Recherche et développement pour la défense Canada | Defence Research and Development Canada

A Survey of Available Technology and Recommendations for Building an iVAC Capability

This work was performed
Under contract

PWGSC's Contract Number: W7701-135551

Scientific Authority:
Alexandre Bergeron-Guyard
(418) 844-4000 x 4107
DRDC – Valcartier Research Centre

And prepared by

Fujitsu Consulting Canada Inc.
2000 Boulevard Lebourgneuf,
Bureau 300, Québec (Québec) G2K 0B8

Version 1.0

Contract Report
DRDC-RDDC-2014-C218
July 2013

History

Version	Description	Author	Date
0.1	Draft	Pierre Jarest	2011-03-07
0.2	Internal review	Andriy Burkov	2013-04-23
0.3	Formatted	Pierre Jarest	2013-04-24
0.5	Internal review	All	2013-04-26
1.0	Final review and delivery	Guy Michaud	2013-07-25

Notes

Table of Contents

List of Figures

List of Tables

Résumé

Ce rapport présente les résultats d'une étude des technologies existantes pour répondre aux exigences des composants primaires pour le système d'analyse virtuelle du renseignement (iVAC). L'équipe a d'abord analysé et compris les exigences de chaque composant. Ensuite, les composants existants répondant à ces exigences, partiellement ou complétement, ont été analysées et comparées afin de fournir des recommandations pour leur utilisation dans le système IVAC. Les composants suivants ont été abordés: la reconnaissance vocale, la compréhension du langage naturel, la synthèse vocale, avatar, le cadre d'intégration et de l'interface utilisateur graphique.

Abstract

This report presents the results of a survey of existing technologies for fulfilling the requirements of primary components for the Intelligence Virtual Analysis Capability (iVAC) system. The team first analyzed and understood requirements for each component. Then, existing components addressing these requirements, at least partially, were analyzed and compared in order to provide recommendations for their use with the iVAC. The following components were considered: Speech Recognition (SR), Natural Language Understanding (NLU), Speech Synthesis (SS), avatar, integration framework and Graphical User Interface (GUI).

Executive Summary

This report is produced in relation to a initiative at Defence R&D Canada (DRDC) Valcartier to develop an Intelligence Virtual Analyst Capability (iVAC) which could be defined as "a computerized software assistant supporting the intelligence analysts in sense making tasks, while being ultimately capable of taking on autonomous analytical tasks in concert with other analysts (virtual or human)" . The capabilities of an iVAC system can be classified into seven broad categories, namely:

- Manage Context
- Acquire Information and Knowledge
- Timely Inform / Communicate / Interact with User
- Learn User and Task Models
- Monitor, Schedule, Manage and Evaluate Activities
- Support Complex Intelligence Tasks
- Interact with Humans and Other Systems

The iVAC is very similar to the domain of Intelligent Software Assistant (ISA) which is a special kind of software agent capable of performing tasks with minimum specific directions from users. Contrary to a conventional software which typically has a graphical user interface or a command line allowing to the user to invoke different functions of the software, an ISA is often controlled using natural language written or spoken phrases, or is capable of detecting users’ intentions by observing its actions. ISA is often seen as proactive software capable of offering its service to the user at the right moment as opposed to reactive software that waits until the user to initiate an interaction and responds accordingly.

This report presents the results of the investigation of the iVAC primary components. The objective was to identify existing technologies and approaches for addressing the needs of the following primary components: Speech Recognition (SR), Natural Language Understanding (NLU), Speech Synthesis (SS), avatar, Graphical User Interface (GUI) and integration framework. For each of them, this report present recommendations regarding the selection of the most appropriate technologies and approaches for building an iVAC prototype.

Based on those recommendations, a development plan will be prepared. This plan will address the development and integration of the iVAC prototype primary components. The development will be performed using an agile methodology, where capabilities will be integrated through a series of sprints. Each sprint will target the integration or development of a set of components, adding new capabilities each time. At the end of the framework development, the prototype will be ready for the integration of iVAC analysis components.

1 Introduction

1.1 Identification

This document is produced by Fujitsu Consulting (Canada) Inc. under contract W7701-135551/A, entitled "Intelligence Virtual Analyst Capability (iVAC) – Framework and Components". It presents the results of the survey and recommendations the primary components of the iVAC framework.

1.2 Overview

This document is organized as follows:

- Section 1 – Introduction: Identifies this document.
- Section 2 – Methodology: Presents the methodology used for performing the survey.
- Section 3 – Speech Recognition Components: Presents the results of the survey for speech recognition components.
- Section 4 – Natural Language Understanding Components: Presents the results of the survey for natural language understanding components.
- Section 5 – Speech Synthesis Components: Presents the results of the survey for speech synthesis components.
- Section 6 – Avatar Components: Presents the results of the survey for avatar components.
- Section 7 – Integration Framework: Presents the results of the survey for the integration framework.
- Section 8 – Graphical User Interface: Presents the results of the survey for graphical user interface components.
- Section 9 – Recommendations: Summarizes the recommendations for each of the analyzed components.
- Section 10 – Conclusion: Provides the conclusion of the survey and way ahead.
- Section 11 – Acronyms: Provides a list of acronyms used in this document.

2 Methodology

In order to conduct the survey of various software products, technologies, problem solving approaches and integration frameworks related to prototyping an iVAC, we mainly aimed at three sources of information: the open Web, private communications with vendors over email or phone, and our experts' knowledge or opinion. Due to the limited time available to do all the research, most of the gathered information came from the Web and expert's knowledge and judgment. We also tried some technologies when demos were available.

In order to identify technologies we first created a list of keywords that could potentially be used to find relevant Web pages. We then submitted these keywords to the Google search engine and, from the search results, we created a list of technologies of interest. We then gathered basic information about each of these technologies and, by looking at the properties of different technologies and by taking into account the iVAC requirements, we established a list of criteria to compare these technologies.

We then searched for more detailed information about each of the technologies. This detailed information should have allowed us to compare the technologies according to the list of comparison criteria. At this stage, we contacted vendors in case that the information about the technology of interest was limited on the open Web.

For certain problems, such as integration of heterogeneous components and natural language understanding, where no single product or technology is capable of fully solving the problem at hand, we asked our experts to apply their knowledge and judgment and to come with several alternative approaches for solving the problem and a review of useful technologies.

The final recommendations were given after consulting all experts and discussion.

3 Speech Recognition (SR) Components

3.1 Open Source

Among the open source speech recognition technologies, only two could potentially respond to the requirements of the contract, i.e. to be able to understand simple commands (following a predefined pattern) and to be used for free speech transcription for more complex interactions.

3.1.1 CMUSphinx

CMUSphinx is an open source speech recognition system developed at Carnegie Mellon University. It contains several independent speech recognizers (most widely used are Pocketsphinx and Sphinx-4) as well as Sphinxtrain, a set of acoustic model training tools. Sphinx-4 is written in the Java programming language and is intended for building desktop applications, while Pocketsphinx is written in C and is better for embedded devices such as smartphones.

Supported languages:

CMUSphinx can be trained to support any language. In order to add new languages, an acoustic model and a language model for the new language have to be provided. These models can be created using the tools bundled with CMUSphinx. By default the following languages are supported: English, Chinese, French, Spanish, German, and Russian (this means that the acoustic and language models for these languages already exist).

Using the tools provided with CMUSphinx, the acoustic models can be adapted to each individual speaker (e.g., to his voice and accent).

Supported modes:

Both command and control and dictation modes are supported.

NLU:

NLU functionality is not included. Simple commands to be recognized can be specified using grammars in JSpeech Grammar Format (JSGF). For more advanced command recognition (e.g., for understanding commands whose meaning depends on a context and/or a history of previous commands) a subsequent text analysis module is required.

API characteristics:

Sphinx-4 comes with a java API allowing using it for building a custom application for recognizing voice commands or for dictation. Pocketsphinx comes with an API for the C programming language.

Strengths/limitations:

The principal strength of CMUSphinx is its open source nature and a wide adoption in research labs and academia. It is highly customizable: by using the included tools, new acoustic and linguistic models can be trained; existing models can be adapted to each individual speaker. Furthermore, Sphinx-4 is written in the Java programming language, so it can be used to build solutions for any modern desktop operating system. It comes with a good amount of documentation and the community on the support forums is active and dynamic.

The main limitations compared with the state-of-the-art Nuance Dragon technology are a limited amount of training data used with the default acoustic model. It also lacks advanced algorithms for speech recognition and acoustic model adaptation, compared to the state-of-the-art commercial technologies.

Maturity:

The CMUSphinx technology is considered to be mature enough for building reliable custom speech recognition based applications. The speech recognition accuracy based on the default acoustic models may not be high enough especially in dictation mode and in languages other than English. However, for short commands based on a limited vocabulary and when adapted to each specific speaker and speaking conditions, good accuracy can be obtained.

Licensing and cost aspects:

Free of charge, a permissive BSD-like license allows modifications of the source code, as well as commercial use or redistribution.

3.1.2 Julius

Julius is another open source speech recognition engine written in the C programming language. It was designed for the Linux operating system, but it has been proven to work on Windows as well.

Supported languages:

By default, Julius comes with the Japanese language support. English acoustic and language models are available from a third party for free non-commercial use. Any other language can be supported by using custom trained acoustic and language models.

Supported modes:

Both the dictation and command and control modes are supported.

NLU:

Julius does not come with an advanced NLU capability. A grammar-based recognition parser named "Julian" is integrated into Julius. Julian is a modified version of Julius that uses hand-designed DFA grammar as a language model. It can be used to build voice command systems of a restricted vocabulary, or spoken dialog system tasks. As with CMUSphinx, for more advanced command recognition a subsequent text analysis module is required.

API characteristics:

Julius engine comes as a library and offers an API for the C programming language.

Strengths/limitations:

Among the strengths of Julius one should note that it is a fast and efficient engine; it is free of charge and open source; it integrates with Windows SAPI (can replace the default speech recognition engine of the Windows SAPI), and supports both dictation and command and control modes.

Julius' principal limitations are the default support of only one language (Japanese), limited availability of language and acoustic models for other languages and an average performance on them; the need of third party tools (such as HTK) for training and adapting models; limited portability (Julius is manly Linux-oriented); a very limited documentation and a lack of support as compared to the commercial engines or even the open source CMUSphinx.

Maturity:

The Julius technology is considered to be mature enough for building reliable custom speech recognition based applications. For getting it effectively working with languages other than Japanese or for adapting acoustic models to each individual speaker, one has to use third-party tools and invest considerable resources.

Licensing and cost aspects:

Free of charge, a permissive BSD-like license allows modifications of the source code, as well as commercial use or redistribution.

3.2 Commercial

The market of commercial speech recognition solutions is dominated by Nuance Communications, the supplier of the speech recognition technology for many of the modern virtual assistant software products including Siri for iOS. In this section, we provide an overview of the Nuance's solution and of several of its competitors.

3.2.1 Nuance Dragon SDK

Dragon NaturallySpeaking is a speech recognition software package developed by Nuance Communications for Windows operating system. There is also a Mac OS version of the software called Dragon Dictate. Nuance also sells an SDK, a software development kit (in two versions: client and server) allowing developers to create custom Windows applications with speech recognition (and synthesis) capabilities powered by the Nuances' Dragon technology. The Server version of the Dragon SDK allows installing the speech recognition software on a server in a network and performing a batch processing of audio recordings. The Client version installs on the user's desktop and is intended for the real-time voice processing including command and control and dictation tasks. In the following, we describe the Client version of the SDK as it is the most appropriate for the goal of the iVAC project.

Supported languages:

English, French, German, Italian, Spanish, Dutch. The accuracy of speech recognition can be increased by adapting the software to each individual speaker's accent and vocabulary.

Supported modes:

Both command and control and dictation modes are supported.

NLU:

In the command and control mode, Dragon can recognize predefined sequences of words or patterns of words as commands. However, a separate NLU module is required to recognize more complex commands as ones that imply a certain context or whose meaning depends on the previously received commands.

API characteristics:

The Dragon SDK Client technology is Windows OS only. It comes as a set of Microsoft ActiveX components, tools and libraries allowing creating fully customized voice enabled application for Windows OS. Developers access ActiveX components using Dragon API and can program using any programming language that ActiveX support.

Strengths/limitations:

Nuance's speech recognition technology is widely accepted as the state-of-the-art. This is the choice for mission critical applications where speed and accuracy are major criteria. For such domains as intelligence analysis, the accuracy of speech recognition is critical because of the complexity of the requests and their high frequency during each analysis session. The SDK also comes with a speech synthesis technology thus allowing the developer having only one technology to work with. This greatly shortens the learning curve for the developers, reduces incompatibility issues and, potentially, cost. A comprehensive and detailed technical documentation is available from the vendor, as well as online and phone based technical support. Software updates are also available.

The main limitations are a relatively high cost (see below), the support of only Windows OS, as well as the fact that ActiveX is only fully supported by the Internet Explorer browser.

Maturity:

This is a mature technology. The current version of the speech recognition technology is 12, while starting at version 10, the technology has already been considered by many individuals (according to the reviews found on the Internet) as the best performing on the market.

Licensing and cost aspects:

In order to develop and use solutions using the SDK, one has to buy a development license for 5000$. Once the application is built and ready to be used, for each speaker a separate user license has to be bought for 324$ per speaker (price decreases as you buy more licenses).

3.2.2 Microsoft Speech Platform

The Microsoft Speech Platform consists of an Application Runtime that provides speech recognition and synthesis functionality, an API for managing the runtime, and Runtime Languages that enable speech recognition and speech synthesis in specific languages. Using the Microsoft Speech Platform, developers can add speech recognition and synthesis functionality to enhance users' interaction with their applications.

Supported languages:

26 Runtime Languages are provided by the vendor, including various accents of English and French.

Supported modes:

Only command and control mode is supported.

NLU:

Grammars can be defined for recognizing phrases meaningful for a given application. No advanced NLU functionality is provided.

API characteristics:

Two APIs are available: native-code API for lower-level programming in C++ and managed-code API for higher-level programming in C# with .NET Framework objects. The managed-code API omits some of the low-level functionality of the native code in return for more efficient programming of speech functionality, and provides ample programming control for all but the most rigorous speech application requirements.

Also included in the SDK, the Microsoft Grammar Development Tools provide a comprehensive set of command-line applications with which developers can validate, analyze, and tune grammars for speech recognition.

Strengths/limitations:

Free of charge and can be distributed without restrictions with custom applications; comes with a comprehensive SDK and technical documentation. The speech synthesis components are included; 26 languages are supported.

Microsoft Speech Platform can operate only of Windows Vista or later version of Windows OS. It integrates well within the Windows ecosystem. The speech recognition engine cannot be trained to better understand specific users. No support service is provided by the vendor.

Maturity:

The Microsoft Speech Platform API and runtime are derived from the Windows Speech API (SAPI, see below) and the speech runtime in Windows, but are primarily intended to support speech applications running as standalone services. The current generation of the SAPI is 5 (started in 2000) and the current version is 5.4 which ships with Windows 7. The technology is considered mature and can be chosen for building enterprise-level Windows applications.

Licensing and cost aspects:

The Microsoft Speech Platform is free of charge. It can be used for building custom applications for Windows OS (Vista or later) and distributed with them with no significant restrictions.

3.2.3 Microsoft Speech API

Microsoft Speech API (SAPI) is a speech recognition technology integrated within Windows OS. It is very similar to the Speech Platform described above. The principal differences, according to the vendor, are listed in the table below:

Table 1: Comparison between Microsoft Speech Platform and SAPI

Feature	Microsoft Speech Platform	Windows/SAPI
Speech Runtime	Available for download from the Microsoft Download Center. Can be redistributed with your applications. Requires the Microsoft Speech Platform SDK and one or more Runtime Languages.	Included in Windows Vista, Windows 7, and Windows Server 2008. Not redistributable.
Language support and speech engines	Supports 26 languages for speech recognition and text-to-speech using redistributable Runtime Languages that you can download to enable speech recognition and TTS for specific languages. You can redistribute these Runtime Languages with your applications. Does not support the speech engines that are included in Windows.	Supports 8 languages for speech recognition and 3 languages for text-to-speech using the speech engines that are included in Windows Vista, Windows 7, and Windows Server 2008. Speech engines are not redistributable.
Speech engine access	Supports exclusive access to a speech engine by one application.	Supports shared access to a speech engine by multiple applications and exclusive access by one application.
Speech Recognition	Optimized to understand variations in speech	Optimized to train speech recognition for a specific user.

	patterns from a diverse population of users for any given language.	
Grammars	Supports command-and-control grammars that define the vocabulary that is meaningful to the application.	Supports free-text dictation and command-and-control grammars.
Audio format	Supports 8-bit audio, automatically downsamples audio files of higher resolution.	Supports 16-bit audio.
Native-code Application Programming Interface (API)	Speech Platform Native-Code API, included in Microsoft Speech Platform SDK. Requires the Microsoft Speech Platform Runtime and one or more Runtime Languages.	SAPI 5.4 for Windows 7 and Windows Server 2008. SAPI 5.3 for Windows Vista.
Managed-code API	Microsoft.Speech namespaces, included in the Microsoft Speech Platform SDK, available for download from the Microsoft Download Center. Requires the Microsoft Speech Platform Runtime and one or more Runtime Languages.	System.Speech namespaces in the .NET Framework versions 3.0 and higher.

Supported languages:

Recognized languages are U.S. English, U.K. English, traditional Chinese, simplified Chinese, Japanese, Spanish, French and German.

Supported modes:

Both command and control and dictation modes are supported.

NLU:

Grammars can be used to improve recognition of simpler commands; however, no advanced NLU functionality is included.

API characteristics:

Native API for C++ programming as well as managed API for managed applications programming (in C# or VB.Net) are available.

Strengths/limitations:

SAPI is a mature technology:

- It is integrated into the Windows OS and can be accessed from multiple programming languages;
- The speech recognition supports both command and control and dictation modes;
- Other speech recognition engines can be used instead of the default ones thus allowing writing code of the application once and then replacing the engine as required;
- The engine can be trained for each individual user;
- The technology includes both speech recognition and text to speech capabilities.

The main limitations are:

- Poorer performance compared to the state-of-the-art commercial products such as Nuance Dragon;
- The engine is integrated into the Windows OS and cannot be distributed with the application;
- Text-to-speech capability supports only few languages by default (models for other languages can be created by developers or acquired from third parties).

Maturity:

The current generation of the SAPI is 5 (started in 2000) and the current version is 5.4 which ships with Windows 7. The technology is considered mature and can be chosen for building enterprise-level Windows applications.

Licensing and cost aspects:

Speech runtime is included into the Windows OS and is not redistributable. The SDK is free of charge and can be redistributed with custom applications for Windows OS.

3.2.4 AT&T Watson

AT&T Watson is a speech recognition engine distributed by AT&T. At the time of writing the current report, we did not get access to the technical documentation on the engine itself due to the legal and time constraints. The below information is based on the description of the web based API (providing restricted capabilities to application developers and requiring internet access to work).

Supported languages:

Only English language is fully supported. The Spanish language is supported in certain contexts.

Supported modes:

The engine is optimized for certain types of requests called contexts. The supported contexts are Web Search, Business Search (points of interests), Voicemail to Text, SMS, Question and Answer, TV, Generic (a general purpose context that can automatically detect and transcribe the English and Spanish languages). The Spanish language is only supported in the Generic mode.

NLU:

No such functionality is provided by the technology.

API characteristics:

Four SDKs are available (HTML5, native Windows, iOS, Android). Using the SDK, an application with speech recognition functionality can be built using C++, C# or VB.Net. In order to be processed, the audio file containing a phrase is sent to a remote Internet server belonging to the vendor, along with the context; then the transcription is returned by the server. The longest phrase supported is four minutes.

Strengths/limitations:

The convenience of a web based API is that it can be easily called directly from a browser based application. Assuming that the iVAC representation layer will be realized as a browser application, this is a big advantage compared to using desktop-oriented speech recognition technologies and allows developing the application or a prototype faster.

The main limitations are:

- Impossibility for training the engine to recognize specific users' accents and vocabularies;
- Dependence on the third-party services and an Internet connection (thus excluding the possibility of building mission critical applications);
- Inability of providing grammars for more precise command and control mode;
- Latency.

Maturity:

The technology is less mature than previously described engines, however it is frequently considered (according to opinions found on the Internet) as a viable and less expensive alternative to the Nuance Communication's web API (based on the Dragon technology). The API of Watson can in particular be used for fast prototyping of voice activated applications.

Licensing and cost aspects:

The free trial period includes 90 days of free of charge access to the API. For 99$/year, one million API calls per month can be bought; one million additional API calls can be pre-purchased for additional $0.007 per call.

3.2.5 Google Speech API

The speech recognition technology of Google is considered very well performing in practice. It is based on the recent advancements in the speech recognition research, such as "deep learning". As the previous AT&T Watson technology, it is a web based API with similar resulting advantages and shortcomings.

Supported languages:

A variety of languages are supported (more than 30) including English and French.

Supported modes:

Short commands mode (without grammar) and continuous mode (for dictation) are available.

NLU:

No such functionality is provided by the technology.

API characteristics:

An API for JavaScript is built-in into the browser. Only Chrome browser version 25 or higher currently supports this API.

Strengths/limitations:

The convenience of a web based API is that it can be easily called directly from a browser based application. Assuming that the iVAC presentation layer will be realized as a browser application, this is a big advantage compared to using desktop-oriented speech recognition technologies and allows developing the application or a prototype faster; both brief (one-shot) speech input (for short commands) and continuous speech input (for dictation) are supported.

The main limitations are:

- Impossibility for training the engine to recognize specific users' accents and vocabularies;
- Dependence on the third-party services and an Internet connection (thus excluding the possibility of building mission critical applications);
- Currently only works in Chrome web browser version 25 and higher; latency;
- Grammars cannot be currently provided for better performance in command-and-control modes;
- No support from the vendor and a relatively scarce documentation.

Maturity:

The technology is less mature as previously described engines, however it is considered to be very precise.

Licensing and cost aspects:

At the moment of writing this report, the API could be accessed free of charge. No official information on the limits of usage or price was found.

4 Natural Language Understanding (NLU) Components

4.1 Description of NLU Component

The NLU component has two main roles to play. Given an input query from the user, it must identify:

1. What is asked of the system (the task to perform)?
2. What are the parameters of that task?

Solving this problem in general is an incredibly hard problem. Apple's Siri or IBM's Watson are two examples of systems that required a substantial effort to develop and that can only understand a restricted range of sentences and queries. So it is worth mentioning that the NLU component that will be developed in this project will invariably need be tailored to the application at hand.

In what follows, we propose three approaches of increasing complexity, both in the nature of the method and in the time that will be required to implement it. Section 4.3 also provides a description of the different open-source libraries that would be used to implement any of the three approaches.

4.2 Description of Approaches

To ground the description of these approaches with a specific example, we will consider the query "*find articles about recent terrorist activity*". In this example, the type of the task is a document search task, and the parameters of the task specify that documents mentioning recent terrorist activity should be returned.

4.2.1 Approach Based on Bag-Of-Words

In its simplest approach, the NLU component would reduce the input sentence as a bag-of-words, i.e. the set of the words contained in the sentence, thus removing the word ordering information within the sentence. This approach dominates information retrieval systems. Given that one of the main tasks of the system will be to retrieve documents, this approach would support the majority of use cases.

To identify the task type, we would associate with each task one or more keywords and look for those keywords within the bag-of-words. In our example, imagine that we had associated the word "*find*" to the task of returning relevant documents for the requested information. Then, since "*find*" is found within the bag-of-words, the system would correctly label the sentence as a document search task.

As for the parameters of the task, we would simply represent them with the bag-of-words of the sentence from which the task keyword (in our example, "find") would have been removed. Since most information retrieval systems only work at the bag-of-words level, this representation of the parameters of the task would be sufficient to perform it.

To support other tasks than document search, we would need to convert them into information retrieval tasks. Let's take the example of the task of buying tickets for a flight between Montréal and Toronto. By treating all possible flights available as a document containing the corresponding city of departure and destination, finding such flights could be achieved by a simple search over these documents.

Strengths/limitations

The main strength of this approach is its simplicity. It would require the least work to implement, using available NLP libraries.

Its main limitation is that the output of this approach is the shallowest (as opposed to deep or rich) of all three approaches, which will limit the accuracy of the different task components of the system.

4.2.2 Approach Based on Parsing with a Hand-Written Grammar

In this section approach, we propose to explicitly constrain the grammatical structure of the possible user queries. This would be achieved by defining a context-free grammar for all acceptable queries. A parser would then be used to derive the syntactic tree of input queries, from which the type and parameters of the task could be directly extracted.

For our running example, consider the following (unrepresentatively simple) context-free grammar:

```
S -> COMMAND PARAMETERS

COMMAND -> 'find' 'articles'

PARAMETERS -> 'about' ABOUT_WHAT

ABOUT_WHAT -> WORD ABOUT_WHAT | WORD

WORD -> 'recent' | 'terrorist' | 'activity'
```

This grammar explicitly separates the type of command (through the non-terminal COMMAND) and its parameters (non-terminal PARAMETERS). Here is the result of parsing the query "*find articles about recent terrorist activity*" using this grammar (generated by NLTK):

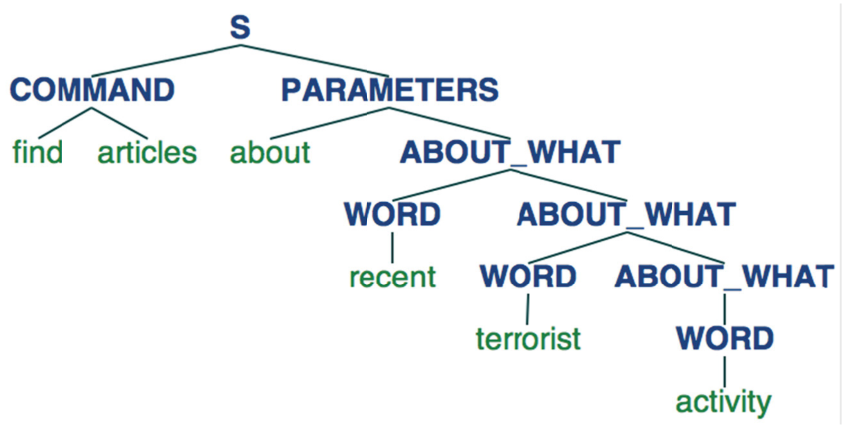


Figure 1: Context-Free Grammar

We see that the sub-tree labeled by the non-terminal `COMMAND` contains the phrase associated with the task type, while the highest sub-tree labeled by the non-terminal `ABOUT_WHAT` covers the phrase identifying the parameters of the document search task.

At this point, we could either return the full parsed tree, or explicitly write rules for extracting type and parameters information from the tree, to return only that information. Since we would have to write the context-free grammar anyways, writing these additional rules would require little extra work.

Strengths/limitations

The strength of this approach is that the output it provides is much more structured than the bag-of-words approach. This should increase the precision of the subcomponents that will accomplish the tasks in question.

Its main limitation is that a context-free grammar will need to be constructed by hand. The consequence is either that the grammar will only cover a limited set of likely user queries, or covering a wide variety of queries will come at the cost of substantial manual labour and trial and error. It should be noted however that the same grammar could be used by the speech recognition component, reducing potential errors in the query interpretation pipeline.

4.2.3 Approach Based on Rich Set of Statistical NLP Annotations

In this last approach, we propose to use the full power of available statistical NLP taggers to extract a rich and flexible set of annotations, from which the task type and parameters should be easy to extract. This approach thus attempts to bring us closer to full query understanding.

To understand an input sentence, we need to decompose its syntactic and semantic structure and identify the relationships between the concepts involved in that sentence. This additional "metadata" about input sentences is key for performing full query understanding, while also allowing us to be more robust to variations in input user queries.

To identify the task type, we could use two methods. The first would be to use a Part-Of-Speech (POS) tag, in order to identify the verbs within the sentence. As in the bag-of-words approach, certain verbs will be associated to different tasks that the system can perform. In the above example, a POS tagger would return the following metadata (generated by the Stanford NLP library):

- find/VB
- articles/NNS
- about/IN
- recent/JJ
- terrorist/JJ
- activity/NN

where each word has been labeled with its part-of-speech. By looking for the verbs corresponding to valid actions for the system, we can easily identify the task to perform. In this example, the task is to "find" something, i.e. search for something.

This method might not work if queries can contain more than a single verb. If we expect such queries to be likely, a second method would be to extract a syntactic parse of the sentence and look for the verb that is closest to the root of the tree. This verb should be the main action verb of the sentence and correspond to the task type. If it does not match any task type keywords, we would go down the parse tree searching for a task type keyword.

It might not always be straightforward to relate a given word to one of the supported tasks. In our example, if we had labeled the task of finding articles as a "*retrieve*" task instead of a "*find*" task, how would we make the connection? For this, we could access the WordNet ontology, which organizes words into a semantic graph and can identify the words that share a given sense. In our example, the words "*retrieve*" and "*find*" do share one of the WordNet senses, so we could infer that the word "*find*" refers to the system task "*retrieve*".

Moreover, since a word can appear in different inflected forms, the extraction of the lemma associated with a given word will be useful in order to match different words that share the same lemma (e.g. "activity" and "activities"). We could also extract the stem (i.e. the base root) of the words. For example, the stem for the words "retrieve" and "retrieval" is "retriev". So, if the task of searching for documents had instead been labeled "retrieval", the NLU component would allow identifying that the word "find" corresponds to this task, even though "find" is a verb and "retrieval" is a noun (WordNet does not share senses between nouns and verbs).

Next comes the hardest part of understanding the query: extracting the parameters of the task to perform. We propose to use a dependency parser, which will identify the set of relationships between the different words. In particular, a dependency parser run on our running example would output the following list of relationships between the words of the query (generated by the Stanford NLP library):

- root(ROOT-0, find-1)
- dobj(find-1, articles-2)
- amod(activity-6, recent-4)
- amod(activity-6, terrorist-5)
- prep_about(find-1, activity-6)

These relationships are given in the form of predicates, where the name of the predicate identifies the type of relationship and the two arguments determine what are the words sharing this relationship.

This representation also allows for the extraction of the parameters of the task, associated with the word "*find*". By following the `dobj` (object) relationship, we find that we are looking for "*articles*" (as opposed to other types of documents). Also, from the relationship `prep_about`, we obtain that we are searching for a type of "*activity*". To get more specific information about the type of activity, we can follow the relationship `amod` (modifier) that tells us that we are looking for activity that is both "*recent*" and related to "*terrorist*".

Different tasks will look for different relationships in the extracted dependencies and some work would go towards either identifying rules for extracting this information for a given task, or towards feeding the dependency graph to a machine learning algorithm that will learn to perform the task appropriately. However, this approach does not require writing a complete grammar for all likely user queries, which would be at least as tedious.

