

# **Künstliche Intelligenz im Weltall: Der Remote Agent**

Ulrich Tschaffon

Proseminar Künstliche Intelligenz  
Universität Ulm

03.06.04

## **Zusammenfassung**

Dieses Dokument liefert eine Beschreibung des Remote Agent, einer intelligenten Steuerungssoftware eines Raumgefährtes, welche 1998 von der NASA auf der Deep Space 1 Mission zum ersten mal erfolgreich getestet wurde. Es wird im einzelnen auf die Notwendigkeit des Remote Agent in der heutigen Weltraumforschung hingewiesen und die Architektur, generelle Konzepte der Realisation des Systems, sowie die Funktionsweisen der einzelnen Komponenten wie der Planer, die Ausführungseinheit und die Zustandsidentifikations- und rekonfigurationskomponente beschrieben, um ein grobes Verständnis der Arbeitsweise des Remote Agent zu vermitteln.

# 1. Einleitung

Der Remote Agent (kurz: der Remote Agent) [1] ist ein intelligentes Steuerungsprogramm eines unbemannten Raumgefährtes, welches eigenständig und ohne ständige Überwachung durch den Menschen festgelegte Missionsziele in vorgegebener Zeit erreichen und dabei auch auf unvorhergesehene Ereignisse schnell und zuverlässig reagieren kann.

Er soll vor allem da zum Einsatz kommen, wo eine schnelle Reaktion auf sich rasch ändernde äußere Einflüsse von Nöten ist und eine bemannte Mission zu gefährlich bzw. unmöglich ist. Denn durch das heutige Verfahren, dem expliziten Anweisen des Raumgefährtes durch Befehle und die damit verbundene Verzögerung durch die Distanz bzw. Störung, ist es zum Beispiel unmöglich, ein sich schnell bewegendes Gefährt um ein zu spät erkanntes Hindernis zu manövrieren oder ein Gefährt zu steuern, welches sich auf der Schattenseite eines Planeten befindet. So hat die NASA zum Beispiel Pläne ein Flugzeug auf dem Mars fliegen zu lassen. Dieses Flugzeug direkt zu steuern ist aufgrund Zeitverzögerung, der dortigen sehr dünnen Atmosphäre und der sich häufig ändernden Einflüsse sehr schwierig.

Das Hauptargument jedoch (für die NASA), welches für den Remote Agent spricht sind die viel geringeren Kosten der Missionen. Bei einer normalen, direkt gesteuerten Weltraummission sind bis zu 300 Menschen am Boden ständig beschäftigt, die erhaltenen Daten auszuwerten, auftretende Probleme zu beseitigen, Flugbahnen zu berechnen, usw. Um nun ein von einem Remote Agent gesteuertes Weltraumgefährt zu betreuen, braucht es nur einen Bruchteil des üblichen Bodenteams, was es den Forschern erlauben würde, viel mehr solcher Operationen gleichzeitig zu betreiben, was wiederum zu mehr Informationen führen würde.

## 1.1. Was wird von Remote Agent verlangt

Ein von einem Remote Agent gesteuertes Raumschiff soll dazu in der Lage sein, langfristige Missionen zuverlässig und selbstständig abzuschließen. Es soll auch in einer unbekanntem Umgebung unter unerwarteten Umständen dazu im Stande sein, die Mission erfolgreich zu beenden. Auftretende Fehler sollen, soweit möglich, selbstständig behoben werden können. Auch soll der Remote Agent mehrere Aktivitäten gleichzeitig ausüben und sie untereinander koordinieren können.

Kurz: Er soll eine menschliche Crew ersetzen bzw. sie später sogar übertreffen.

## 1.2. Wichtige Prinzipien zur Umsetzung des Remote Agent

Die Architektur des Remote Agent unterscheidet sich in drei Eigenschaften von „herkömmlichen“ künstlich intelligenten Steuerungsprogrammen. Dies ist zum einen das *modellbasierte Programmieren*, welches beschreibende Modelle als Vorlage für die verschiedenen Programmfunktionsweisen verwendet, die *Echtzeitberechnung von Lösungen*, welche das autonome Reagieren auf unvorhergesehene Fehler ermöglicht und die *zielorientierte, wiederholte Ausführung*, welche eine zuverlässige Befehlsausführung gewährleisten soll.

