Design And Analysis Of Agent Architecture For Dashboard Platform Domain Services

V.S. Raghunathan, A. Sathya, N. Devi

Received 28 March 2017; Accepted 7 June 2017; Available online 12 June 2017

ABSTRACT

Construction by Configuration (CbC) brings in a complete change in Software Development Life Cycle. CbC influences every step of Software Development Life Cycle to bring down the cost and time taken for each step. CbC can also be applied for a) User Interfaces b) Syntax validations c) Semantic validations (policies) and d) Domain data Services. CbC at the front end takes care of User Interfaces, Syntax and Semantic validations. CbC at the back end manages the Domain data services using an Agent. As per SOA Architecture, the domain data services are defined as RESTful web services for data Push / Pull from domain database. This paper analyses the Agent architecture improving Performance by taking two case studies a) Analysis on Screened Vs Suspected Oral Pre-Cancer survey b) Analysis on Symptoms Oral Pre-Cancer Survey. The details of Optimizations in terms of cost, time are analyzed and detailed.


INTRODUCTION

The Dashboard Platform Services (DPS) is Copyright © No. SW-8923 / 2016 of National Informatics Centre which is got CbC implemented in the Front End. As part of this paper detailed Back End Agent was built, Tested and Evaluated Performance against the routine design and development of RESTful web service for each and every Dashboard requirement. DPS is easy to use Platform Service, where the user can easily configure their dashboards having readily available themes, also having the variety of charts and the layouts. DPS is a real-time user Interface which shows the graphical presentation of the user information like a data and the charts. In layouts, user can able to configure the one or more charts. The charts are of two categories: Single Series and Multi-Series supported by Dashboard Platform Service. Rendering of the charts are made available in Dashboard Platform Services based on open source / GPL (General Public License) scripts that are listed below:

- Google Chart
- High Chart
- JQ Plot
- D3js

To design a new dashboard using Dashboard Platform Services the following steps are necessary to configure the required Dashboard:

1. Register a new Dashboard
Dashboard Platform Services allows to design, customize, register, authenticate and also to deploy the web services. The Themes, Layouts, labels, header and footer of the dashboard are customized and the preview of the dashboard is available under preview dashboard. Once the dashboard is customized, the dashboard specification and data specification are available for download and use. Data specification is used for rendering the dashboard on the fly using the customized options in the CbC architecture applied in the Front End. Dashboard delivers Dashboard specification and data Specification. The data specification is available as an XML payload to fetch the data from the domain. Once the Domain web service is written, tested and validated then these web services are registered in the DPS platform. Registration of web services with necessary authentication details is part of an integration of the dashboard with the domain. After successful Integration the actual dashboard preview with live data from the domain using the web services that were registered later. Dashboard Platform Services is designed using loosely coupled architecture separating the specifications of dashboard, Data Specifications for Dashboard and open RESTful APIs to carry the Dashboard specifications and Data Specifications as required. The Rendering engine is based on the open source tools such as 1. Google Chart, 2. High Chart, 3. JQ Plot, 4. D3js.

A. Drawbacks of the existing model of Dashboard Platform Services:
- Web services have to be developed which requires more time.
- Skilled people are required to carry out the task.
- Managing the database becomes difficult from the novice users.
- More resources are required

B. Construction by Configuration:
Construction by Configuration (CbC) allows the re-usability [10] of abstraction, which has to be configured to adapt them to their local circumstance of use. By applying the Construction by Configuration (CbC) in the platform, it would allow the services to be designed and developed through reuse. The construction to be reduced by the configuration of well-defined components.

C. CbC in Dashboard Platform Services:
Dashboard Platform services is a loosely coupled architecture. Separating the data layer (heterogeneous databases, Middle layer (Open Standard, Scalable, restful web service into open XML payload) and Front-end services (open standard HTML5, CSS3, JavaScript, and Native apps on multiple platforms).