Other types of metadata we will want to consider outputting are:

- **Recognized named entities:** If a word in a query corresponds to an organization or a person, we would like to tag this word as such, so that subsequent manipulations of the query can be informed that this word is a named entity.
- **Semantic role labeling (shallow semantic parsing):** This is the task of identifying the verbs in a sentence and linking each with the chunks in the sentence that play a semantic role with respect to that verb. For example, the semantic role labeling output for the query "*I want to find articles about recent terrorist activity*" would correspond to the following (taken from the University of Illinois web demo):

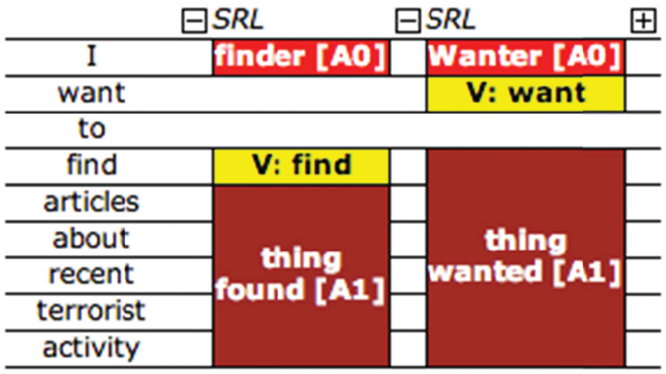


Figure 2: Semantic role labeling

Most relevant in our case, the "thing to be found" has been clearly identified as "*articles about recent terrorist activity*". However, since a shallow parsing is produced, more parsing of the chunk playing the role of "thing to be found" would be required in order to get a structured representation of what it corresponds to. The dependency parser output could be used in conjunction with this output in order to extract that information. The information added by the semantic role labeler is that roles are now explicitly semantically labeled, as opposed to syntactically labeled. For example, the relation labeled A0 is always the subject (the agent of the action), while A1 is always the subject (the patient of the action). There are other labeled relationships, such as the instrument (A2), starting point (A3), ending point (A4) as well as several modifiers (AM) specifying the location (ARGM-LOC) or time (ARGM-TMP).

We note that this approach could also be used to extract the same metadata from the collection of documents over which document retrieval would be performed as well. This would allow search to match specific metadata between the query and the documents and allow for a higher precision in the search.

Moreover, since this approach uses trainable statistical taggers and parsers, it would be possible to tune the NLU component to ensure that it works for certain types of queries. We would only need to collect and label a small set of representative user queries, which would cover the majority of use cases of the system. By adding this training data to the publically available labeled data and training our own taggers and parsers, we would obtain an NLU component tuned to our application.

To sum up, this approach will take a simple sequence of words and output the same sequence of words but augmented with POS tags, lemmas, stems, and dependency relations between words. It should ideally also support the extraction of named entities and semantic roles.

Strengths/limitations

The main strength of this approach is its flexibility. Unlike the bag-of-words or grammar approach, this approach should be more robust to word inflexions, synonymic word variations or syntactic sentence variations.

Its limitation is its lack of transparency. The behavior of statistical NLP taggers and parsers is not perfectly predictable. In the grammar approach, we know exactly what types of queries will be correctly parsed. However, from such a grammar, we could generate all possible queries along with their target syntactic parse and tune the NLU component to ensure that it works as required for these cases. Unlike the grammar approach, which will only work on a predefined set of query structures that the user will be forced to use, this approach should be able to generalize correctly outside that set.

4.3 Open source libraries for NLU component

Here are the most promising open source libraries we identified for this component. For each library, we identify whether or not it includes the following NLP components referenced in any of the three proposed approach for NLU:

- Stemmer or lemmatizer
- Part-Of-Speech tagger
- Named entity tagger
- Parser for hand-written grammar
- Syntactic parser (constituent parsing)
- Semantic role labeling
- Dependency parser
- Semantic ontology (WordNet)

4.3.1 DKPro Core

DKPro Core is a Java library for NLP algorithms. There are two versions: one with an APL license and one (larger) with a GPL license. It is mainly an interface to multiple NLP libraries. There is also DKPro Similarity, which is a complement to DKPro Core and that computes the similarity between two words or texts.

Supported NLP components:

- Stemmer or lemmatizer: **yes** (Snowball, TreeTagger, etc.)
- Part-Of-Speech tagger: **yes** (Stanford POS tagger, OpenNLP POS tagger, etc.)
- Named entity tagger: **yes** (Stanford named entity recognizer)
- Parser for hand-written grammar: **maybe** (by using Stanford parser on an hand-written PCFG)
- Syntactic parser (constituent parsing): **yes** (Stanford parser, Berkley parser, OpenNLP parser)
- Dependency parser: **yes** (MaltParser, Stanford parser)
- Semantic role labeling: **no**
- Semantic ontology (WordNet): **yes** (through DKPro Similarity)

API characteristics

DKPro Core comes with a Java API.

Supported languages

DKPro comes with models trained for many different languages, including English, French, German, Chinese, and more. All NLP components are supported in English. In French, there are no named entity taggers, semantic ontology or semantic role labeler.

Maturity

The development of this library started more than 5 years ago. It is also well documented. It is developed by the Ubiquitous Knowledge Processing (UKP) lab at TU Darmstadt in Germany. The group received two IBM's 2008 Unstructured Information Analytics (UIA) Awards for their work related to this library.

This all suggests that this library is fairly mature.

Licensing/cost

Free of charge, a permissive Apache license allows modifications of the source code, as well as commercial use or redistribution. Some of its interfaced components however use a more restricted GPL license.

Strengths/limitations

The main strength of DKPro is that it already interfaces with most NLP components we would need for the project. Hence, it has done some of the work we would necessarily need to do if we were to opt for an approach that relies on many statistical NLP methods. It is also usable within the UIMA information extraction framework, which is the framework behind IBM’s Jeopardy Watson system.

Its disadvantage is that it is not an NLP library per se, but an interface to many other libraries. Hence, it has many dependencies.

4.3.2 OpenNLP

Apache OpenNLP is a Java library supporting many statistical NLP components.

Supported NLP components:

- Stemmer or lemmatizer: **no**
- Part-Of-Speech tagger: **yes**
- Named entity tagger: **yes**
- Parser for hand-written grammar: **no**
- Syntactic parser (constituent parsing): **yes**
- Dependency parser: **no**
- Semantic role labeling: **no**
- Semantic ontology (WordNet): **no**

API characteristics

OpenNLP comes with a Java API.

Supported languages

OpenNLP provides trained models for English, but not French. Information about other languages can be obtained here: http://opennlp.sourceforge.net/models-1.5/.

Maturity

OpenNLP is mature and has a stable release since 2011. It is developed through the Apache Software Foundation.

Licensing/cost

Free of charge, a permissive Apache license allows modifications of the source code, as well as commercial use or redistribution.

Strengths/limitations

The advantage of OpenNLP is that it is a true NLP library, and not simply an interface to other libraries. It is also within both UIMA and GATE (another popular information extraction framework)

However, it is much less complete than DKPro Core, which interfaces with OpenNLP and many other libraries.

4.3.3 GATE

GATE is an information extraction platform, which supports the use of many different NLP annotation needed for information extraction. Unlike UIMA, which is an only a framework, GATE includes plugins for different NLP components or interfaces to other NLP libraries.

Supported NLP components:

- Stemmer or lemmatizer: **yes** (Snowball)
- Part-Of-Speech tagger: **yes** (ANNIE, Stanford POS tagger)
- Named entity tagger: **no**
- Parser for hand-written grammar: **no** (but allows for detecting regular expressions)
- Syntactic parser (constituent parsing): **yes** (Stanford parser)
- Dependency parser: **yes** (Stanford dependency parser)
- Semantic role labeling: **no**
- Semantic ontology (WordNet): **yes**

API characteristics

GATE comes with a Java API.

Supported languages

Mainly English.

Maturity

GATE is mature: its initial release dating back to 1995 and it is in use in the industry.

Licensing/cost

Free of charge, a permissive LGPL license allows modifications of the source code (though without any change to the license), as well as commercial use or redistribution (under LGPL license).

Strengths/limitations

GATE's strength is that it is a full information extraction framework, of the type needed to build a question answering system "à la Watson".

However, it supports only a limited amount of statistical NLP components that we would like to use.

4.3.4 Stanford CoreNLP

Stanford CoreNLP is a Java NLP library, from one of the best NLP lab in the world. It is often interfaced by other NLP libraries.

Supported NLP components:

- Stemmer or lemmatizer: **yes**
- Part-Of-Speech tagger: **yes**
- Named entity tagger: **yes**
- Parser for hand-written grammar: **maybe** (by writing a CFG as a PCFG and using the parser)
- Syntactic parser (constituent parsing): **yes**
- Dependency parser: **yes**
- Semantic role labeling: **no**
- Semantic ontology (WordNet): **no**

API characteristics

Comes with a Java API. It can also be used as a command-line tool. Wrappers for Python, Perl, Ruby and Scala are also available.

Supported languages

Main English, but includes POS tagger and syntactic parser for French.

Maturity

Though its initial release is recent (2010), the fact that so many other libraries interface it suggests it is an excellent NLP library.

Licensing/cost

Free of charge, uses a GPL license that allows modifications of the source but makes it difficult to incorporate into a commercial product that doesn't also have a GPL license.

Strengths/limitations

Stanford CoreNLP probably provides the best NLP components, in terms of accuracy.

However, it does not support all components we would like to use. We will probably end up using it, but perhaps through another, more complete NLP library.

4.3.5 ClearTK

ClearTK is an interface to many NLP tools as well as machine learning libraries. It supports the OpenNLP, Stanford CoreNLP, the MaltParser, the Snowball, and many others. It also supports many machine learning libraries, such as SVMlight, LIBSVM, OpenNLP MaxEnt, MALLET and GRMM, which allows for the development of new statistical NLP taggers. ClearTK was developed for the UIMA information extraction framework.

Supported NLP components:

- Stemmer or lemmatizer: **yes**
- Part-Of-Speech tagger: **yes** (Stanford POS tagger)
- Named entity tagger: **yes** (Stanford named entity recognizer)
- Parser for hand-written grammar: **maybe** (by writing a CFG as a PCFG and using the Stanford parser)
- Syntactic parser (constituent parsing): **yes** (Stanford parser)
- Dependency parser: **yes** (MaltParser, Stanford parser)
- Semantic role labeling: **yes** (ClearTK semantic role labeler)
- Semantic ontology (WordNet): **no**

API characteristics

Comes with a Java API.

Supported languages

Since ClearTK interfaces with the same libraries that DKPro interfaces with, it can use the same trained models and support the same languages.

Maturity

ClearTK has mostly been used in academic projects. However, we read that it is fairly stable and follows a rigorous development process (around 70% of the project is covered by unit tests).

Licensing/cost

Free of charge, a permissive BSD-3 license allows modifications of the source, as well as commercial use or redistribution.

Strengths/limitations

The strengths and limitations of ClearTK are similar to those of DKPro. Both cover many components, can be used within UIMA, but have many dependencies. ClearTK actually covers one more NLP component, a semantic role labeler, but DKPro seems a bit more mature.

4.3.6 ClearNLP

ClearNLP is a Java NLP library, developed at the University of Massachusetts Amherst and the University of Colorado Boulder.

Supported NLP components:

- Stemmer or lemmatizer: **yes**
- Part-Of-Speech tagger: **yes**

- Named entity tagger: **no**
- Parser for hand-written grammar: **no**
- Syntactic parser (constituent parsing): **no**
- Dependency parser: **yes**
- Semantic role labeling: **yes**
- Semantic ontology (WordNet): **no**

API characteristics

Comes with a Java API.

Supported languages

English only.

Maturity

Seems to be used mostly for research. We did find it interfaced by many other libraries.

Licensing/cost

Free of charge, a permissive Apache license allows modifications of the source code, as well as commercial use or redistribution.

Strengths/limitations

This library should be compared to the Stanford CoreNLP library, as both were developed in an academic research lab. Its strength is probably that it is the only library mentioned in this report that provides a semantic role labeling algorithm. However, it does not seem to be used as much as the Stanford library.

4.3.7 NLTK

NLTK is a Python NLP library.

Supported NLP components:

- Stemmer or lemmatizer: **yes** (Porter, Snowball)
- Part-Of-Speech tagger: **yes**
- Named entity tagger: **yes**
- Parser for hand-written grammar: **yes**
- Syntactic parser (constituent parsing): **no**
- Dependency parser: **yes** (MaltParser)
- Semantic role labeling: **no**
- Semantic ontology (WordNet): **yes**

API characteristics

Comes with a Python API.

Supported languages

English only, except for stemming (since Snowball supports many languages).

Maturity

NLTK is very mature. It also comes with a free book and is well documented.

Licensing/cost

Free of charge, a permissive Apache license allows modifications of the source code, as well as commercial use or redistribution.

Strengths/limitations

NLTK is the only library among those mentioned in this report that directly supports the parsing of sentences from a hand-written context-free grammar. It supports a fairly large number of NLP components and is very user friendly.

Its main disadvantage might be that its API is in Python. That being said, implementing an interface with it should be feasible.

4.4 Commercial Libraries for NLU Component

There are enough of good open source NLP libraries available that in our opinion there is no need to consider closed-source or paid commercial libraries. An example of a commercial NLP is LingPipe. . It can be used free of charge under certain conditions, but to incorporate it into a commercial product, a license of up to 40 000$ must be purchased (http://alias-i.com/lingpipe/).

5 Speech Synthesis (SS) Components

In this section we review several prominent speech synthesis technologies, open source ones and those coming from commercial vendors. We compare them in following terms: API characteristics, supported languages, maturity, licensing and cost aspects; we also give their main strengths and limitations.

5.1 Open source

5.1.1 Flite

Flite is a small, fast run-time speech synthesis engine. It is designed for embedded systems like PDAs and smartphones as well as large server installations which must serve synthesis to many ports. Flite is part of the suite of free speech synthesis tools which include Edinburgh University's Festival Speech Synthesis System (see below) and Carnegie Mellon University's FestVox project, which provides tools, scripts, and documentation for building synthetic voices.

API characteristics:

Flite is written in ANSI C, and is designed to be portable to a variety of platforms. Flite comes as a library that can be linked into other programs and it includes two simple voices with the distribution. Included with the distribution is also a small example executable that allows synthesis of strings of text and text files from the command line.

Supported languages:

By default Flite supports only English with a capability of adding other languages using external tools.

Maturity:

Flite is commonly considered a mature technology for building speech synthesis enabled application for embedded systems.

Licensing and cost aspects:

The software is open source and free of charge and can be used for building commercial applications with no significant restrictions.

Strengths and limitations:

Fast and lightweight, portable, mature, free of charge and open source with a permissive license.

The synthetic voices sound artificial, comparatively to certain of the reviewed commercial products, only English language is supported by default.

5.1.2 Festival

According to Wikipedia, the Festival Speech Synthesis System is a general multi-lingual speech synthesis system developed at the University of

Edinburgh. Substantial contributions have also been provided by Carnegie Mellon University and other sites. "It offers a full text to speech system with various APIs, as well as an environment for development and research of speech synthesis techniques. It is written in C++ with a Scheme-like command interpreter for general customization and extension."

API characteristics:

Festival supports a variety of APIs including a Scheme API, a Shell API, a server/client API, a C/C++ API, Java Speech API (JSAPI).

Supported languages:

Festival is designed to support multiple languages, and comes with support for English including British and American pronunciation, Welsh, and Spanish. There are also voice packages for several other languages, such as Castilian Spanish, Czech, Finnish, Hindi, Italian, Marathi, Polish, Russian and Telugu.

Maturity:

The technology is considered to be mature. The system is primarily developed under Unix (Linux, FreeBSD and Solaris), however the system has been ported to Windows and is used in a number of real applications on that platform. Although there is full support for Windows, the system is not considered to be as stable and as mature on Windows as on Unix-like platforms.

Licensing and cost aspects:

The technology is free of charge and is distributed under a free software license similar to the BSD License.

Strengths and limitations:

Free software with multi-lingual speech synthesis capabilities that runs on multiple-platforms offering black box text to speech, as well as an open architecture and flexible APIs. Festival is designed as a component of large speech technology systems.

The Windows version of the software is not as stable as a Unix one. The synthetic voices sound artificial, comparatively to certain of the reviewed commercial products.

5.1.3 FreeTTS

FreeTTS is an open source speech synthesis system written in the Java programming language. It is based on Flite (see above).

API characteristics:

FreeTTS is an implementation of Sun's Java Speech API (JSAPI).

Supported languages:

By default FreeTTS supports only English with a capability of adding other languages using external tools.

Maturity:

The software is considered mature. However, no new versions of the software were added since 2009.

Licensing and cost aspects:

Free of charge and open source with a permissive BSD-like license allowing using the technology for building custom applications with no significant restrictions.

Strengths and limitations:

Portable, fast, free of charge and open source.

Comes only with the English language by default. The synthetic voices sound artificial, compared to some of the reviewed commercial products.

5.1.4 eSpeak

eSpeak is a compact open source speech synthesizer for Linux, Windows, and other platforms.

API characteristics:

eSpeak can be used as a command-line program, or as a shared library. Supports both Windows and Linux OS. On Windows, it can be used as a shared library (a DLL), as a command-line tool, or can be accessed from the Windows SAPI interface.

Supported languages:

eSpeak does text to speech synthesis for the following languages (with varying quality): Afrikaans, Albanian, Armenian, Cantonese, Catalan, Croatian, Czech, Danish, Dutch, English, Esperanto, Estonian, Finnish, French, Georgian, German, Greek, Hindi, Hungarian, Icelandic, Indonesian, Italian, Kannada, Kurdish, Latvian, Lojban, Macedonian, Malayalam, Mandarin, Norwegian, Polish, Portuguese, Romanian, Russian, Serbian, Slovak, Spanish, Swahili, Swedish, Tamil, Turkish, Vietnamese, Welsh.

Maturity:

A mature technology; it is widely used on Linux operating systems; has also been used by Google Translate.

Licensing and cost aspects:

eSpeak is distributed under the GNU GPLv3+ license, which means that the software is free of charge and open source. However, the usage of the software is limited and allows using it for creating custom software that has to be distributed under a compatible license.

Strengths and limitations:

A wide choice of supported languages, good documentation, portability and integrality with other software, open source and free. However, the quality of speech varies greatly from one language to another; the perceived quality of the default voices is lower than that of other open source technologies seen above.

5.2 Commercial

Several well established and widely used commercial speech synthesis solutions are available on the market. They usually better quality speech synthesis capabilities than open source technologies reviewed above.

5.2.1 Neospeech VoiceText

Neospeech VoiceText is one of the leading commercial speech synthesis technologies providing high quality and very naturally sounding voices.

VoiceText text to speech (TTS) Engine/SDK offers single-user applications as well as Server/SDK for multi-user applications. VoiceText Engine/SDK supports SAPI 5.1 and C-based Application Programming Interfaces (APIs). VoiceText Server is an add-on software component to VoiceText Engine that manages multi-thread, multi-voice TTS services in a multi-access/user environment. Using the provided SDK, generally VoiceText Server is used for building custom server applications such as telephony systems, alert systems, etc. VoiceText Server/SDK supports C, COM, Java, and .NET-based Application Programming Interfaces (APIs).

Windows, Linux and several embedded environments are supported.

Based on the VoiceText technology, NeoSpeech TTS Web Services offer a hosted online speech synthesis solution to those who wish to develop a speech-enabled website, service, or application; Internet connection is required to use these Web services.

VoiceText comes with a support of English, Korean, Japanese, Chinese, and Spanish voices. The French language is not supported.

Neospeech VoiceText offers the most naturally sounding voices according to Fujitsu's internal review; multiple operation systems supported, multiple languages, and an SDK that support several APIs as well as Windows SAPI.

5.2.2 Nuance Vocalizer

Nuance Vocalizer is another leading technology of speech synthesis developed by Nuance Communications: the same company producing the Dragon speech recognition technology.

The Foundation SpeechObjects, Nuance's Java-based speech application development framework, defines classes that let developers create specific TTS prompt objects from a simple text string; Nuance Voice Web Server can be used to generate TTS prompts from VoiceXML applications.

Nuance Vocalizer features 63 voices supporting 43 different languages and dialects around the world. Both English and French are supported in multiple accents.

Vocalizer is a stand-alone product and does not come with an SDK.

5.2.3 Microsoft Speech Platform and SAPI

Microsoft Speech Platform supports 26 languages for speech synthesis. Microsoft SAPI supports three languages for speech synthesis: English, Chinese and Japanese.

5.2.4 Nuance Dragon SDK Client

Nuance Dragon SDK Client (see the Speech Recognition Components section) comes with the speech synthesis capability and supports vocal output in French, Dutch, Spanish, Italian, German, and English US/UK.

5.2.5 Cereproc

The CereVoice Engine SDK is a cross-platform text-to-speech (TTS) software development kit, enabling developers to add TTS output to their application. A simple, yet powerful, C/C++ API, CereVoice Engine, is provided across Windows, Mac OS X, iPhone OS, and Linux platforms. The CereVoice Engine API includes a Python wrapper to ease development of new applications. Supported languages: English, Dutch, French, Catalan, Spanish, Italian, and German.

The cost of the software is as follows: $1095 USD for the SDK, plus $400 per user of the software, plus support for the SDK at $595 per year. For example, for 10 users and one developer seat: $5690 for the first year, then $1690 per year, for the SDK and support of the voice. An additional user of the app would cost $400; if the number of users is greater than 49, then the price becomes to $375 per user.

5.2.6 Acapela

Acapela offers both client (Windows and Mac OS) and server (Windows and Linux) versions of SDK. Available in 32 and 64 bits; has several APIs (a proprietary API, Microsoft SAPI 4, Microsoft SAPI 5.1 & 5.3) and one common API for Mac OS and Windows allowing dual platform development; several programming languages supported (C, C++, VB (Through ActiveX), Delphi, Java); 30 languages available.

Pricing for the desktop SDK is $2000 (USD). Runtime licenses cost $200/unit for language for small volumes (1-49 units). Pricing for the Server SDK is $2,800. Runtime licenses cost $690/channel. A channel = 1 X real-time TTS output.

6 Avatar components

In this section, we describe technologies that can be used to build the Avatar functionality of the iVAC. We start by describing the open source software.

6.1 Open Source

6.1.1 Microsoft Agent

Table 2: Score Summary of Microsoft Agent.

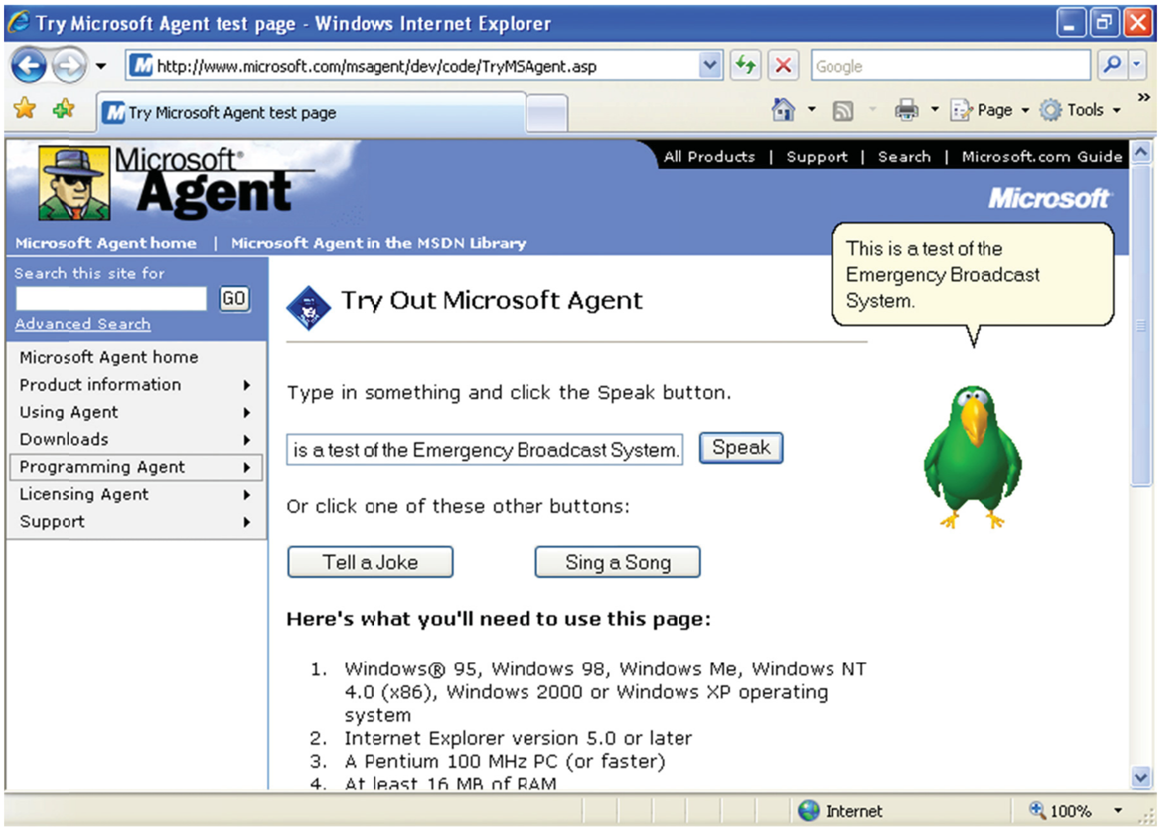
Score Summary	
Features coverage for Avatar dev.	★★★★★
Ease of integration	★★
API Availability and documentation	★★★★
Interactions with third-party applications	★★★★
Web integration possibility	★★★★★
Maturity / Long time support	(discontinued)
Dependent of external proprietary services (server farm)	no

Description

Microsoft Agent is a technology developed by Microsoft which employs animated characters, text-to-speech engines, and speech recognition software to enhance interaction with computer users. Thus it is an example of an embodied agent. It comes preinstalled as part of Microsoft Windows 2000 through Windows Vista. Microsoft Agent functionality is exposed as an ActiveX control that can be used by web pages.

Microsoft Agent is a technology that provides a foundation for more natural ways for people to communicate with their computers. It is a set of software services that enable developers to incorporate interactive animated characters into their applications and Web pages. These characters can speak, via a text-to-speech engine or recorded audio, and even accept spoken voice commands. Microsoft Agent empowers developers to extend the user interface beyond the conventional mouse and keyboard interactions prevalent today.

Microsoft Agent is not open source, but many open source alternative support existing Microsoft Agent version 2.0 characters, including Microsoft Office Assistant characters.


Try Microsoft Agent test page - Windows Internet Explorer
http://www.microsoft.com/msagent/dev/code/TryMSAgent.asp
Try Microsoft Agent test page
Microsoft Agent
All Products | Support | Search | Microsoft.com Guide
Microsoft Agent home | Microsoft Agent in the MSDN Library
Search this site for
GO
Advanced Search
Microsoft Agent home
Product information
Using Agent
Downloads
Programming Agent
Licensing Agent
Support
This is a test of the Emergency Broadcast System.
Speak
Tell a Joke
Sing a Song
Internet
100%

Try Out Microsoft Agent

Type in something and click the Speak button.

is a test of the Emergency Broadcast System.

Or click one of these other buttons:

Here's what you'll need to use this page:

1. Windows® 95, Windows 98, Windows Me, Windows NT 4.0 (x86), Windows 2000 or Windows XP operating system
2. Internet Explorer version 5.0 or later
3. A Pentium 100 MHz PC (or faster)
4. At least 16 MB of RAM

Figure 3: Microsoft Agent Try Out in Microsoft Explorer Web browser

API characteristics

- Multiple avatar characters available, including an Agent Character Editor software to create new characters;
- Interaction with a speech recognition engine;
- Interaction with a text-to-speech engine;
- Support of spoken voice commands.

Development

Microsoft Agent is designed primarily for developers who use languages or environments which support COM or Microsoft ActiveX control interfaces. These include:

- Microsoft Visual Studio (Visual C++, Visual Basic);
- Microsoft Office (Visual Basic for Applications);
- Microsoft Internet Explorer (Visual Basic Scripting Edition or Microsoft Jscript);
- Microsoft Windows Script Host (ActiveX Scripting Language). Microsoft Agent can be integrated into other software. One example of such integration is the "Desktop GMail Alert System"

(http://www.codeproject.com/Articles/14774/Desktop-GMail-Alert-System).

Maturity

Microsoft Agent software is in end-of-life cycle, beginning with Windows 7, it will not be included or supported in future versions of the Microsoft Windows operating system. Microsoft encourages Microsoft Agent application developers and redistributors to evaluate their activities in light of this decision.

Licensing / cost

Free

Strengths

Can be integrated into Web browser.In order to use a MS Agent character in Web browser, the animation must be available on the client computer. This can be done either by installing the complete character on the client computer, or by remotely invoking specific animation behaviors via HTTP requests. In web-based applications, a problem with the use of these characters is that they are not universally-installed on all client systems, and HTTP-based invocations of animation behaviors can be bandwidth-intensive. Remote invocation of ActiveX controls may also be blocked by user security settings, preventing execution of animation effects. However, for the purposes of the iVAC project, the character can be installed on each analyst's desktop.

Limitations

- This software can be employed in the project, but is discontinued.
- The software is limited to Microsoft Windows Operation System and ActiveX Controls.

6.1.1.1 Double Agent

Description

Double Agent is an open source alternative to Microsoft Agent that allows Agent applications to work on Windows 7 and beyond. It emulates the Microsoft Agent server and the Microsoft Agent ActiveX control. It supports existing Microsoft Agent characters, including Microsoft Office Assistant characters. It implements the Agent Server and Agent ActiveX Control programming interfaces, so it can be used with no changes to existing applications (http://www.cinnamonsoftware.com/double_agent.htm).

Maturity

The project is mature.

Licensing / cost

Free

Limitations

- DoubleAgent relies on the discontinued Microsoft® Agent software.
- Limited to Microsoft Windows Operation System and ActiveX Controls.

6.1.1.2 Microsoft Agent Plugin for Mozilla

Description

The Microsoft Agent plugin for Mozilla Firefox allows the ActiveX Controls needed by Microsoft Agent 2.0 to run within certain versions of the Mozilla Firefox web browser. This plugin is based on the original Netscape 7 ActiveX plug-in developed by Netscape Communications Corporation that allows the ActiveX controls needed by Microsoft Agent to be able to run in Firefox (http://hoagiebot.furrynet.com/foxeenet/goodies/index.html#FIREFOX).

API characteristics

ActiveX controls that allow communication between the Microsoft Agent software and Firefox web browser.

Maturity

The Microsoft Agent Plugin for Mozilla is an experimental project and is not met for production purpose.

Licensing / cost

Free

Strengths

- Integration to Web browser.

Limitations

- DoubleAgent relies on the discontinued Microsoft Agent software.
- The plugin is limited to specific versions of the Mozilla Firefox browser.
- Limited to Microsoft Windows Operation System and ActiveX controls.

6.1.2 ClippyJS

Table 3: Score Summary of ClippyJS.