### 1.2.1. Modellbasiertes Programmieren

Um für ein neues Projekt nicht immer eine neue Software schreiben zu müssen, was erhebliche Kosten und Mühen verursacht, wurde beim Remote Agent die sogenannte modellbasierte Programmierung angewandt. So basieren alle Programme der einzelnen Hardwarekomponenten auf vorher ausgearbeiteten Modellen, welche die Funktionsweise des Programms bereits klar festlegen. Auch

wenn man aufgrund der unterschiedlichen Hardware nicht verhindern kann jedes mal ein neues hardwarespezifische Programm zu schreiben, so kann man die in den Modellen grundsätzliche Arbeitsweise oft wiederverwenden. Da diese Modelle relativ einfach zu verstehen sind, können verschiedene Mitarbeiter anhand dieses Modells nachvollziehen, was ein Programm macht bzw. welche Übergabeparameter es benötigt ohne das Programm selber wirklich zu verstehen. Auf diese Weise lassen sich Missverständnisse und die daraus resultierenden Fehler relativ früh erkennen und beseitigen.

Es benutzen auch Teile der Software des Remote Agent diese kodierten Modelle um zum Beispiel Fehler zu finden, wie das Zustandsidentifikations- und -rekonfigurationskomponente. Denn anhand dieser Modelle erkennt das Modul die Zusammenhänge verschiedener Komponenten und kann so Lösungen für eventuelle Probleme finden.

### **1.2.2. Echtzeitberechnung von Lösungen**

Da der Remote Agent auch völlig autonom mit unvorhergesehenen Problemen fertig werden soll, muss er beim Eintreten von Ereignissen die Lösung in Echtzeit berechnen. Dies bringt natürlich die Schwierigkeit mit sich, die Berechnung in einem angemessenen Zeitrahmen bzw. sie überhaupt (bei nicht zu lösenden Problemen) zu beenden.

### **1.2.3. Zielorientierte, wiederholte Ausführung**

Herkömmliche Raumgefährte werden durch direkte Befehle gesteuert, die dann zu einem festgelegten Zeitpunkt ausgeführt werden. Diese Art von Befehlsausführung ist nicht besonders flexibel gegenüber auftretende Fehler. Um die Zuverlässigkeit dieser Befehlsausführung zu erhöhen, wurde dazu übergegangen nur noch einen zu erreichenden Wert anzugeben, welcher in einer Schleife mit dem aktuellen Wert verglichen wird und bei einer zu großen Abweichung korrigiert wird (*feedback control*).

Der Remote Agent geht noch ein wenig weiter. Es wird nur ein Ziel und ein Zeitrahmen festgelegt und es wird ein Plan aufgestellt um dieses Ziel zu erreichen. Die Einhaltung des Planes wird ununterbrochen überwacht indem der aktuelle Status des Raumgefährtes mit dem zu erreichenden verglichen wird. Bei einer Abweichung über den Toleranzbereich hinaus wird sofort eine Korrektur vorgenommen.

## **2. Architektur des Remote Agent**

Um einen Remote Agent zu erhalten, welcher autonome Operationen mit festen Ressourcenbeschränkungen in einem festen Zeitrahmen bewerkstelligen kann, brauchen wir einen zeitlichen *Planer* mit einem zugehörigen *Missions Manager*, welche die vorhandenen Ressourcen verwalten und Pläne zum Erreichen der vorgegebenen Ziele erstellen. Um diese erstellten Pläne zuverlässig und in der vorgegebenen Zeit auszuführen ist die *Ausführungseinheit* nötig. Die modellbasierte *Zustandsidentifikations- und -rekonfigurationskomponente* dient zur Fehlererkennung und schneller Fehlerbehandlung der Hardwarekomponenten. Die Architektur des Remote Agent und dessen Einbindung in die Steuerungssoftware des Raumgefährtes zeigt die Abbildung 1.