Fig. 1: Dashboard Platform Services Architecture

D. DPS in a CbC based Platform:
- Development for Dashboard requirement is not necessary. Instead, it can be easily configured using the available themes and a wide variety of charts in DPS.
Completely Open Source, reusable and vendor neutral.
- Given services can be depicted in periodic interval both as data and chart.
- It can be integrated with any Application using any technology.
- Scalable Design and Architecture
- Loosely Coupled Architecture with three layers such as Application Layer, Web Services Layer and Data Layer

E. Agent:
The Agent is a computer system [3] that is situated in some environment which is capable of autonomous action in the environment in order to meet its design objectives. The Agents [13] may be usefully classified according to the subset of these properties that they enjoy. Every agent, by our definition, satisfies the four properties are Reactive, Autonomous, Goal-Oriented, and Temporally Continuous. Adding other properties produces potentially useful classes of agents, for example, mobile, learning agents. Thus a hierarchical classification based on set inclusion occurs naturally. Mobile, learning agents are then a subclass of mobile agents. Agent is intelligent when it learns from an environment and uses the learning's in the subsequent interactions – Intelligent agents. In other words, it is the artificial intelligent. Also called an agent [13] is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.

The KidSim Agent explains:
"Define an agent [13] as a persistent software entity dedicated to a specific purpose. The word 'Persistent' which distinguishes agents from sub-routines; agents have their own ideas about how to accomplish tasks, their own agendas. 'Special purpose' distinguishes them from entire multifunction applications; agents are typically much smaller". The agent services [8] are readily available to their clients for the purpose of web service deployment.

F. Need for Agent Services:
- The current state of service delivery platform be taken to the next level by facilitating the agent service with the following features
- To promote scalability in terms of supported user and to enhance the performance of service delivery platform.
- To facilitate the ability of customization.
- To provide flexibility for data movement
- To adopt Construction by Configuration and agility.

G. Agent for Dashboard Platform Services:
Agent for DPS accepts Data specifications configured using Dashboard Platform Services and Delivers the desired one or more web service from the respective domain. The Purpose of an agent is to deliver the web services.

The agents can receive needed requirements that can be provided in the form of the input and process these requirements as a programmed to create a Service ID [9]. To get the required information the inputs are passed as the parameters. The final output is generated when the process completion is web services. The agents can have the capacity to act like the query builder and to fetch information and to execute these queries from the domain server [13]. Finally, the web services generated as an output should be verified for the format and thereby maintain the standardization it allows the services.

The e-Governance [11] with their advent of highly scalable RESTful services under service oriented architecture (SOA) used to provide the variety of application. By establishing the Interoperability and scalability these approaches may uses loosely coupled integration of Services. Since the delivery [12] of the service is required in a short time which leads to dissatisfaction with the existing business process, it has to become a necessity to re-engineer the strategy to 'Construct and Configure'.

MATERIALS AND METHODS

A. Agent Architecture in Dashboard Platform Services:
In the dashboard platform services where the requirements of the particular system is analyzed and the agent can act as the request manager. The Loosely Coupled Agent Architecture is depicted below:
The Dashboard Specification defines the format of data fetch from domain by using the SOA architecture which is Scalable RESTful services. To define the dashboard specification requires the various parameters namely, Dashboard Type, Layout, No. of charts and more. After defining the dashboard specification the user has been testing the database connectivity of Domain server, then the user can create data view in database (offline).

After creation of view the user have to test the view. Based on the no. of charts the views can be created for the Standard Dashboard type and for Drilldown and Dependent Dashboard type Single view can be created. The no. of Views is proportional to the Standard Dashboard chart. Test one or more views depending on the Dashboard Type. After successful view testing then the user can able to download the Web application ARchive.

Then deploy and test the Web application ARchive. After testing the Web application ARchive the URI is updated against the desired dashboard service. After updated in the Dashboard Platform Services the Dashboard is Ready to use.

