A CEA Practical Implementation: Activity Box

Project submitted in partial fulfillment of the requirements for the

Degree of

Master of Engineering in Technology Innovation Management

By: Daniel Cardenas

For: Professor Steven Muegge

Carleton University, Ottawa, Ontario, Canada

December 2009

Abstract

This report examines the implementation of Communication-Enabled-Applications (CEA) features within an existing commercial application called Activity Box, using the assets provided by the Coral CEA sandbox. Activity Box is an online system that manages the reservation of recreational activities, and it has been in operation since December 2008. The project evaluates the implementation within Activity Box of two CEA capabilities named Call-A-Customer and Call-An-Operator, both features associated with particular events in the application. To effectively communicate with Coral CEA modules without negatively affecting Activity Box, we have created a Windows service that acts like an agent whose job is to buffer all interactions between the Activity Box and Coral CEA platform. This agent is offered as a contribution to the Coral CEA ecosystem so other members can take advantage of it.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. The Company

Rezact Inc. is a small company located in Mont-Tremblant, Quebec, a city famous for being one of the biggest ski resorts in the east of North America. Relatively new, Rezact started its operations in 2006 with the purpose of designing and implementing a new reservation system that could allow its main client, The Activity Center, to manage the reservation of different types of recreational activities offered in the region. Rezact currently employs three people.

1.2. Business Model

The business model on which Rezact, and consequently Activity Box, operate, is rather simple, and involves three entities. On one side, there are customers staying in the resort for a brief period of time, usually a week or less. These customers are mostly composed of families wishing to do something more during their stay in the resort besides skiing. On the other hand, there are small companies (called Operators), usually family-operated, that provide recreational activities to these customers. The operators are spread throughout a relatively wide geographical area around the resort, and have historically faced the challenge of reaching potential customers and attracting them to their businesses. The activities they provide can be as diverse as spa and massage packages, dog sledding, horseback ridings, helicopter tours, etc., but the common factor among them is their dependency on resources with limited capacity, like a dog sled that can accommodate up to three people, or a

massage experience that requires a certified massage therapist. The total capacity of all resources is what determines the availability of a certain activity at any given date and time.

To provide a better experience to their customers, a few years ago the resort created a company called The Activity Center whose mission it is to centralize on a single point all tasks related to the reservation of activities. The Activity Center behaves as a reseller, selling activities to customers on behalf of the operators while charging them a commission for the service. Under this model, operators that would otherwise struggle to attract customers can reach a lucrative segment for a small commission, and the resort can keep its customers satisfied. The model is shown in figure 1.

Until December 2008, the reservation of activities was a manual process: it involved the reseller communicating with each and every one of the operators by phone every time a customer was interested on reserving a particular activity, only to verify that the operator had enough availability to serve the number of participants required by the customer. Once the availability had been established, the reseller continued the negotiation with the customer until final payment was processed. Then, a confirmation was manually sent by fax to the operator of the chosen activity indicating the details of the new reservation. This was a tedious and slow procedure that negatively affected sales, especially during high season periods when customer demand increases exponentially compared to regular seasons. Activity Box was created to provide a common platform, shared by both the reseller and the operators

that could control resource availability and increase revenues by allowing more reservations to be placed in the network.

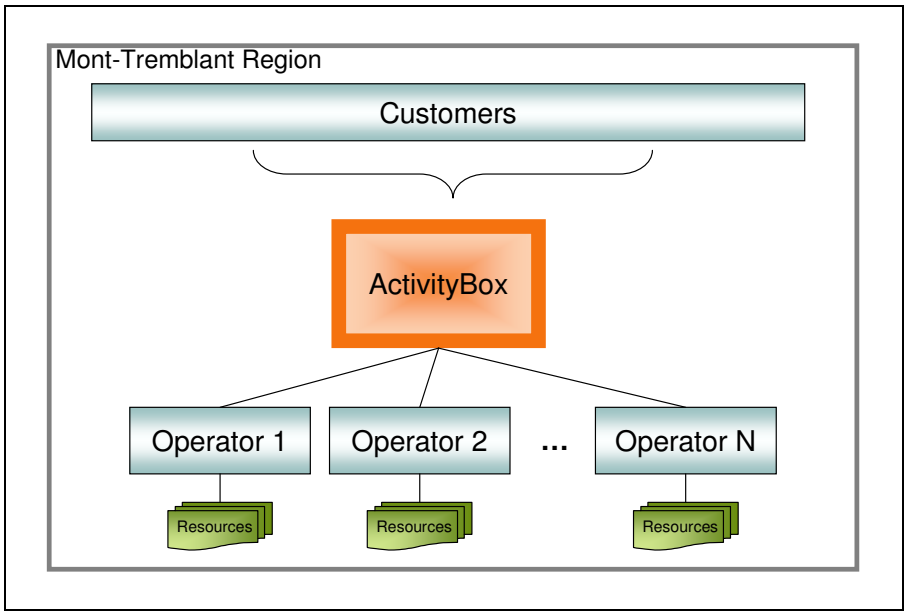

Figure 1. Rezact Business Model

1.3. Activity Box

Activity Box is an online reservation system that is currently being used in the Mont-Tremblant area in Quebec. Activity Box manages reservations for various types of recreational activities, including events with a venue capacity (e.g. concerts, races), activities requiring pieces of equipment (e.g. dog sledding, horseback riding), activities requiring people such as guides or instructors (e.g. rock climbing, guided tours), and activities requiring both people and equipment (e.g. spa services that require a massage therapist and a room). The activities are managed by local operators, most of them small companies that have the know-how and the resources to operate a particular activity. Activity Box helps them to control the inventory of

resources allocated to all of their activities defined within the system. It currently serves more than 33 operators managing over 300 activities that are sold by a network of 30 resellers. Since Activity Box was launched in December 2008, it has processed more than 15,000 reservations representing over 3 million dollars.