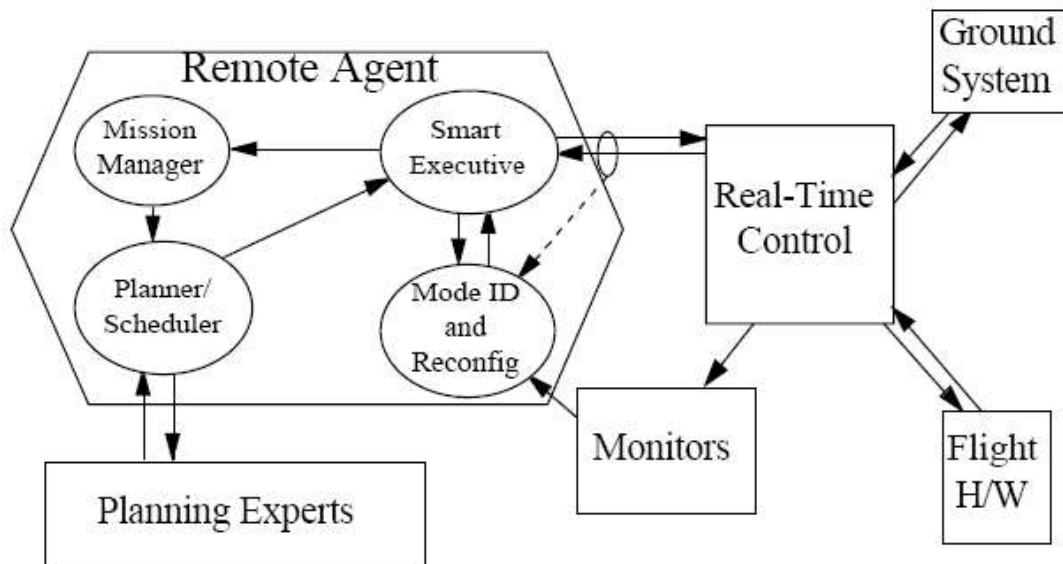


Abbildung 1: Die Architektur des Remote Agent innerhalb der Steuerungssoftware

## 2.1. Planer und Missions Manager

Der Mission Manager verwaltet die langfristigen Ziele, die Mission des Raumschiffes, jeweils unterteilt in mehrere Abschnitte. Diese Ziele werden anfangs vom Bodenpersonal festgelegt und können auch später von der Erde aus nach Belieben geändert bzw. erweitert werden. Auf Anfrage der Ausführungseinheit formuliert der Missions Manager ein zeitlich beschränktes Planproblem, welches an den Planer übergeben wird. Dieser erstellt aufgrund dieses Planproblems mittels einer Backtrackingmethode einen flexiblen Plan, welchen er an die Ausführungseinheit übergibt. Der Plan legt nur die Aktivität der beteiligten Komponenten im Verlaufe des Plans fest, so dass Details der Ausführungseinheit überlassen werden können, was die Flexibilität des Plans erheblich erhöht. Außerdem legt der Plan noch fest, wann der nächste Plan vom Planer erstellt werden soll, damit nach der Abarbeitung des Planes durch die Ausführungseinheit bereits der nächste bereit steht.

Andere On-Board Softwaresysteme, sog. Expertensysteme beteiligen sich ebenfalls unterstützend am Planungsprozess, doch sie beschränken sich nur auf gewisse Spezialgebiete in welchen komplizierte Rechnungen nötig sind (z.B.: Benzinverbrauch).

## 2.2. Die Ausführungseinheit

Die Ausführungseinheit übersetzt den erhaltenen Plan in Befehle des Echtzeitsteuerungssystems und führt ihn unter den angegebenen Beschränkungen aus. Dadurch, dass der Planer meist nur „grobe Richtlinien“ zum Ausführen eines Planes übergibt, hat die Ausführungseinheit die Möglichkeit verschiedene Wege zur Erfüllung eines Zieltes zu gehen. Außerdem überprüft sie die Durchführung der eigenen Befehle durch Bestätigung des Befehlsempfänger direkt oder durch Bestätigung von der Zustandsidentifikations- und rekonfigurationskomponente. So kann die Ausführungseinheit zum Beispiel überprüfen, ob der Befehl eine Kamera anzuschalten auch wirklich durchgeführt wird, indem es von der Zustandsidentifikations- und rekonfigurationskomponente den Status der Kamera abfragt.

Sollte die Ausführungseinheit beim Ausführen des Planes auf Probleme stoßen, so kann sie es mit

einem anderen Lösungsansatz versuchen oder die Zustandsrekonfigurationskomponente aufrufen, um dieses Problem zu lösen.

Wenn sie vom aktuellen Plan instruiert wird einen neuen Plan anzufragen, wird der aktuelle Zustand des Raumgefährtes an den Missions Manager übergeben und auf den neuen Plan gewartet.

Sollte es unmöglich sein, den Plan auszuführen, bricht sie den Plan ab, stoppt sämtliche Aktivitäten, versetzt das System in einen sicheren und stabilen Zustand und übergibt den momentanen Zustand an den Missions Manager mit der Anfrage nach einem neuen Plan.

