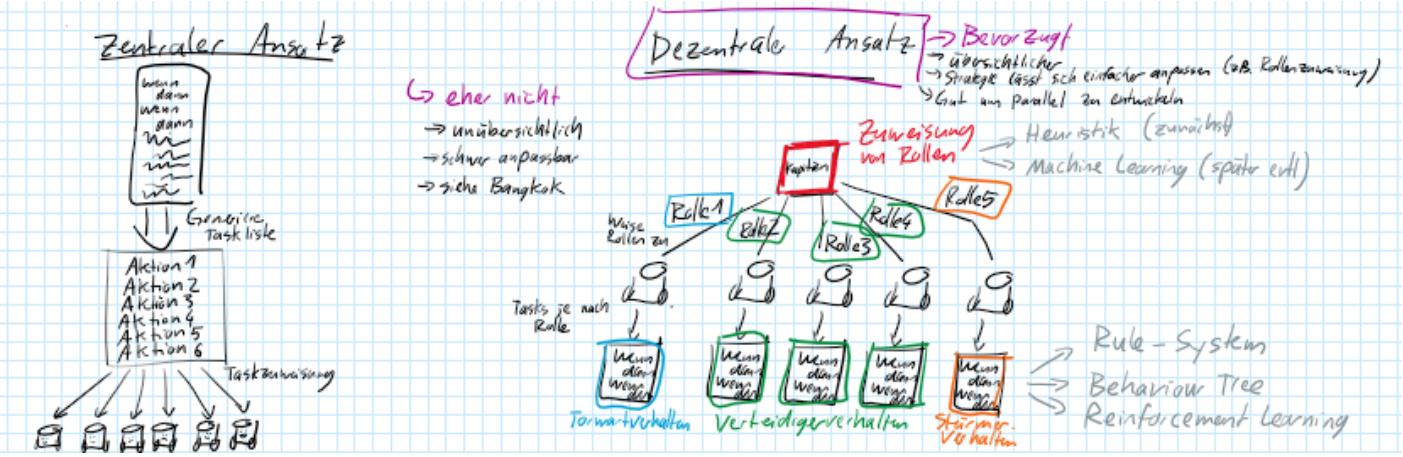
Grundprinzip und Architektur

Unter Taskmanager Bangkok wurde der ursprüngliche Ansatz beschrieben, die Strategie über ein einziges *Rule-System* zu realisieren. Aus den dort beschriebenen Problemen wurden Anforderungen für eine neue Architektur abgeleitet, die die Entwicklung, Anpassbarkeit und Erweiterung des Systems vereinfachen sollen. Die Ziele lassen sich folgendermaßen klassifizieren:

- *Mehrschrittiger Entscheidungsprozess*; Hiermit soll eine bessere Kapselung der Module erreicht werden, wodurch diese unabhängig voneinander entwickelt werden können. Außerdem soll nach dem "Teile und Herrsche"-Prinzip das große Problem der Strategie in mehrere Teilprobleme aufgeteilt werden, um die Lösbarkeit des Problems zu vereinfachen. Der wesentliche Ablauf der Task-Zuweisung lässt sich folgendermaßen beschreiben:
  1. Analyse des Spielfeldes
  2. Festlegung der Rollenverteilung (Anzahl Angreifer/Verteidiger...)
  3. Zuweisung der Rollen je nach gewünschter Rollenverteilung und Roboterposition
  4. Festlegung der auszuführenden Tasks in Abhängigkeit der Rollenverteilung
  5. Zuweisung der Tasks zu den verfügbaren Robotern in Abhängigkeit der Rollenverteilung
- *Kapselung der Module*; Roboter die an derselben Aufgabe arbeiten sollten besser kooperieren, während Roboter die andere Aufgaben ausführen ignoriert und nicht in die Entscheidungsfindung einbezogen werden sollten. Dies vereinfacht einerseits die Entwicklung, da nicht stets das Gesamtsystem betrachtet werden muss, um Änderungen an kleineren Teilaufgaben vorzunehmen und verbessert gleichzeitig die Kooperation, da Roboter innerhalb einer Gruppe sich besser abstimmen können. Die Gruppen werden dabei durch den Role Manager definiert.
- *Optimierung der Zuweisung*; Damit die Roboter als Team spielen, ist es notwendig dass sie gemeinsam spielen und ihre Mitspieler bei den Entscheidungen berücksichtigen. Da prinzipiell alle Roboter hardwaretechnisch gleich aufgebaut sind, kann jeder Roboter jeden Task gleich gut ausführen. Dies ermöglicht auch eine optimierte Zuweisung, bei der zunächst die auszuführenden Tasks ermittelt werden und die tatsächliche Zuweisung dieser zu den Robotern erst im nachfolgenden Schritt erfolgt. Dadurch kann der Gesamtweg des Teams minimiert werden, was in kürzeren Fahrtwegen und somit einer höheren Spieldynamik resultiert.