The system has been developed following the Software-as-a-Service paradigm, which means that Activity Box is offered as service rather than a product, and charges its users a commission for its utilization. Under this approach, the system can support an unlimited number of regions, operators and resellers regardless of their physical location. In consequence, the application can be used not only in the Tremblant area, but in every tourist destination (like Vancouver or Florida) where activities are provided. The extended model is shown in figure 2.

Activity Box currently consists of a reseller module that allows reservation agents to book activities on behalf of the customers. This same module is used by operators to check their reservations, make reservations for themselves and some other administrative tasks. Future plans include online reservation kiosks that will be deployed in a particular area that will allow customers to reserve their own activities as well as a public online portal to enable customers to reserve activities from any place even before they arrive at their destinations[1].

[1] It should be mentioned that a commercial deployment of Activity Box utilizing Coral CEA assets is already planned. However, this implementation is currently (as of December 2009) beyond the scope of the project itself

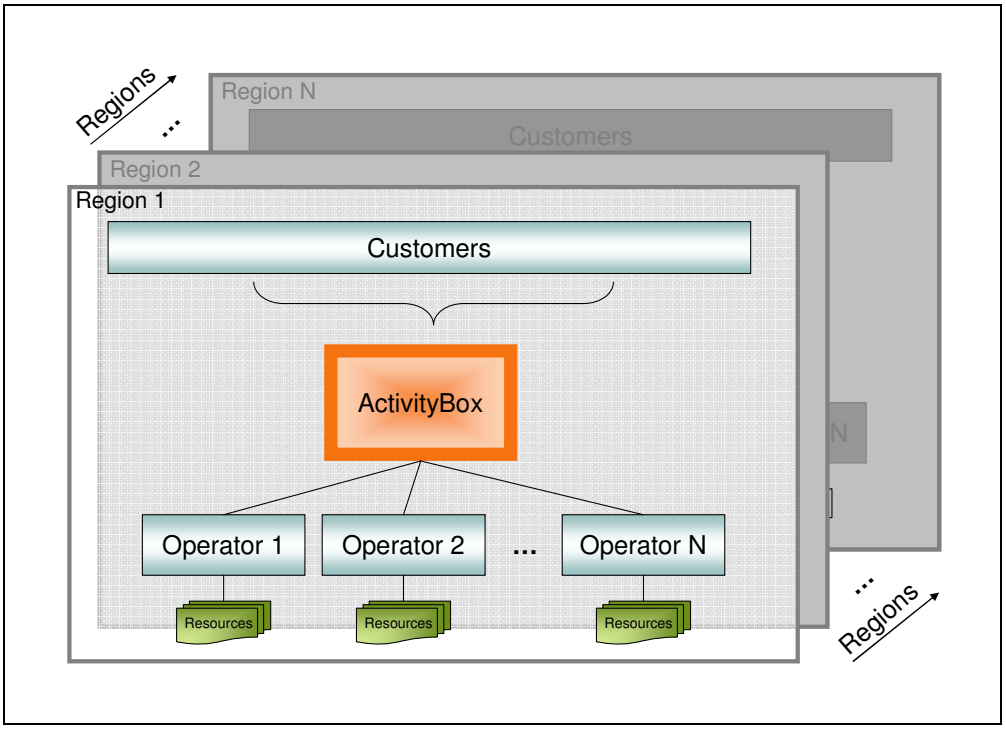

Figure 2. Activity Box Extended Model

1.4. Coral CEA

Coral CEA is a not-for-profit organization that assists companies with the implementation and commercialization of Communications-Enabled Applications (CEA). Coral CEA is part of a business ecosystem anchored around CEA functionalities that are offered as building blocks, out-of-the-box components that link the capabilities and intelligence of networks platforms with the power of current applications to provide a new set of features and functionalities. Coral CEA counts amongst its founding members companies with extensive technology expertise, like IBM, Nortel and Eclipse, and organizations like Carleton University and the Information Technology Association of Canada (ITAC).

1.5. Organization of this Report

This report is organized in six major sections:

- Section 2: Project Definition and Scope presents the goal and expected outcomes of the project, as well as its relevance to the parties involved. It also details the rationale behind each of the decisions taken while defining the scope of the project and the development platform
- Section 3: The Literature review presented on this report is constrained by only those subjects directly related to the project. In this section we present a brief analysis of business ecosystems and how small companies can leverage its value
- Section 4: This section presents a high level design of the project. We describe the limitations found in the Coral CEA platform and how we solved them for the purposes of the demo. One of the key elements of the solution, the Coral CEA Agent, is also presented in this section
- Section 5: Project Implementation describes in great detail how each one of the elements that comprise the solution was implemented. This section shows how to create a proxy class based on Coral CEA service definitions, the implementation details of the agent used to handle the communication with Coral CEA services, and the structure of the table that links client applications with the agent
- Section 6: This final section presents our conclusions and the recommendations based on our findings during the project, for both the sponsor company and Coral CEA platform

2. Project Definition and Scope

This section presents the key elements that define the project and its scope. We describe in detail what the deliverables are, why are they important to both Rezact and Coral CEA and how the project can contribute to these stakeholders. Finally, we analyze the different alternatives we considered when we defined the scope and platform of the project, which we believe could be valuable for future implementers of Coral CEA features.