## 2.3. Zustandsidentifikations- und rekonfigurationskomponente

Die Zustandsidentifikations- und rekonfigurationskomponente benutzt ein einziges, beschreibendes Modell des Raumschiffes, welches die Funktionsweisen und Zusammenhänge der einzelnen Komponenten umfasst. Die Identifikationskomponente macht die wahrscheinlichste Raumschiffskonfiguration aus, indem es die aktuellen Monitoraten (Monitore überwachen die Zustände einzelner Komponenten) und die Befehle an das Echtzeitsteuerungssystem mit dem zugehörigen Status vergleicht und jede Änderung der Ausführungskomponente meldet.

Die Rekonfigurationskomponente benutzt das Modell des Raumgefährtes um die günstigste Methode zu finden, die gewünschte Funktionalität durch Hardwarerekonfiguration oder durch Reparieren der fehlerhaften Komponente wieder herzustellen. Beim erfolgreichen Beheben des Fehlers wird die Ausführungseinheit von der Rekonfigurationskomponente angewiesen, den zuvor nicht ausführbaren Befehl zu wiederholen.

## 3. Planer und Missions Manager

Da es sehr schwierig ist einen Plan über einen größeren Zeitraum bis ins kleinste Detail zu planen ohne irgendwelche Details zu übersehen, versucht der Remote Agent periodisch zu planen, also einen ausgedehnten Schritt in mehrere kleinere Planungsschritte zu unterteilen, ein jeder mit seinen eigenen Beschränkungen. Dies kann jedoch wieder zu Problemen führen, da sich z.B. ein Fehler im ersten Schritt auf einen späteren Schritt auswirken kann. Wird beispielsweise am Anfang zu viel Brennstoff fürs Beschleunigen verbraucht, so fehlt dieser Brennstoff später für das Bremsen. Hier kommt der Missions Manager ins Spiel. Dieser legt dem Planer Beschränkungen so auf, dass er nur Pläne generieren kann, die den Gesamtverlauf der Mission nicht gefährden.

Der Planer bekommt also vom Missions Manager die Bedingungen und die Ziele, die im zu konstruierenden Plan eingehalten werden müssen, sowie den aktuellen Status des Raumgefährtes von der Ausführungseinheit und erstellt daraus einen Plan, welcher ausgeführt werden muss, um diese Ziele zu erreichen.

Um einen Plan zu konstruieren wird eine sogenannte *planning engine* für jedes zu lösende Problem neu mit dem Modell, der Beschreibung und Vorgehensweisen der betreffenden Anwendung konfiguriert und ein Plan mittels *backtracking* erstellt.

In einem solchen Plan müssen die verschiedenen möglichen Eigenschaften der Software und Hardware des Raumschiffes beschrieben werden können, wie z.B.

- Bedingungen für Zustände / Handlungen (die Kamera muss an sein, um ein Foto machen zu können)
- die fortgeschrittene Zeit
- Ressourcenbeschränkungen

- ständig parallel laufende Programme. Diese Programme können nie terminieren, sondern nur zwischen verschiedenen Zuständen wechseln (wie die Antriebskontrolleinheit)
- funktionale Abhängigkeiten
- langfristig verwendete Parameter (Brennstoffverbrauch)
- der Planer muss in der Lage sein, mit sog. Expertensystemen zu kommunizieren und von ihnen Ziele zu übernehmen, welche in den Plan eingebunden werden sollen

Um nun so ein Modell zu beschreiben, welches die Funktionsweise der zugehörigen Anwendung möglichst genau angeben soll, verwendet der Planer zwei strukturelle Prinzipien:

1. *Das Zustandsvariablenprinzip (state variable principle)*: Die Evolution eines Systems über eine Zeitspanne wird durch einen endlichen Satz an Zustandsvariablen beschrieben.

Jede Zustandsvariable kann zu einem Zeitpunkt nur einen Wert haben. Auch muss jedes im Plan vorkommende Literal genau einer Zustandsvariable zugeordnet sein. Das Literal ist also der die Belegung einer Zustandsvariable zu einem bestimmten Zeitpunkt. Ein Plan ist somit eine zeitliche Abfolge von Änderungen aller Systemzustandsvariablen über die Gesamtdauer des Planes.

Zum Beispiel kann man mit diesem Prinzip in der „Affe und Bananen“ Welt (in dieser Welt gibt es nur einen Affen, Bananen und einen Felsen) alle möglichen Zustände mittels folgender Zustandsvariablen beschrieben werden: der Ort des Affen, der Ort des Felsens, der Ort der Bananen und die Höhe des Affen (befindet sich der Affe auf dem Boden, klettert er gerade auf den Felsen oder sitzt er auf dem Felsen).