Die grobe Architekturänderung ist in der nachfolgenden Abbildung gezeigt. Links befindet sich das zuvor genutzte Rule-System, bei der ein komplexes Rule-System die Aufgaben für alle Roboter generiert. Rechts ist der für Bordeaux 23 dargestellte Ansatz, bei der die Roboter Rollen erhalten. Roboter mit derselben Rolle befinden sich in derselben Gruppe und arbeiten gemeinsam an einer Aufgabe, wie der Verteidigung des Strafraums.



Eine weitere Wesentliche Neuerung ist, dass die *Rule-Systeme* zur Steuerung der einzelenen Rollen durch *Zustandsautomaten* ersetzte wurden. Ein wesentlicher Vorteil dieses Schrittes besteht darin, dass einerseits besser kontrolliert werden kann, welche Zustände in welche anderen Zustände übergehen dürfen und andererseits, dass *Hysteresen* verwendet werden können. Diese Schritte stabilisieren das System und verhindern so den ständigen Wechsel von Tasks, falls ein instabiler Zustand (zb. durch Gegnerbewegungen/Rauschen/schlechter Datenqualität/...) eintreten kann, bei denen die Roboter sehr schnell zwischen Zuständen wechseln und daher letztlich keine der Aktionen ausführen.

> **Rule-System**:
> Ein Rule-System ist eine Architektur zur Steuerung von Systemen. Hierbei werden verschiedene Regeln definiert, die an Aktionen gekoppelt sind. Jede Rule besitzt zudem eine Priorität, wobei die Regeln in absteigender Priorität geprüft werden. Ist die Bedingung einer Regel erfüllt, so wird die zugehörige Aktion ausgeführt. Weitere Infos: Wikipedia, Complexica.

> **Zustandsautomat**:
> Ein Zustandsautomat (engl. State Machine) ist eine Architektur zur Steuerung von Systemen. Dabei werden verschiedene Zustände sowie Transitionen zwischen diesen definiert. Das System verbleibt in einem Zustand, bis die Bedingung an einer dort entspringenden Transition erfüllt ist. Jedem Zustand können Aktionen zugeordnet werden, um so ein Verhalten zu steuern. Weitere Infos: Wikipedia, Medium.

> **Hysterese**:
> Eine Hysterese beschreibt ein Prinzip, mit dem sich Zustandsänderungen kontrollieren und stabilieren lassen können. Beispiel: Wird ein Zustand bei

> gewechselt, wenn x > 2 ist, soll dieser erst wieder verlassen werden wenn x < 1 ist. Flackert der Wert zwischen 1,8 und 2,2 so bleibt der Zustand stabil. Weitere Infos: Max Planck Gesellschaft: Hysterese.

Revision #3
Created 13 February 2025 15:09:18 by Malte Sparenborg
Updated 13 February 2025 15:12:12 by Malte Sparenborg