Score Summary	
Features coverage for Avatar dev.	★★ (Front-End Interface only)
Ease of integration	★★★★★
API Availability and documentation	★★★★★
Interactions with third-party applications	★★★★★
Web integration possibility	★★★★★
Maturity / Long time support	★★★★★
Dependent of external proprietary services (server farm)	no

Description

Clippy.js is a full JavaScript implementation of Microsoft Agent, ready to be embedded in any website. ClippyJS offers the simpler possible Web integration of a virtual agent (https://www.smore.com/clippy-js).

Agents are composed of a PNG file with multiple image sequences that represents the frames of each animation related to agent action (greating, searching, pointing left…). The animations are configured from jQuery CSS effects by showing or looping part of the PNG image in appropriate order.

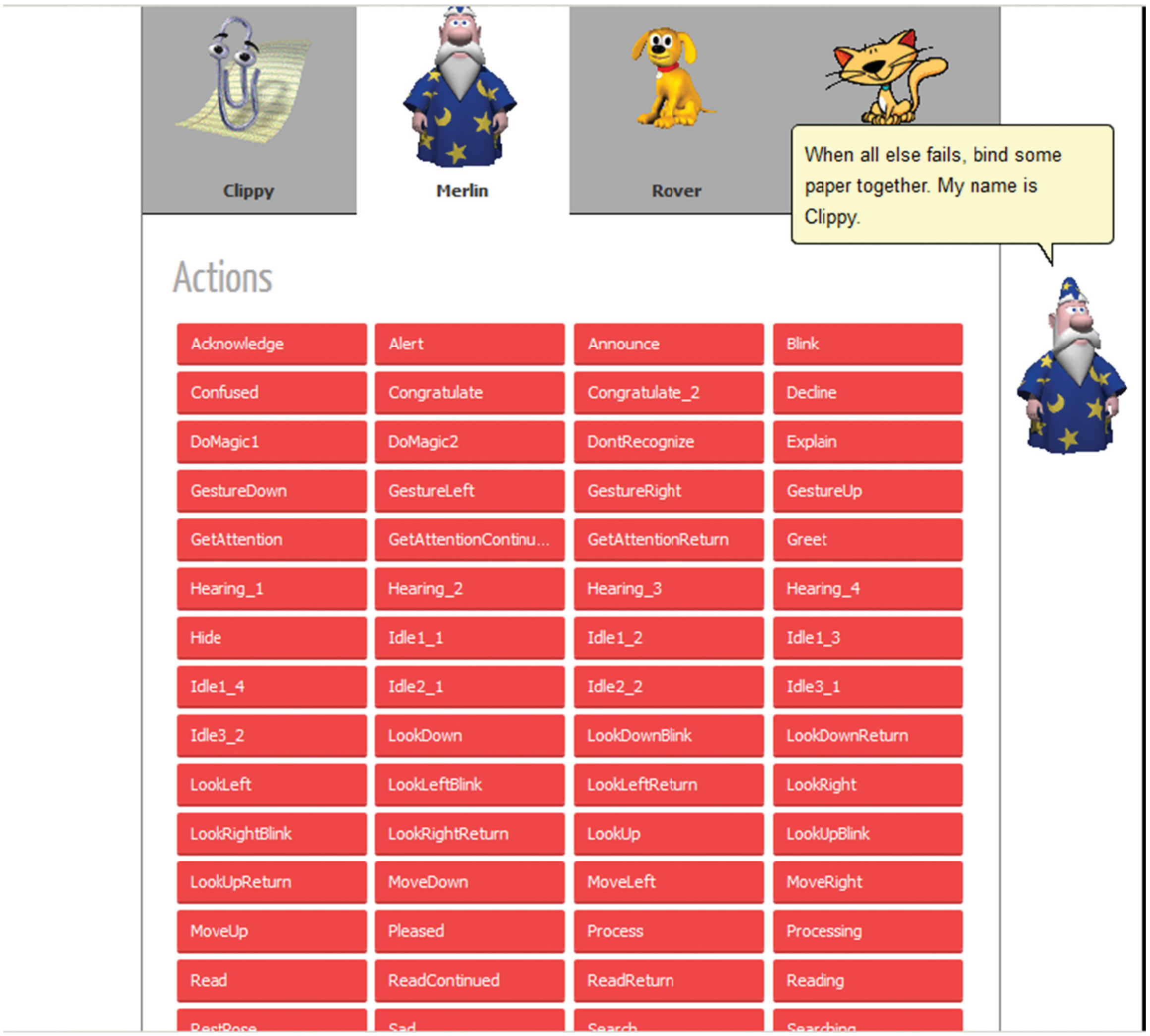

When all else fails, bind some paper together. My name is Clippy.

Figure 4: ClippyJS Available Animations for Merlin Agent

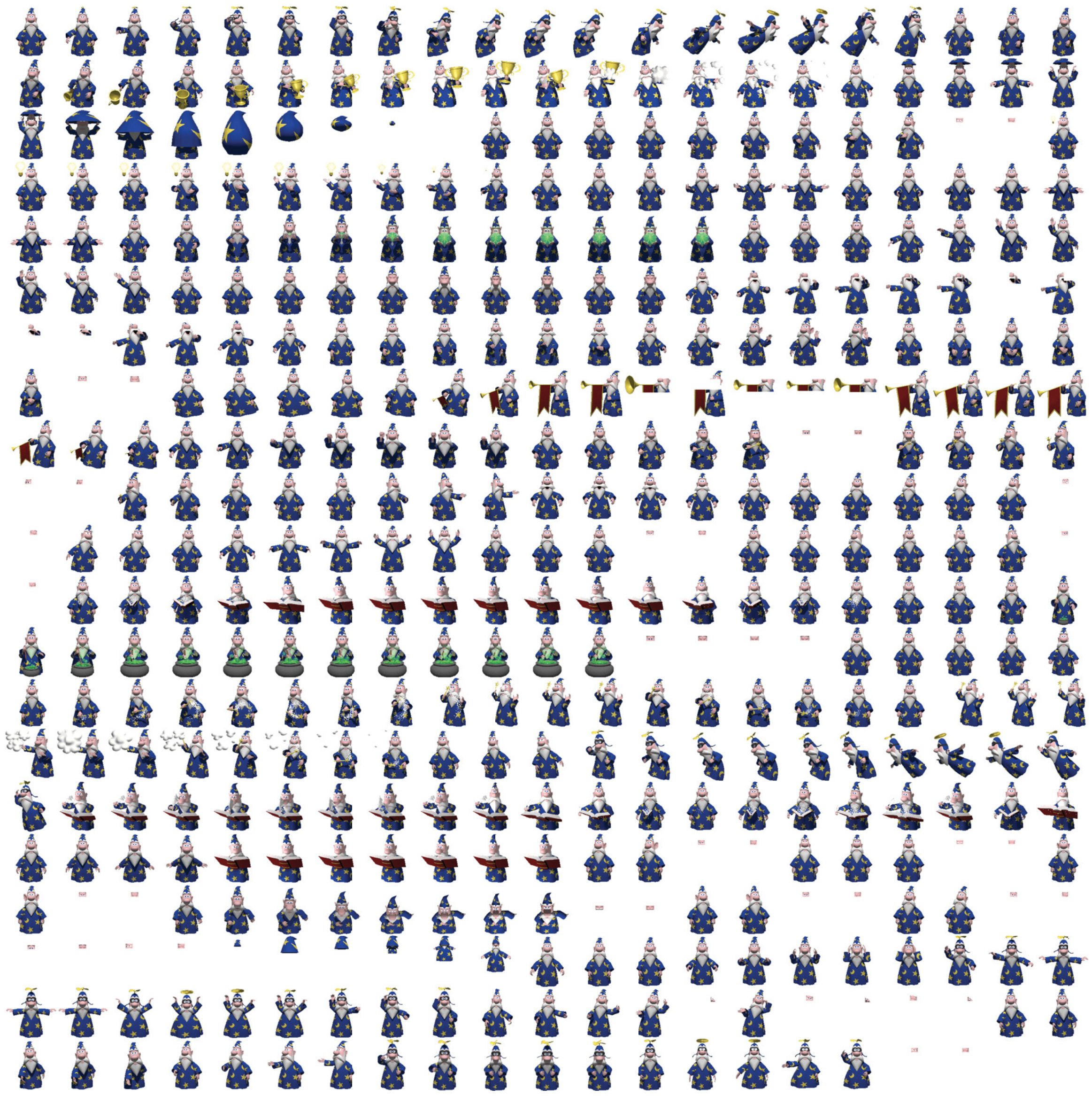


Figure 5: ClippyJS PNG image Map for Animation Purpose

API characteristics

- JavaScript jQuery library, HTML, CSS.
- Agents are composed of multiple sets of animations (PNG frames sequences).
- Animations are launched trough simple JavaScript calls. For example, it comes with 73 animations for Merlin character (see figure above).
- Agent ”speaks" to the user in a speech balloon.
- Agent is displayed on top of other Web page layers and is moveable.

Development

ClippysJS is developed with standard Web technologies and can be extended, personalized and connected to any technologies for any purposes.

Integration with Web services

ClippysJS communication with Web services and actions call could be triggered both ways through websocket (Socket.IO, http://socket.io). This could allow real time animation on actions such as user's speech command recognition and text-to-speech output to the user.

Maturity

Base on mature and simple Web standards.

Licensing / cost

Free

Strengths

- Lightweight and simple Web integration.
- Easy integration and customization.
- Possible integration of real time communication with web services.
- No software or environment dependencies.

Limitations

- This script only generate front-end avatar representation, it must rely on external software / web service for text-to-speech, voice command and speech recognition.
- ClippyJS does not include phonemes (Lip-Synching) animations.

6.1.2.1 Custom jQuery alternative

Base on the same technologies and principles of ClippysJS some interesting Avatar rendering alternatives could be explored.

- Create our own PNG image map by making a photo shoot of a real person.
- Create a set of static images representing the status of the task.
- Create video sequence from a real person and attach them to action (similar to viClone).
- Create Flash embed application with animated sequence.

6.1.3 Need to know about Chatbots technologies ambiguity

Chatbots uses third-party applications or static images for avatar representation. They do not offer avatar services as needed in IVAC.

A chatter robot is a type of conversational agent, a computer program designed to simulate an intelligent conversation with one or more human users via auditory or textual methods.

For example, ALICEBOT use OddCast Flash Widget for avatar representation.

6.2 Commercial

6.2.1 Guil3D - Virtual Assistant Denise

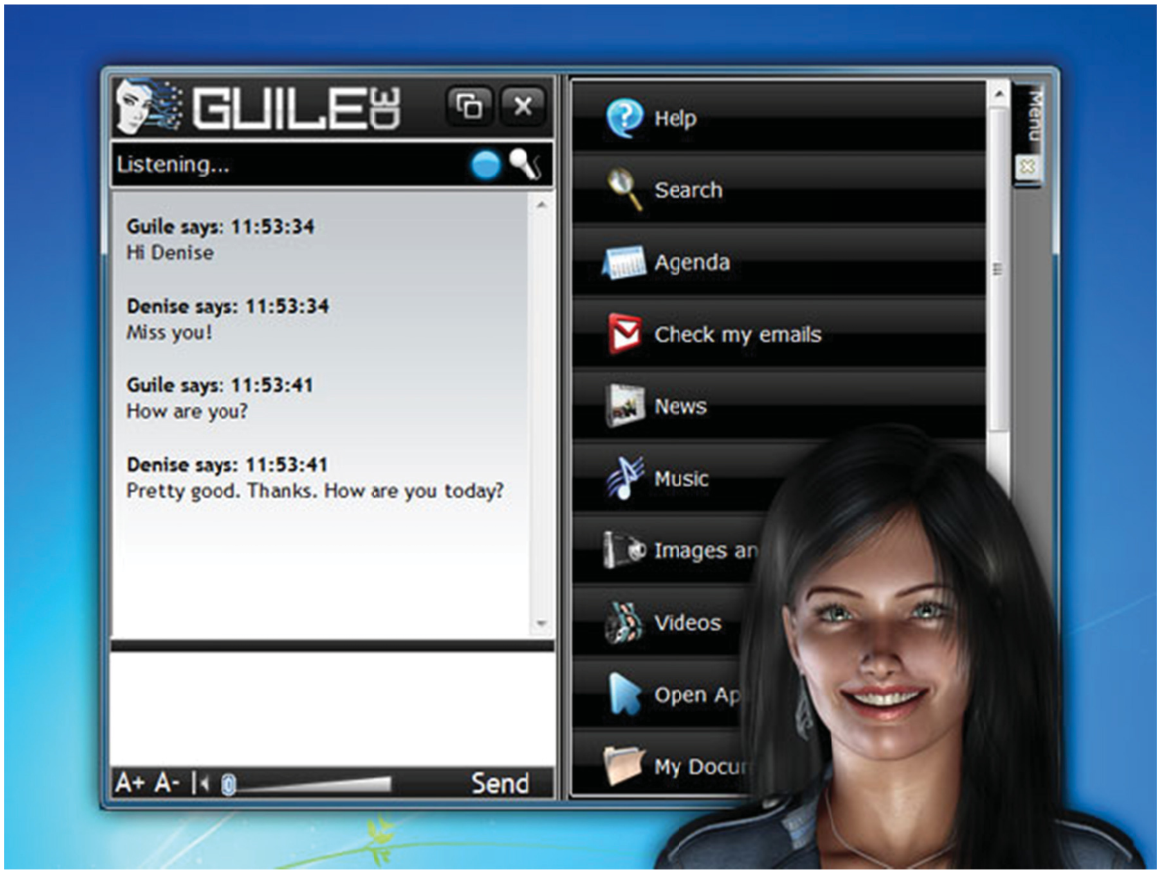
Table 4: Score Summary of Denise Virtual Assistant.

Score Summary	
Features coverage for Avatar dev.	★★★★★
Ease of integration	★★★
API Availability and documentation	★★★
Interactions with third-party applications	★★★
Web integration possibility	★
Maturity / Long time support	★★★
Dependent of external proprietary services (server farm)	no

Description

Denise is a Virtual Assistant Windows desktop software with some artificial intelligence capabilities. Denise is a full feature application whose main function is to assist users in human-computer interaction. Denise can search the web, explore and play multimedia files, read and answer e-mails, schedule and remind appointments, get and read RSS feeds aloud, run computer applications, speak aloud Skype messages and has several other features (https://guile3d.com/en/product/).

The software has built-in natural language understanding capability, thus it understands various commands uttered by the user. Denise mimics a real human being, using facial recognition (said to be available soon), text-to-speech and speech recognition technology to identify users, understand speech questions, search for the best answers and speak aloud important information and search results. The artificial intelligence capability of Denise can learn, adapt, be improved and modified by itself or by users to achieve specific tasks. Denise comes with a very natural synthetized English voice to convert any text to speech in an almost female human like voice (by using the Nuance's Dragon technology) and also comes with an English Speech Recognition engine for Speech Recognition. There is also the option to install Denise having Speech Recognition and TTS voices for French, German, Italian and Spanish (also based on the Nuance's Dragon technology).



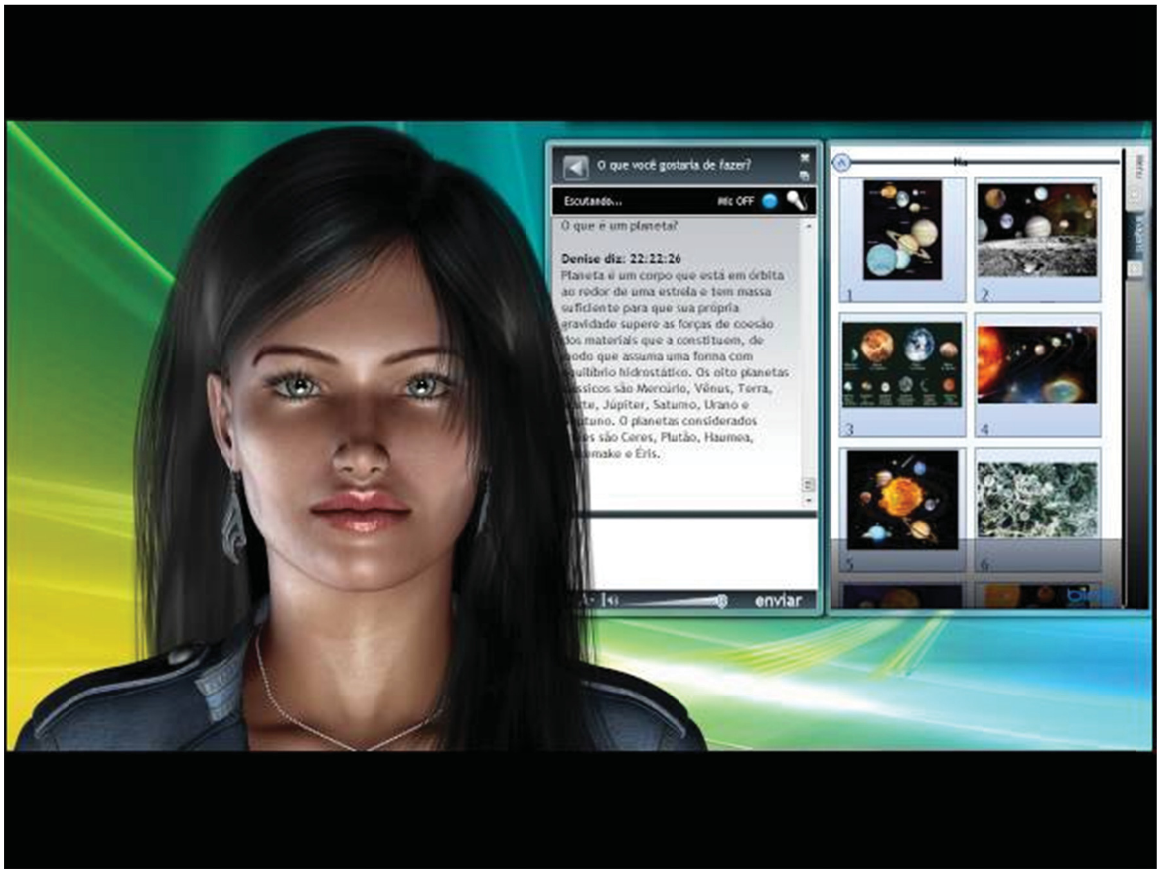
Guile says: 11:53:34
Hi Denise

Denise says: 11:53:34
Miss you!

Guile says: 11:53:41
How are you?

Denise says: 11:53:41
Pretty good. Thanks. How are you today?

Figure 6: Denise Interface



O que é um planeta?

Denise diz: 22:22:26
Planeta é um corpo que está em órbita ao redor de uma estrela e tem massa suficiente para que sua própria gravidade supere as forças de coesão dos materiais que a constituem, de odo que assuma uma forma com uilíbrio hidrostático. Os oito planetas ssicos são Mercúrio, Vênus, Terra, rte, Júpiter, Saturno, Urano e tuno. O planetas considerados es são Ceres, Plutão, Haumea, emake e Éris.

Figure 7: Denise Image Search Results

API characteristics

- Real-time graphical interface;
- Speech recognition in command mode;
- High quality text to speech with a US English female voice;
- AIML (http://en.wikipedia.org/wiki/AIML) custom tags;
- Artificial Intelligence Engine;
- Artificial Intelligence Editor;
- Use of Custom AIML sets in any language;
- Possibility to use synthesized TTS voices in multiple languages
- Editable English Commands AIML sets;
- Speech recognition in dictation ;
- Possibility to build and customize voice commands for better speech recognition;
- A partially available SDK;
- E-mail / Forum Support;
- E-mail / Forum / System Analysts Support;
- Integration with 3rd part application;

- Microsoft Kinects Interface.

Development

SDK is partially available.

Currently available tools:

- AIML custom tags SDK (to customize Denise's artificial intelligence);
- AIML brain editor SDK (to create and edit new knowledge bases);
- Kiosk Studio (to make PowerPoint like presentations).

Maturity

Guil3D - Virtual Assistant Denise out of the box functionalities seems to be mature for desktop purposes. The project seems to be a one man project from Guile Lindroth.

Licensing / cost

- Denise Platinum : $ 120.00;
- Denise Business : $ 820.00 (includes all features).

Strengths

- Includes all major features requested by an avatar, such as voice recognition and commands, text-to-speech and 3D human representation.

Limitations

- Only Desktop client integration is available in the current version.
  - Web integration is possible trough external Web service such as Pandorabots Chatbot Hosting Service (http://www.pandorabots.com).
- Guile 3D Studio uses a proprietary Web Browser application to surf the web.
- Denise interactions with Web HTML / JavaScript components seem not possible or limited to its own Web browser and navigation purpose (http://guile3d.com/en/help/manual/webbrowser.html);
- Ask Denise to search in a custom set of data through a Web service or external desktop application seems not possible or not documented.
- Limited to Microsoft Windows Operation System.
- Denise requires a computer with full access to the Internet to install and run Virtual Assistant Denise.

6.2.2 OddCast – Avatar Studio & SitePal

Table 5: Score Summary of Avatar Studio.

Score Summary

Score Summary	
Features coverage for Avatar dev.	★★★★★
Ease of integration	★★★
API Availability and documentation	★★★★
Interactions with third-party applications	★★★★
Web integration possibility	★★★★★
Maturity / Long time support	★★★★★
Dependent of external proprietary services (server farm)	Yes

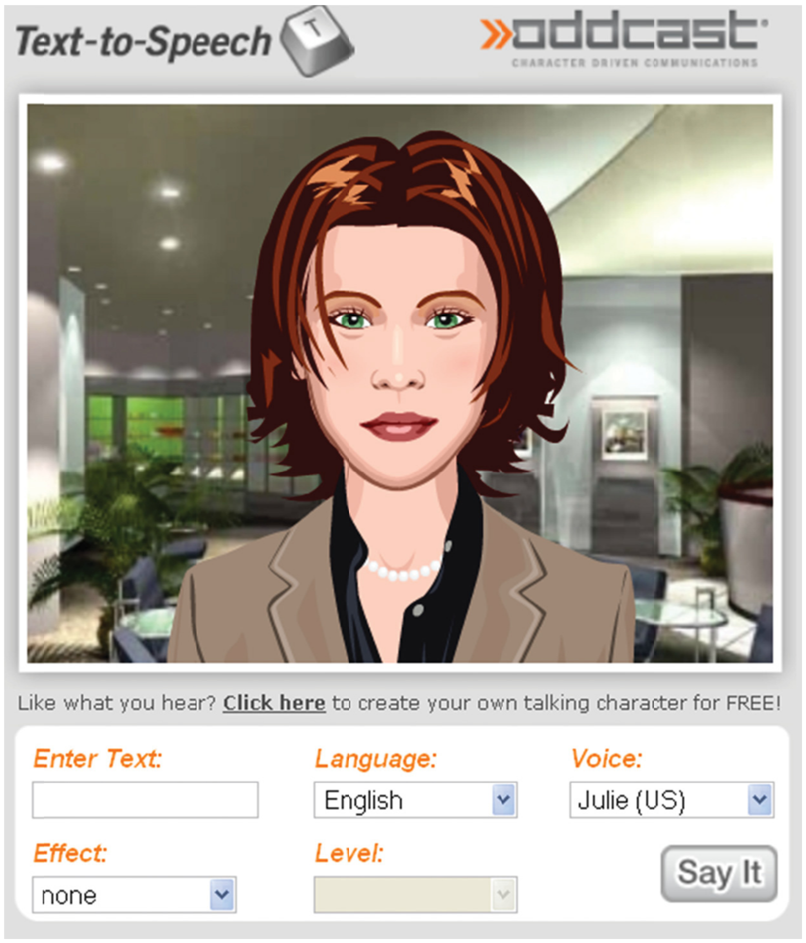
Description

The Oddcast suite of technologies represents a powerful and scalable platform for the deployment, management, and tracking of multi-media campaigns. Built on a Flash-based ASP model, the platform provides stutter-free experiences for even low bandwidth users, requires no special plug-ins, provides precise real-time reporting, and supports an unlimited number of concurrent users by way of their application server farm (http://www.oddcast.com/).

API characteristics

- Character

Illustrated Characters: Fully animated and customizable characters. For example:



Like what you hear? Click here to create your own talking character for FREE!

Enter Text:
Language: English
Voice: Julie (US)
Effect: none
Level:

Figure 8: Avatar built with Oddcast Avatar Studio

3D PhotoFace functionality that allows transforming any photo into a 3D avatar; With 3D PhotoFace, users can upload a head shot to create a fully rendered 3D face. Then, they can easily manipulate the features: age, emotional expression, makeup and accessories. Users can also alter nose, eyes, mouth, and more, for example:

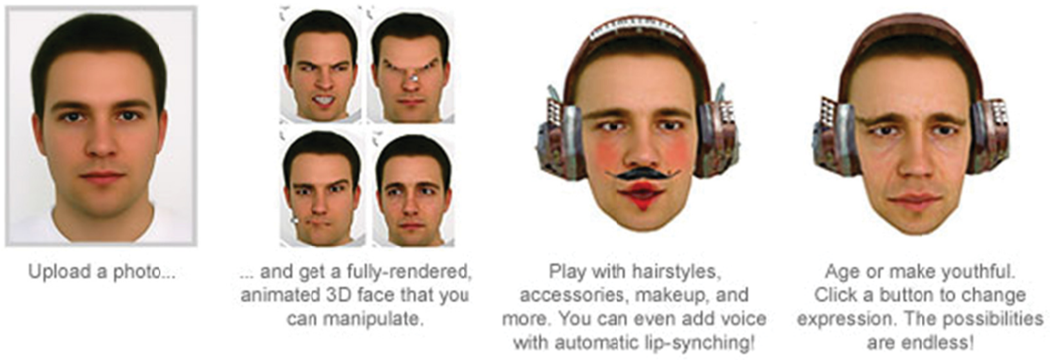

Upload a photo...
... and get a fully-rendered, animated 3D face that you can manipulate.
Play with hairstyles, accessories, makeup, and more. You can even add voice with automatic lip-synching!
Age or make youthful. Click a button to change expression. The possibilities are endless!

Figure 9: Full Body 3D Avatars

- Full Body 3D Avatars functionality allows creating fully customizable 3D characters;
- Voice and Audio features:
  - User's voice can be integrated with the avatar;
  - Text-to-Speech functionality allows avatar characters to speak any text dynamically, in real time with accurate lip-synching. Special effects can also be applied to the audio, including emotive cues and expressions. This functionality comes as an alternative or compliment to a real human voices, and is often used in conjunction with our Record by Phone and Record by Mic technologies. TTS is available in 25 languages, including translation from and back to English or other languages.

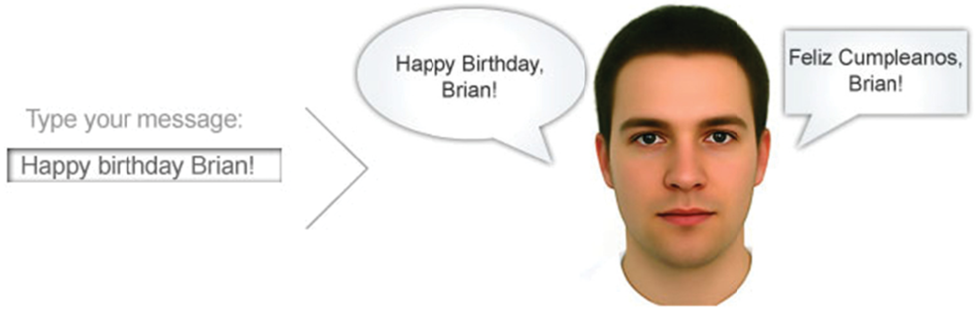


Figure 10: Avatar Happy Face

  - TTS Translator allows for text in English and certain other languages to be transformed into spoken audio for all of the above languages, or for reverse-translation for those languages.
  - Powerful emotive cues enable users to customize the delivery of their text, controlling emotional content (laughing, crying, and sneezing) and character behavior (when combined with one of our character technologies).
  - Users can manipulate the speed, pitch, and tone of their audio playback, and can also add special effects (robotic voices).
  - No need for any additional plug-in for authoring or experiencing TTS speech, just Flash or JavaScript capability.
  - APIs can be made available to developers.
    Voice Recognition enables an application to respond according to users' spoken commands. Users can interact with applications in conversational format, or direct the application's goals and behaviors.
    Head Tracking enables an application to respond according to users' facial features and movements. Head Tracking technology tracks the face, nose, eyes, mouth, and other features of the face using only a webcam and flash.

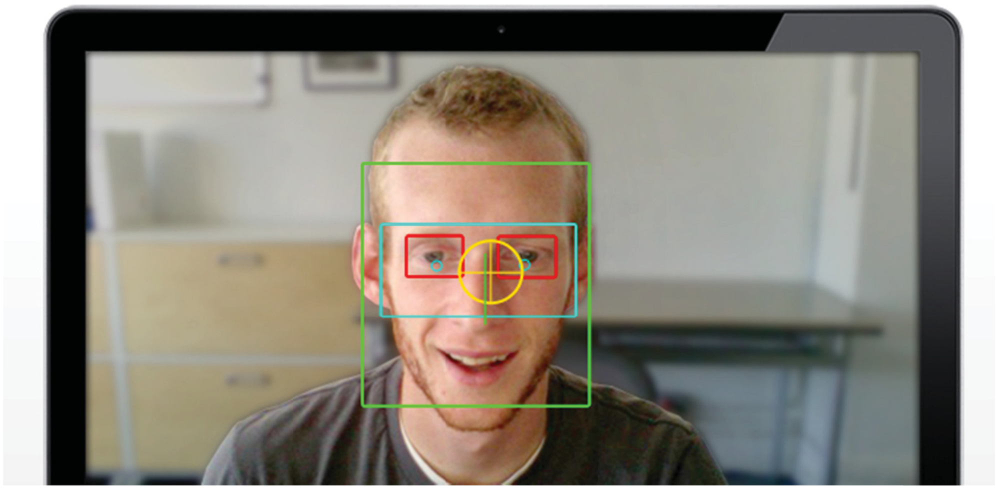


Figure 11: Avatar

Developer API Packages

OddCast offers well documented API for all suite modules.

Technology:

- Adobe Flash SWF file, Action Script 3: widget & avatar container;
- JavaScript: user control, interaction and configuration;
- HTML: front end embedded integration.
  Text to Speech API:
  - Allows connecting the Oddcast TTS utility to any dynamic web application;
  - Supports both Flash & JavaScript;
  - Available both as Server-side API and Flash component API;
  - Supports Mobile implementation via Server-to-Server API.
- Voice Toolkit API:
  Voice-recording services and telephony capability are implemented and executed on the Oddcast server farm, where we have allocated massively-scalable computing capacity capable of serving millions of requests per day. By taking advantage of the economy of scale embodied by our infrastructure, Oddcast is able to offer this functionality as a cloud-computing service at an affordable price.
- Audio Engine API: Lets users create and mix audio;
- AV Capture API: Capture and share entire app experiences.
  The AV Capture Module supports an ActionScript 3.0 API;
- Text to Sing API: Build your own app with our Text to Sing API;
- Face Detection API: Technology that finds and tracks your head.
  Oddcast provides a client side faceless software object (agent), to be loaded by licensee host application from Oddcast servers. The agent can locate and track a face if it appears within the Flash player webcam feed.