2. *Das Merkmalprinzip (token principle)*: Es muss keine Unterscheidung von Zustandsliteralen und Handlungsliteralen gemacht werden. Ein einziges Literal, das *token* reicht aus um die Veränderung von Zustandsvariablen über eine Zeit zu beschreiben.

Dieses Prinzip widerspricht dem klassischen Planungsgedanken, bei dem zwischen Veränderung und Zuständen unterschieden wird. Der Planer verwendet einfach eine einzige Art von Literal, welches die Änderungen der alles beschreibenden Zustandsvariablen beschreibt. Auf diese Weise erhält man ein einfaches Verfahren, den Plan zu beschreiben bzw. zu erstellen. Sämtliche Zustandsänderungen in einem Plan werden also mit Hilfe von *token* dargestellt.

Um zu gewährleisten, dass ein Plan zuverlässig ausgeführt wird, formuliert der Planer die Pläne als „*constraint networks*“, was soviel heißt wie „Netzwerk aus Bedingungen“. Die Pläne stellen also mehr eine Art Hülle von Bedingungen dar, welche in einem gewissen Zeitrahmen erfüllt sein sollen. So legt ein Plan beispielsweise bei einer Beschleunigung des Weltraumgefährtes nicht fest, wann genau wie stark beschleunigt werden muss, sondern nur von wann bis wann eine gewisse Geschwindigkeit erreicht werden soll. Dies lässt der Ausführungseinheit einen Spielraum um auf bei der Ausführung des Planes unerwartete Ereignisse zu reagieren. So kann sie eine andere Tätigkeit vorschieben, zuerst einen Fehler beheben, usw.

### **Angewandte generierende Planung**

In Abbildung 2 beschreibt wie der Planer einen Plan erstellt. Falls ein Teil des Planes Fehler hat erweitert er das Netzwerk aus Bedingungen um diesen Fehler zu beheben. Danach überprüft der Planer die Zusammensetzung des Teilplans wiederum auf Fehler. Sollte er immer noch einen Fehler aufweisen, so macht er die vorige Korrektur rückgängig und probiert eine andere Methode (*backtracking*). Ist der Teilplan fehlerfrei, wird er zurückgegeben.