The CbC based Agent [1] is programmed with the below requirement:

- Agent work with different platforms.
- Register the Service ID, Service type (Master, transmission, Multi-media, Dashboard etc.,)[10]
- Bundles DPS service-id.
- Create and monitor Agent database
- UI [10] for Agent user client to configure, enable, Test, Deploy, Buffering and monitor the registered request
- Service type may involve namely pull, authentication, response token and more. description, authorization, monitoring the status, filter condition.
- Service specification with chart, chart layout, a chart type, dimension, and measures.
- User preference to create and to customize views
- Processes involved in Tomcat services namely Invoke, configure and provide security fixes.
- Test and deploy view for web services.
- Test and deploy final web services.

The agents are built from the components that are customizable and reusable. By reducing the construction of an agent the approach allows the configuration of well-defined components and relationship between them. The agent [13] have the capability to act as the query builder to fetch the information from the domain server and to execute these queries. The output generated web service can be verified for the format and thereby allows the services to maintain the standardization.

B. Agent Work flow:

Takes Connection string from domain and creates final restful web services to provide data in desired XML format. The user has to proceed with username and Password for authentication purpose. Service ID will be
listed in a combo, based on service ID selection Service Name, Classification, sub classification, Layout, chart type, Level, chart based On(Date Wise, Non Date Wise) details.

**Fig. 3: Agent Workflow**

**RESULTS AND DISCUSSION**

Optimization Using CbC

Applying [10] Construction by Configuration in the platform, well-defined components reducing the time needed to design and develop the application services. By reducing the time complexity the major activity is using the CbC agent. The following steps are the various levels used for the optimization of time and cost resources.

A. **Various levels of optimization:**
   1. CbC based architecture of DPS with web services for data push and push services.
   2. CbC based Agent for web services automation
   3. Optimization using store and forward method (Buffering) for repeated Requests.
   4. Additional optimization by tuning the database (Indexing)
   5. Additional optimization as a part of Agent using multi-dimensional tables (cube data / Big data analytics)

**Level 0 - CbC based architecture of DPS with web services for data push and push services:**

The user can able to configure the dashboard having readily available themes, charts and layouts. The data required for dashboard or retrieve using the domain web services at runtime. By using the data retrieved from the domain using web services the preview of dashboard is made available. Though using CbC architecture dashboard is configured the web services are to be designed, developed and deploy.

**Level 1 - CbC based Agent for web services automation:**

The time consumed for design, development and deployment of web services can be replaced with CbC Agent.
Fig. 4: CbC based Agent for web services automation

This optimizes the time and cost associated with deploying each and every web services that are required for the dashboard. Using the CbC Agent the dashboard web services requirements can be configured easily and made available instantly. Thereby reducing the time to a logistics.

**Level 2 - Optimization using store and forward method (Buffering) for repeated Requests:**

As the dashboard is viewed and re-viewed repeatedly at multiple levels it is suggested to use store and forward method to optimize the dashboard hits. This method is one of the additional optimization for repeated requests.

**Level 3 - Additional optimization for tuning the database (Indexing):**

Indexes are special lookup [6] tables that the database search engine can use to speed up data retrieval. Simply put, an index is a pointer to data in a table.

Creation of Index:

\[ \text{CREATE INDEX index_name ON table_name;} \]

Once an index [7] is created, no further intervention is required: the system will update the index when the table is modified, and it will use the index in queries when it thinks doing so would be more efficient than a sequential table scan. Creating an index on a large table can take a long time. By default, PostgreSQL allows reads (SELECT statements) to occur on the table in parallel with index creation, but writing queries of Insert, Delete and Update are blocked until the index build is finished.

**Level 4 - Additional Optimization As A Part Of Agent Using Multi-Dimensional Tables (Cube Data / Big Data Analytics):**

This technique is one of the extra optimizations as a part of Agent used for the drill down charts which are essentially multi-dimensional data. Multidimensional [4] Online Analytical Processing Server (MOLAP) uses the array-based multidimensional storage engines for multidimensional views of data. With multidimensional data stores, the storage utilization may be low if the data set is sparse. Therefore, many MOLAP server use two levels of data storage representation to handle dense and sparse data sets.