The agent supports an ActionScript API that allows host app to specify the following parameters:

- Webcam dimensions
- Stage reference
- Webcam analysis rate
- The agent sends periodic events (at the specified rate) to continuously inform the host app of the location of the face in the frame. Facial points tracked include: eyes, nose, mouth, chin and several others.
- Avatar Studio: Avatars that you create and control

  Oddcast Character Studio provides companies with an easy-to-use authoring tool to create and embed customized, speaking animated characters within HTML pages, ad banners and Flash movies.

  With an intuitive online editor, Character Studio lets you create and update characters, backgrounds and audio messages. HTML, JavaScript and ActionScript programmers can utilize APIs to create advanced interactions with users. Developers can use APIs to integrate the avatars into unique business logics or databases and generate dynamic and personalized speech on the fly.

  Character Studio is not designed for end-users to create avatars, but rather for developers and businesses to integrate avatars into their user experiences. To enable end-users to create white label avatars on your site, the Avatar Framework is required.

- Avatar Framework API: Avatars for community site members

Maturity

OddCast Suite is a proven and mature product.

Deployment Options & Purchasing

Following consultation, OddCast can recommend the best technology and customization options and provide a formal quote.

Annual licensing of Character Studio starts at $5,000.

Text to Speech functionality can be incorporated into any Oddcast custom application. Text to Speech is also available to developers building their own applications, and APIs are available to integrate the module with third-party applications.

Strengths

- Full web technologies and web browser integration capabilities and compatibilities.
- Stunning visual Avatar 2D / 3D representation.
- Available and well documented API for each module.
- OddCast modules can be integrated with other applications.
- Integration on all major OS:
  - Windows 98, ME, NT, 2000, XP, Vista.
  - Mac OS 8.1 & up or OS X 10.2 & up.

▫ Linux (not officially support playback)

Limitations

- All OddCast services are implemented and executed on the OddCast external server farm. Internet access is required.

6.2.3 viClone Personalized Assistance

Table 6: Score Summary of viClone.

Score Summary	
Features coverage for Avatar dev.	★★★★
Ease of integration	★★★
API Availability and documentation	★
Interactions with third-party applications	★★
Web integration possibility	★★★★★
Maturity / Long time support	★★★
Dependent of external proprietary services (server farm)	yes

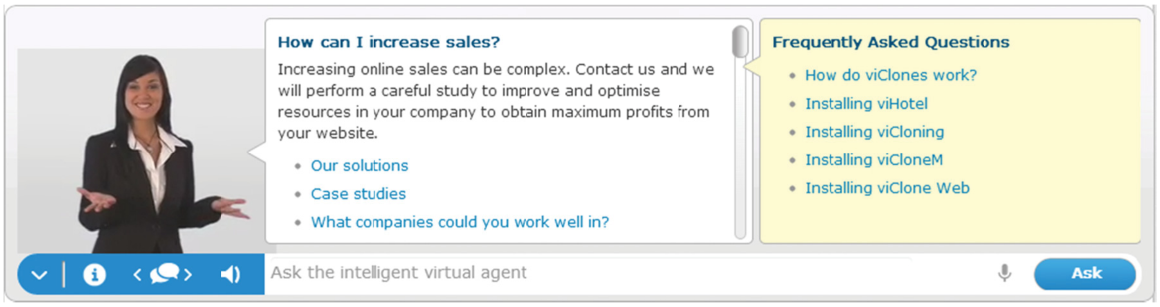
Description

viClone is an Intelligent Virtual Agent using natural language processing and artificial intelligence, capable of providing assistance as if it were another employee in your company.

viClone offer personalized customer assistance and support in real time. It can attend to all your critical mass of current and potential customers at the same time, in several languages, allowing less common questions and issues to be handled by a small amount of staff.

viClone uses video sequences combine with text-to-speech to create realistic Web base Avatar.

viClone Avatar chat box



How can I increase sales?

Increasing online sales can be complex. Contact us and we will perform a careful study to improve and optimise resources in your company to obtain maximum profits from your website.

- Our solutions
- Case studies
- What companies could you work well in?

Frequently Asked Questions

- How do viClones work?
- Installing viHotel
- Installing viCloning
- Installing viCloneM
- Installing viClone Web

Figure 12: viClone Avatar Chat Box[1]

API characteristics

- Natural Language Processing
- HTML5 technology which displays correctly from IE, Firefox, Chrome, Safari and Opera on your computer or any iOS or Android based mobile device.
- Multiple languages available for client chat.
- Statistical customer reports.

Developer API Packages

No documented API.

Strengths

- Full web technologies and web browser integration capabilities and compatibilities.
- Realistic visual human representation (video).

Limitations

- Annual licensing is high.
- All viClone services are implemented and executed on their external server farm.

[1] http://www.viclone.net

7 Integration Framework

7.1 Challenges

Developing the iVAC requires a well-designed architecture that supports the integration of different applications, systems, services, libraries and tools; the management of different kind of data stores, delivers great quality of service and provides, in nature, the capabilities of accommodating heterogeneous components. Moreover, the system will potentially need to interact with Big Data processing system, provide online machine learning capabilities, interact with Speech Recognition modules, be able to interface both ISTIP and VOILA and must be scalable to support multiple users and requests at the same time.

Then, the main goal is to provide integration architecture that satisfies not only today needs of the iVAC system but that can easily integrate new applications, systems, services, modules or software libraries. Such integration architecture can be seen as a blueprint that guides the development and the evolution of the iVAC system.

Over the years, different integration architecture and patterns are emerged in the Enterprise Application Integration landscape. The main fundamental integration architecture variants are:

- **Multi-tier Distributed architecture:** generally known as n-tier system architecture, is a highly scalable and available distributed architecture in which the presentation, the business processing and the data management are logically separated. For example, an application that uses middleware to service data requests between a user and a database employs multi-tier architecture. The most widespread use of multi-tier architecture is the three-tier architecture;
- **Point-to-Point:** a collection of independent systems, which are connected through a network;
- **Hub-and-Spoke:** this integration architecture represents a further stage in the evolution of application and system integration, in which a central hub takes over responsibility for communications;
- **Pipeline:** In pipeline architecture, independent systems along the value-added chain are integrated using a message bus. The bus capability is the result of interfaces to the central bus being installed in a distributed manner through the communication network, which gives applications local access to a bus interface. Different applications are integrated to form a functioning whole by means of distributed and independent service calls that are orchestrated through an ESB and, if necessary, a process engine;
- **Service integration:** in the services integration, integration is made by putting a service layer as a central point where the applications are connected. Each application functionality is externalized as a service that can be consumed by others;

- **Service-Oriented Architecture:** Service-orientation presents an ideal vision of a world in which resources are cleanly partitioned and consistently represented. When applied to IT architecture, service-orientation establishes a universal model in which automation logic and even business logic conform to this vision. This model applies equally to a task, a solution, an enterprise, a community, and beyond.

In this section, we will concentrate only on three of them, «n-tier distributed architecture», «Service Integration approach» and «SOA Integration approach» and then make a recommendation. Because they are not only the most appropriate or relevant for our case but also modern applications integration architecture follow one of them or a mix of the then.

7.2 Proposed Integration Approaches

7.2.1 Multi-tier Distributed Architecture

This approach involves breaking the system in different tier that can be physically distributed over the network. By breaking up an application into tiers, developers only have to modify or add a specific layer, rather than have to rewrite the entire application over. Generally, such architecture contains a presentation tier, a business or data access tier, and a data tier. Each tier can be clustered in order to bring more processing power to the overall system.

This king of architecture can scale either horizontally (by adding more servers) or vertically (by adding more resources – memory, CPU[2]) or both. It can serve well the needs of the iVAC system.

The diagrams below depict this approach:

[2] CPU= Central Processing Unit

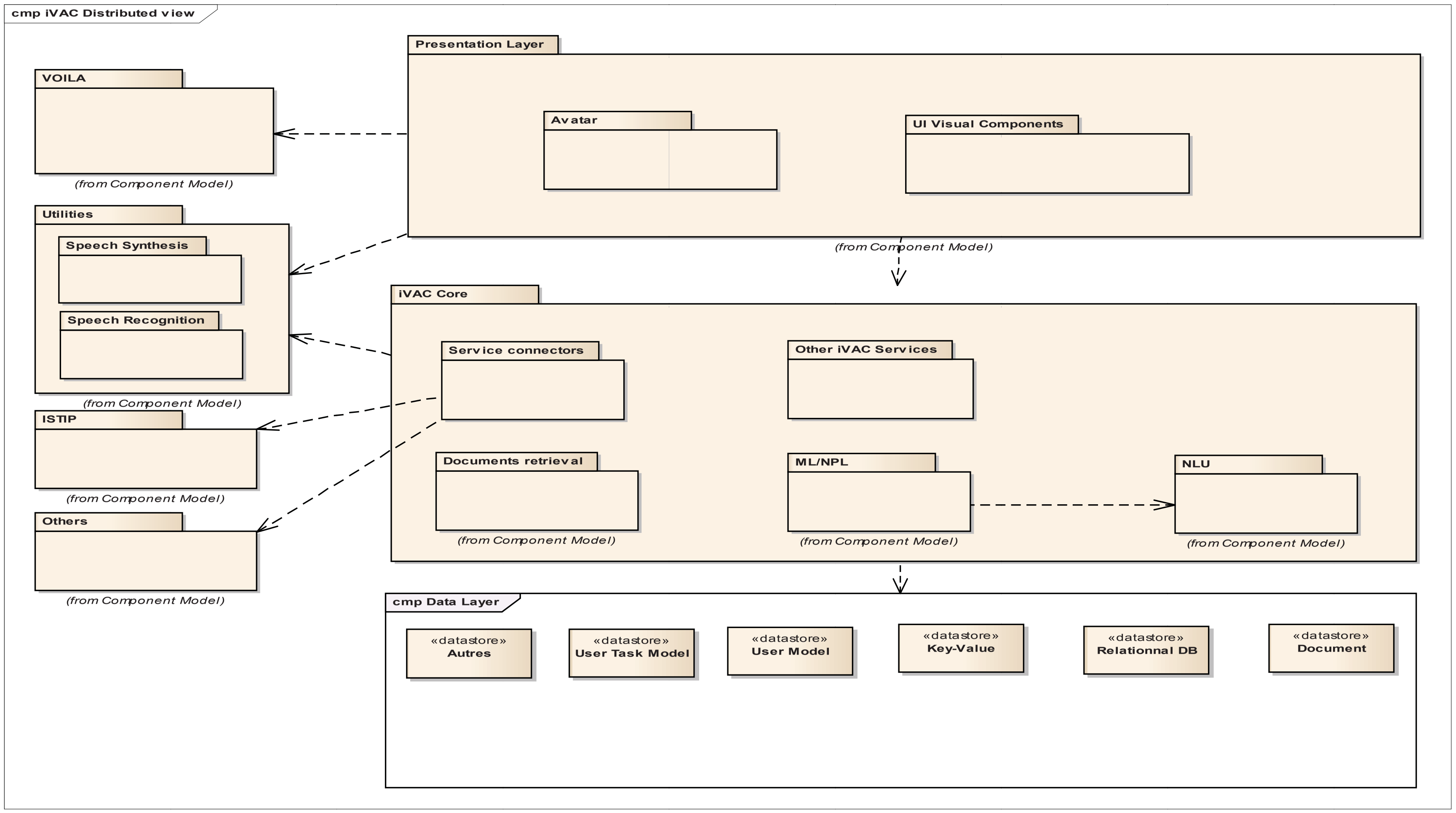


Figure 13: Multi-layer architecture

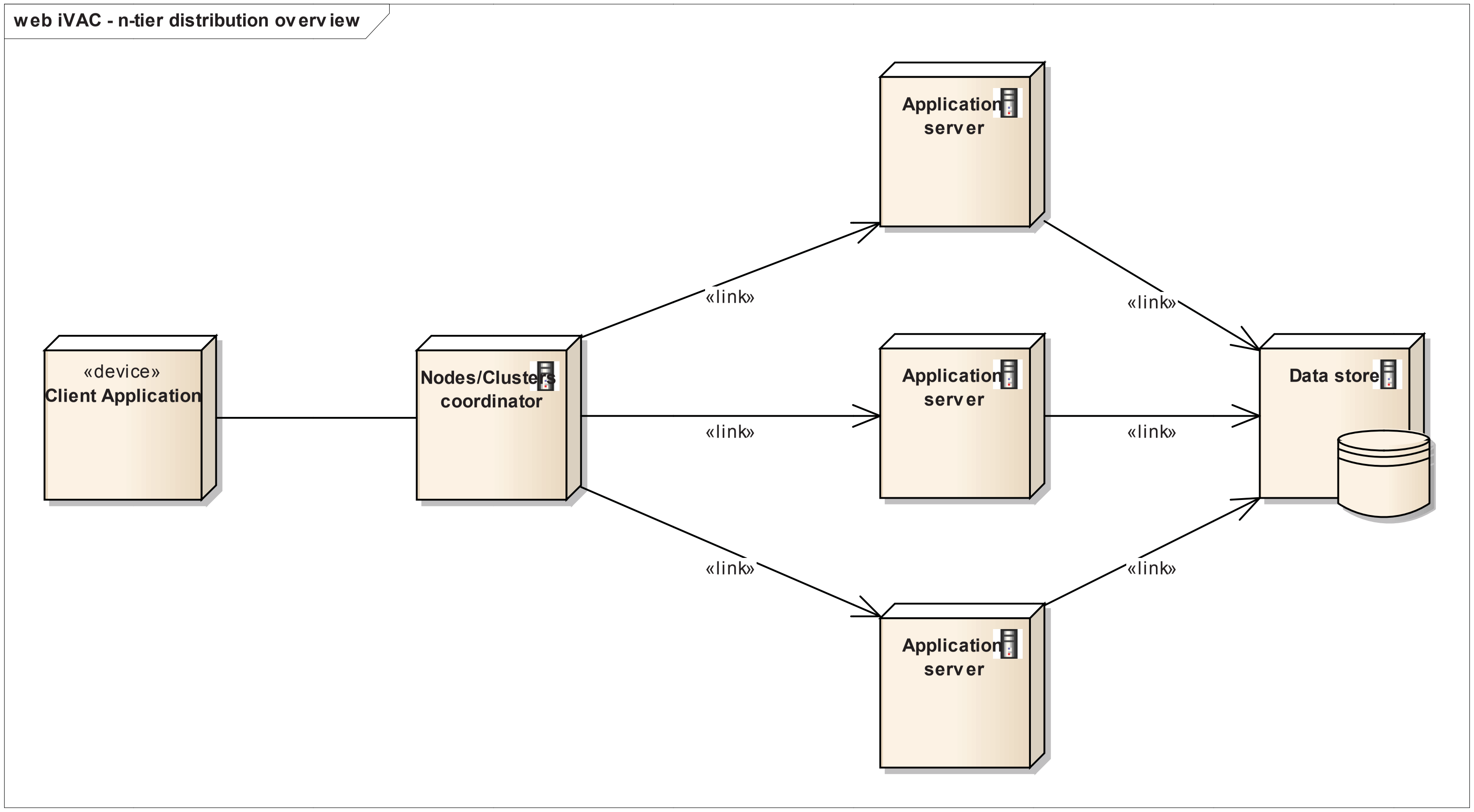


Figure 14: n-tiers distribution overview

7.2.1.1 Description

As depicted in the diagram above, the multi-tier architecture approach shows a typical layered architecture in which there are:

- **Presentation layer:** regrouping all UI components, this layer is the interface between the client (the user, the analyst) and the iVAC system. This layer may interact with the VOiiLA services. It can be designed according the MVC design pattern;
- **Business layer:** all the iVAC core business functionalities and capabilities are part of this layer. The presentation layer delegates every request processing to this layer. Business components on this layer may require interaction with ISTIP services in order to process correctly a request. If so, facilities provided by the ISTIP services coupled with some well-chosen design patterns will guide such an integration;
- **Data layer:** this layer contains all the data stores used to deliver the iVAC system. There can be any type: relational, Key-Value, Document and so on.

One of the most powerful features of a layered architecture is its distribution capability. Indeed necessary, each layer can be deployed on different servers and the servers can be clustered in order to bring more processing power, scalability and great availability to the system. The diagram in Figure 14 is provided as an example of such distribution.

7.2.2 Service Integration Approach

This approach involves the integration of applications through a service layer where services are aggregated, composed and consumed as needed. In «Service Integration approach», the applications are not directly connected to each other as it would be the case in an approach like «Point-to-Point integration» but interact through the service layer. When needed, an application calls a service[3] by sending a request through a well-defined interface. The service processes the request and decides whether to respond by itself or to invoke other services to make more processing in order to properly response to the request. When the application's interfaces are different and / or incompatible with each other and then no Web service is available, service adapters are developed to allow the integration.

Today's major development platforms, either the «.Net» platform or Java and particularly Java Enterprise Edition (Java EE), provide all the necessary mechanisms to support such integration architecture. In this architecture, the applications rely on the application server on which it is deployed, the middleware (the application server for Java EE or IIS /WAS/AppFabric and WCF for the «. NET Framework») to provide high availability and great quality of service and low level services, like:

- Transaction management;

[3] The term service here does not mean Web service rather any piece of software that delivers a value to the user. It can be implemented as a component (component-based, Java EE, Net) or as a Web service.

- Multithreading and concurrency management;
- Location transparency;
- Security management;
- Synchronous and Asynchronous mechanism;
- Etc.

Using some well-chosen design patterns improve the overall quality of such architecture. Even if either Java EE or «. Net» platform can be used indifferently to deliver any system based on this integration approach. However, for a «Service Integration approach», we recommend using Java EE to build the iVAC. The reasons for this recommendation are, among others, the following:

- DRDC dispose a large infrastructure based on Java EE;
- It is required to make maximum use of DRDC ISTIP services;
- It is required to integrate to VOiiLA services; most VOiiLA services are Java-based;
- the nature of the research that conducted by the DRDC and the availability of APIs commonly used in the context of the so-called research;
- expertise of the development team already in place
- etc.

The diagram below depicts this approach:

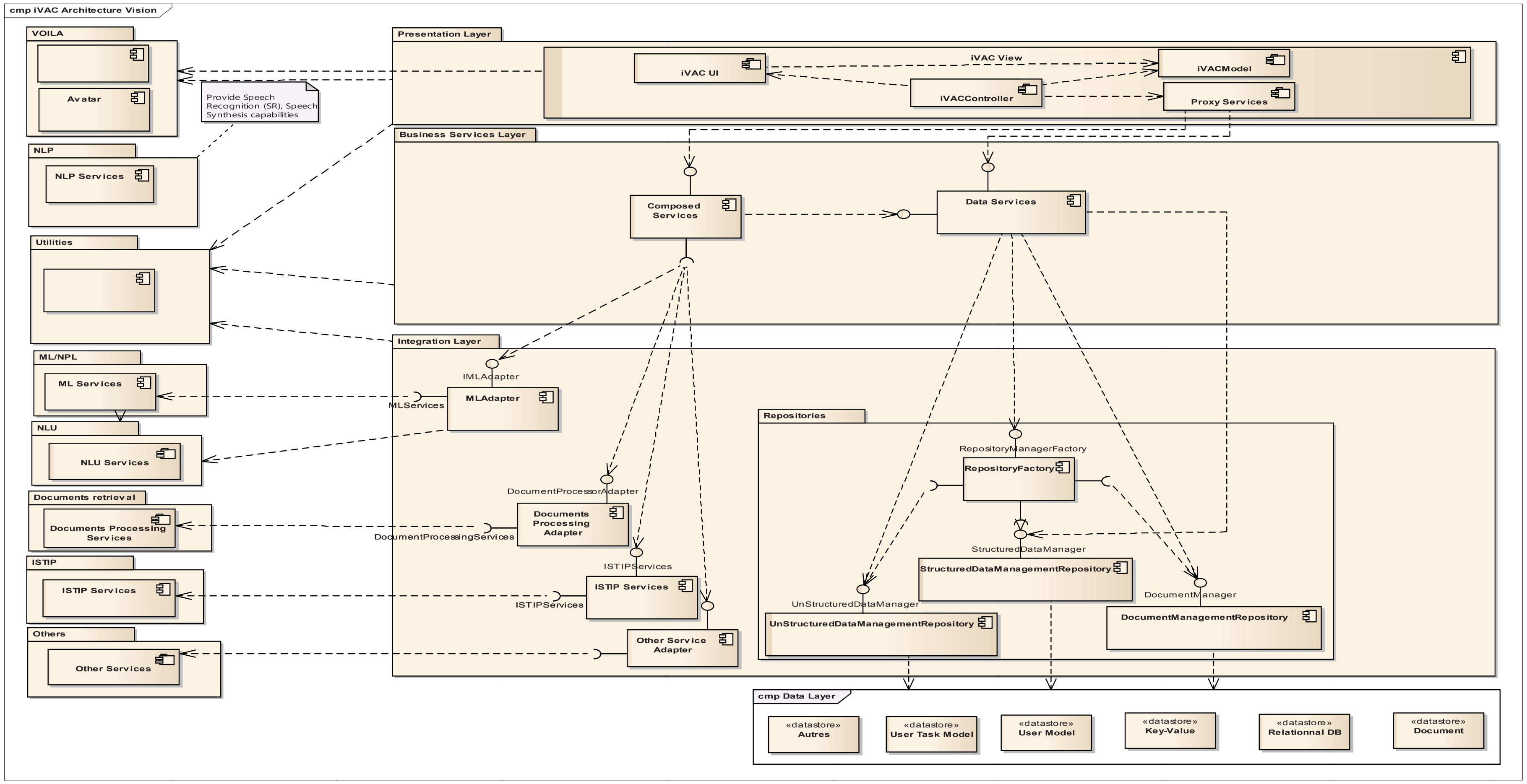


Figure 15: Service-Based Integration Approach (Service-Based SOA)

7.2.2.1 Description

As depicted in the diagram above, the service-based integration approach shows a typical layered architecture in which there are:

- **Presentation layer:** designed according the MVC design pattern, this layer is the interface between the client (the user, the analyst) and the iVAC system. This layer interacts with the VOiiLA services;
- **Business layer:** the business comprises business services that can be:
  - **Simple business services**: service that exposes a simple functionality, algorithm. This king of service doesn't interact with other services the deliver its capabilities. There are very reusable;
  - **Data services:** provide data manipulation services known as CRUD operation (create, read, update and delete). To provide the services, data services rely on services provided by the repositories in the integration layer. There are very reusable;
  - **Composed services:** composed services also known as task services use service composition to deliver a business service. To deliver its capabilities, a composed service can use data services, simple business even other composed service. Doing that hides service complexity for the service consumer by providing a simple service interface to get access to the business service; that span provide a simple service interface. There are less reusable;
  - **Process services:** process services are process-driven services also known as long running services. There are composed by other services (data services, simple business services, composed services even other process services). There are less reusable.
- **Integration layer**: generally, it is where integration happens. In this layer we find:
  - **Data integration:** integration different data sources;
  - **Application integration:** integration with other applications;
  - **Service integration:** integration with services provided by other applications, departments, and partners and so on. Each library, API and system to be integrated to the iVAC will happen in this layer.

7.2.3 Service Oriented Architecture

This integration approach involves the use of Service-Oriented Architecture (SOA) to deliver the iVAC system. Service-Oriented Architecture is a paradigm for the realization and maintenance of business processes that spam large distributed system. It is based on three major technical concepts: services, interoperability through an Enterprise Service Bus (ESB) and loose coupling[4]. The ESB is the best solution to bridge heterogeneous technologies by leveraging SOA. The best way in which is can accomplish this is to abstract all of the components. SOA is not about technology but it is an end-to-end approach to the IT system landscape as the support function for business processes.

[4] Nicolai M. Josuttis, SOA in Practice: The Art of Distributed System Design, O'Reilly, 2007

Since building the «iVAC» system requires the integration of the different applications, systems, services and some API that provide specific functionalities, working on multiple data sources, using an enterprise Service Bus (ESB) to glue everything together becomes very interesting to deliver a more scalable architecture. In such approach, the ESB is used as the central integration component for service calls. It becomes the backbone of the overall architecture.

The diagram below depicts this approach:

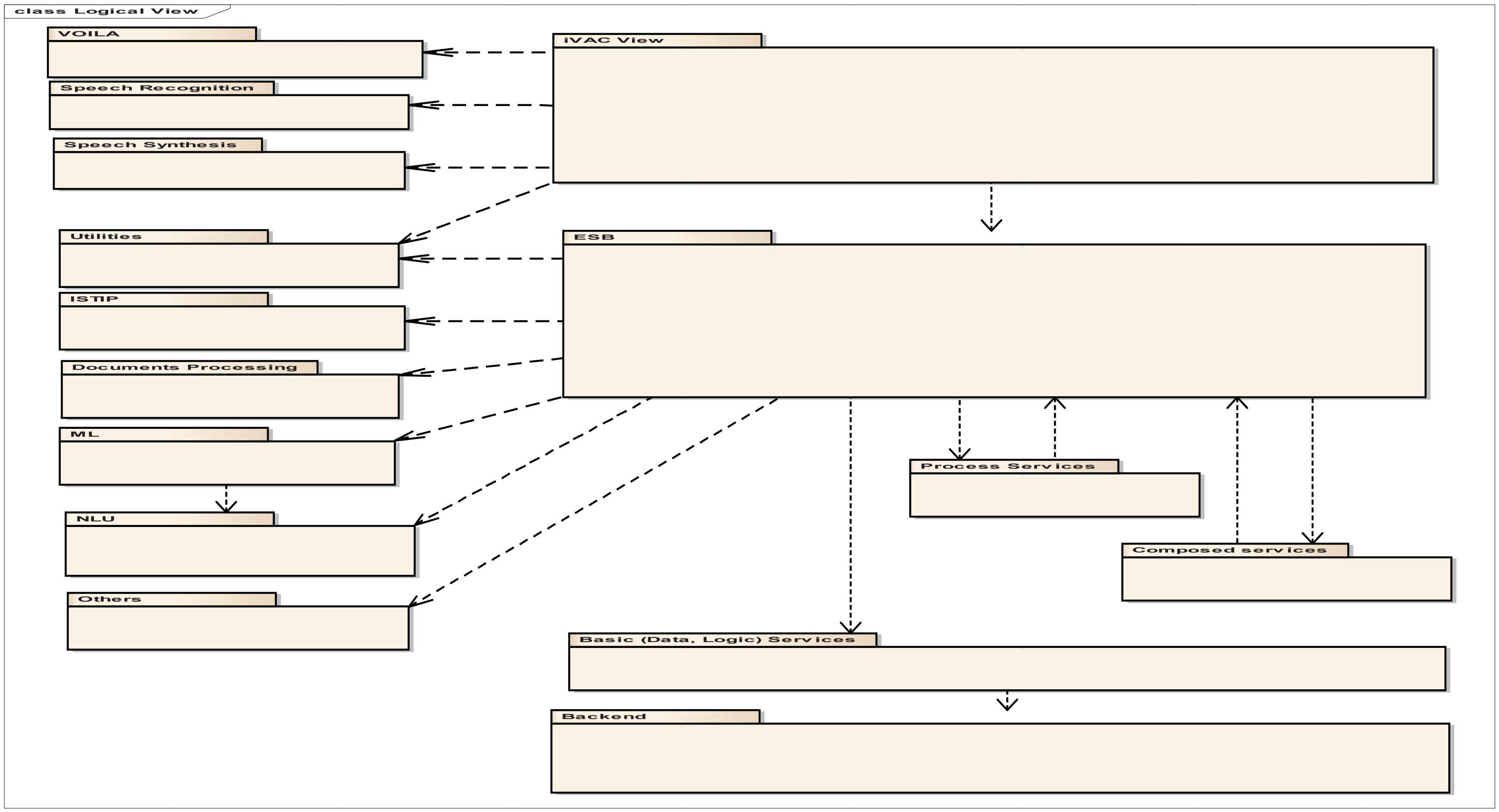


Figure 16: SOA Integration Approach Based on an Enterprise Service Bus (ESB)

7.2.3.1 Description

This architecture shows the same layered architecture as in 1.2.1 where services are grouped by category and intend and are localized in their own layer. The major difference in this architecture is the introduction of the Enterprise Service Bus (ESB) as the central component where integration happens and where services are created, composed, grouped and consumed. Doing that, completely decouple service consumer from service provider. It becomes easier to create new service from existing ones and to integrate legacy systems.

7.3 Candidate Frameworks

7.3.1 ISTIP and VOiiLA

7.3.1.1 ISTIP

Intelligence Science and Technology Integration Platform (ISTIP) is service-oriented architecture integration platform that provide services in intelligence science domain. ISTIP is made of services organized into different logical layers:

- Data services layer: services provided basic data operation (create, read, update, delete «CRUD»);
- Task services (business services): provide business services like «intelligence service: for examples Case-Based reasoning and Rule-based reasoning services»
- Process services: provide more process-driven business services like «Multi Reasoning Inference (MRI)»

Most ISTIP services are developed with J2EE (Java 2 Enterprise Edition) technologies and are deployed on several version of the JBoss Application Server (version 5 and 7). ISTIP is connected to different data stores to fully deliver its capabilities.

7.3.1.2 VOiiLA

The core VOiiLA architecture is based on the «Widgets Application Shell» whose primary purpose is to allow the creation of multiple applications from a base application shell. According to the VOiiLA document, the main goals of this framework are:

- Provide ability to create new web application easily;
- Reuse existing user interface components (mainly widgets) and control services;
- Provide interoperability between VOiiLA and the ISTIP layer by mean of control services.
- VOiiLA components are built with different web-based framework and both server-side and client-side scripting tools like:
- PHP;
- Google Web Toolkit (GWT);
- Ozone Widget Framework (OWF);
- JavaScript, DOJO, Ajax;
- Etc.

7.3.1.3 Interfacing with ISTIP

Every service developed for the iVAC system must be able to be integrated easily with ISTIP and iVAC services must be able to interact with ISTIP services. To achieve this goal, every iVAC service will provide well-defined service interface and use standard-based approach in their development. A least, these services will be WS-I[5] basic profile compliant. Doing this warranty a well and easy integration with ISTIP. As mentioned earlier, ISTIP is a SOA-based integration platform; the iVAC must interact with ISTIP services through their defined service interfaces. When a well-define interface doesn’t exist for an ISTIP service, a service connector can be used to achieve this goal.