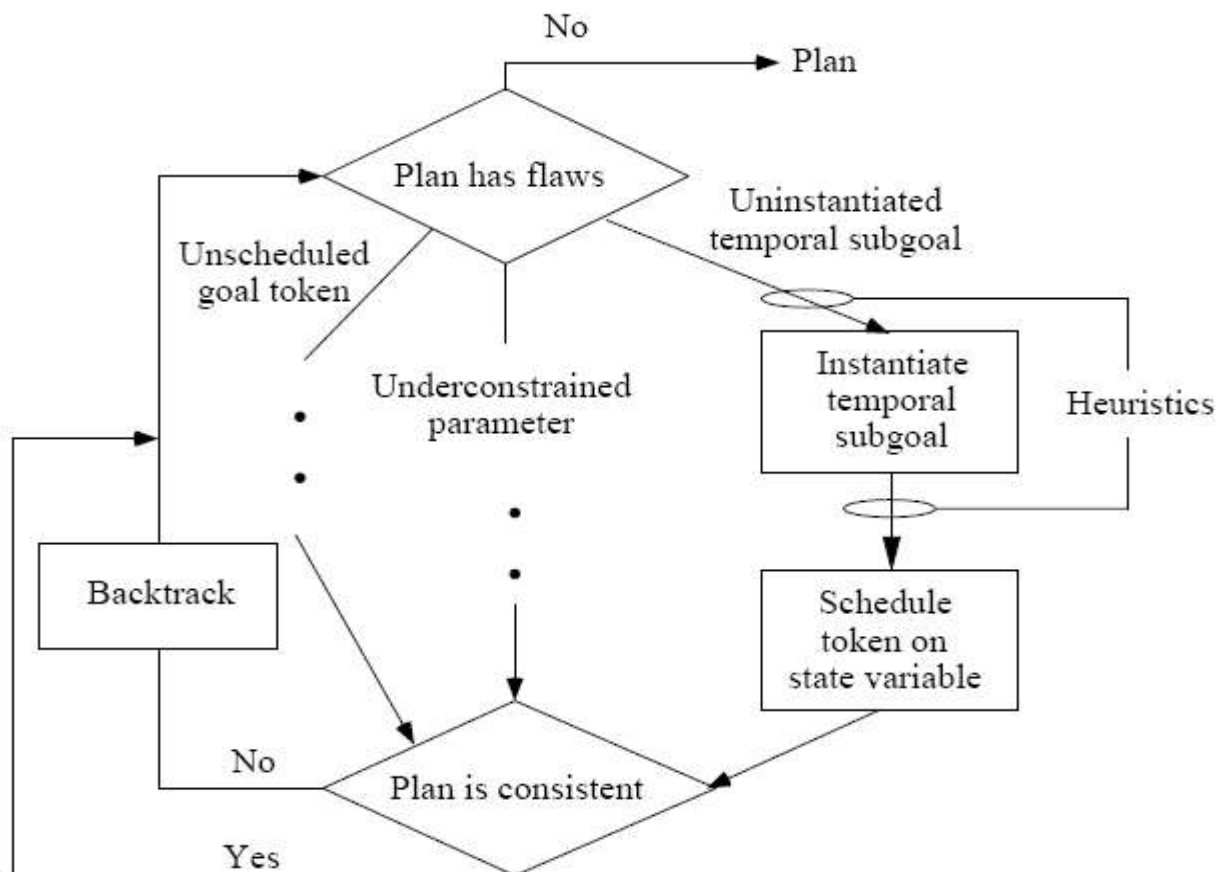


Abbildung 2: Problembeseitigungszyklus des Planers

Der Planer setzt den letztendlichen Plan also aus vielen kleinen atomaren, selbst erzeugten Plänen zusammen und nicht aus atomaren Bausteinen. Dieser wird dann an die Ausführungseinheit übergeben.

## 4. Die Ausführungseinheit

Die Ausführungseinheit lässt sich kurz gefasst als robustes, ereignisgesteuertes und zielorientiertes Ausführungssystem beschreiben. Sie fordert Pläne vom Planer an und führt sie aus. In diesen Plänen können gleichzeitig und voneinander abhängig arbeitenden Prozesse vorkommen, deren erfolgreiche Durchführung, Dauer und Ergebnisse noch im Unklaren liegen können. Sie unterstützt eine Sprache zur Festlegung von Prozesszielen, welche bei der Planübersetzung von der Ausführungseinheit in kleinere Prozesse aufgeteilt werden und gleichzeitig abgearbeitet werden können. Dies erhöht die Abstraktionsebene, auf der die Ausführungseinheit die zu erreichenden Ziele festlegen muss (*constraint networks*). Desweiteren gibt es eine enge Integration der Aufschlüsselung der gerade auszuführenden Prozesse und der Fehlermeldungen durch die Zustandsidentifikations- und rekonfigurationskomponente in die Ausführungseinheit (*feedback*), was es der Ausführungseinheit ermöglicht auf Fehlerereignisse schnell reagieren zu können.

## 4.1. Periodisches Planen über ausgedehnte Missionen

Die Ausführungseinheit muss bei dem Planer regelmäßig nach neuen Plänen anfragen und der Planer mit anderen auszuführenden Aufgaben koordinieren. So verbraucht die Erstellung eines Planes einen nicht unerheblichen Teil der verfügbaren Energie und Rechenleistung und kann somit nicht zu einem beliebigen Zeitpunkt geschehen.

