager mit Hilfe der Ressource Manager der einbezogenen Knoten, ob ausreichend Verarbeitungsressourcen zur Verfügung stehen. Da für die Servicerobotikanwendung keine harten Echtzeitanforderungen gestellt werden, genügt es, dass der PnP Manager die Komponenten gleichmäßig anhand ihrer geschätzten Berechnungskomplexität auf die 3 Knoten des Care-O-bot 3 verteilt.
6. Nach erfolgreicher Prüfung wird der Komponentengraph in entsprechende Teilgraphen unterteilt, die an die jeweiligen PnP Clients der Knoten gesendet werden. Ggf. werden außerdem die noch nicht aktiven, relevanten Softwarekomponenten gestartet.
7. Nach erfolgreicher Konfiguration schicken die PnP Clients eine Bestätigung an den PnP Manager.
8. Der PnP Manager aktiviert schließlich die Papierkorbleerungsanwendung, welche zentral als eine Anwendung modelliert ist, die direkt in einer bestimmten Abfolge Befehle an die eingesetzten Komponenten detect_trash_bin und clear_trash_bin kommandiert.

2.6.5. Nutzung der Navigation mit verminderter Anzahl an Laserscannern

Zur Demonstration von Plug&Play auf Softwarekomponentenebene eignet sich die Navigationsfähigkeit sehr gut, die mit unterschiedlichen Strategien implementiert sein kann. Größere Systeme haben oft die Möglichkeit, die mobile Plattform omnidirektional, also in beliebiger Weise verfahren zu können, z.B. auch seitlich. Damit solch ein Vorgang sicher ist, muss Sensorik Hindernisse in allen Bewegungsrichtungen wahrnehmen können. Kann dies nicht sichergestellt werden, weil z.B. nur frontal ein Laserscanner zur Verfügung steht, so sollte besser auf eine simplere Navigation zurückgegriffen werden, z.B. eine differentielle, die den Roboter hauptsächlich in die Vorwärtsrichtung kommandiert. Sind alle Algorithmen der Art Navigation in Komponenten mit gleichen Schnittstellen untergebracht, so können diese je nach Roboterausstattung problemlos von der Intelligenten Steuerung oder dem Anwendungsentwickler gegeneinander ausgetauscht werden. Folgende technische Abläufe stellen die Nutzung der passenden Softwarekomponente unter Nutzung der Intelligenten Steuerung dar. Als Beispielanwendung kann die Fußbodenreinigung dienen.

1. Der Nutzer fordert bei der Intelligenten Steuerung die Generierung einer Anwendung zur Fußbodenreinigung an.

2. Die Intelligente Steuerung fordert beim PnP Manager über die Funktion generateGraph den Komponenten- und Kommunikationsgraphen an. Dieser enthält im momentanen Systemzustand die notwendigen Funktionalitäten detect_dirt_visually und vacuum_clean_room_focused, da diese Komponenten bereits früher installiert wurden.
3. Die Intelligente Steuerung versucht nun mit den im System bzw. im Komponentenrepository (z.B. online Komponentendatenbank) vorhandenen Komponenten ein Programm zu generieren, welches das gewünschte Ziel erreichen kann.
    a. Der Reinigungsvorgang wird durch den Ausführungswunsch der Komponente vacuum_clean_room_focused vom Benutzer spezifiziert.
    b. Über den Fähigkeitenidentifizierer aus dem Typmanifest der Komponente vacuum_clean_room_focused liest die Intelligente Steuerung die Anforderungen und Abhängigkeiten aus der Fähigkeitendatenbank.
    c. Die zur Zielerreichung geeignete Komponente vacuum_clean_room_focused ist bereits auf dem System installiert.
    d. Unter den Anforderungen und Abhängigkeiten dieser Komponente findet sich die Anforderung, dass mit einem Erzeuger einer Karte der Verschmutzungen verschaltet werden muss. Im aktuellen Komponentengraphen wird dafür bereits die Komponente detect_dirt_visually vorgeschlagen. Weiterhin stellt die Intelligente Steuerung fest, dass beide Komponenten eine Navigationskomponente benötigen (detect_dirt_visually indirekt über die Inspektionskomponente). Als mögliche Kandidaten stehen laut Komponentengraphen die Komponenten move_base_omnidirectional und move_base_differential zur Verfügung. Im Normalfall stehen am Roboter 3 Laserscanner zur Verfügung, die alle Richtungen abdecken; somit wird sich für die erste Variante entschieden, da sie eine größere Übereinstimmung mit den Fähigkeiten der vorhandenen Hardware hat. Steht dagegen nur noch ein Laserscanner an der Vorderseite zur Verfügung, so ist die Anforderung von move_base_omnidirectional nach 360° Sensorabdeckung verletzt. Die Intelligente Steuerung wählt nun automatisch die nächstbeste Methode, die mit der vorhandenen Hardware eine adäquate Zielerreichung ermöglicht, welche durch move_base_differential gegeben ist. Die Schnittstellen zu andere Komponenten sowie zu Anwendungen sind identisch, daher ändert sich an der übrigen Programmverschaltung nichts. Die Prüfung der weiteren Abhängigkeiten aller relevanten Module wirft keine Unzulänglichkeiten auf.
    e. Es sind alle notwendigen Module bereits im System vorhanden, daher ist die Arbeit der Intelligenten Steuerung hiermit beendet.
4. Die Intelligente Steuerung hat damit eine zielführende Anwendung erstellen können. Diese wird zusammen mit dem anwendungsspezifischen Komponenten- und Kommunikationsgraphen zurück an den PnP Manager gemeldet.
5. Der PnP Manager veranlasst beim Network Configuration Calculator die Prüfung der neuen Konfiguration. Dabei werden die physikalischen Verbindungen sowie Marshaller und Demarshaller eingerichtet. Darüber hinaus prüft der PnP Manager mit

Hilfe der Ressource Manager der einbezogenen Knoten, ob ausreichend Verarbeitungsressourcen zur Verfügung stehen. Da für die Servicerobotikanwendung keine harten Echtzeitanforderungen gestellt werden, genügt es, dass der PnP Manager die Komponenten gleichmäßig anhand ihrer geschätzten Berechnungskomplexität auf die 3 Knoten des Care-O-bot 3 verteilt.

6. Nach erfolgreicher Prüfung wird der Komponentengraph in entsprechende Teilgraphen unterteilt, die an die jeweiligen PnP Clients der Knoten gesendet werden. Ggf. werden außerdem die noch nicht aktiven, relevanten Softwarekomponenten gestartet.
7. Nach erfolgreicher Konfiguration schicken die PnP Clients eine Bestätigung an den PnP Manager.
8. Der PnP Manager aktiviert schließlich die Reinigungsanwendung, welche zentral als eine Anwendung modelliert ist, die direkt in einer bestimmten Abfolge Befehle an die eingesetzten Komponenten detect_dirt_visually und vacuum_clean_room_focused kommandiert.