Here two operations might be used for the OLAP operations they are slice and dice operations. The slice selects one particular dimensions from a given cube and provide a new sub-cube. And the dice selects two or more dimensions for a given cube and provide a new sub cube.

OLAP Cube: To provide a quick response to queries by dimension and measure an OLAP cube [2] is the best technology used to stores the data in an optimized way. In terms of development, learning and project time. It will return benefits in optimized response time to analyze the large amount of data. This capability can result in insights that drive actions and decisions that enable a large organizational productivity, cost saving.

**Case Study And Analysis:**

For the purpose of analysis two case studies namely

a) Analysis on Screened vs Suspected

b) Analysis on Symptoms are considered.

Screened Vs suspected case study deals with a number of persons screened and examination the number of
people who are suspected for oral pre-cancer. This defines the possibility of a number of cases to be examined further for diagnosis. The second case study deals with symptoms and influence of symptoms on different types of cancers.

A. Case 1 - Dashboard Analysis On Screened Vs Suspected (In Minutes):

In case 1 the dashboard analysis on screened vs suspected by writing the web service takes a long time. Then by using the CbC agent reduces the 99.65% in standard, 99.44% in dependent, 99.30% in drill down dashboard type. Whereas by using the CbC agent + Buffer optimization is 99.97% in standard, 99.97% in dependent, 99.62% in drill down. Additional optimization using CbC agent + Buffer optimization + database tuning may further optimization is 99.97% in standard, 99.99% in dependent and 99.95% in drill down dashboard type. Extra optimization using the multi-dimensional cube for drilldown analysis may increase the performance of the database to retrieve the data quickly. In the below experiment it is very clear that CbC agent which adopts CbC technology, Clearly, optimizes the time of the Dashboard rendering service. This is true for the different dashboard types (Standard, Dependent, Drill down)

Table 1: Case 1 Optimization Analysis

<table>
<thead>
<tr>
<th>Dashboard Type</th>
<th>Normal Data Push / Pull Services (t) In Minutes</th>
<th>CbC Agent</th>
<th>CbC Agent + Buffer Optimization</th>
<th>CbC Agent + Buffer Optimization + Database Tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value (t1) in min</td>
<td>% of Improvement</td>
<td>Value (t2) in min</td>
<td>% of Improvement</td>
</tr>
<tr>
<td>STANDARD</td>
<td>4320</td>
<td>15</td>
<td>99.65%</td>
<td>1.04</td>
</tr>
<tr>
<td>DEPENDENT</td>
<td>5760</td>
<td>32</td>
<td>99.44%</td>
<td>1.33</td>
</tr>
<tr>
<td>DRILLDOWN</td>
<td>8640</td>
<td>60</td>
<td>99.30%</td>
<td>6.32</td>
</tr>
<tr>
<td>TOTAL</td>
<td>18720</td>
<td>107</td>
<td>98.84%</td>
<td>8.69</td>
</tr>
</tbody>
</table>

The average percentage of improvement by applying CbC is 99.46% other than applying CbC store and forward technique (Buffering) and database tuning has been applied to achieve further improvement it is observed that after apply buffering it improve further to the average of 99.84% and after applying database tuning (indexing) the improvement is 99.97%

**CbC optimization**

*Case Study: Oral Pre-Cancer Screened Vs Suspected*

![Figure 5: Chart for case 1 optimization analysis](image)
B. Case 2 - Dashboard Analysis On Symptoms (In Minutes):

In case 2 the dashboard analysis on symptoms by writing the web service takes a long time. Then by using the CbC agent reduces the 99.61% in standard, 99.38% in dependent, 98.95% in drill down dashboard type. Whereas by using the CbC agent + Buffer optimization is 99.97% in standard, 99.91% in dependent, 99.56% in drill down. Additional optimization using CbC agent + Buffer optimization + database tuning may further optimization is 99.98% in standard, 99.94% in dependent and 99.95% in drill down dashboard type. Extra optimization using the multi-dimensional cube for drill down analysis may increase the performance of the database to retrieve the data quickly.