2.1. Objective

The main objective of the project is to analyze the implications and the technical and business-related steps required to implement CEA capabilities to an existing commercial application called Activity Box using Coral CEA assets.

2.2. Deliverables

This project has three clearly defined deliverables:

1. A demo application that will consist of a limited version of Activity Box implementing the following two CEA functionalities on a testing environment:

- ✓ Call-A-Customer: Sometimes reservations get cancelled because of unforeseen reasons, like poor weather or a broken piece of equipment. When this event happens, the customer needs to be immediately notified so they can decide whether to rebook for another date or to receive a refund for the value of the cancelled activity. Customer satisfaction relies on the

effectiveness and promptness of these communications, so we propose an automatic notification to customers that a reservation has been cancelled

- ✓ Call-An-Operator: This is the reverse of the previous scenario. Here, a customer cancels a reservation and the operator needs to be notified. This particular case is especially important when cancellations are done at the last moment in high season, where the operator is most likely operating at the edge of its capacity. If they are promptly notified of the event, they can react accordingly and allocate the newly freed resource for newly arriving customers

2. The current implementation of Coral CEA methods has certain limitations that could negatively affect a client application. To overcome them, we have developed a software agent that will act as a buffer between any client application and Coral CEA platform. This agent takes the form of a Microsoft Windows service, and its function is to periodically look for new call requests from the application and, when found, to forward them to the corresponding Coral CEA APIs. It will also update back to the application the status of any requested call

3. This report actually constitutes the third deliverable of the project. Here we will detail not only the steps, problems and solutions found during the implementation phase of the project, but also the rationale behind the decisions on each phase. More technical notes will also be included in the appendixes

2.3. Relevance

2.3.1. Relevance to Coral CEA

- ✓ Instead of creating a demo starting from the Coral CEA platform, this project follows the opposite approach. We take an existing, commercial application and we create a demo based on that. This allows us to put into practice CEA concepts and benefits using Coral CEA assets on a real application
- ✓ The reservation of recreational activities is the forgotten area in the tourism industry that represents a huge market in terms of revenue. We hope that by showing a practical implementation of Coral CEA assets, more members will see the benefit and become themselves members of the ecosystem, thus increasing the value of the Coral CEA platform

2.3.2. Relevance to Rezact

- ✓ By eventually implementing the CEA features described before in the production environment, we could greatly improve the quality of the service Activity Box provides to its users, both customers and operators. Rezact will benefit from having the testing phase of this project carried out before even attempting to do it for real
- ✓ Placing a link to a demo version of Activity Box on Coral CEA portal, we are increasing the exposure of the application, which in turn could bring more clients to start doing business with Rezact

- ✓ Rezact is a small company, and as such, it would have had great difficulty having access to the specialized set of assets that Coral CEA offers. The opportunity to work hands-on with this new technology certainly places Rezact as a company one step beyond its competition

2.4. Contributions

2.4.1. For Rezact

This project demonstrates new product capabilities that will be greatly valued by Rezact's users, and develops knowledge and expertise to enable rapid commercial deployment. Implementing communication-enabled features could increase the efficiency of the entire reservation process by providing convenience and new capabilities to customers and activity operators. No competitor currently offers this advantage. In summary, we think that Rezact can benefit from this project in the following ways:

- ✓ Rezact will gain from the opportunity to test Coral CEA features embedded into Activity Box for future deployment in production. As we will see later, this testing will enable the company to evaluate the decision whether or not to implement Coral CEA features in the future
- ✓ At the present stage, Rezact can greatly benefit from having a new set of "tangible" features. By showing how customer service can be improved if CEA capabilities were enabled, Rezact could increment its perceived value to potential investors

- ✓ As part of this project, Rezact will receive the equivalent of an implementation plan that will facilitate any eventual migration from the demo to the production environment

2.4.2. For Coral CEA

By the time this project is finished, Rezact and Activity Box will be amongst the first external users of Coral CEA building blocks. Having lead users that can pave the way for other members to follow is a sign of a healthy ecosystem. We believe that we can contribute to Coral CEA ecosystem health in the following ways:

- ✓ A practical attempt to implement Coral CEA features from "top-to-bottom", that is to say, utilizing an existing commercial application as the starting point to create a demo, will give us a deep understanding of the complete set of procedures needed for any company to successfully implement CEA capabilities. Moreover, it will enable us to provide to Coral CEA with some recommendations or suggestions about functionalities that would be good to have in order to increase the value of its services
- ✓ The client agent that is part of our deliverables is designed in such a way that it can be easily implemented by any other company trying to interact with Coral CEA APIs. Additionally, this tool could also serve as a general framework for Coral CEA itself to implement the agent's buffer mechanism as part of its own service should it desire to do so

2.5. Scope and Platform Definitions

In this section, we will describe the rationale behind some of the decisions taken when defining the scope of the project, whose main deliverable was a demo application that needed to satisfy two basic requirements:

1) To show how a company could solve real problems with the utilization of CEA capabilities exposed by the Coral CEA sandbox

2) To show other potential Coral CEA members the value of using CEA assets within their organizations.