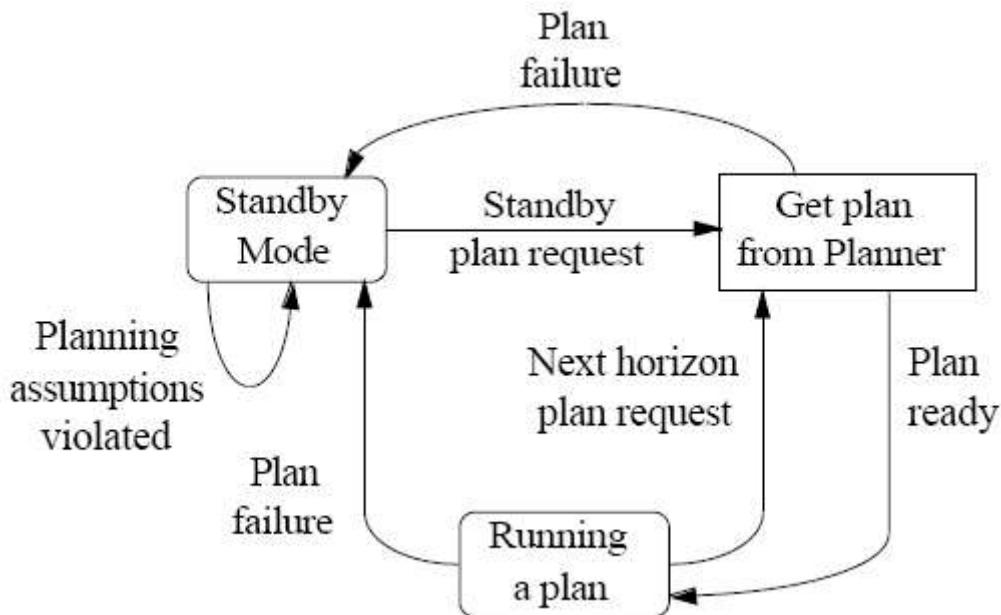


Abbildung 3: Periodisches Planen

Abbildung 3 zeigt die beiden häufigsten Wege des periodischen Planens. Wird ein Plan fehlerfrei ausgeführt, so wird der Planer von der Ausführungseinheit zu einem vordefinierten Punkt benachrichtigt einen neuen Plan zu erstellen. Dieser wird dann nahtlos an den vorherigen angefügt.

Schlägt die Ausführung eines Planes fehl, weil ein unbehebbarer Fehler aufgetreten ist, so werden sämtliche laufenden Prozesse von der Ausführungseinheit abgebrochen und das System geht in einen Standbymode über. Danach fragt die Ausführungseinheit nach einem neuen Plan von dem Planer (und informiert ihn über mögliche Einschränkungen des Systems, wie z.B. ein defekter Antrieb, welche bei der erneuten Planung berücksichtigt werden müssen) und führt diesen sofort nach dessen Bereitstellung aus.

Nun ist klar, dass das Planen selbst natürlich auch geplant werden muss. Denn einen Plan zu erstellen setzt genau wie eine andere Tätigkeit gewisse Bedingungen voraus, um erfolgreich zu sein. So muss ein Plan in einer gewissen Zeitspanne bereitstehen um eine optimale Auslastung der Ausführungseinheit zu erhalten. Auch muss der Planer von einer festen Ausgangslage bestimmter Zustände ausgehen können, denn bei sich ständig ändernden Zuständen ist es schwer einen einwandfreien Plan zu erstellen. Damit dieser rechtzeitig erstellt werden kann, wird im vorigen Plan bereits festgelegt, wann mit dem Planen des nächsten Planes angefangen werden soll. Beim Ausführen des Planes wird also das Erstellen des nächsten Planes von der Ausführungseinheit genauso behandelt wie jede andere mögliche Aktivität auch.

## 4.2. Robuste Ausführung eines Planes

Die Ausführungseinheit muss Pläne auch bei auftretenden Fehlern erfolgreich ausführen können. Sollte es nun passieren, dass das System auf einen Fehler stößt bevor die Planungsphase für den folgenden Plan abgeschlossen ist, wird die laufende Planungsphase abgebrochen und das System in einen Standbyzustand versetzt, in welchem es die Planungsphase erneut beginnt. Angesichts der nicht ganz unerheblichen Planungszeit von durchschnittlich 8 Stunden pro Plan kann ein solcher Fehler sehr schnell zu einer Verfehlung von Teilzielen oder ganzen Missionen führen. So könnten zum Beispiel keine Fotos von einem schnell vorbei fliegenden Kometen gemacht werden, da aufgrund eines Fehlers erst ein neuer Plan entworfen werden muss. Demzufolge sollten Pläne möglichst immer erfolgreich ausgeführt werden können, selbst bei auftretenden Fehlern.

Die Ausführungszeit schätzt deswegen immer die Ausführungszeit von einzelnen Prozessen vor deren Ausführung ab. Dies stellt sicher, dass ein Prozess, der exakt zu einem bestimmten Zeitpunkt ausgeführt sein muss, abgeschlossen ist und sich nicht auf Grund seiner eigenen, vorher oft unbekanntes Ausführungszeit verzögert.

Bei bereits in der Planungsphase absehbaren Unsicherheiten, wie zum Beispiel die Frage wie lange genau ein Antrieb angeschaltet sein muss, um eine bestimmte Endgeschwindigkeit zu erreichen, baut der Planer Prozesse in den Plan ein, die von der Ausführungseinheit nicht unbedingt ausgeführt werden müssen. Die letztendliche Entscheidung darüber fällt die Ausführungseinheit, indem es zur Laufzeit die Bedingungen überprüft und bei Bedarf einen solchen Prozess startet. Dies erhöht die Flexibilität und Stabilität eines Plans enorm.