Table II: Case 2 Optimization Analysis

<table>
<thead>
<tr>
<th>Dashboard Type</th>
<th>Normal Data Push / Pull Services (t) in Minutes</th>
<th>CbC Agent</th>
<th>CbC Agent + Buffer Optimization</th>
<th>CbC Agent + Buffer Optimization + Database Tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value (t1) in min</td>
<td>% of Improvement</td>
<td>Value (t2) in min</td>
<td>% of Improvement</td>
</tr>
<tr>
<td>STANDARD</td>
<td>5760</td>
<td>22</td>
<td>99.61%</td>
<td>1.28</td>
</tr>
<tr>
<td>DEPENDENT</td>
<td>6480</td>
<td>40</td>
<td>99.38%</td>
<td>5.67</td>
</tr>
<tr>
<td>DRILLDOWN</td>
<td>11520</td>
<td>120</td>
<td>98.95%</td>
<td>50</td>
</tr>
<tr>
<td>TOTAL</td>
<td>23760</td>
<td>182</td>
<td>96.95%</td>
<td>56.95</td>
</tr>
</tbody>
</table>

The average percentage of improvement by applying CbC is 99.31% other than applying CbC store and forward technique (Buffering) and database tuning has been applied to achieve further improvement it is observed that after apply buffering it is improve further to the average of 99.81% and after applying database tuning (indexing) the improvement is 99.95%.

![CbC optimization](chart.png)

Case Study: Oral Pre-Cancer Symptoms

Fig. 6: Chart for case 2 optimization analysis

If we take ‘t’ as time, ‘c’ as cost the optimization is f(c,t). Where cost is based on involves cost on resources (man power) used for design, development, testing and development. If the time taken for one web service is \( t_{ws1} \) and if there are “n” number of web service the time taken is...
\[ \sum_{i=1}^{n} twsi. \]  

The cost involved for web services ws1 is c1. Then

\[ \sum_{i=1}^{no of web services} c1 \cdot twsi. \]  

For case study 1 - The total optimization on after applying 3 levels of optimization is

\[ t_1(\Sigma(wsii)) - t_3(\Sigma(wsii)) \text{ where } t_1 < t_2 < t_3 < t_4 \]

Thus in case study 1 considering all complexities in standard, dependent and drill down charts the time gets reduced by 99.95 % in the sequence 18720>107>34.37>4.813

For case study 2 - The total optimization on after applying 3 levels of optimization is

\[ t_1(\Sigma(wsii)) - t_3(\Sigma(wsii)) \text{ where } t_1 < t_2 < t_3 < t_4 \]

Thus in case study 2 considering all complexities in standard, dependent and drill down charts the time gets reduced by 99.97 % in the sequence 23760>182>56.95>9.623

**Conclusion:**

This paper details the architecture of the Agent with a live deployment of delivery of services through an Agent and analyses the responses to optimize the time and cost. The improved and optimized Agent architecture provides an effective cost and time optimization while delivering dashboard Services using Agent, which is the back end CbC for Dashboard Platform Services. CbC at Front-end implemented as Drupal Content Management System (CMS) generates the dashboard specification. The improved and Optimized Agent at the Back-end configures the database connection and the necessary data Push / Pull Services. The improved Agent architecture analyzed in this paper is flexible, Scalable and reusable. As CbC has been applied both the Front and also for the Back end can be used for providing SaaS (Software as a Service) in different domains for multiple application requirements. The case study referred is a proof to strengthen the need and effective use of CbC.

**REFERENCES**

1. Anitha et al., 2016. "Developing A Cbc Based Agent For Pre-Oral-Cancer-Analytics - A Case Study " In proceedings of National Conference On Innovative Computing Techniques NCICT.
10. Raghunathan, et al., 2015. " Construction By Configuration (Cbc) In E-Governance Service Delivery Platform "In International journal of applied engineering research, ISSN 0973-4562 10(0.5)
13. Stan franklin and art graesser, 2005. “is it an agent, or just a program?:a taxonomy for autonomous agents” In Incs, 1193: 21-35.