Furthermore, the demo needed to be small enough so that a non-trained user could easily navigate his way through the system, but at the same time complete enough so that the user could appreciate the benefits obtained by using CEA capabilities. Considering these two factors, a decision was made to replicate a reduced version of the current system in a testing environment. The demo of the system will have reduced functionality in order to facilitate user interaction, but it will be using all the internal components that are used by the real application. This approach had the advantage of requiring a simple reduction of user interface elements by keeping the internal elements that would eventually be migrated to a production environment. The demo would show the following two CEA features implemented into the system: Call-A-Customer and Call-A-Operator, already explained in section 2.2. A third CEA feature involving video conferencing, which was intended to be used by online kiosks,

was discarded because there is currently no real system that could take advantage of such a feature[2].

For the implementation of the demo, the platform in which Activity Box had been developed had to be considered. Activity Box is an online web application that was developed using Asp.Net and a MS-SQL 2005 database. The majority of the application logic resides on the database, so it was paramount to be able to reutilize the database layer. This approach had the following advantages:

a) Reduces the implementation time since all the availability and reservation routines are already coded

b) It easy modification of any information (operators, start times, activities, etc) using the existing setup interface pointing to the demo database

c) There is no need to acquire a new license if the database is installed on Carleton or Coral CEA servers. A free reduced version of MS-SQL server (called Express Edition) is available for download

The main disadvantage of this approach is that a windows-based computer would be needed to host the database, which could be a scare resource.

For the demo application itself, the following options were considered:

1) To use a completely different programming language, like Ruby. The application could be then hosted on the cloud. Learning a new programming

[2] Online kiosks that allow customers to book their own activities are planned to be developed and deployed in 2010

language was considered to be a benefit. A disadvantage was the reduced portability of the demo to the production environment of the real system

2) To use Asp.Net platform to be hosted on Carleton servers. This would require a windows-based computer externally accessible where IIS could be installed

3) Deploy the demo on Rezact's server using a public URL based on the current system (coraldemo.activitybox.ca). This approach had the main advantage of letting us test the feasibility of using Coral APIs in the real system and facilitating the migration of programmed CEA features from the demo to the production environment.

Ian Bothwell, Application Architect and Lead Technician at Coral CEA, was asked about the alternatives outlined above. His answers and suggestions are summarized below:

1) Coral CEA does not have servers available to host applications developed by members

2) Continue to use the current language platform (Asp.Net) to minimize the work in porting a prototype to production

3) Create a virtual host (coraldemo.activitybox.ca) in Rezact's production server

4) If utilizing Rezact's server is not an option, there is the possibility of deploying the .Net application using Mod Mono on Apache (http://mono-project.com). This would allow us to use any Linux box with Apache

Following Ian's suggestions, and in view of the evident advantages, it was decided to use Rezact server to host the demo application (coraldemo.activitybox.ca), and the demo would be a reduced version of the real system with limited capabilities in order to facilitate its utilization by non-trained users. The platform would be kept the same, that is to say, Asp.net using a MS-SQL database.

3. Literature Review

In this section, we will explore how the academic literature could help us solve two questions: a) Why Rezact and Activity Box should use Coral CEA platform to extend its services, and b) How Coral CEA and other member companies could benefit from having Rezact as contributing user? The first question will be answered by analyzing how business ecosystems are formed and the relationship between keystone companies and niche players. The latter will be framed around lead-users theories and the benefits provided to the ecosystem when innovations are driven by lead users.

3.1. Business Ecosystems and Platforms

The basic concepts of business ecosystems are provided by Moore (1993), who compares a network of companies working in collaborative ways as that of a natural ecosystem. Moore asserts that innovative business cannot live in a vacuum: they must attract capital, partners, suppliers and customers to create self-sustainable business communities where member companies co-evolve capabilities around a new innovation and help the ecosystem to improve because the ecosystem's success is attached to the company own success.

This view is extended by Iansiti & Levien in their book "The Keystone Advantage" (2004). According to the authors, in every ecosystem coexist "keystone species" and "niche players". Keystone companies deploy strategies that actively shape and

regulate the health of a business ecosystem by providing stable and predictable platforms over which other members can develop and flourish. At the same time, the authors consider that the majority of business ecosystems are formed by small companies or niche players that, when analyzed individually, seem to have little or no influence in the ecosystem, but whose performance and effectiveness affect the entire network because they provide essential product and services to the ecosystem. Niche players are usually located at the edges of the ecosystem, where new services are offered and new markets are explored, providing a healthy diversity to the ecosystem.

The same authors, in their article "Strategy as Ecology" (2004), argue that the success of a company depends on the collective health of the ecosystem that influence the creation and the delivery of innovations. Each member of a business ecosystem shares the fate of the community as a whole and to increase an ecosystem's overall health, keystone organizations should create platforms that allow other members –niche players- of the same ecosystem to share the revenues and increase their own performance.

It is under this light that Coral CEA can be seen as a keystone entity, providing the platform and infrastructure needed by the ecosystem to grow in a healthy way. Rezact, with its product Activity Box, acts as a niche player that offers to the tourism industry a new set of services – CEA capabilities – that no other competitor can currently offer without incurring in prohibiting developing and implementation costs.