Meistens aber tritt ein Fehler in einem bestimmten Teil eines Systems auf, welcher durch ein Reset des Systems behoben werden könnte. Allerdings würden bei diesem Reset auch Teilsysteme betroffen, die eigentlich einwandfrei funktionieren und dadurch könnten unter Umständen neue Probleme entstehen. Deswegen müssen beim Beheben des Fehlers immer die Zusammenhänge zwischen der fehlerhaften Komponente und dem Rest des Systems beachtet werden. Da all diese Methoden der Fehlerkorrektur zur Laufzeit nicht unbedingt trivial sind und die Ausführungseinheit ohnehin schon mit dem Ausführen des Planes ausgelastet ist, wird es bei der Fehlerbehebung stark von der *Zustandsidentifikations- und rekonfigurationskomponente* unterstützt.

## 5. Die Zustandsidentifikations- und rekonfigurationskomponente

Die Zustandsidentifikations- und rekonfigurationskomponente basiert auf einem einzigen beschreibenden, strukturellen Modell des Raumschiffes. Anhand dieses Modells kann sie sämtliche Zusammenhänge der Hard- bzw. Software des Raumgefährtes nachvollziehen und mögliche Fehler entdecken.

Die Identifikationskomponente verfügt über die Möglichkeit, sich verändernde Konfigurationen der Hardware und Software des Raumschiffes aufgrund der Ausführung von Befehlen oder dem Auftreten von Fehlern zu verfolgen. Anhand des beschreibenden Modells und der auszuführenden Befehle durch die Ausführungseinheit errechnet sie die wahrscheinlichste Konfiguration und vergleicht diese mit den aktuell verzeichneten Daten. Abweichungen zwischen den vorhergesagten und den tatsächlichen Daten signalisieren einen Fehler. Die Identifikationskomponente lokalisiert den Fehler, schränkt ihn ein und bestimmt den Ursprung des Fehlers. Des weiteren unterrichtet sie die Ausführungseinheit darüber, ob ein Befehl erfolgreich ausgeführt worden ist und erlaubt sogenanntes *monitoring*, also die Verfolgung der Ausführung eines Planes durch die Ausführungseinheit, was zum Nachvollziehen der von dem Remote Agent eingeleiteten Schritte durch das Bodenteam enorm hilfreich ist.

Wenn die aktuelle Konfiguration des Raumschiffes nicht der erwünschten Konfiguration entspricht,

kann die Rekonfigurationskomponente den günstigsten Weg feststellen um die gewünschte Konfiguration zu erreichen. Auch kann sie alternative Konfigurationen finden, um ein gewünschtes Ziel dennoch zu erreichen, wenn die eigentlich dafür vorgesehene Konfiguration auch dann einen Fehler aufweist.

Sollte keine Rekonfiguration möglich sein, so versetzt die Rekonfigurationskomponente das Raumschiff in eine Standbykonfiguration und erwartet Anweisungen vom Planer oder dem Bodenteam.

## 6. Zusammenfassung

Um beim Remote Agent über längere Zeit verlässliche, autonome Operationen mit festen Fristen, Ressourcenbeschränkungen und gleichzeitige Aktivität unter eng vernetzten Teilsystemen zu erreichen wurden die Prinzipien des *modellbasierten Programmieren* und des *zielorientierten, wiederholten Ausführen* verwendet. Die daraus resultierende Architektur beinhaltet das *zeitlich begrenzte Planen*, die *robuste Ausführung* und die *modellbasierte Zustandsidentifikations- und rekonfigurationskomponente*.

Der Remote Agent wurde bereits 1999 über einen kurzen Zeitraum von einer Woche erfolgreich getestet. Verbesserungsansätze sind die Adaptivität, mit der der Remote Agent in unbekanntem Umgebungen reagiert, die Schnelligkeit des Planentwurfes und das eigene Verständnis ihrer Funktionsweise und damit eine schnellere und zuverlässigere Fehlerkorrektur.

Ziel in der Zukunft ist es, einmal ganze Flotten durch einen Remote Agent gesteuerten Sonden im Weltall zu unterhalten, welche selbständig Informationen beschaffen, diese untereinander austauschen und sich bei größeren Missionen gegenseitig ergänzen und unterstützen. Mit einer solchen Flotte würde sich der Wahrnehmungsbereich der Forschung im Weltall um ein Vielfaches erweitern.

## Quellenangabe

- [1] Nicola Muscettola (Recom Technologies), P. Pandurang Nayak (RIACS), Barney Pell (RIACS), Brian C. Williams, Remote Agent: To Boldly Go Where No AI System Has Gone Before, Bericht, NASA Ames Research Center, 1998.