Can Solution Manager Diagnostics be a Self Managed Decision System in the future?

SAP DEVELOPER NETWORK

SDN Community Contribution

(This is not an official SAP document.)

Disclaimer & Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.

Can Solution Manager Diagnostics be a Self Managed Decision System in the future?

SAP DEVELOPER NETWORK

Applies To:

Solution Manager Diagnostics

Summary

This article is written to debate if agent based mono-dialog systems can eventually emerge as a Self Managed system. This is just an attempt to foresee how SMD and its agents become 'intelligent agents'.

By: Venkat Mahalingam

Company: Active Global Support, SAP America Inc

Date: 23 Sep 2004

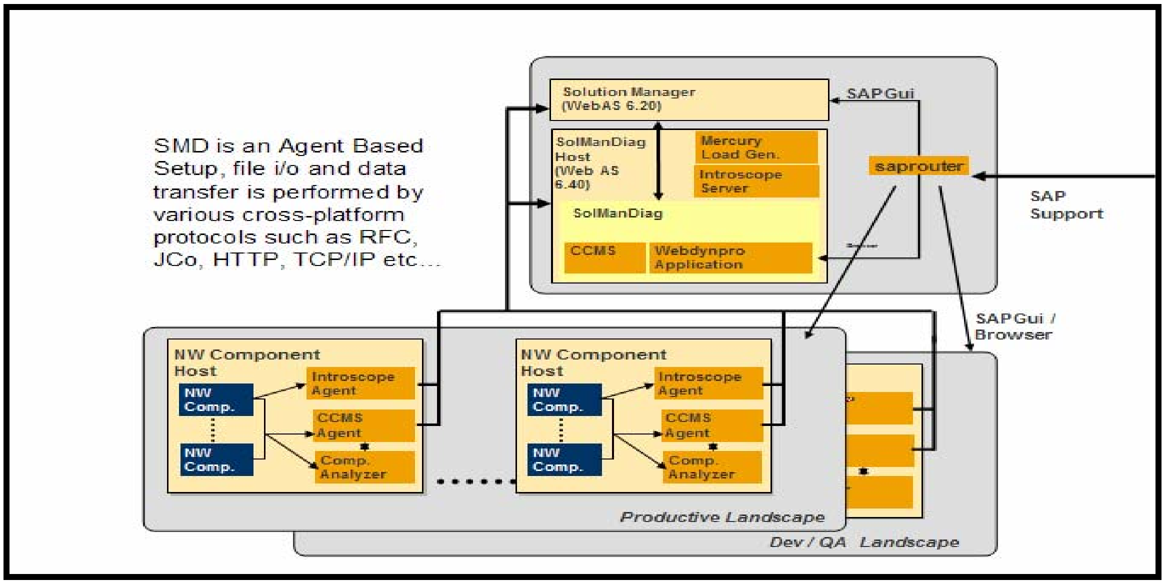
Introduction

Solution Manager Diagnostics (or SMD), is a new dimension to Java application monitoring and troubleshooting. SMD is an evolution; its core concepts and ideology trace itself back to the initial days of R/2 and R/3. SMD is one of the finest examples of re-factoring existing systems to provide simple granular solutions to complex paradigms.

There still remains a question; can an agent-based system like SMD (Solution Manager in the future) be a self-managed decision based system reducing human administrative and intelligence efforts?

Let us revisit the architecture of SMD and devise a theoretical prototype as to which parts can be remodeled for intelligent behavior.

Figure 1 – SMD Layout



SMD is an Agent Based Setup, file i/o and data transfer is performed by various cross-platform protocols such as RFC, JCo, HTTP, TCP/IP etc…

Figure 1, depicts in short the current architecture of the SMD System. The section which hosts the "Solution Manager (WebAs 6.20)" and the "Solution Manager Diagnostics" is the Monitoring System. The Landscapes below it (Dev/QA/Production) are the monitored systems.

As described in the footnote of Figure 1, the SMD system currently operates based on agent inputs via different protocols. These agents report to a central repository, which then is used to analyze the logged responses respectively. For instance, the CCMS Agent's destination system or repository is the in-built CCMS system. Similarly, the Introscope agent reports to the Introscope Enterprise Manger.