The answer to the question of why Activity Box should extend its services based on Coral CEA platform can be found by reviewing Cusumano and Gawer (2002). Their main conclusion is that the combined efforts of platform leaders – as Coral CEA - and complementary innovators – as Rezact – can increase the potential market size for everyone. Rezact, working in isolation, would not be able to extend its range of services to include CEA capabilities –and subsequently increase its potential market size- without the platform support provided by Coral CEA. While the benefits of being part of a much larger ecosystem as that of Coral CEA are evident for Rezact, it might not be so for Coral CEA itself. Rezact needs to show proof of its individual value to the platform leader. We will explore how Rezact is valuable for Coral CEA in the next section.

3.2. Lead Users Theory

According to Von Hippel (1986 & 2005), empirical studies have shown that between 10 and 40 percent of users develop and modify products for their own use. These lead users have two predominant characteristics. First, they are ahead in the market in the sense that they have needs that will be later experienced by the majority of "regular" users, and as such, they realize solutions that most users will need in the future. Second, lead users expect to have important profits from the solutions they find to these needs. If a solution is perceived to be valuable enough to increase profit levels, the lead user will be eager to start the development process of the innovation.

The author defines innovation attractiveness as the combined value of the novelty of the innovation itself and the probability that this innovation becomes the de-facto standard of a particular market area. He found that innovations developed by lead users usually have higher innovation attractiveness indexes, in other words, lead-user innovations are more commercially attractive than manufactures innovations.

Cusumano and Gawer (2002) suggest that the trick to being a successful niche player is to always have "peanuts to offer the elephant", that is to say, to create products or services that continually enhance the value of the platform. We believe that a deliverable like that of the agent proposed in this project can certainly constitute a valuable "peanut" that could make Rezact more attractive to Coral CEA.

Rezact, by being one of the first users of the technology and services offered by Coral CEA sandbox, stands in a perfect position to realize the needs of future users of the platform, thus providing a valuable input to the platform leader and, subsequently, to the entire ecosystem. Furthermore, the reservation of recreational activities constitutes a new market segment for Coral CEA that is being explored by Rezact acting as a niche player, an exploration that could also led to further growth to the Coral CEA ecosystem.

4. Project Design

The following section covers the main aspects related with the high-level design of the project. We include here a detailed analysis of the limitations found in the current implementation of Coral CEA methods, limitations that forced us to devise a solution in the form of an agent that would handle all interactions with the Coral CEA sandbox. The design of this agent is also covered in this section.

4.1. Coral CEA APIs

Coral CEA's services are provided as web services using SOAP as the communication and implementation protocol. The platform provides several services, but for this project we are only interested in the communication entry points provided by the Third Party Call Control V3 (TPCv3) API, which allows the creation of communication links between one or multiple endpoints. The second Coral CEA service of interest for this project is the Audio Call service, which allows an application to play a pre-recorded message to participants on an existing call, and to monitor the status of the audio message requested.

In the case of ActivityBox, the existing functionality requires the application to separately "call" a customer or an operator and inform them about an event on the system. There is currently no requirement to connect these two parties together (although nothing prevents Rezact from implementing this particular feature in the future). For this reason, we have not included Coral CEA's Third Party Call Control V2 (TPCv2) API, which allows for calls to be created between two parties, first one

named the “originator” that places the call that is going to be answered by the “terminating” party.

4.2. Coral CEA services

An initial review of Coral CEA technical documentation showed to us that certain complementary services were not currently provided by the platform, or they were provided in a limited way. This section details these findings as they were discussed with Ian Bothwell, Application Architect and Lead Technician at Coral CEA, as well as the decisions taken to overcome these issues.

4.2.1. Multi-language Text-To-Speech. Functionalities like Call-A-Customer and Call-An-Operator require that the message transmitted to the intended party contains certain basic information, including, but not limited to, the name of the activity, the reservation number and the start date and time of the activity. Activity Box currently manages over two hundred and fifty activities for both winter and summer seasons, so if only recorded audio messages could be used, it becomes impractical to create hundreds of audio messages for each activity in the system, not to mention the need to create a new audio track every time a new activity is added to the system. Additionally, any communication to either the customer or the operator needs to be provided in the receiver's language, which in Activity Box domain is either English or French.

The ability to convert text-to-speech (TTS) is an important feature needed by Activity Box. Moreover, we strongly believe that for any TTS functionality in the Cora CEA platform to be usable for commercial purposes, it needs to support both official languages in Canada, English and French. Unfortunately, such a component is not yet ready on the current Coral CEA sandbox. Ian Bothwell estimates that an English-based TTS feature could be implemented within six months, with a French-based version considered as a possibility.

Although the lack of TTS support on the Coral CEA platform places a real restriction for its applicability in the production environment of Activity Box, we implemented a workaround for the demo. We considered two alternatives:

a) To drastically reduce the number of operators and activities to be shown on the demo (to just 4 activities in total), which would allow us to still include the required information in the call without having to record hundreds of audio messages

b) To create a general recorded message that would only notify that a certain event has occurred, like a cancellation, without any reference to the activity or the date

We selected the first alternative because we think that having detailed messages being delivered to customers and operators represents a closer scenario to the one we would have in production.