[5] WS-I: Web Service Interoperability, http://en.wikipedia.org/wiki/WS-I_Basic_Profile

7.3.1.4 Interfacing with VOiiLA

It is very important to see the integration of iVAC and VOiiLA at two different levels:

- The integration of iVAC services to VOiiLA;
- The integration of iVAC visual components to VOiiLA.

7.3.1.4.1 Interfacing iVAC Business Services with VOiiLA

VOiiLA components (widget and/or controller services) will be able to interact with iVAC functionalities by mean of services. Each iVAC service will provide a well-defined service interface that VOiiLA components can use to interact with iVAC.

7.3.1.4.2 Interfacing iVAC Visual Components with VOiiLA

VOiiLA is a web-based integration platform. In order for the iVAC visual components to be able to integrate to VOiiLA, they must be web-based. Any web framework compliant HTML 4/5 that support JavaScript, CSS[6] and Ajax[7] can be used to deliver the iVAC visual components. Every components build according to such framework will be able to be integrated to VOiiLA.

7.3.2 Primary Candidate Framework 1 (Software)

As described in «1.2.1 – Multi-tier distributed architecture» and depicted in figure «1 and 2», this framework make use of a well suitable web framework[8] for the UI aspect of the iVAC and a distributed multi-tier approach to build the iVAC system. For doing this, emphasis will be made on software products that are mainly distributed by nature and designed to solve the same kind of problem as the purpose of the iVAC:

- **UIMA[9]**: a set of standards-based frameworks, infrastructure and components that facilitate the analysis and annotation of an array of unstructured content (such as text, audio and video). UIMA can be used for real-time content analytics[10] and natural language processing. A UIM application might ingest plain text and identify entities, such as persons, places, organizations; or relations, such as works-for or located-at;
- **UIMA AS[11] Scaleout**: UIMA AS is the Apache[12] next generation scalability replacement for the Collection Processing Manager (CPM). UIMA AS provides a more flexible and powerful scaleout capability, and extends support to the UIMA components not supported by the CPM, the flow controller and CAS multiplier. UIMA components can be run within UIMA AS with no code or component descriptor changes. UIMA AS is built to empower the UIMA capability to process collection of documents more rapidly and to augment the overall scalability, reliability and availability of the system. UIMA AS relies on Apache ActiveMQ for its scaleout capability;

[6] CSS= Cascading Style Sheet

[7] Ajax = Asynchronous JavaScript and XML

[8] See 6 for the recommended UI Framework

[9] UIMA or Unstructured Information Management Architecture, http://uima.apache.org/

[10] As it has done for Watson in Jeopardy's competition, February 2011, http://www-03.ibm.com/innovation/us/watson/science-behind_watson.shtml, https://blogs.apache.org/foundation/entry/apache_innovation_bolsters_ibm_s

[11] UIMA AS or UIMA Asynchronous Scaleout, http://uima.apache.org/doc-uimaas-what.html

[12] http://www.apache.org/

- **Apache Hadoop[13]:** a software framework that enables data-intensive distributed applications to work with thousands of nodes and petabytes of data. Generally seen as a foundation of Cloud computing, Apache Hadoop enables application access, sort, and process data in a massively parallel system. According to the software's web site,

  Apache Hadoop is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

To deliver its parallel processing power, Hadoop relies on some powerful core features:

- **Hadoop Distributed File System (HDFS**): A distributed file system that provides high-throughput access to application data;
- **Hadoop YARN**: A framework for job scheduling and cluster resource management;
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.
- **Apache ActiveMQ[14]**: is a popular and powerful open source messaging and integration server. This message broker is based on JMS (Java Message Service) and supports many cross language clients and protocols[15];
- **Apache Zookeeper[16]**: is a distributed coordinator server that can be used in a distributed environment to keep server nodes, clusters in synch. It can be used as a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications;
- **Apache Mahout[17]:** is a machine learning library designed to build scalable machine learning system. Its core algorithms for clustering, classification and batch based collaborative filtering are implemented on top of Apache Hadoop using the map/reduce paradigm;
- **Apache Tomcat**[18]: is as open source web container that implements the Java EE Servlet and Java Server Pages specification[19]. As such, it can be used as presentation server to service dynamic web components and/or web services components. It can be very helpful to publish the components developed for the iVAC as services in order to facilitate integration with other systems.

7.3.2.1 Alternatives

Even if the software system or libraries presented above can fulfil the iVAC needs, here are some alternatives that may be considered to augment, correct or even replace some of them:

[13] http://hadoop.apache.org/

[14] http://activemq.apache.org/

[15] http://activemq.apache.org/cross-language-clients.html

[16] http://zookeeper.apache.org/

[17] http://mahout.apache.org/

[18] http://tomcat.apache.org/

[19] http://wiki.apache.org/tomcat/Specifications/

- Gate[20]: General Architecture for Text Engineering (Gate) is an infrastructure for developing and deploying software components that process human language. It is nearly 15 years old and is in active use for all types of computational task involving human language. GATE excels at text analysis of all shapes and sizes. It is used for different king of projects: from large corporations to small start-ups, from multi-million research consortia to undergraduate projects. According to the software brochure, key features of Gate are:
    - Component-based development reduces the systems integration overhead in collaborative research.
    - Automatic performance measurement of Language Engineering (LE) components promotes quantitative comparative evaluation.
    - Distinction between low-level tasks such as data storage, data visualisation, discovery and loading of components and the high-level language processing tasks.
    - Clean separation between data structures and algorithms that process human language.
    - Consistent use of standard mechanisms for components to communicate data about language, and use of open standards such as Unicode and XML.
    - Insulation from idiosyncratic data formats (GATE performs automatic format conversion and enables uniform access to linguistic data).
    - Provision of a baseline set of LE components that can be extended and/or replaced by users as required
- **Kafka[21]:** is a real-time messaging system. According the product's web site:

  Kafka is designed to support the following:
    - Persistent messaging with O (1)[22] disk structures that provide constant time performance even with many TB of stored messages.
    - High-throughput: even with very modest hardware Kafka can support hundreds of thousands of messages per second.
    - Explicit support for partitioning messages over Kafka servers and distributing consumption over a cluster of consumer machines while maintaining per-partition ordering semantics.
    - Support for parallel data load into Hadoop.

Kafka provides publish-subscribe solution that can handle all activity stream data and processing on a consumer-scale web site. Kafka aims to unify offline and online processing by providing a mechanism for parallel load into Hadoop as well as the ability to partition real-time consumption over a cluster of machines. This product can be used to augment the messaging capabilities of the iVAC framework.

- **Storm[23]**: in a human – iVAC interaction, real-time processing capability is very important. Storm can bring this capability to the iVAC overall architecture. As specified in the product's web site:

[20] http://gate.ac.uk/

[21] http://kafka.apache.org/

[22] O(1) is an Asymptotic complexity measurement of a process or algorithm. This is a constant value meaning the performance is always constant.

Storm is a free and open source distributed real-time computation system. Storm makes it easy to reliably process unbounded streams of data, doing for real-time processing what Hadoop did for batch processing. Storm is simple, can be used with any programming language. It has many use cases: real-time analytics, online machine learning, continuous computation, distributed RPC, ETL, and more. It is scalable, fault-tolerant, guarantees that the data will be processed.

- **Trident**: it is an extension of Storm that makes it an easy-to-use distributed real-time analytics framework for Big Data processing. Trident is a high-level abstraction for doing real-time computing on top of Storm. It allows to seamlessly intermixing high throughput (millions of messages per second) and stateful stream processing with low latency distributed querying. Like a query language, Trident has joins, aggregations, grouping, functions, and filters. In addition to these, Trident adds primitives for doing stateful, incremental processing on top of any database or persistence store.
- **Cloudera Hadoop**: special Hadoop distribution from Cloudera[24]. It brings real-time capability to Hadoop. Cloudera products are distributed in two forms: community and enterprise.
- **Myrrix**: is a complete real-time, scalable clustering and recommender system, evolved from Apache Mahout. Just as we take for granted easy access to powerful, economical storage and computing today, Myrrix will let take for granted easy access to large-scale «Big Learning» from data. According the product’s web site, here is the rational behind the creation of that system:

  Apache Mahout already provides popular infrastructure for building real-time recommender engines for small- and medium-sized data via the Taste subproject. Separately, it provides infrastructure for computing recommendation and clustering models over massive data sets in batch using Hadoop. But both elements are needed at the same time, in one cooperating system not separately. Myrrix has both, as two cooperating layers:

  - a Serving Layer, which answers requests, records input and provides recommendations and clusters in real-time, and
  - a Computation Layer, which does the heavy-lifting in the background to update the machine learning model for the Serving Layer, using parallelized machine learning algorithms.

  Myrrix brings those two elements together into one complete, ready-to-run system. Its Serving Layer is an HTTP server with a REST-style API, and can receive updates and compute updated results in milliseconds. It also communicates with the Computation Layer, a series of jobs that run on a Hadoop cluster, to run the larger-scale machine learning algorithms and fully update the underlying machine learning models periodically.
- **Jubatus**[25]: Jubatus is a distributed processing framework and streaming machine learning library. Jubatus includes these functionalities:
  - Online Machine Learning Library: Classification, Regression, Recommendation, Graph Mining, Anomaly Detection

[23] http://storm-project.net/

[24] http://www.cloudera.com/content/cloudera/en/home.html

[25] http://jubat.us/en/

- Feature Vector Converter (fv_converter): Data Preprocess and Feature Extraction
- Framework for Distributed Online Machine Learning with Fault Tolerance

From the product web site

There many similar points between Hadoop/Mahout and Jubatus. These are scalable and run on commodity hardware. However, Hadoop is not equipped with sophisticated machine learning algorithms since most of the algorithms do not fit its MapReduce paradigm. Though Apache Mahout is also a Hadoop-based machine learning platform, online processing of data streams is still out of the scope.

Jubatus processes data in online manner, and achieve high throughput and low latency. To achieve these features together, Jubatus uses a unique loosely model synchronization for scale out and fast model sharing in distributed environments.

Jubatus processes all data in memory, and focus on operations for data analysis.

- **Jetty[26]:** is a pure Java-based HTTP server and Java Servlet container. Jetty is developed as a free and open source project as part of the Eclipse Foundation. The web server is used in products such as ActiveMQ, Alfresco[27], Apache Geronimo[28], Apache Maven[29], Google App Engine[30], Eclipse[31] to name a few. Jetty is also the server in open source projects such as Lift[32], Eucalyptus[33], Red5[34] and Hadoop. Jetty supports the latest Java Servlet API (with JSP support) as well as protocols SPDY[35] and WebSocket[36].

7.3.2.2 Advantages

- Performance improvement;
- Highly scalable and available architecture;
- Low-latency;
- Make extensive use of software and/or libraries which are distributed by nature (or design).

7.3.2.3 Disadvantages

- Extremely complex to manage and maintain;
- Integrate those software components and system libraries can be very challenging;

26 http://www.eclipse.org/jetty/

27 http://www.alfresco.com/

28 http://geronimo.apache.org

29 http://maven.apache.org

30 https://cloud.google.com/products/

31 http://www.eclipse.org

32 http://en.wikipedia.org/wiki/Lift_(web_framework)

33 http://en.wikipedia.org/wiki/Eucalyptus_(computing)

34 http://en.wikipedia.org/wiki/Red5_(media_server)

35 http://en.wikipedia.org/wiki/SPDY

36 http://en.wikipedia.org/wiki/WebSocket

- Maintenance and extension of such a system can be difficult in term of time and effort;
- Could be an error prone approach;
- Can require significant resources and complex and rare or unavailable expertise.

7.3.3 Primary Candidate Framework 2

As described in «1.2.2 – Service integration approach» and depicted in Figure 15: Service-Based Integration Approach (Service-Based SOA), this framework make use of a well suitable web framework[37] for the UI aspect of the iVAC and a service-based approach to link or bind the iVAC services to other systems/components/applications and libraries. The Java mainly Java Enterprise Edition is the preferred platform to build the iVAC following this approach. We will need:

- A full certified Java EE (5 preferably 6) application Server to host the iVAC services and provide basic services (JBoss 5 and 7 are in use at the DRDC);
- A certified Java EE (5 preferably 6) web container to deploy the iVAC UI components (Tomcat is in use at the DRDC);
- Rely on the full stack Java EE technologies to deliver a full featured iVAC services. Name a few of them:
    - EJB (Java Enterprise Bean, Stateless Session Bean, Stateful Session Bean «where needed», MDB «Message-Driven Bean»)
    - Timer service (for scheduled tasks);
    - JAX-WS (Java XML Web Service);
    - JAX-RS (Restful Web Service);
    - JAXB (Java Architecture for XML Binding);
    - JPA (Java Persistence API);
    - JTA/JTS (Java Transaction API/Java Transaction Services)
    - JCA (Java Connector Architecture);
    - All XML-like technologies;
    - Etc.

All those stuffs are provided by design by the application server and the Java EE infrastructure;

- Develop a service adapter (or connector) to integrate each components/system that doesn't provide a compatible interface. The service adapter will provide well-defined interface that can be used by any iVAC's service (self, composed or process service). This service adapter must not be accessible directly by other systems but through iVAC services;
- Develop a data transformation service (Data Mapper) to transform back and forth data from and/or to each system to be integrated;
- Develop a service adapter for intelligence software libraries;
- When a well-defined service interface is provided for an application/system/services, it will be used as it is;

[37] See 6 for the recommended UI Framework

- Using JMS (Java Message Service) and a server-side component (MDB, Message Driven Bean) to provide asynchronous capabilities and to connect to long running system;
- Using JPA/JDBC to connect to each relational data store for data manipulation;
- Using the facilities provided or supported by the other data stores (document, Key-value, unstructured and so one) to integrate them;
- Using JCA (Java Connector Architecture) or other mean where needed or required to integrate more complex and uncommon (unusual piece of) software system.

7.3.3.1.1 Advantages

- Application integration is made through a service layer, no direct relationship between applications;
- No need to learn a new language or Framework; the development team is up and running;
- No need for rewriting applications; when required the application's functionalities are exposed as services;
- Greater use of the expertise developed over the projects and years;
- An excellent base to a more complete well integrated SOA architecture;
- Make extensive use of the DRDC infrastructure, components and services.

7.3.3.1.2 Disadvantages

- More there are systems to integrate more the architecture becomes complex and difficult to manage and maintain;
- Require a lot of manual data transformation to facilitate the communication between the integrated systems; because of leak of automatic data transformation features and data standardization like a canonical data model;
- Encourage excessive use service adapter ; no built-in facilities to make it more easier to integrate a piece of software system or applications;
- Does not facilitate agility and can be costly in the long term (need a connector every time you need to integrate an atypical application);
- Time consuming;

7.3.4 Primary Candidate Framework 3

Using a well-suitable web based framework for the UI components and the SOA integration paradigm to deliver the iVAC system.

Although SOA is technically made of services, in this approach, an Enterprise Service Bus (ESB) will be used as the central point of the integration architecture (see section 1.2.3, Figure 16: SOA Integration Approach Based on an Enterprise Service Bus (ESB)). This component brings another abstraction layer on the overall architecture. The encapsulation of services by the ESB means that the client application does not need to know anything about the location of the services, or the different communication protocols used to call them. The ESB enables the shared, enterprise-wide, and even intra-enterprise use of services and separate business processes from the relevant service implementations[38]. Then, it completely decouples the service consumer (the client) from the service provider

[38] Lee et al. 2003

in such a way that the service consumer doesn't need to know anything about the service provider.

The main functionalities provided by any ESB product fall into those:

- Build and deploy services;
- Encapsulate legacy systems;
- Route messages;
- Transform message formats;
- Perform protocol mediation.

There are many ESB products on the market both commercial and Open source that can be used to fulfil this integration approach. But, only the so-called «open source» ESB are considered. Among them, a few were analysed:

- **OpenESB**: OpenESB39 is a Java based open source enterprise service bus. It can be used as a platform for both Enterprise Application Integration and Service Oriented Application. OpenESB allows integrating legacy systems, external and internal partners and new and custom development. OpenESB is relying on standard JBI (Java Business Integration), XML, XML Schema, WSDL, BPEL and composite application that provides simplicity, efficiency, and long-term durability. Since the acquisition of Sun Microsystem by Oracle in January 2010, this product is maintained by the OpenESB community that was created for that purpose;
- **MuleESB**: Mule40 is a lightweight enterprise service bus (ESB) and integration framework. It can handle services and applications using disparate transport and messaging technologies. The platform is Java-based, but can broker interactions between other platforms such as .NET using web service or socket. The architecture is a scalable, highly distributable object broker that can seamlessly handle interactions across legacy systems, in-house applications, and almost all modern transports and protocols. It is well-featured and integrated ESB product. But, some of the most interesting features like the web-based management console and others required licence fee (see annex);
- **FuseESB**: Fuse41 ESB is an open source integration platform based on Apache ServiceMix42 that supports JBI and OSGi43 for use in enterprise IT organizations. Fuse ESB is robust SOA infrastructure that provides a standardized methodology, server, and tools to integrate components in mission-critical applications. Fuse ESB has a pluggable architecture that allows organizations to use their preferred service solution in their SOA. Any standard JBI or OSGi-compliant service engine or binding component – including BPEL, XSLT or JMX engines - may be deployed to a Fuse ESB container, and Fuse ESB components may be deployed to other ESBs. Fuse ESB is part of a family of open source SOA infrastructure tools that include Fuse Message Broker (enterprise release of Apache ActiveMQ), Fuse Services Framework (enterprise release of Apache CFX) and Fuse Mediator Router (enterprise release of Apache

[39] http://www.open-esb.net/

[40] http://www.mulesoft.org/

[41] http://fusesource.com/products/enterprise-servicemix/

[42] http://servicemix.apache.org/

[43] OSGi= Open Services Gateway initiative

Camel). Recently, Fuse ESB was been integrated to Red Hat and became Red Hat JBoss Fuse. Since this integration, Fuse ESB tends to be closely bound to the overall JBoss products;

- **WSO2 ESB**: The WSO2 Enterprise Service Bus (ESB), from WSO2, an open source application development software company, is a simple, lightweight and high performance enterprise service bus (ESB), based on the Apache Synapse44 enterprise service bus, providing enhanced management, development and configuration support as well as SOA Governance capabilities. According to the WSO2's web site45, Ebay uses WSO2 Enterprise Service Bus as one of the key elements in its transaction software, which continuously executes $2,000 worth of transactions per second and over 1 billion transactions per day.
  The WSO2 Enterprise Service Bus (WSO2 ESB) supports the creation of Proxy Services graphically, which allows users to easily create virtual services on the ESB layer to front existing services. Existing services are SOAP, Plain Old XML (POX)/REST services over HTTP/S, as well as SOAP or legacy services over JMS, Apache VFS46 file systems, Mail systems (such as Post Office Protocol (POP3), Internet Message Access Protocol (IMAP), and Simple Mail Transfer Protocol (SMTP)), Financial Information Exchange (FIX), Hessian, and Advanced Message Queuing Protocol (AMQP). There is any JMS Message Broker integrated into the product. Before using JMS messaging capabilities in the product, at least Apache ActiveMQ47 must be installed. Even if the product can be ran on most operating systems, Linux/Solaris48 are the recommended ones for a production environment;
- **JBoss ESB**: the JBoss Enterprise SOA Platform (or JBoss SOA Platform) is free software/open-source Java EE-based Service Oriented Architecture(SOA) software. It is part of the JBoss Enterprise Middleware portfolio of software. The JBoss Enterprise SOA Platform enables enterprises to integrate services, handle business events, and automate business processes, linking IT resources, data, services and applications. Because it is Java-based, the JBoss application server operates cross-platform: usable on any operating system that supports Java. The software is Enterprise Application Integration (EAI) or business integration software. As any enterprise service bus (ESB) software, JBoss ESB is used to map the Service-Oriented Infrastructure (SOI) and Service-Oriented Architecture (SOA) concepts onto a concrete implementation. The software is middleware used to connect systems together, especially non-interoperable systems;
- **Ultra ESB**: new in the Enterprise Service Bus (ESB) landscape, UltraESB is a lightweight enterprise service bus (ESB) capable of supporting many transports and message formats natively. It allows messages to be mediated via Java or JSR 223 scripting languages49 over the JDK support. It is the first ESB to claim support for

[44] http://synapse.apache.org/

[45] http://wso2.com/products/enterprise-service-bus/, last consultation date: 2013-03-27

[46] Apache VFS file systems support: s-ftp, file, zip, tar, gz, webdav, and cifs

[47] ActiveMQ: a popular and powerful messaging and intergration patterns server, http://activemq.apache.org/ , last consultation date : 2013-03-27

[48] http://docs.wso2.org/wiki/display/ESB460/Installation+Prerequisites , last consultation date: 2013-03-27

[49] See the API documentation : http://api.adroitlogic.org/

Zero-Copy proxying50 of messages. The initial version was published first in January 2010 and the code was subsequently open sourced under the OSI approved Affero General Public License AGPL51 in August 2010.

7.3.4.1 ESB Comparison

Here some elements of comparison between open source ESB products. This is based on Forester publication and other studies[52]. This table will help understanding our recommendation.

Table 7: Comparison of ESB Systems.

	WSO2 ESB and SOA Platform	Mule ESB	FuseSource ESB (Red Hat JBoss Fuse)	Adroit Logic Ultra ESB	JBoss ESB and SOA Platform
Supports Enterprise Integration Patterns	Yes	Yes	Yes	Yes	Yes
Delivers all required ESB features (i.e. web services, message transformation, protocol mediation, content routing)	Yes	Yes	Yes	Yes	Yes
Offers a complete and cohesive SOA Platform (i.e. ESB, Message Broker, Governance Registry, Business Process Server, Data Services Server, Application Server)	Yes	No	No	No	No
SOA Governance	Yes	No	No	No	No
Graphical ESB Development Workbench	Yes	Yes[53]	Yes	No	Yes
Based on a composable architecture	Yes	No	No	No	No
Cloud integration platform offering (iPaaS)	Yes	Yes	No	No	No
Cloud Connectors and Legacy Adapters	Yes	Yes	No	No	Yes
Performance	High	Moderate	Moderate	High	Moderate
Security and Identity Management	Yes	Limited	Limited	Limited	Limited
Open Business Model	Yes	Yes	Yes	Yes	No
Licencing	Apache V2.0	CPAI[54] Or Commercial	Apache V2.0 Or Commercial	AGPL[55] Or Zero-dollar non-GPL commercial	Apache V2.0 Or Commercial

[50] http://www.prweb.com/releases/2010/01/prweb3462154.htm , consultation date : 2013-04-03

[51] http://en.wikipedia.org/wiki/Affero_General_Public_License

[52] http://blog.cobia.net/cobiacomm/2012/08/01/esb-comparison/ , consultation date : 2013-04-03

[53] Mule ESB provides the friendliest graphical interface for both the development environment and the administrative user interface. But, they require licence fees.

[54] CPAL= Common Public Attribution License.

	WSO2 ESB and SOA Platform	Mule ESB	FuseSource ESB (Red Hat JBoss Fuse)	Adroit Logic Ultra ESB	JBoss ESB and SOA Platform
Future evolution and maintenance	Yes	Yes	May be[56]	Yes	May be[57]

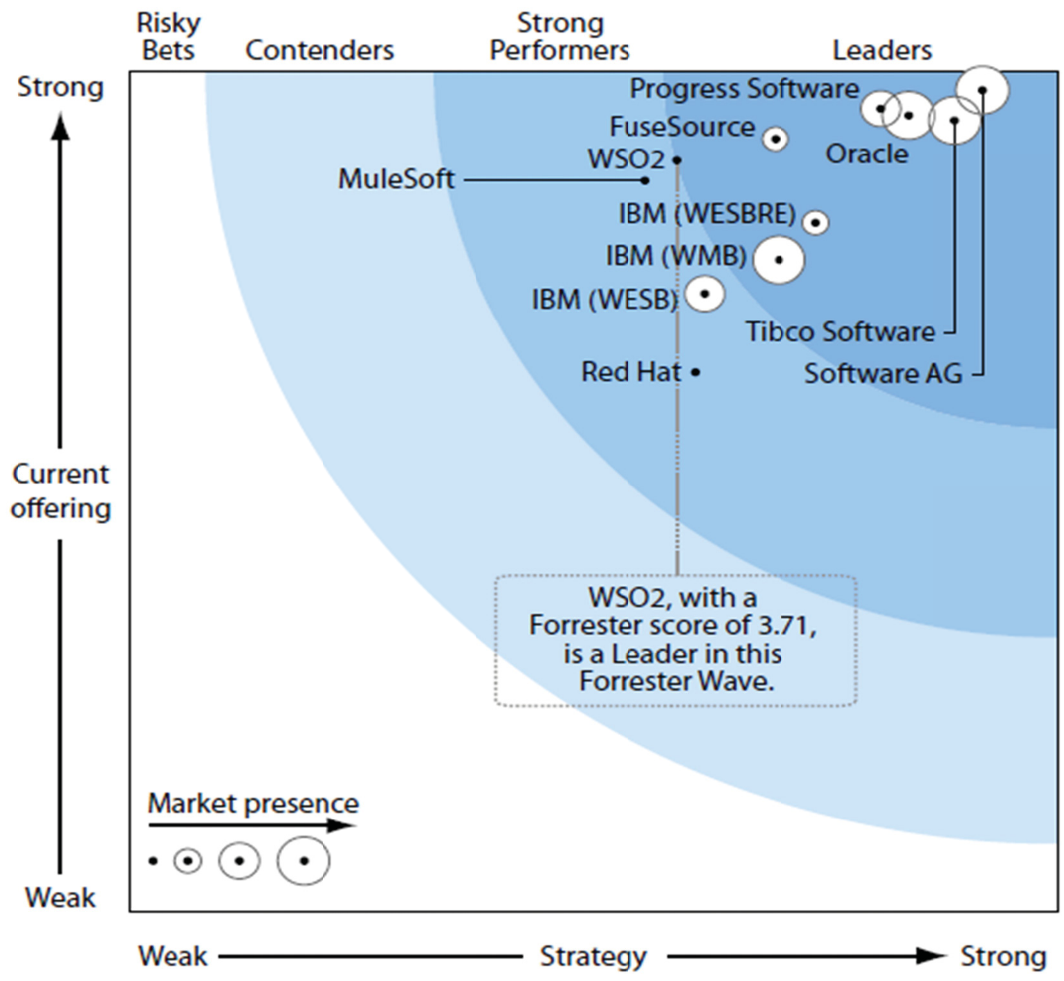


Figure 17: Forrester Wave™: Enterprise Service Bus, Q2 '11, April 25, 2011

7.3.4.2 Advantages

- Application integration is made through the bus;

---

[55] AGPL= Affero General Public Licence for development and pre-production, and zero-dollar non-GPL commercial licence for production. Both require filling a licence agreement Internet form.

[56] Since Fuse ESB become Red Hat JBoss Fuse, it is not clear that Red Hat will continue to maintain two ESB products: JBoss ESB and Red Hat JBoss Fuse.

[57] It is possible that Red Hat discontinues one of its two ESB products (JBoss Fuse and JBoss ESB).

- Facilitate a more agile development approach by letting services composition directly over the bus;
- Less development effort;
- Very flexible architecture;
- Fast time to market;
- Provide mechanisms for building more loose coupled and process-driven applications;
- Compliant with standards;
- Low fallow-up costs.

7.3.4.3 Disadvantages

- Startup and infrastructure costs may be high;
- The learning curve can be important for both the development and the support teams;
- A new infrastructure software system to manage;
- Can be overkill in some cases;
- Requires a comprehensive SOA strategy and governance.

7.3.4.4 ESB Recommendation

Delivering the iVAC integration architecture regarding SOA perspective and based on the comparison table provided, WSO2 ESB is a more featured Enterprise Service Bus (ESB). It is our choice for this approach. This product provides:

- An Eclipse-based visual development tool to build system, processes and services;
- An easy to use visual management console for deploying, managing and motoring applications and services;
- Automatic data transformation features;
- Support many languages;
- Built-in support for multiple data base systems;
- Extensive use of standard;
- Etc.

7.3.5 Other possibilities

Using the Spring Framework to deliver a more lightweight integration architecture for building the iVAC system according to the «Service Integration Approach».

According the spring Framework documentation[58]:

The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform. Spring includes:

- Flexible dependency injection with XML and annotation-based configuration styles;
- Advanced support for aspect-oriented programming with proxy-based and AspectJ-based variants;

[58] http://www.springsource.org/spring-framework, consultation date: 2013-03-26

- Support for declarative transactions, declarative caching, declarative validation, and declarative formatting;
- Powerful abstractions for working with common Java EE specifications such as JDBC, JPA, JTA and JMS;
- A flexible web framework for building RESTful MVC applications and service endpoints;
- Rich testing facilities for unit tests as well as for integration tests.

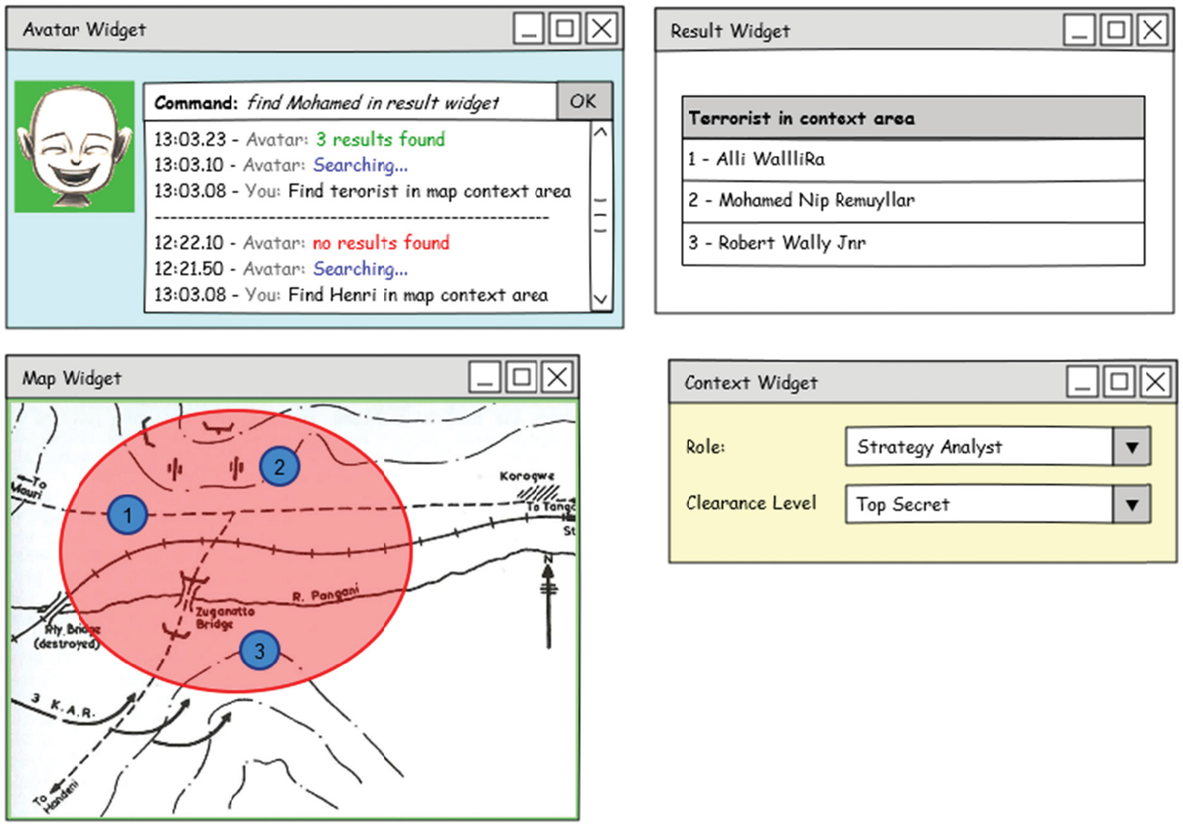
Spring is modular in design, allowing for incremental adoption of individual parts such as the core container or the JDBC support. While all Spring services are a perfect fit for the Spring core container, many services can also be used in a programmatic fashion outside of the container.