An Idea

If we rebuild a threadbare input and output system (based on the agent – repository architecture pair), and enhance the skeleton to think and react, I think we will have a self managed decision system. What remains is a language, an agent language which is already gaining a lot of popularity in mainstream Computer Science and Artificial Intelligence. As quoted,

'[Agents] communicate with their peers by exchanging messages in an expressive "agent communication language". While agents can be as simple as subroutines, typically they are larger entities with some sort of persistent control. **(Genesereth and Ketchpel, 1994, P48*)'***

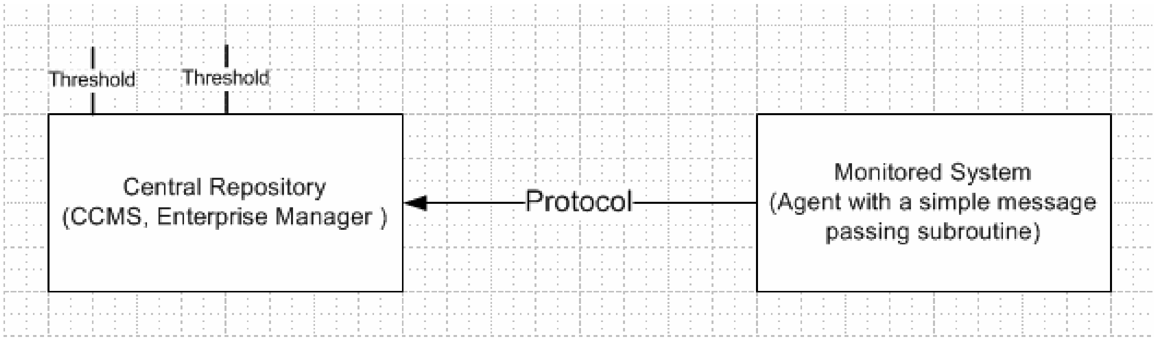
The reason why the statement sounds metaphorical is because Agents can be as simple as subroutines or even like softbots (software robots). The definition of agents has two notions; the first one is weak and relatively uncontentious; the second is stronger and more contentious.

Let's discuss each one in brief. The weak notion is used to describe an agent which is a *small piece of hardware or a software based entity* **(as agents in SMD in our case)** which have imbibed properties such as **autonomy, social ability, reactivity and pro-activeness**. For example, a concurrent UNIX process which encapsulates some state to do message passing to other interfaces in the same environment which is evolved from an object based concurrent programming paradigm.

The stronger notion is used to describe agents which in theoretical AI (artificial intelligence), describes an agent which has mentalistic notions such as knowledge, belief, intention and obligation. Some researchers also call them emotional agents (our agents are not emotional, probably we are!!)

Now let's visualize the skeleton which I was talking about –

Current Concept


Threshold
Threshold
Central Repository
(CCMS, Enterprise Manager)
Protocol
Monitored System
(Agent with a simple message passing subroutine)

The handshaking is as follows –

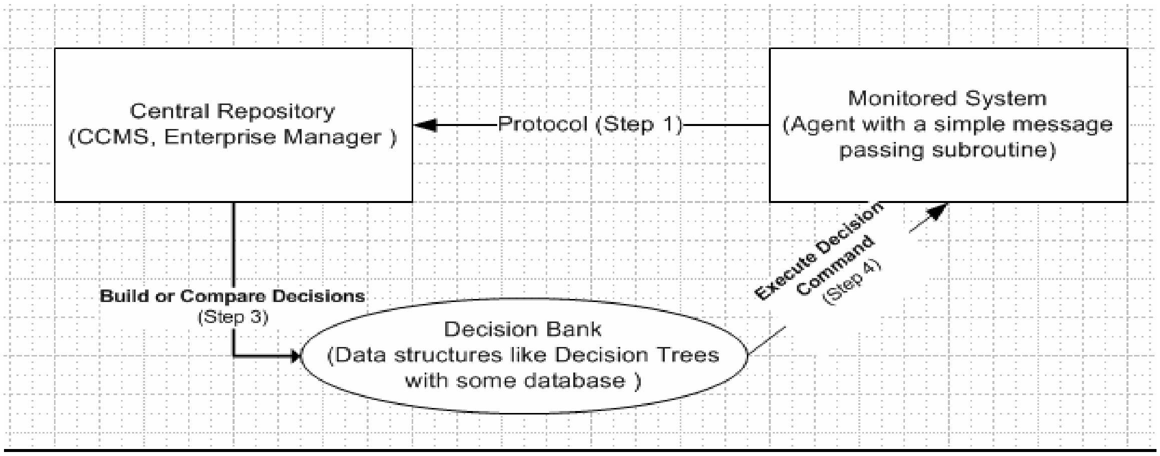
1. The message passing from the agent is mono-directional. Data is collected at the monitored system's end based on criteria and requirements. The data is persisted and parsed by a supported protocol, then transported to the Central Repository for analysis.
2. Based on some thresholds set in the central repository, the data points coming from the agent triggers certain alarms.
3. This methodology of communication is based on a generic client server computing model.

So what can be improved, hypothetically speaking in this implementation is that the knowledge used to decipher those raised alarms in order to diagnose the problem is lost with every new human intervention. There is no data structure which records those steps taken to analyze and troubleshoot the issue at hand.