4.2.2. Synchronous calls. Coral CEA APIs are synchronous, which means that the client application needs to wait to receive some form of response from Coral CEA APIs. We believe that this approach could adversely affect the application by creating contention on Activity Box web server and reducing the future scalability of the system if it is to be implemented within the normal activity reservation process. Calls to Coral CEA APIs thus need to be decoupled from the web application itself so the performance and the user interaction are not affected in any way.

4.2.3. Call confirmations. Each call attempt originated from Activity Box needs to be logged for auditing purposes, so resellers can verify if a customer or an operator has been notified about a particular event, like a cancellation. Coral CEA APIs implement a series of response codes that can be provided on demand to client components like Activity Box. To obtain the current and final status of a call, the client application needs to periodically poll Coral CEA servers to obtain the status of each requested call. Furthermore, the status of the call, which plays a recorded audio track, is kept on Coral CEA servers for only ten minutes. Ian Bothwell mentioned that there are plans to create a "push" mechanism by which the client becomes a server listening for status updates sent by Coral CEA. However, there is no yet a clear time frame for this functionality to be implemented.

4.2.4. Concurrent calls. The current limit on the number of concurrent calls in the Coral CEA sandbox is estimated to be eight. Future limits for commercial applications will depend on the underlying infrastructure deployed. Although a detailed analysis of

physical implementations in terms of communication capabilities is beyond the scope of this project, we need to provide a buffer mechanism that could manage a very limited number of phone lines (whether traditional or VoIP lines) to operate. Since Coral CEA APIs do not provide any queue mechanism to handle this type of situations, any commercial solution, including Activity Box should consider this limitation within its design.

4.2.5. Additional services. Some other complementary services like the ability to send faxes, to schedule calls to be executed at a particular time and date, to send text messages to cell phones or to allow interaction between the party called and the platform (like requesting the message to be repeated) are currently not provided within the Coral CEA sandbox. Ian Bothwell commented that some of the features mentioned above could be considered if there are an important number of Coral CEA members requesting those functionalities.

4.3. Coral CEA Client Agent

For time-sensitive applications, like Activity Box reservation system where the response time is a crucial factor, waiting for a third-party to answer requests could have negative effects. The synchronous method offered by Coral CEA could generate contention and scalability issues on Activity Box. Moreover, the polling system provided by Coral CEA platform to obtain the status of any requested communication falls outside a traditional web-driven architecture. Finally, limitations on the number of concurrent calls that can be put through the platform at any given time would

force additional validations and control-like procedures on the end-user layer. For these reasons we have decided to separate as much as possible any Coral CEA implementation from the application itself to keep Activity Box isolated from any operational problem.

One way we suggest is to create an agent that serves as an intermediary between Activity Box and Coral CEA APIs. The concept proposed follows this strategy: every time an event that requires CEA capabilities is detected, Activity Box would save into a common database table called CEAQueue the request for an outgoing call, including, for demo purposes, the name of the audio file associated with the event (e.g. a cancellation). The agent, implemented as a Windows service, would be continuously querying this table to detect any new request. When a new request is found, the agent will submit the request to Coral CEA APIs, and it will keep polling Coral CEA server to obtain an updated status of the request. Each status change is updated back into the CEAQueue table so the client application can be kept informed of the status of any call. The general model of this design is shown in figure 3.

By decoupling Activity Box from Coral CEA interactions, we are not only providing a safety net in case something goes wrong, but also mitigating the modifications involved on the client application since they will most likely involve just an extra SQL-like instruction on the database level needed to insert the request into the CEAQueue table, keeping the user and business layers mostly untouched. To provide updated

calling status information to the users, Activity Box would only need to query this new local table without having to make continuous remote calls to Coral CEA APIs.