Spring applications can be deployed as standalone applications on Tomcat Server and/or on a full featured Java EE application server such as IBM WebSphere, JBoss, and Oracle Weblogic.

The main difference between this approach and the service integration approach based on Java EE technologies stack (see primary candidate framework), is a more lightweight development framework and model by using more simple Java-based development (no EJB and other Java EE heavyweight components). Then, it shares likely the same advantages and trade-off. Adding, Spring-based development approach doesn’t require a full Java EE Application, a web container like Tomcat is sufficient.

8 Graphical User Interface

8.1 IVAC's GUI Layout Proposal



Command: *find Mohamed in result widget*

13:03.23 - Avatar: 3 results found
13:03.10 - Avatar: Searching...
13:03.08 - You: Find terorist in map context area

12:22.10 - Avatar: no results found
12:21.50 - Avatar: Searching...
13:03.08 - You: Find Henri in map context area

Terrorist in context area
1 - Alli WallliRa
2 - Mohamed Nip Remuyllar
3 - Robert Wally Jnr

Role:	Strategy Analyst
Clearance Level	Top Secret

Figure 18: Avatar GUI Sample

8.1.1 Avatar Widget

The Avatar Widget is composed of the following elements.

Avatar agent embodiment

The Avatar is a visual representation of a human or other personage. The Avatar should be able to give visual feedback to the user about ingoing tasks, requests status and results. The visual feedback could be a mix of facial expressions, head, eyes and body movements. Symbols, colors and sounds should also be added for more clarity.

In the best scenario, the Avatar could be a fully three-dimensional animated character, but a set of static images, colors and symbols could be as efficient.

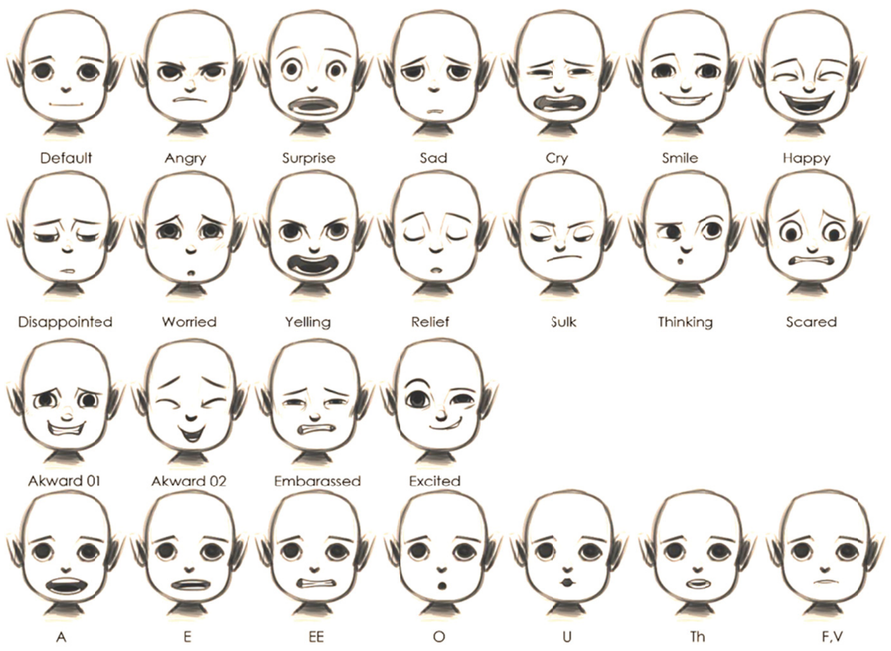


Figure 19: Some faces

Case examples:

Search	Eyes look up and move from side to side, one eyebrow is upper than the other. A loading spinner could be added for clarity.	Thinking		Blue
Request not understood, No result found	Eyes look down, eyebrows angle down. A red triangle shaped exclamation is added.	Disappointed	⚠	Red
Result found	Mouth wide open, eyes closed (in contrast of search expression). A green circle is added with the number of results.	Happy	202	Green

Figure 20: Other status: listening to voice command, refine your search, speaking

- **Voice command capability**

The user should be able to make natural language requests to the avatar system through voice commands. The Avatar should give visual feedback to user when listening mode is active.

- Avatar text-to-speech phonemes lip-syncing

  In the best scenario, the animated Avatar should be able to synchronize lips movements with text-to-speech voice from the Avatar system or at least move lips in the same time frame.

- Text command/question input

  User should be able to ask questions and launch commands as text string to the Avatar system.
  An input text field and a "launch analyze" button are added to the widget for that purpose.

- Conversation History List

  The history / logs of "conversations" (requests and responses) from the user and the Avatar system should be available for the user. A chat box like list is shown below the text input command. Text transcript of voice conversations should also be logged in the chat box.

Result Widget

The Avatar system must be able to visually render output results from user request. The results could be a generated table / grid object or a list of items.

Map Widget

In this layout proposition, the map widget is used as a context for data retrieval in voice or text requests to the Avatar system.

Context Widget

The context widget goal is to inform the system about the user. Knowledge about user role, clearance level and localization could modify Avatar system response and interaction. For example: Top secret information cannot be divulgate to a recruit.

8.2 VOiiLA's GUI

Minimal Requirements for Integration with VOiiLA

VOiiLA environment offers low limitations for applications and widgets integration.
Any Web technologies can be added to the environment.

What to look for

- Browser version compatibility & limitation.
  Ex: ActiveX may be limited to Microsoft Internet Explorer browser and client running on Intel x86 hardware.
  Advance HTML5 canvas and data storage will not work in old browser like IE6.
- Too many library inclusions at load time will slow down the application. Ex: adding Prototype JavaScript library to VOiiLA would not be a good choice. Try using already include jQuery and Dojo instead.
- External Web services (Software as a Service (SaaS)) must be avoided. VOiiLA has no access to the Web.

- Even if desktop applications could communicate with VOiiLA trough ActiveX, Java applet or WebORB Integration Server like software, desktop applications can not be fully integrated in VOiiLA. A desktop application needs to be installed on each client machines thus reducing the portability and augmenting the deployment complexity.
- Reuse existing widgets, images, CSS & Web services is a good practice.

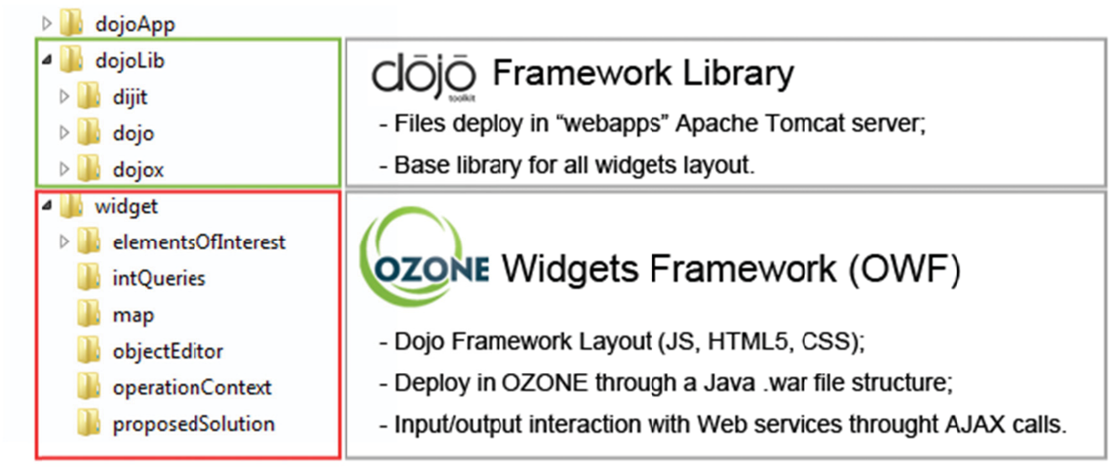
Technology description and role

Table 8: Web-based GUI Technologies

Technology	Description	Role
Web Browser	Internet Explorer, Firefox, Chrome, Safari, Opera	Render web content UI.
HTML	Text Markup	Define base layout and objects of the document.
CSS	Cascading Style Sheet	Visual styles for HTML layout and objects (color, position, etc…).
Widget Framework Controller	Ozone Widget Framework (OWF)	Framework for visually organizing and laying out lightweight web applications (widgets) within user's browser. Provides infrastructure services to facilitate development of workflows and presentation-tier application integration.
JavaScript & JavaScript Library	jQuery	Manipulate document DOM elements: event binding, animation, variable & object manipulation
JavaScript Framework & UI	Sencha EXT JS, Dojo Toolkit	Rich GUI objects and manipulations: Grid, Tree, Graph, Window display, etc...
AJAX	Asynchronous JavaScript and XML method	Interact and manipulate DOM elements asynchronously (without refreshing the document). Call an external script, page or web service with call back functionalities.

JSON	Lightweight text encapsulated representation of data	Data transportation and representation through Ajax and web services.
JAVA	Server side object-oriented computer programming language	Widget encapsulation for OWF. Server side web services programming.

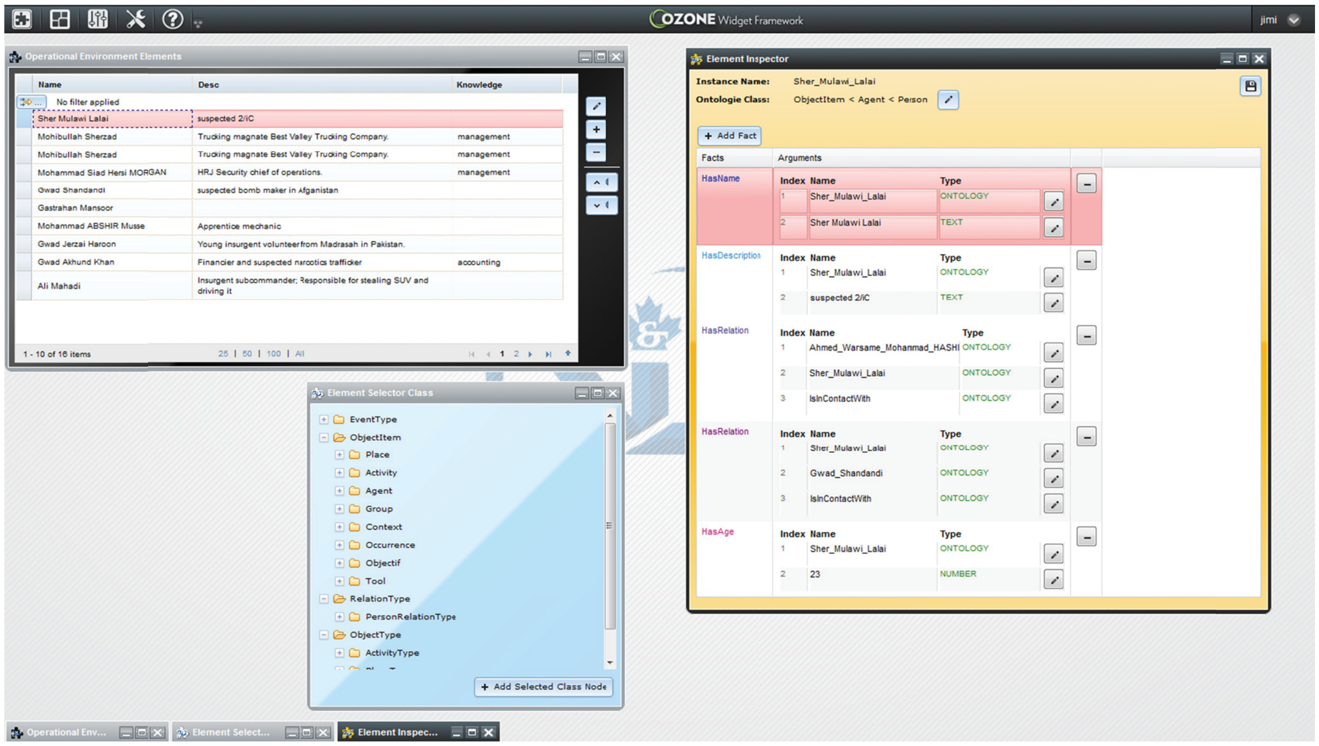
Widgets OWF integration in the context of SIIP project



- Files deploy in "webapps" Apache Tomcat server;
- Base library for all widgets layout.

- Dojo Framework Layout (JS, HTML5, CSS);
- Deploy in OZONE through a Java .war file structure;
- Input/output interaction with Web services throught AJAX calls.

Figure 21: Visual Web browser outputs of Widgets



Operational Environment Elements

No filter applied

Name	Desc	Knowledge
Sher Mulawi Lalai	suspected 2/iC	
Mohibullah Sherzad	Trucking magnate Best Valley Trucking Company.	management
Mohibullah Sherzad	Trucking magnate Best Valley Trucking Company.	management
Mohammad Siad Hersi MORGAN	HRJ Security chief of operations.	management
Gwad Shandandi	suspected bomb maker in Afganistan	
Gastrahan Mansoor		
Mohammad ABSHIR Musse	Apprentice mechanic	
Gwad Jerzai Haroon	Young insurgent volunteerfrom Madrasah in Pakistan.	
Gwad Akhund Khan	Financier and suspected narcotics trafficker	accounting
Ali Mahadi	Insurgent subcommander; Responsible for stealing SUV and driving it	

1 - 10 of 16 items 25 | 50 | 100 | All 1 2

Element Selector Class

- EventType
- ObjectItem
  - Place
  - Activity
  - Agent
  - Group
  - Context
  - Occurrence
  - Objectif
  - Tool
- RelationType
  - PersonRelationType
- ObjectType
  - ActivityType

+ Add Selected Class Node

Element Inspector

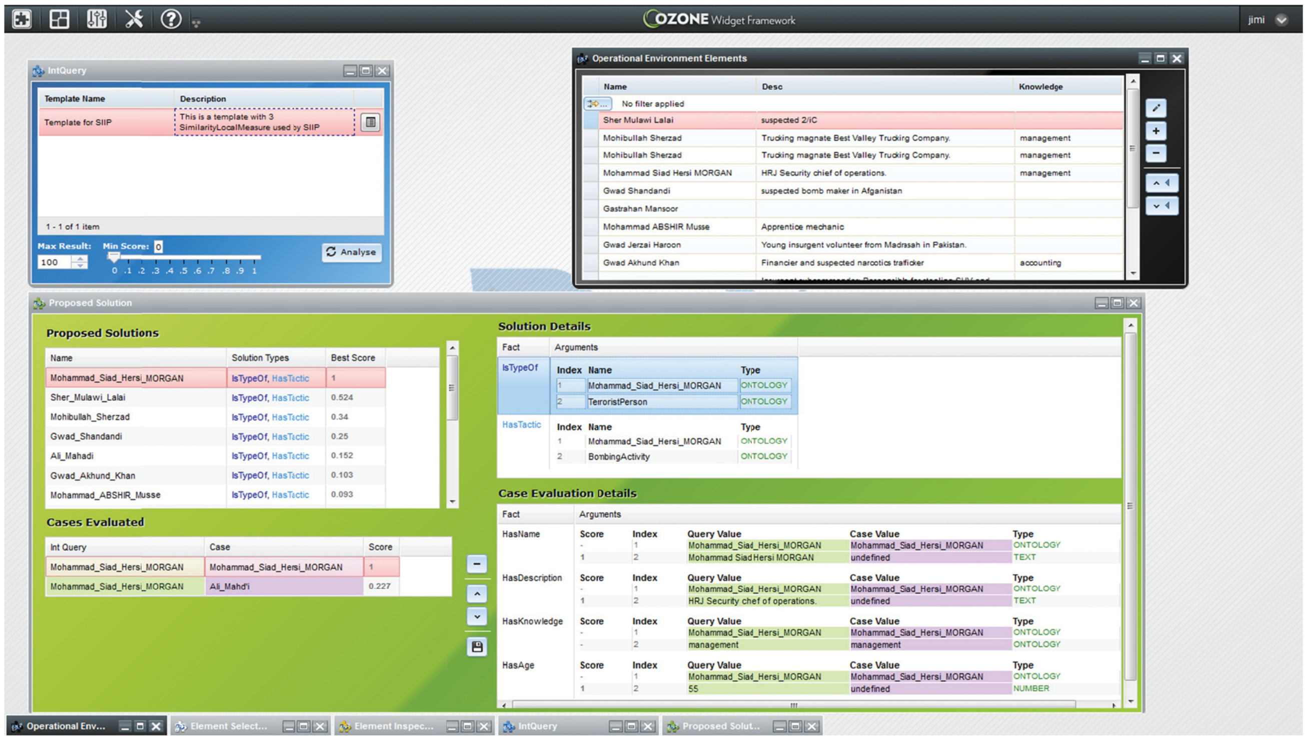
Instance Name: Sher_Mulawi_Lalai
Ontologie Class: ObjectItem < Agent < Person

+ Add Fact

Facts	Index	Name	Type
HasName	1	Sher_Mulawi_Lalai	ONTOLOGY
	2	Sher Mulawi Lalai	TEXT
HasDescription	1	Sher_Mulawi_Lalai	ONTOLOGY
	2	suspected 2/iC	TEXT
HasRelation	1	Ahmed_Warsame_Mohammad_HASHI	ONTOLOGY
	2	Sher_Mulawi_Lalai	ONTOLOGY
	3	IsInContactWith	ONTOLOGY
HasRelation	1	Sher_Mulawi_Lalai	ONTOLOGY
	2	Gwad_Shandandi	ONTOLOGY
	3	IsInContactWith	ONTOLOGY
HasAge	1	Sher_Mulawi_Lalai	ONTOLOGY
	2	23	NUMBER

Figure 22 Widgets: Operational Environment Elements

Widgets: IntQuery, Operational Environment Elements, Proposed Solution.



IntQuery

Template Name	Description
Template for SIIP	This is a template with 3 SimilarityLocalMeasure used by SIIP

1 - 1 of 1 item

Max Result: 100 Min Score: 0 Analyse

Operational Environment Elements

No filter applied

Name	Desc	Knowledge
Sher Mulawi Lalai	suspected 2/iC	
Mohibullah Sherzad	Trucking magnate Best Valley Truckirg Company.	management
Mohibullah Sherzad	Trucking magnate Best Valley Truckirg Company.	management
Mohammad Siad Hersi MORGAN	HRJ Security chief of operations.	management
Gwad Shandandi	suspected bomb maker in Afganistan	
Gastrahan Mansoor		
Mohammad ABSHIR Musse	Apprentice mechanic	
Gwad Jerzai Haroon	Young insurgent volunteer from Madrasah in Pakistan.	
Gwad Akhund Khan	Financier and suspected narcotics trafficker	accounting

Proposed Solution

Proposed Solutions

Name	Solution Types	Best Score
Mohammad_Siad_Hersi_MORGAN	IsTypeOf, HasTactic	1
Sher_Mulawi_Lalai	IsTypeOf, HasTactic	0.524
Mohibullah_Sherzad	IsTypeOf, HasTactic	0.34
Gwad_Shandandi	IsTypeOf, HasTactic	0.25
Ali_Mahadi	IsTypeOf, HasTactic	0.152
Gwad_Akhund_Khan	IsTypeOf, HasTactic	0.103
Mohammad_ABSHIR_Musse	IsTypeOf, HasTactic	0.093

Cases Evaluated

Int Query	Case	Score
Mohammad_Siad_Hersi_MORGAN	Mohammad_Siad_Hersi_MORGAN	1
Mohammad_Siad_Hersi_MORGAN	Ali_Mahd'i	0.227

Solution Details

Fact	Index	Name	Type
IsTypeOf	1	Mohammad_Siad_Hersi_MORGAN	ONTOLOGY
	2	TerroristPerson	ONTOLOGY
HasTactic	1	Mohammad_Siad_Hersi_MORGAN	ONTOLOGY
	2	BombingActivity	ONTOLOGY

Case Evaluation Details

Fact	Score	Index	Query Value	Case Value	Type
HasName	-	1	Mohammad_Siad_Hersi_MORGAN	Mohammad_Siad_Hersi_MORGAN	ONTOLOGY
	1	2	Mohammad Siad Hersi MORGAN	undefined	TEXT
HasDescription	-	1	Mohammad_Siad_Hersi_MORGAN	Mohammad_Siad_Hersi_MORGAN	ONTOLOGY
	1	2	HRJ Security chef of operations.	undefined	TEXT
HasKnowledge	-	1	Mohammad_Siad_Hersi_MORGAN	Mohammad_Siad_Hersi_MORGAN	ONTOLOGY
	-	2	management	management	ONTOLOGY
HasAge	-	1	Mohammad_Siad_Hersi_MORGAN	Mohammad_Siad_Hersi_MORGAN	ONTOLOGY
	1	2	55	undefined	NUMBER

Figure 23: Widgets: IntQuery, Operational Environment Elements and Proposed Solution.

8.3 GUI alternatives

8.3.1 Web GUI: HTML, JavaScript, AJAX & Websocket

Table 9: Score Summary of a Web GUI

Score Summary	
Integrability with VOiiLA	★★★★★
Capability of dynamically representing the actual state of the iVAC	★★★★★
Capability to capture user's context	★★★★
Capability of integrating the avatar	★★★★
Capability of integrating SR and SS components	★★★
Ease of implementation	★★★★
Low integration time	★★★★

Technology description and role

Table 10: Technology Description And Role

Web Browser	Internet Explorer, Firefox, Chrome, Safari, Opera	Render web content UI.
HTML	Text Markup	Define base layout and objects of the document.
CSS	Cascading Style Sheet	Visual styles for HTML layout and objects (color, position, etc…).
JavaScript & JavaScript Library	jQuery, Prototype,	Manipulate document DOM elements: event binding, animation, variable & object manipulation
JavaScript Framework & UI	Sencha EXT JS, Dojo Toolkit, Kendo UI, jQuery UI	Rich GUI objects and manipulations: Grid, Tree, Graph, Window display, etc...
AJAX	Asynchronous JavaScript and XML method	Interact and manipulate DOM elements asynchronously (without refreshing the document). Call an external script, page or web service with call back functionalities.
JSON, XML	Lightweight text encapsulated representation of data	Data transportation and representation through Ajax and web services.
Web Socket, Node Server, Socket.IO	TCP-based protocol with handshake interpreted by HTTP servers as an Upgrade request	Makes possible more interaction between a browser and a web site, facilitating live content and the creation of real-time apps.
Adobe Flash ***(optional alternative)***	SWF files can contain animations or applets of varying degrees of interactivity and	Display animated vector graphics on the Web.

	function multimedia, vector graphics and ActionScript	
Java applet ***(optional alternative)***	Java applets can be part of a web page and executed by the Java Virtual Machine (JVM) in a process separate from the web browser. (3D hardware acceleration)	Display non-trivial, computation intensive visualizations on the Web.

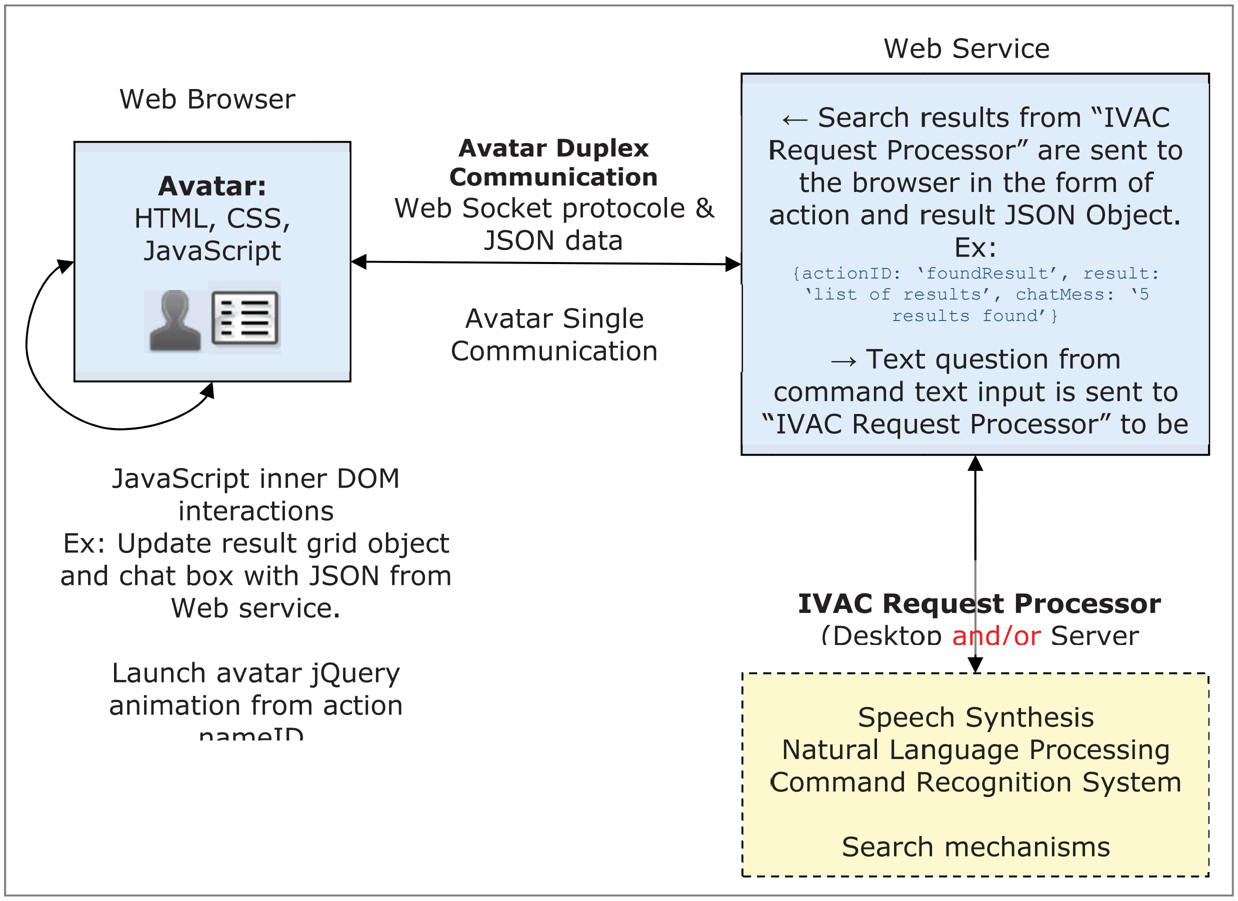
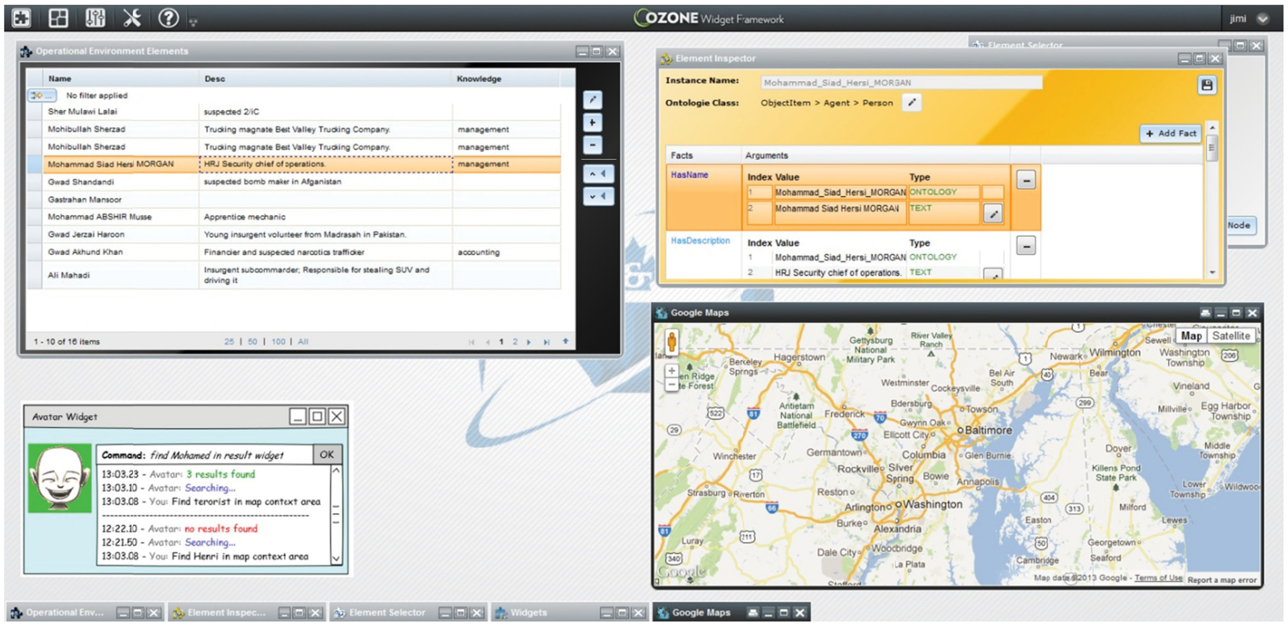

Web Service
Web Browser
Avatar: HTML, CSS, JavaScript
Avatar Duplex Communication
Web Socket protocole & JSON data
Avatar Single Communication
← Search results from "IVAC Request Processor" are sent to the browser in the form of action and result JSON Object. Ex:
{actionID: 'foundResult', result: 'list of results', chatMess: '5 results found'}
→ Text question from command text input is sent to "IVAC Request Processor" to be
JavaScript inner DOM interactions
Ex: Update result grid object and chat box with JSON from Web service.
Launch avatar jQuery animation from action nameID
IVAC Request Processor
(Desktop and/or Server
Speech Synthesis
Natural Language Processing
Command Recognition System
Search mechanisms

Figure 24: Avatar GUI Interactions Model Summary



Name	Desc	Knowledge
No filter applied		
Sher Mulawi Lalai	suspected 2/iC	
Mohibullah Sherzad	Trucking magnate Best Valley Trucking Company.	management
Mohibullah Sherzad	Trucking magnate Best Valley Trucking Company.	management
Mohammad Siad Hersi MORGAN	HRJ Security chief of operations.	management
Gwad Shandandi	suspected bomb maker in Afganistan	
Gastrahan Mansoor		
Mohammad ABSHIR Musse	Apprentice mechanic	
Gwad Jerzai Haroon	Young insurgent volunteer from Madrasah in Pakistan.	
Gwad Akhund Khan	Financier and suspected narcotics trafficker	accounting
Ali Mahadi	Insurgent subcommarder; Responsible for stealing SUV and driving it	

1 - 10 of 16 items   25 | 50 | 100 | All   1 2

Instance Name: Mohammad_Siad_Hersi_MORGAN
Ontologie Class: ObjectItem > Agent > Person

+ Add Fact

Facts	Arguments		
HasName	Index	Value	Type
	1	Mohammad_Siad_Hersi_MORGAN	ONTOLOGY
	2	Mohammad Siad Hersi MORGAN	TEXT
HasDescription	Index	Value	Type
	1	Mohammad_Siad_Hersi_MORGAN	ONTOLOGY
	2	HRJ Security chief of operations.	TEXT

Command: *find Mohamed in result widget*

```
13:03.23 - Avatar: 3 results found
13:03.10 - Avatar: Searching...
13:03.08 - You: Find terorist in map context area
---------------------------------------------------
12:22.10 - Avatar: no results found
12:21.50 - Avatar: Searching...
13:03.08 - You: Find Henri in map context area
```

Figure 25: Visual Example of Web Avatar in VOiiLA's OWF Widget

Integrability with VOiiLA

★★★★★

Given that this alternative offers all web technologies and mostly the same as SIIP VOiiLA project, it has the top integrability score.