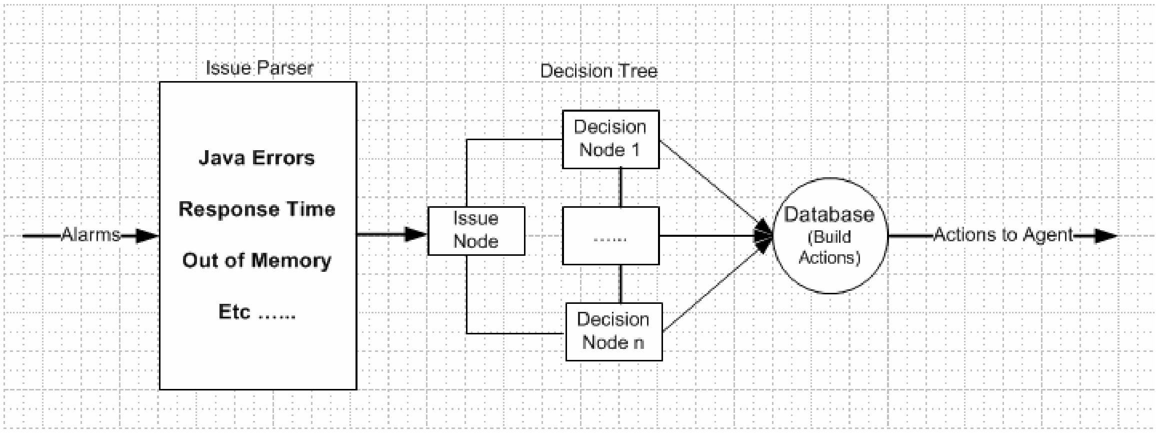
The Decision Making Concept

I incorporated a "Decision Bank" which is a simple Decision Tree which builds a knowledge base on a database. The process of decision building will enable some heuristics capabilities.

Based on the navigation on the decision tree the database will be queried for the appropriate action to execute.


Central Repository
(CCMS, Enterprise Manager)
Protocol (Step 1)
Monitored System
(Agent with a simple message passing subroutine)
Build or Compare Decisions
(Step 3)
Decision Bank
(Data structures like Decision Trees with some database)
Execute Decision Command
(Step 4)

Zooming on the “**Decision Bank**”,

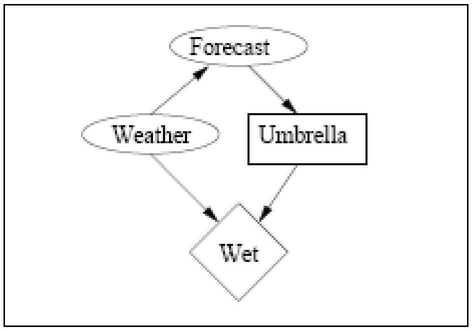


Just as a hypothetical example, let’s say the solution to a certain problem is adjusting the configuration files. The *Issue Parser* will parse the log files or a trace file along with a timestamp when the alarm is raised, and traverse the issue node and the decision tree which finally states that a certain property of a property file has to be changed from 512M to 1024M. The database is queried for the appropriate command to be executed for a decision to be made in the monitored system.

Now let’s concentrate on making the **agent** a decisive entity and see how we can debate some intelligence in it. I think I will dig into a subject which only modern Computer Science addresses, something new known as “**Decision Theory and the language of adaptive agent software**”. These days we deal with complex environments, hence the related problems are also complex. If we peruse the inherent properties of a complex environment we discover that it is Uncertain, Nondeterministic, Dynamic and Unique.

We must also understand as to how an Adaptive environment differs from a traditional environment. An adaptive environment chooses preferences as opposed to specifications. Tradeoffs are made in one during design time while in runtime in the other. There is a continuous change of aspirations in an adaptive environment. With respect to day to day operation, results in an adaptive environment depend on a probabilistic model (degree of belief) and it also has to deal with a wide range of contingencies based on dynamic computations. Keeping these criteria in mind, any mono-directive agent can be made intelligent and decisive.

An Adaptive Example



Let's analyze a small example. The diagram shown above is the process of making a derived decision based on some available inputs. Once the agent knows that the ***Forecast*** process determines ***Wet Weather,*** the agent seeks out for an umbrella. A similar concept is also used in building game architecture. In our case the decision bank will present some inputs based on the processing which takes place in the issue parser and the tree. The agent has to determine which option will alleviate the problem, and the steps taken can also be iterative.

Last but not the least, the prospects of SMD is unlimited. The tool can further developed and enhanced to solve more permutations and combinations of problems.

Author Bio

Venkat Mahalingam works for SAP America in Active Global Support Organization as a Sr Support Consultant with a total IT experience of 5 years. His main focus in SAP is Solution Manager Diagnostics and Enterprise Portal. His previous experience involves programming in C and C++ for stock trading applications. He is an avid gamer, plays an instrument and sketches in his spare time.