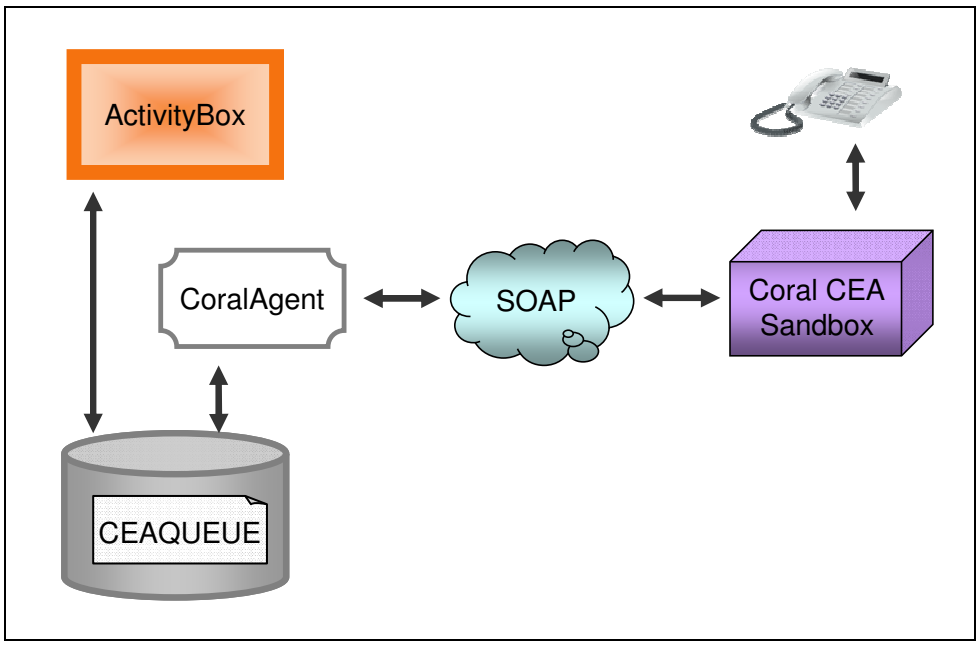

Figure 3. General Design of the Agent

Moreover, by having this design, we could easily provide Coral CEA functionalities to other applications as long as they are hosted on the same machine. The current limitation by which Coral CEA only stores call information for a limited amount of time, ranging from 30 seconds to 10 minutes, will also be solved by storing call status locally. We also believe that this approach (a windows-based service) could be, eventually, offered to other Coral CEA clients that face the same challenges, or even be used by Coral CEA itself as a general framework to implement its own buffer mechanism.

5. Project Implementation

The details of the implementation of the project are described in this section. Here, we outline the steps needed to create a proxy class around Coral CEA web services and describe the structure of the common table that is used to interact with Coral CEA Agent. Finally, we explain how the configuration settings could make the agent a customizable tool useful to other Coral CEA members in similar scenarios.

5.1. Coral CEA Service Wrapper

Coral CEA services are provided as web services in WSDL format. WSDL stands for Web Services Description Language and is an XML-based language that provides a model for describing Web services. Under the .Net framework used on this project, there are two ways to consume a web service. The first one is to directly reference the resource, usually as a web reference in Microsoft terminology, and the other alternative is to use the "wsdl.exe" utility that is shipped with Microsoft Visual Studio to create a class wrapper around the web service, which can later be referenced by the client application. We believe that the second approach, that is to say, the class wrapper, is a much cleaner way to interact with a web service because it isolates the client application from any changes on the web service itself. It also permits the reutilization of the wrapper class among several clients.

In order to create a class wrapper, we need first to obtain the service definition from Coral CEA. They are provided in xml format, and the list of files that are required for

each service is detailed on Coral CEA documentation. For our demo, we required the definitions for “Third Party Call Control V3.0” and “Audio Call” services. Once the files are downloaded to a local directory, we need to remove the “xml” extension. Then, using a Visual Studio command prompt, navigate to where the files are located, and execute the wsdl tool by using the following command:

```
wsdl.exe /l:VB /n:WSCoralCEA_Call /out:TPCCv3_CallSession.vb
parlayx_third_party_call_service_3_4.wsdl parlayx_third_party_call_interface_3_4.wsdl
parlayx_common_faults_3_0.wsdl parlayx_common_types_3_1.xsd
```

where:

/l:VB: Specifies the language to use for the generated proxy class. You can specify CS (C#; default), VB (Visual Basic), JS (JScript) or VJS (Visual J#) as the language argument.

/n:WSCoralCEA_Call: Specifies the namespace for the generated proxy class.

/out:TPCCv3_CallSession.vb: Specifies the file (or directory) in which to save the generated proxy code. If omitted, the tool will use the web service name for the file name.

The remaining parameters just indicate to the tool the additional files that need to be included in the proxy class. After executing the command, the following result should appear:

```
Microsoft (R) Web Services Description Language Utility
[Microsoft (R) .NET Framework, Version 2.0.50727.42]
Copyright (C) Microsoft Corporation. All rights reserved.
Writing file 'TPCCv3_CallSession.vb'.
```

Then repeat a similar process for the Audio service with this command:

```
wsdl.exe /l:VB /n:WSCoralCEA_Audio /out:TPCCv3_Audio.vb
parlayx_audio_call_play_media_service_3_2.wsdl parlayx_audio_call_play_media_interface_3_2.wsdl
parlayx_audio_call_types_3_2.xsd parlayx_common_faults_3_0.wsdl parlayx_common_types_3_1.xsd
```

Next, create a class library in Visual Studio, which will generate a dll file. We named ours "WSCoral". Then copy the files generated by the wsdl tool to the working directory of the WSCoral class and include them in the project. Make sure the library has the following references specified:

```
System
System.Data
System.Web
System.Web.Services
System.XML
```

Finally, build the library, which will result in a file called "WSCoral.dll". We will later use this wrapper class within the Coral CEA Agent.

5.2. Coral CEAQueue Table

The purpose of this table is to serve as a point of contact between any client application and the agent or service that will directly interact with Coral CEA methods. Since Activity Box uses a MS-SQL database, we defined the table using that MS-SQL data types, but it could be easily converted to any other data provider.

The design of this table allows us to request calls to the agent, which in turn will request them to Coral CEA. Currently, we only support one participant at a time, but

the design can be extended to support multiple participants per call. One of the additional features added to the basic Coral CEA service when using this table in conjunction with the agent is the ability to specify the priority in which we want the calls to be requested. This option is useful in scenarios where there are few lines available and we need to ensure that the most important notifications or messages are requested first. In the Activity Box case, we use the priority feature to place cancellation calls at the top of the list, followed by confirmations of new reservations.

The CEAQueue table is also used to indicate to the agent which audio file we want to be played once the call is connected. The audio files are associated with the different activities on the system, as well as for each of the events that we are controlling. Our design also contemplates the capability to try a call two or more times if the initial attempt was unsuccessful for a number of reasons other than the participant hanging up the call. We are also providing a set of User-Defined-Fields, or UDFs, which allow client applications to attach useful information to each call that can be later used to retrieve particular calls. In Activity Box, for example, we have used two of these fields to indicate the reservation number and the name of the recipient of the call.