Capability of dynamically representing the actual state of the iVAC

★★★★★

Web alternative does not include SR and SS components, external desktop of server side software will be require for this purpose (e.g. Nuance Dragon). In this context, real-time duplex communication with the GUI, services and desktop software and the availability of API connectors is critical for the Avatar to give accurate feedback to the user.

Capability to capture user's context

★★★★

Base user's context is provided by getting static information about the logged user. User's interactions with the Web GUI give the system in time information about the context.

The capability of capturing a complex user context such as: face recognition, eye gazing, tone of voice and voice recognition depend directly on the capability of the Avatar Services, third-party software and hardware availability (microphone, camera).

Capability of integrating the avatar

★★★★

Web GUI alternatives allow multiple way of integrating the Avatar. Limitations come with the capability of the Avatar system to make real-time communication with the GUI and the availability of API connectors.

Examples of possible limitations:

- Avatar representation may not be able to lip-sync text-to-speech sound output because of a missing third-party API communication junction point.
- Low speed of communications between the Web GUI and Avatar system may cause lags in Avatar real-time animation.

Capability of integrating SR and SS components

★★★★

Web GUI does not include SR and SS components, but it is possible to communicate with them through Web services.

Disadvantages and possible ways of overcoming them

- Full dependency on Avatar system components, API and communication efficiency.

Ease of implementation

★★★★

This GUI alternative is probably the simplest and the most versatile implementation of an avatar in the context of VOiiLA.

Low integration time

★★★★

Lower integration time because VOiiLA’s environment technologies are well known and widgets from SIIP project might be reusable.

Availability, licensing and associated cost

Free - Open source.

8.3.2 Desktop Avatar with Web GUI Alternative

Table 11: Score Summary of a Desktop Avatar and Web GUI

Score Summary	
Integrability with VOiiLA	★★
Capability of dynamically representing the actual state of the iVAC	★★★
Capability to capture user's context	★★★★
Capability of integrating the avatar	★★★★★
Capability of integrating SR and SS components	★★★★★
Ease of implementation	★★
Low integration time	★★

Technology description and role

Technology	Description	Role
Desktop Application	Compiled Desktop Client Application Bundle. Flex, Java, C, C++.	Avatar animation, SR, SS, NTL, Search, etc.
Web Browser	Internet Explorer, Firefox, Chrome, Safari, Opera	Render web content UI.
HTML	Text Markup	Define base layout and objects of the document.
CSS	Cascading Style Sheet	Visual styles for HTML layout and objects (color, position, etc…).
JavaScript & JavaScript Library	jQuery, Prototype,	Manipulate document DOM elements: event binding, animation, variable & object manipulation
JavaScript Framework & UI	Sencha EXT JS, Dojo Toolkit, Kendo UI, jQuery UI	Rich GUI objects and manipulations: Grid, Tree, Graph, Window display, etc...

AJAX	Asynchronous JavaScript and XML method	Interact and manipulate DOM elements asynchronously (without refreshing the document). Call an external script, page or web service with call back functionalities.
JSON, XML	Lightweight text encapsulated representation of data	Data transportation and representation through Ajax and web services.
Web Socket, Node Server, Socket.IO	TCP-based protocol with handshake interpreted by HTTP servers as an Upgrade request	Makes possible more interaction between a browser and a web site, facilitating live content and the creation of real-time apps.

Figure 26: Technology description and role

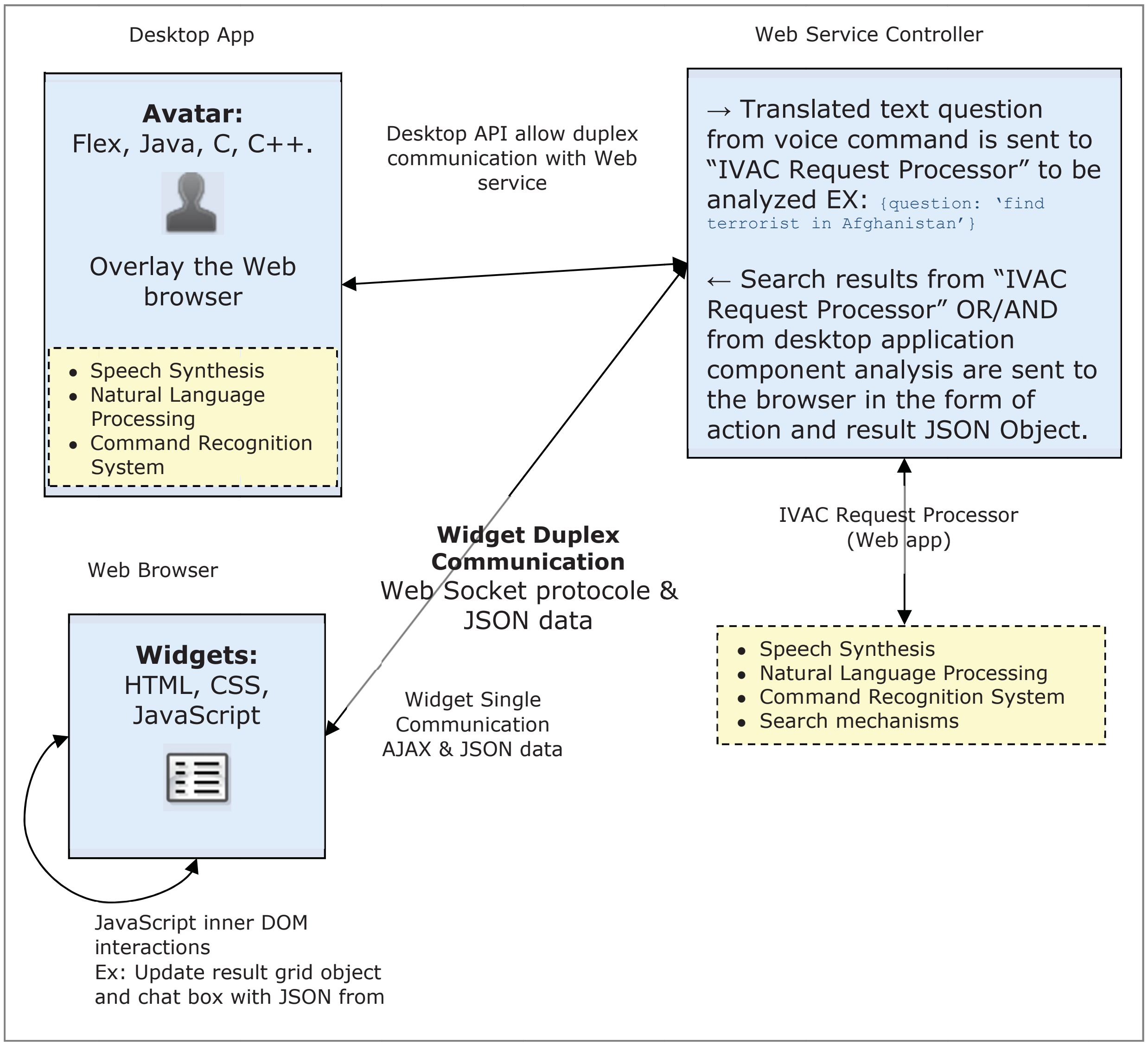
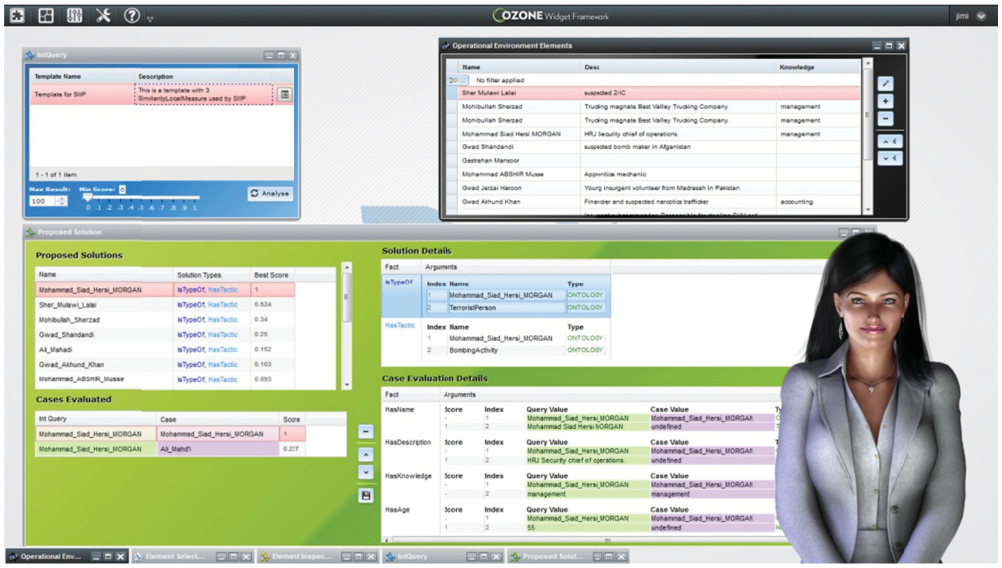

Desktop App
Web Service Controller
Avatar:
Flex, Java, C, C++.
Overlay the Web browser
Speech Synthesis
Natural Language Processing
Command Recognition System
Desktop API allow duplex communication with Web service
→ Translated text question from voice command is sent to "IVAC Request Processor" to be analyzed EX: {question: 'find terrorist in Afghanistan'}
← Search results from "IVAC Request Processor" OR/AND from desktop application component analysis are sent to the browser in the form of action and result JSON Object.
IVAC Request Processor (Web app)
Speech Synthesis
Natural Language Processing
Command Recognition System
Search mechanisms
Web Browser
Widget Duplex Communication
Web Socket protocole & JSON data
Widgets:
HTML, CSS, JavaScript
Widget Single Communication AJAX & JSON data
JavaScript inner DOM interactions
Ex: Update result grid object and chat box with JSON from

Figure 27: Avatar Desktop with Web GUI interactions model summary

This model shows that SS, NLP, SR and other mechanisms (CBR) could be part of the desktop application and/or deploy on a web server.



IntQuery

Template Name	Description
Template for SIIP	This is a template with 3 SimilarityLocalMeasure used by SIIP

1 - 1 of 1 item

Max Result: 100 Min Score: 0

Operational Environment Elements

Name	Desc	Knowledge
No filter applied		
Sher Mulawi Lalai	suspected 2/IC	
Mohibullah Sherzad	Trucking magnate Best Valley Trucking Company.	management
Mohibullah Sherzad	Trucking magnate Best Valley Trucking Company.	management
Mohammad Siad Hersi MORGAN	HRJ Security chief of operations.	management
Gwad Shandandi	suspected bomb maker in Afganistan	
Gastrahan Mansoor		
Mohammad ABSHIR Musse	Apprentice mechanic	
Gwad Jerzai Haroon	Young insurgent volunteer from Madrasah in Pakistan.	
Gwad Akhund Khan	Financier and suspected narcotics trafficker	accounting

Proposed Solutions

Name	Solution Types	Best Score
Mohammad_Siad_Hersi_MORGAN	IsTypeOf, HasTactic	1
Sher_Mulawi_Lalai	IsTypeOf, HasTactic	0.524
Mohibullah_Sherzad	IsTypeOf, HasTactic	0.34
Gwad_Shandandi	IsTypeOf, HasTactic	0.25
Ali_Mahadi	IsTypeOf, HasTactic	0.152
Gwad_Akhund_Khan	IsTypeOf, HasTactic	0.103
Mohammad_ABSHIR_Musse	IsTypeOf, HasTactic	0.093

Cases Evaluated

Int Query	Case	Score
Mohammad_Siad_Hersi_MORGAN	Mohammad_Siad_Hersi_MORGAN	1
Mohammad_Siad_Hersi_MORGAN	Ali_Mahdi	0.227

Solution Details

Fact	Index	Name	Type
IsTypeOf	1	Mohammad_Siad_Hersi_MORGAN	ONTOLOGY
	2	TerroristPerson	ONTOLOGY
HasTactic	1	Mohammad_Siad_Hersi_MORGAN	ONTOLOGY
	2	BombingActivity	ONTOLOGY

Case Evaluation Details

Fact	Score	Index	Query Value	Case Value
HasName	-	1	Mohammad_Siad_Hersi_MORGAN	Mohammad_Siad_Hersi_MORGA
	1	2	Mohammad Siad Hersi MORGAN	undefined
HasDescription	-	1	Mohammad_Siad_Hersi_MORGAN	Mohammad_Siad_Hersi_MORGA
	1	2	HRJ Security chief of operations.	undefined
HasKnowledge	-	1	Mohammad_Siad_Hersi_MORGAN	Mohammad_Siad_Hersi_MORGA
	-	2	management	management
HasAge	-	1	Mohammad_Siad_Hersi_MORGAN	Mohammad_Siad_Hersi_MORGA
	1	2	55	undefined

Figure 28: Visual Example of Desktop Avatar on Top of VOiiLA's OWF Widget in Firefox Browser

Integrability with VOiiLA

★★

Desktop application needs to be installed outside VOiiLA on each client machine.

Capability of dynamically representing the actual state of the iVAC

★★★

The capability depends directly on the capability of the Avatar Services to catch events and communicate status information between the desktop application and the browser. Adding a desktop layer add complexity to the render of query result.
Example: Desktop Avatar should respond to return result by saying "5 results found", while the browser widget should be updated with the JSON results.

Capability to capture user's context

★★★★

Base user's context is provided by getting static information about the logged user. User's interactions with the Web GUI give the system in time information about the context.

The capability of capturing a complex user context such as: face recognition, eye gazing, tone of voice and voice recognition depend directly on the capability of the Avatar Services, third-party software and hardware availability (microphone, camera).

Capability of integrating the avatar

★★★★★

Desktop GUI alternatives allow for a more complex and dynamic render of the Avatar. Limitations come with the capability of the Avatar system to make real-time communication with the GUI and the availability of API connectors.

Examples of possible limitations:

- Avatar representation may not be able to lip-sync text-to-speech sound output because of a missing third-party API communication junction point.
- Low speed of communications between the Web GUI (widgets) and Avatar system may cause lags in Avatar real-time animation.

Capability of integrating SR and SS components

★★★★★

Desktop GUI often includes SR and SS components.

Disadvantages and possible ways of overcoming them

- Full dependency on Avatar system components, API and communication efficiency.

Ease of implementation

★

Mix of desktop and Web GUI alternative is probably the most complex.

Low integration time

★★

Integrate efficient communication within existing desktop application API, Web service and Web widgets will be time consuming.

Availability, licensing and associated cost

Desktop Avatar solutions have a license cost.

8.3.3 Full Desktop GUI Alternative

Table 12: Score Summary of a Full Desktop GUI

Score Summary	
Integrability with VOiiLA	none
Capability of dynamically representing the actual state of the iVAC	★★★★
Capability to capture user's context	★★★★
Capability of integrating the avatar	★★★★★
Capability of integrating SR and SS components	★★★★★
Ease of implementation	★★★
Low integration time	★

Technology description and role

Technology	Description	Role
Desktop Application	Compiled Desktop Client Application	Avatar animation, SR, SS, NTL, Search, etc.

	Bundle. Adobe Air, Java, C, C++.	
JSON, XML	Lightweight text encapsulated representation of data	Data transportation and representation through web services.
Web Socket, Node Server, Socket.IO	TCP-based protocol with handshake interpreted by HTTP servers as an Upgrade request	Makes possible more interaction between a desktop application and a web service, facilitating live content and the creation of real-time apps.

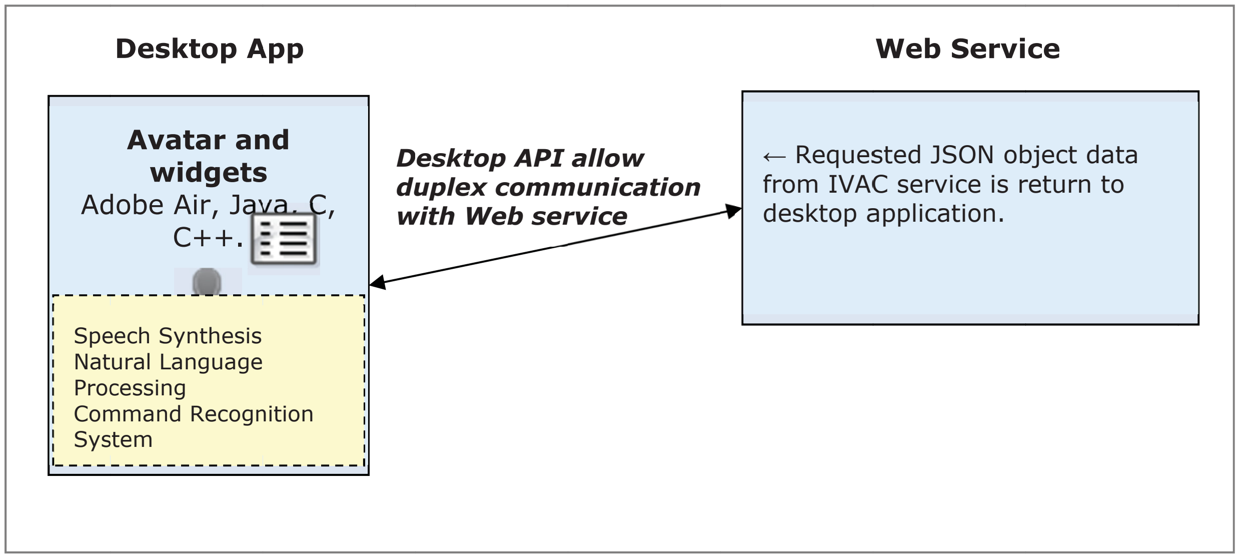

Desktop App
Web Service
Avatar and widgets
Adobe Air, Java, C, C++.
Speech Synthesis
Natural Language Processing
Command Recognition System
Desktop API allow duplex communication with Web service
← Requested JSON object data from IVAC service is return to desktop application.

Figure 29: Full Avatar and widgets desktop model summary

This model shows that SS, NLP, SR and other mechanisms (CBR) could be part of the desktop application. Request could be send to IVAC service to retrieve information needed for desktop search purpose.

Desktop application layout offers more capabilities and performance than Web layout.

Figure 30: Examples of complex desktop application integration with Adobe Air

Integrability with VOiiLA

Desktop application needs to be installed outside VOiiLA on each client machine.

Capability of dynamically representing the actual state of the iVAC

★★★★

The capability depends directly on the capability of the desktop application to communicate with IVAC Web service to gather data for local manipulations.

Capability to capture user's context

★★★★

Base user's context is provided by getting static information about the logged user. User's interactions with the desktop GUI give the system in time information about the context.

The capability of capturing a complex user context such as: face recognition, eye gazing, tone of voice and voice recognition depend directly on the capability of the Avatar Services, third-party software and hardware availability (microphone, camera).

Capability of integrating the avatar

★★★★★

Desktop GUI alternatives allow more complex and dynamic render of the Avatar and widgets.

Here are some advantages of desktop integration versus the Web one:

- Local storage on hard-drive.
- More memory swapping and CPU power availability versus Web browser.
- Access to 2D/3D acceleration card calculation performance.
- Possible deployment on mobile devices in native applications (via Adobe Air).

Capability of integrating SR and SS components

★★★★★

Desktop GUI often includes SR and SS components.

Ease of implementation

★★★

Full desktop GUI alternative offers the simplest integration with third-party software and interactions with hardware devices.

Low integration time

★

Because no desktop integration is possible within VOiiLA web GUI, all widgets should be developed from scratch.Note that this alternative implies that a new desktop application has to be developed.

Availability, licensing and associated cost

There is a license cost for development IDE.For instance, Flash Builder, Flash Catalyst, Flash Professional, and Dreamweaver all provide support for developing and exporting AIR applications.

9 Recommendations

9.1 NLP Component Recommendation

9.1.1 SR Component Recommendation

The open source speech recognition technology that can we recommend (if using open source technologies is a strict requirement) is CMUSphinx. It is a mature technology, with a Java based implementation and a reach API. However, as was discussed with the TA, one of important requirements for the SR functionality is that it works well in most circumstances. The acoustic models coming with the CMUSphinx system were trained on a limited amount of data and the technology does not have implemented some of the more advanced algorithms. As a consequence, the performance "out-of-the-box" is not perfect in all scenarios. The acoustic models can be trained based on more data, and can be adapted to each individual user. However, this would require significant investments in terms of time and money that, in our opinion, is not possible to do within the scope of the current contract.

The state-of-the-art technology of speech recognition today is a commercial Nuance Dragon technology developed by Nuance Communications Inc. It works well out-of-the-box and comes with a reasonable pricing model and a good SDK. It also comes with a good built-in speech synthesis functionality which can be used with the iVAC instead of using another heterogeneous component for that purpose. We recommend this technology as the primary choice for implementing the SR functionality. The only limiting disadvantage of this choice is the availability of this technology on Microsoft Windows OS only. However, since the desktop OS at DRDC Valcartier is Windows, this limitation is not considered as significant.

Another option is Microsoft Speech Platform / SAPI, if the project budget or other considerations do not allow acquiring the Nuance Dragon technology. It is free of charge and it works relatively well out of the box, but in many cases it is significantly less precise than Nuance Dragon.

9.1.2 SS Component Recommendation

All reviewed open source speech synthesis technologies had one significant disadvantage: the sound of the synthesized voice was very artificial and sometimes difficult to recognize. This would be a serious limitation to intelligence analyst's daily work.

Among the commercial technologies, the most naturally sounding voices were observed when testing the Neospeech VoiceText technology. Unfortunately, the vendor was not able to provide us with precise pricing/licensing information. Other comparable technologies in terms of voice quality are Cereproc, Acapela and Nuance Dragon. We recommend Nuance Dragon as the primary choice since it has been recommended as the primary choice of the speech recognition technology for the iVAC.

As in the case of the speech recognition technologies, Microsoft Speech Platform / SAPI could also be considered as a workable alternative to Nuance Dragon in case of budget or other constraints, especially if it will be selected for implementing the speech recognition functionality of the iVAC.

9.1.3 NLU Component Recommendation

Since a friendly, plug-and-play solution for the NLU component is not available, we have laid out three possible approaches, of increasing complexity, which we propose to implement: a **bag-of-words approach**, a **hand-written grammar approach** and a **statistical NLP approach**. Each approach can be understood as a subset of the more complex ones. The simplest one will allow us to deal successfully with the simplest user commands and, as we move to the more sophisticated approaches, we will be able to properly interpret more and more user commands.

Hence we propose a natural progression, in which we would start by implementing the first approach, then move to the second approach and finally spend the rest of available project's resources on perfecting the third and last approach. As pointed out in our report, the third approach is the most sophisticated one. It is at the frontier of modern research in NLP (Watson, Siri, etc.). To implement a statistical NLU component that can understand improvised user commands, large training sets and extensive experimentations would be required. Given the resources made available for this project, it might reveal difficult to implement a component that would reach high performance levels. However, any progress towards this goal will be reflected by an increase in the robustness of the NLU component and will be beneficial to the iVAC system.

Implementing the bag-of-words approach will be trivial and should not require any sophisticated NLP libraries. It only requires processing strings and lists, which is fully supported by any modern programming language.

Implementing the hand-written grammar approach will require work towards (1) defining a grammar over accepted iVAC commands and (2) parsing incoming commands using this grammar. (1) will require that we experiment with different grammars, something our team's expertise will fully allow. For (2), we propose to use the Python NLTK library, which we found to be the most convivial implementation of a hand-written, context free grammar parser. We would thus implement a bridge between the Python library and the iVAC system, which will require expertise that is also fully covered by our team. Hence, we believe that a functional implementation of this component, which would augment the bag-of-words approach and increase the performance of the NLU component, is an achievable objective within the project's time and resource budget.

Finally, time and resource permitting, we recommend the development of a statistical NLP approach that would deal with the most sophisticated user commands. We will rely on the ClearTK and ClearNLP libraries, which cover most of our statistical NLP needs and are in Java (and thus will be straightforward to use with the iVAC system). Our expertise in NLP and Machine Learning will allow us to make progress towards a more robust NLU component.

9.2 Avatar Component Recommendation

No existing avatar technology fully meeting requirements was found during this study. The closest open source option was ClippyJS, but its usage in the iVAC context is not deemed appropriate.

It is therefore recommended to develop a simple avatar in a Web GUI using the jQuery library, based on ClippyJS. The objective of this simple avatar will be to provide feedback on the processing currently being performed by the iVAC system.

9.3 GUI Recommendation

For the iVAC prototype, it is recommended to develop a web-based GUI with proven technologies (HTML, JavaScript, AJAX and web socket). The advantage of this approach is that it will provide complete integration within the VOiiLA environment. Thus, the iVAC prototype could leverage existing widgets developed by DRDC in past and current projects related to the ITI.

9.4 Integration Framework Recommendation

Considering the iVAC system from the performance point of view, the distributed architecture based on UIMA – AS and Hadoop Distributed File System (HDFS) (Primary Candidate Framework 1) is the best choice. However, it is the most complicated to implement, to manage and maintain. Integrating such a system with other applications, systems or services will be very challenging.

The Primary Candidate Framework 3 (SOA with an ESB) would be the best candidate considering

- The long-term objective of the iVAC system
- The requirement to deliver an integration architecture that must be able to integrate different kinds of software systems, applications, services, software libraries and others
- Connecting to different kinds of data sources
- Support different communication protocol.

WSO2 ESB can be used as the backbone of this architecture, but this solution can suffer performance issues that need to be taken into account. Secondly, considering the learning curve needed to master the ESB product in order to be able to correctly develop on it and to exploit its inherent capabilities, we could not recommend this solution for the purpose of the prototype. It can, however, be considered as a long term solution for the iVAC.

According to the analysis above, we recommend the service-based integration approach (Primary Candidate Framework 2) for the prototype of the iVAC system. The development must be done with the evolution through a full SOA system in mind. Everything that needs to be developed for the prototype must be done according to this vision.

10 Conclusion

iVAC, the Intelligent Virtual Analyst Capability, is intended to be a military intelligence analyst's virtual assistant. The first step for building such a capability is to develop a technological framework consisting of:

1. A Natural Language Processing (NLP) component capable of listening to the user's speech, recognize the text, translate the text into the commands, and talk to the user using a synthetized voice.
2. An Avatar component capable of mimicking (or visually "saying") the utterance generated by the speech synthesis capability and producing a certain number of expressions based on different variables.
3. A fast and scalable software backbone capable of integrating iVAC components, data sources and other DRDC Valcartier services (e.g., ISTIP and VOiiLA).
4. A flexible GUI supporting the use and demonstration of different iVAC components.

In this report, we presented the results of a research conducted to compare various candidate technologies and approaches that can be used to build the technological framework of the iVAC. These include:

- Speech synthesis technologies,
- Speech recognition technologies,
- Natural language understanding approaches,
- Avatar technologies,
- Software integration frameworks,
- and others.

Based on our experts' knowledge and expertise, we selected technologies and approaches as candidates for implementing the technological framework of the iVAC. The present report will serve as the basis for the next steps of the work on the implementation of the iVAC which consist in the development planning and the development of the framework.

11 Acronyms

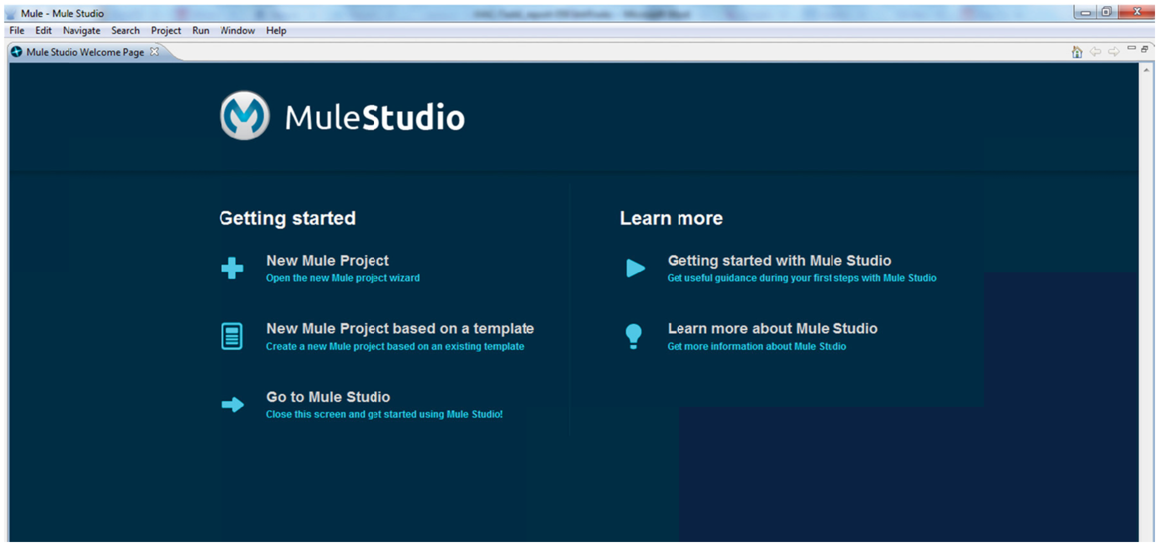
Acronym	Definition
AGPL	Affero General Public Licence
AIML	Artificial Intelligence Markup Language
AJAX	Asynchronous JavaScript and XML
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
BPEL	Business Process Execution Language
BSD	Berkeley Software Distribution
CAS	Common Analysis Structure
CBR	Case Based Reasoning
CFG	Context Free Grammar
CPAL	Common Public Attribution License
CPM	Collection Processing Manager
CPU	Central Processing Unit
CRUD	Create, Read, Update and Delete
CSS	Cascading Style Sheet
DFA	Deterministic Finite state Automaton
DLL	Dynamic Link Library
DOM	Document Object Model
DRDC	Defence Research and Development Canada
EAI	Enterprise Application Integration
ESB	Enterprise Service Bus
FIX	Financial Information Exchange
GRMM	Graphical Models in Mallet
GUI	Graphical User Interface
GWT	Google Web Toolkit
HDFS	Hadoop Distributed File System
HTK	Hidden Markov Model Toolkit
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IE	Internet Explorer

Acronym	**Definition**
IMAP	Internet Message Access Protocol
ISTIP	Intelligence Science and Technology Integration Platform
IT	Information Technology
ITI	Integration Technology Infrastructure
iVAC	Intelligence Virtual Analysis Capability
JAX-RS	Java XML RESTful Service
JAX-WS	Java XML Web Service
JAXB	Java Architecture for XML Binding
JCA	Java Connector Architecture
JDBC	Java Database Connectivity
JDK	Java Development Kit
JEE	Java Enterprise Edition
JMS	Java Message Service
JMX	Java Management Extensions
JPA	Java Persistence API
JS	JavaScript
JSAPI	Java Speech API
JSGF	JSpeech Grammar Format
JSON	JavaScript Object Notation
JTA	Java Transaction API
JTS	Java Transaction Services
JVM	Java Virtual Machine
LE	Language Engineering
LIBSVM	Library for Support Vector Machines
MALLET	MAchine Learning for LanguagE Toolkit
MDB	Message Driven Bean
MS	Microsoft
MVC	Model, View, Controller
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
NLU	Natural Language Understanding
OS	Operating System

Acronym	Definition
OWF	Ozone Widget Framework
PCFG	Probabilistic Context Free Grammar
PNG	Portable Network Graphics
POP	Post Office Protocol
POS	Part-of-Speech
POX	Plain Old XML
REST	Representational State Transfer
SaaS	Software as a Service
SAPI	Speech Application Programming Interface
SDK	Software Development Kit
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOI	Service Oriented Infrastructure
SR	Speech Recognition
SS	Speech Synthesis
SWF	Small Web Format
TCP	Transmission Control Protocol
TTS	Text to Speech
UI	User Interface
UIA	Unstructured Information Analytics
UIMA	Unstructured Information Management Applications
UIMA AS	Unstructured Information Management Applications Asynchronous Scaleout
UK	United Kingdom
UKP	Ubiquitous Knowledge Processing
US	United States
USD	United States Dollar
VB	Visual Basic
VOiiLA	Visionary Overarching Interaction Interface Layer for Analysis
WCF	Windows Communication Framework
WS-I	Web Service Interoperability

Acronym	Definition
WSDL	Web Services Description Language
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

Annex A – Mule ESB Development Tool


Mule - Mule Studio
File Edit Navigate Search Project Run Window Help
Mule Studio Welcome Page
MuleStudio
Getting started
New Mule Project
New Mule Project based on a template
Go to Mule Studio
Learn more
Getting started with Mule Studio
Learn more about Mule Studio

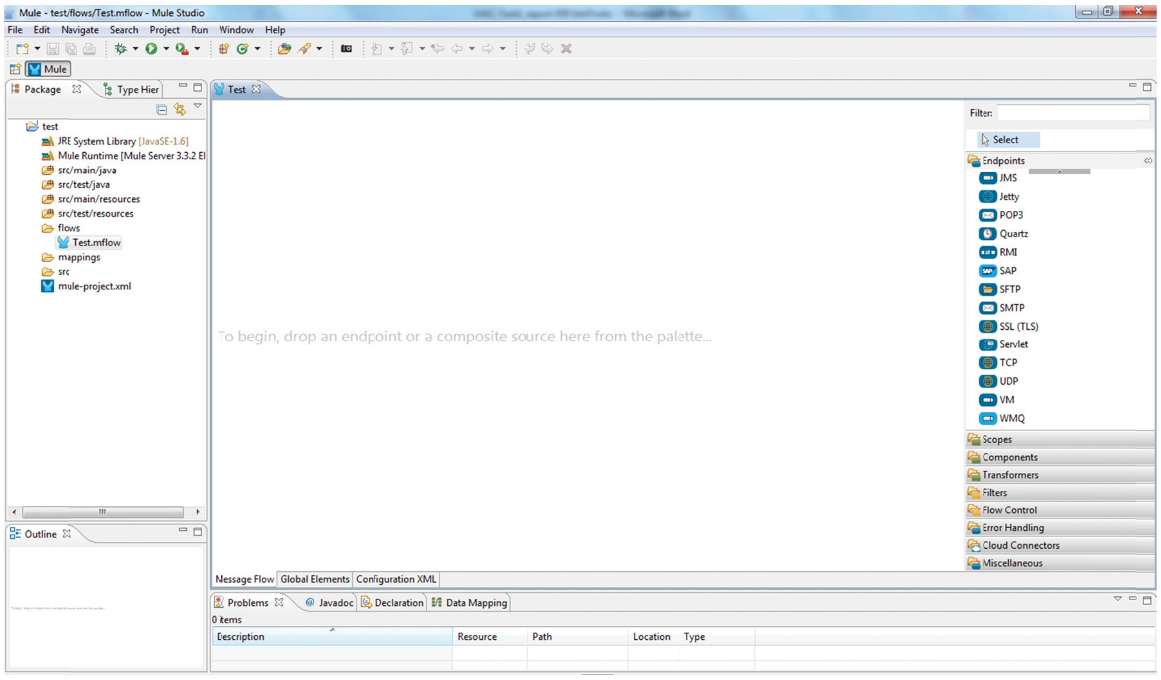
Open the new Mule project wizard

Create a new Mule project based on an existing template

Close this screen and get started using Mule Studio!

Get useful guidance during your first steps with Mule Studio

Get more information about Mule Studio



To begin, drop an endpoint or a composite source here from the palette...

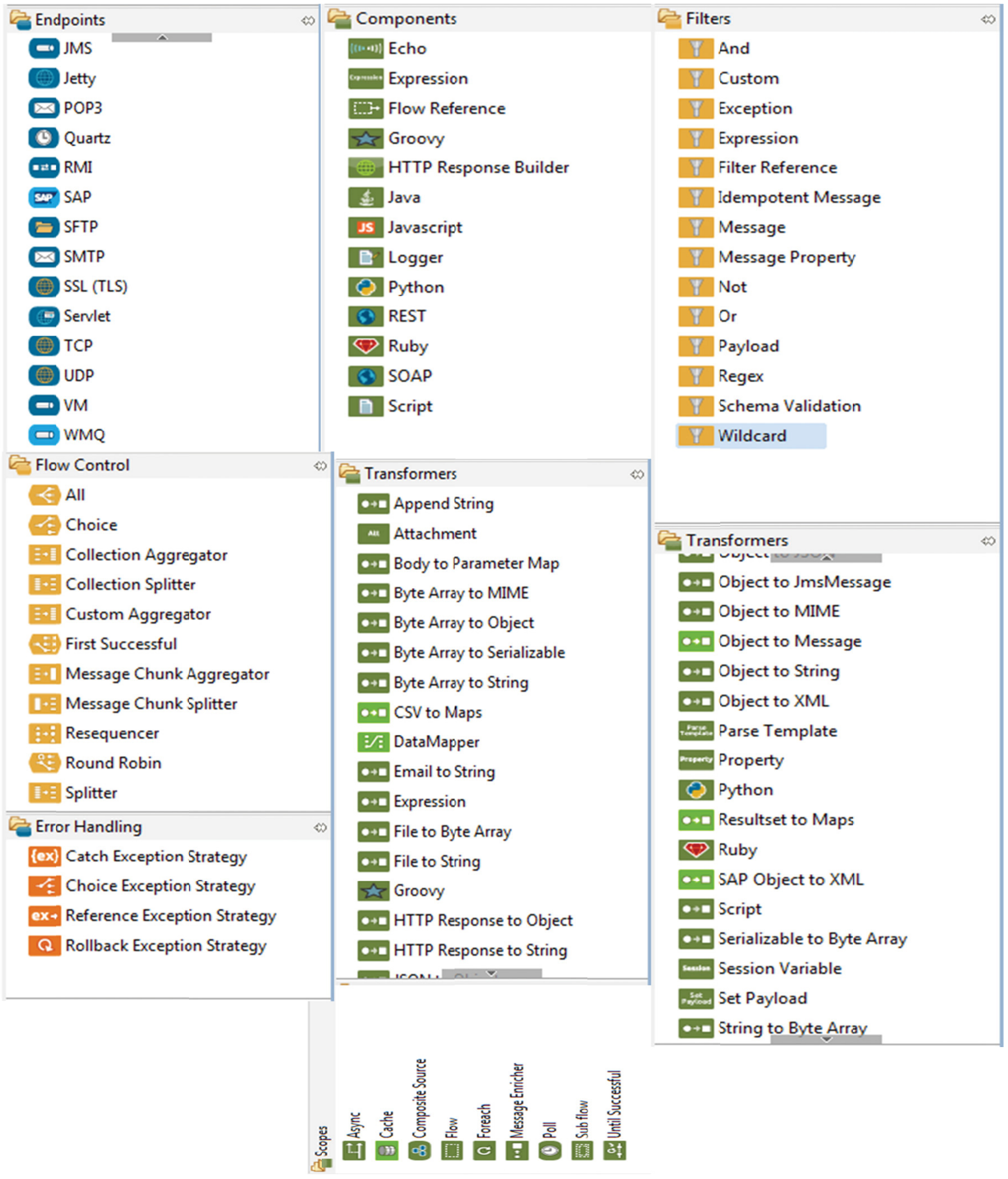
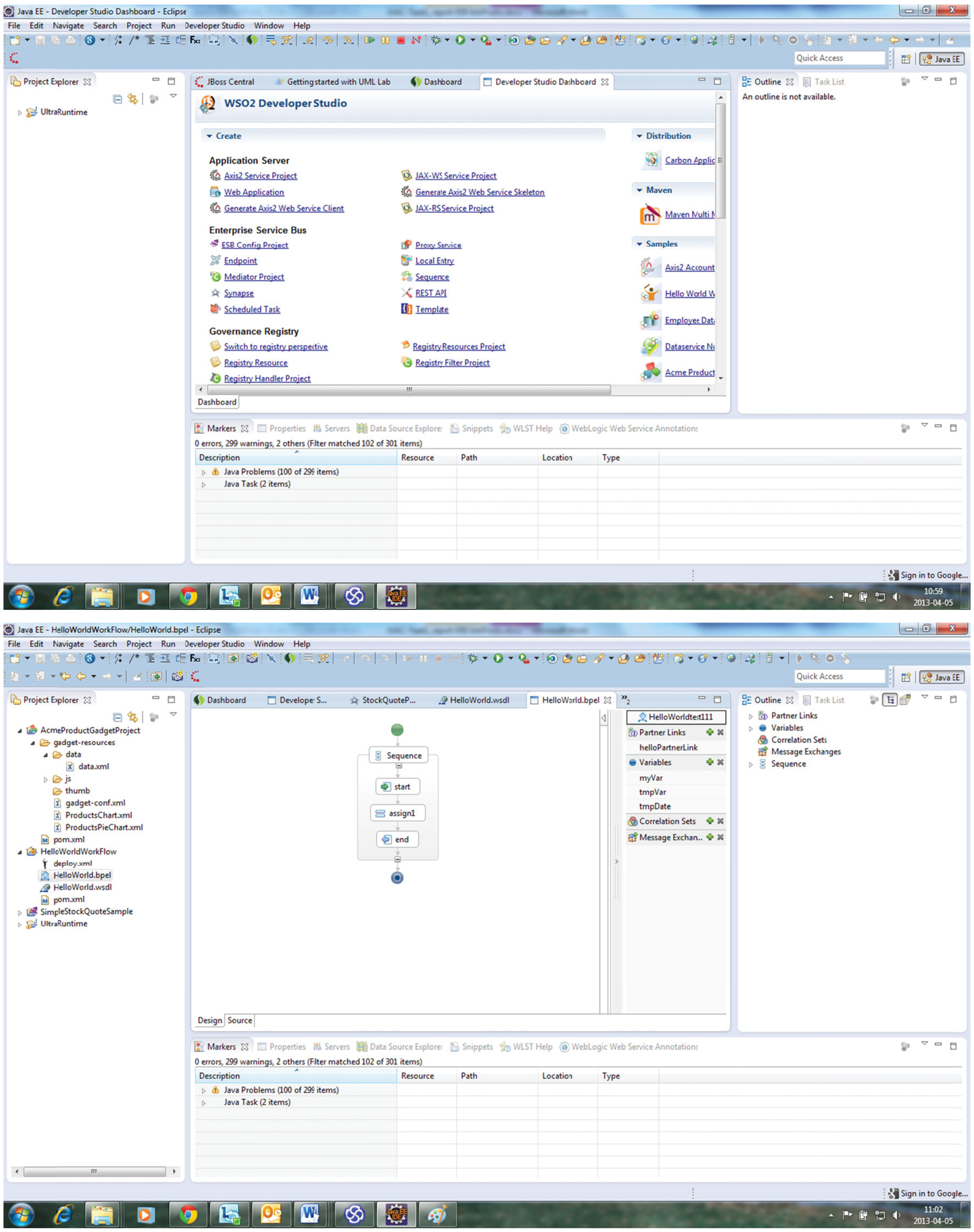
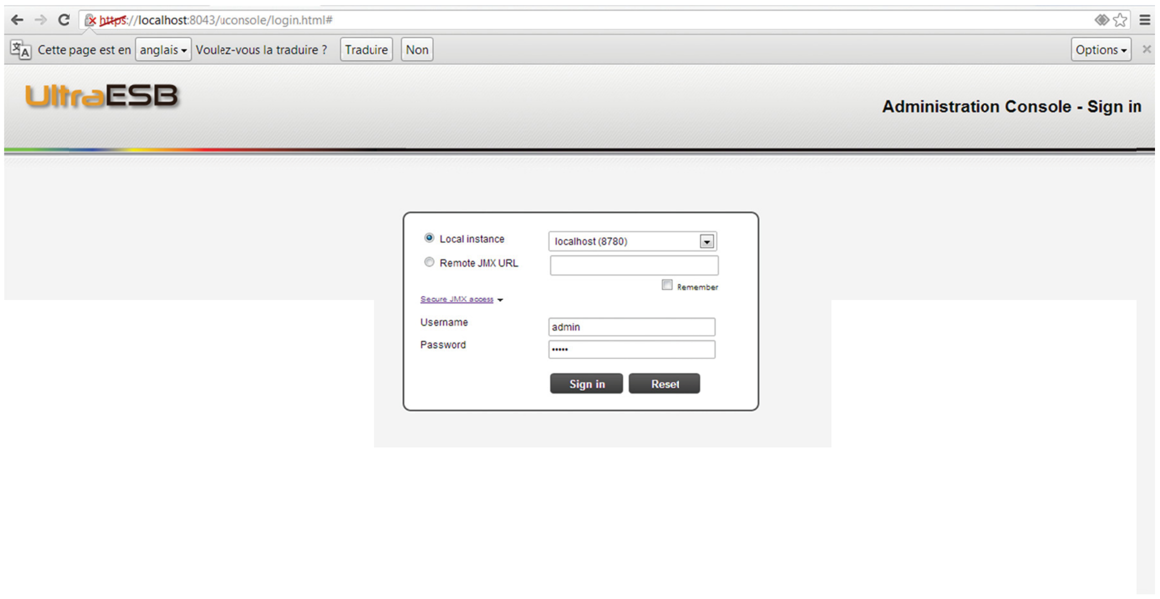


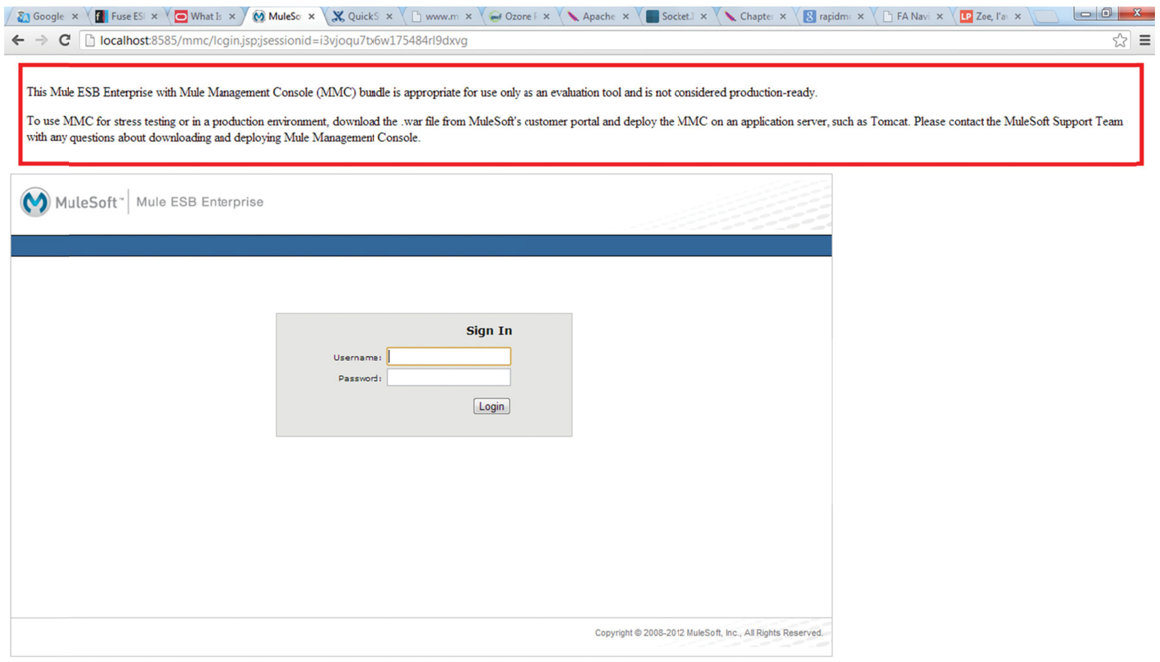
Figure 31: Mule Studio Development Tools



0 errors, 299 warnings, 2 others (Filter matched 102 of 301 items)

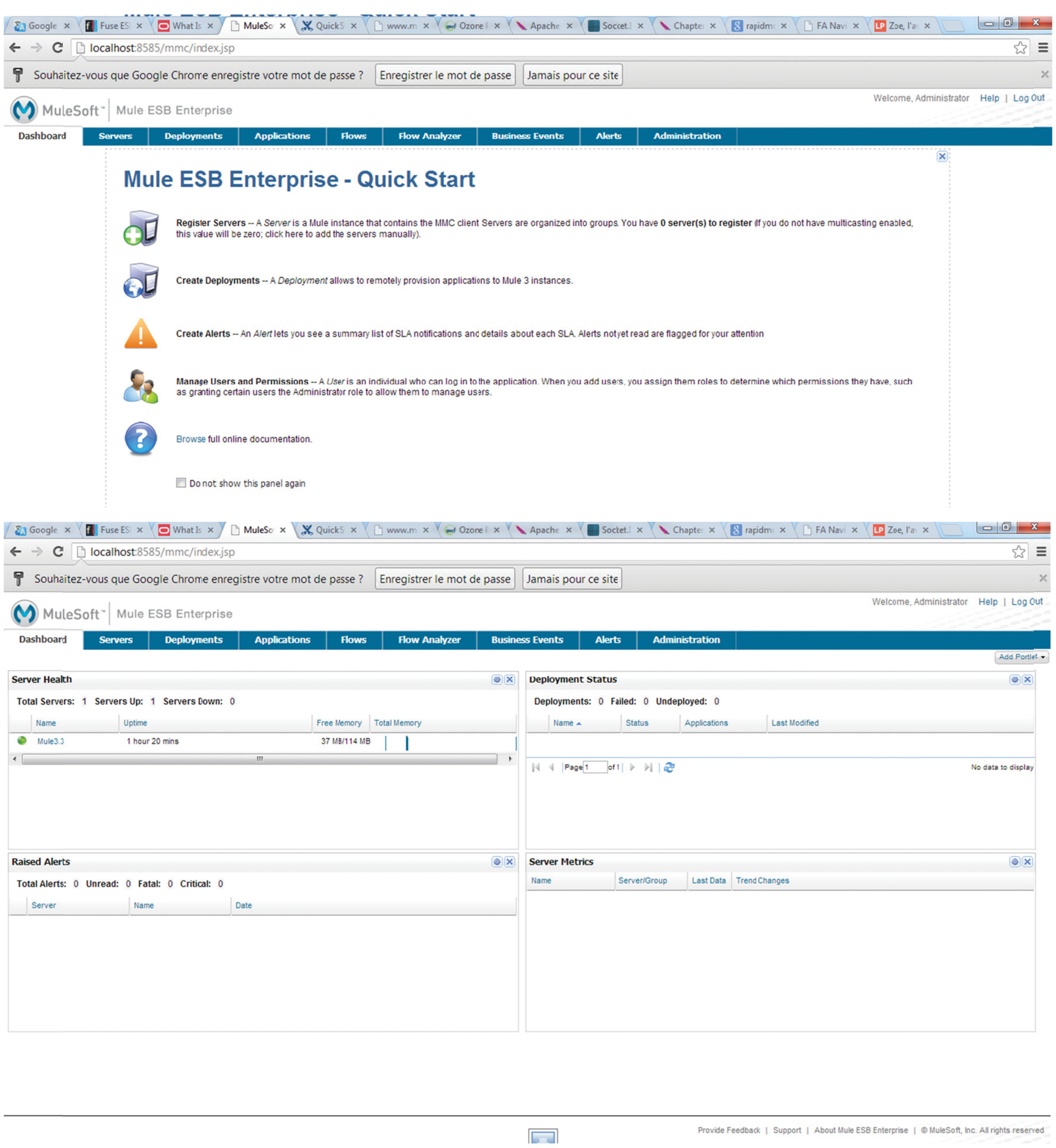
Figure 32: WSO2 - Development Tools


https://localhost:8043/uconsole/login.html#
Cette page est en anglais Voulez-vous la traduire ? Traduire Non Options
UltraESB
Administration Console - Sign in
Local instance
localhost (8780)
Remote JMX URL
Remember
Secure JMX access
Username
admin
Password
Sign in
Reset


localhost:8585/mmc/login.jsp;jsessionid=i3vjoqu7b6w175484rl9dxvg
MuleSoft
Mule ESB Enterprise
Sign In
Username:
Password:
Login
Copyright © 2008-2012 MuleSoft, Inc., All Rights Reserved.

This Mule ESB Enterprise with Mule Management Console (MMC) bundle is appropriate for use only as an evaluation tool and is not considered production-ready.

To use MMC for stress testing or in a production environment, download the .war file from MuleSoft's customer portal and deploy the MMC on an application server, such as Tomcat. Please contact the MuleSoft Support Team with any questions about downloading and deploying Mule Management Console.


localhost:8585/mmc/index.jsp
Souhaitez-vous que Google Chrome enregistre votre mot de passe ?
Enregistrer le mot de passe
Jamais pour ce site
MuleSoft
Mule ESB Enterprise
Welcome, Administrator Help | Log Out
Dashboard
Servers
Deployments
Applications
Flows
Flow Analyzer
Business Events
Alerts
Administration
Add Portlet
Provide Feedback | Support | About Mule ESB Enterprise | © MuleSoft, Inc. All rights reserved

Mule ESB Enterprise - Quick Start

Register Servers -- A *Server* is a Mule instance that contains the MMC client Servers are organized into groups. You have **0 server(s) to register** (if you do not have multicasting enabled, this value will be zero; click here to add the servers manually).

Create Deployments -- A *Deployment* allows to remotely provision applications to Mule 3 instances.

Create Alerts -- An *Alert* lets you see a summary list of SLA notifications and details about each SLA. Alerts not yet read are flagged for your attention

Manage Users and Permissions -- A *User* is an individual who can log in to the application. When you add users, you assign them roles to determine which permissions they have, such as granting certain users the Administrator role to allow them to manage users.

Browse full online documentation.

☐ Do not show this panel again

Server Health

Total Servers: 1 Servers Up: 1 Servers Down: 0

Name	Uptime	Free Memory	Total Memory
● Mule3.3	1 hour 20 mins	37 MB/114 MB	

Deployment Status

Deployments: 0 Failed: 0 Undeployed: 0

Name	Status	Applications	Last Modified

Page 1 of 1 No data to display

Raised Alerts

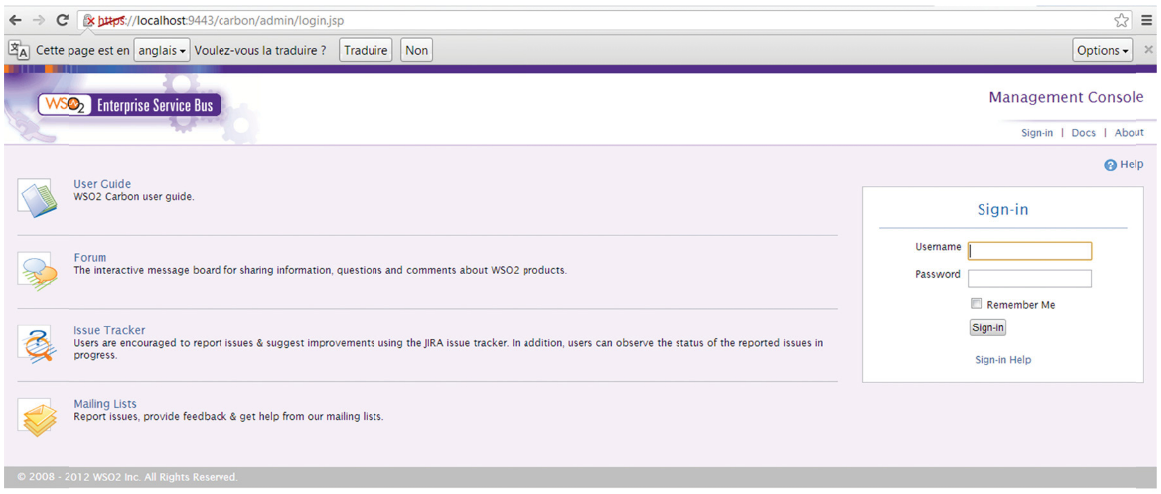
Total Alerts: 0 Unread: 0 Fatal: 0 Critical: 0

Server	Name	Date

Server Metrics

Name	Server/Group	Last Data	Trend Changes

Figure 33: Mule ESB Admin Console



~~https~~://localhost:9443/carbon/admin/login.jsp

Cette page est en anglais Voulez-vous la traduire ? Traduire Non Options

Management Console

Sign-in | Docs | About

Help

User Guide
WSO2 Carbon user guide.

Forum
The interactive message board for sharing information, questions and comments about WSO2 products.

Issue Tracker
Users are encouraged to report issues & suggest improvements using the JIRA issue tracker. In addition, users can observe the status of the reported issues in progress.

Mailing Lists
Report issues, provide feedback & get help from our mailing lists.

Sign-in

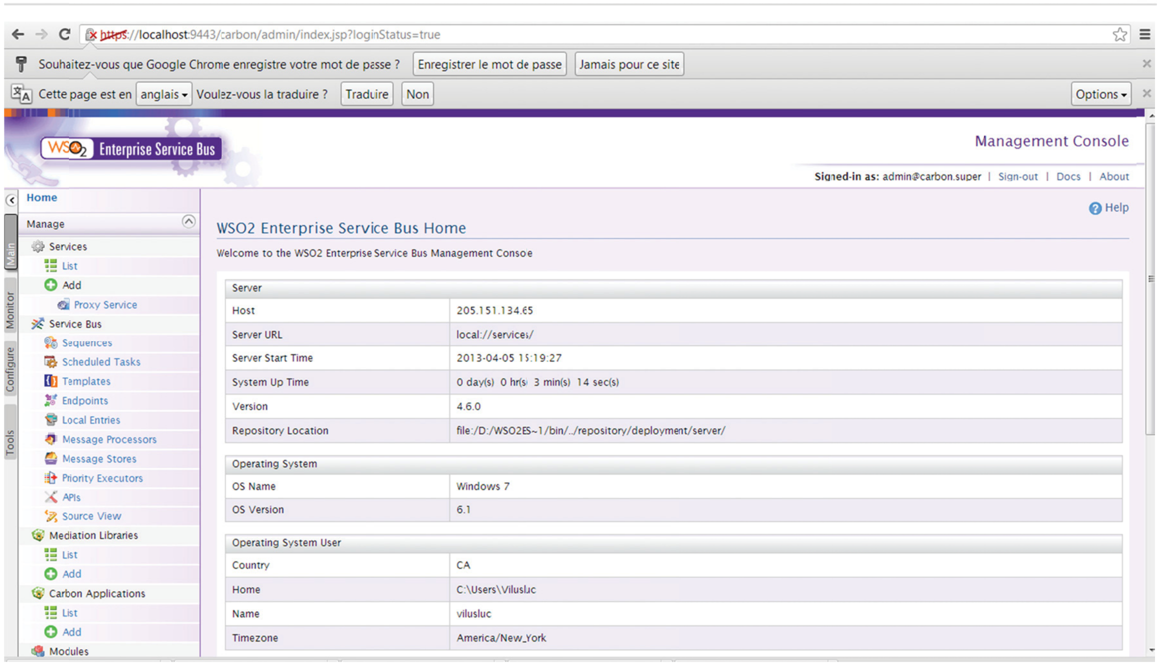
Username	
Password	

Remember Me

Sign-in

Sign-in Help

© 2008 - 2012 WSO2 Inc. All Rights Reserved.



~~https~~://localhost:9443/carbon/admin/index.jsp?loginStatus=true

Souhaitez-vous que Google Chrome enregistre votre mot de passe ? Enregistrer le mot de passe Jamais pour ce site

Cette page est en anglais Voulez-vous la traduire ? Traduire Non Options

Management Console

Signed-in as: admin@carbon.super | Sign-out | Docs | About

Help

WSO2 Enterprise Service Bus Home

Welcome to the WSO2 Enterprise Service Bus Management Console

Server	
Host	205.151.134.65
Server URL	local://services/
Server Start Time	2013-04-05 15:19:27
System Up Time	0 day(s) 0 hr(s) 3 min(s) 14 sec(s)
Version	4.6.0
Repository Location	file:/D:/WSO2ES~1/bin/../repository/deployment/server/

Operating System	
OS Name	Windows 7
OS Version	6.1

Operating System User	
Country	CA
Home	C:\Users\Vilusluc
Name	vilusluc
Timezone	America/New_York

Figure 34: WSO2 Admin Console