The structure of the table follows the terminology of Coral CEA properties, but we have simplified its design and added a few extra fields to achieve the behavior described earlier. The general structure is shown on the next page (Table 1).

Field	Description
CallSessionIdentifier varchar(50) NOT NULL	Unique identifier (returned by the makeCallSession operation) that identifies the call session
CallParticipantStatus int NOT NULL DEFAULT 99	Status of the call participant. Possible values are: 99:Call Pending 0:callParticipantInitial (call is in progress to the participant) 1:callParticipantConnected (participant is active in the call) 2:callParticipantTerminated (call to the participant has ended)
CallParticipantStartTime varchar(50) NULL	The date and time when the call participant was added to the call. This parameter is only returned if the call to the participant was established successfully (that is, if call participant status is not callParticipantInitial)
CallParticipantDuration int NULL DEFAULT 0	The duration (in seconds) of the participant's involvement in the call. This parameter is only relevant if the participant is no longer on the call (that is, if call participant status is callParticipantTerminated)
CallParticipantTerminationCause int NULL	The cause of the termination of the call. This parameter is only relevant if callStatus = callTerminated. Possible values are: 0:CallingPartyNoAnswer 1:CalledPartyNoAnswer 2:CallingPartyBusy 3:CalledPartyBusy 4:CallingPartyNotReachable 5:CalledPartyNotReachable 6:CallHangUp 7:CallAborted
LastCallFaultCode varchar(500) NULL	Contains the last call error message from Coral CEA
CallParticipant varchar(50) NULL	Address of the participant to be included in the call
AudioURL varchar(500) NULL	Name of the audio content file to be played
AudioIdentifier varchar(50)	Unique identifier for the play message request (returned by the playAudioMessage operation)
AudioStatus	Status of the audio message for the participant. Possible values

int NULL	are:0:Played,1:Playing,2:Pending,3:Error
Attempts int 0	Number of call attempts that the agent have already made to Coral CEA
NextAttemptUTC datetime GETUTCDATE()	Indicates when the next attempt should be made to place the call, in UTC format
CallPriority int NOT NULL DEFAULT 1	Indicates the priority in which calls should be requested. The higher the value, the higher the priority

Table 1. Structure of CEAQueue Table

5.3. Coral CEA Agent

One of the main components of the solution is the agent used to handle all communications with Coral CEA platform. The main concept is to have this agent continuously running and periodically querying the CEAQueue table on the database looking for new calls to make. Once a call request is found, it places the request to Coral CEA and keeps polling the Coral CEA web service for the status of the call, while at the same time updates such status back into the CEAQueue table so the client application can have the latest information. When the call is completed, the agent updates the termination status back to the CEAQueue table, and places the next call in the list, if any. The pseudo code of agent is shown below:

```
List = GetList(CEAQueue)
For Each Call on List
        Request Call to CoralCEA
        Update(CEAQueue)
        If CallStatus = Initializing Then
                Update(CEAQueue)
                Do while ParticipantStatus <> Terminated
                        If ParticipantStatus = Connected
                                Request Audio to CoralCEA
```

```
                                Update(CEAQueue)
                        End If
                        Get ParticipantStatus
                End Do
                Update(CEAQueue)
        End If
Next Call on List
```

Amongst its functionalities, the agent is able to try a call several times if the communication did not go through due to technical reasons other than the customer hanging up, and can be configured to place calls only between certain times of the day to prevent the application from calling customers at disturbing times in the night. This agent also encapsulates some of the internal mechanisms needed to interact with Coral CEA and exposes them as configurable parameters that can be changed on the configuration file. The list of settings that can be used is shown in table 2.

Parameter	Default Value	Description
DBPollInterval	30	Indicates the interval in seconds in which the agent should query the CEAQueue table in the database searching for new requests
ACEServerURL	Determined by Coral CEA	Coral CEA service endpoint
ACEAudioServerURL	Determined by Coral CEA	Coral CEA service endpoint for playing pre-recorded messages
ACEUser	Determined by Coral CEA	Network credentials to communicate with Coral CEA
ACEPassword	Determined by Coral CEA	Network credentials to communicate with Coral CEA
ACETimeout	20	Time in seconds before an attempt to communicate to Coral CEA expires
CallAttempts	4	Maximum number of attempts the agent should try to place a call through Coral CEA

LogEntries	1	Indicates whether or not to log entries in the event log of the server (0:NO, 1:YES)
StartHour	8	Starting time in which the agent should start placing calls through Coral CEA
EndHour	20	Ending time in which the agent should stop placing calls through Coral CEA
ParticipantPrefix	Sip:9	Any string that needs to be placed before a regular phone number
ParticipantSuffix	@134.117.254.226	Any string that needs to be placed after a regular phone number
AudioURLPrefix	play=	Any string that needs to be placed before the audio file name
AudioURLSuffix	;early=no;locale=en_us	Any string that needs to be placed after the audio file name

Table 2. Custom settings for Coral CEA Agent

The agent has been implemented as a Windows service developed in Visual Studio. It also needs a reference to the Coral CEA Service Wrapper "WSCoral" described earlier in section 5.1. The CoralCEAAgent windows service requires an installation program that is also included with the project. The source code for the agent and the database routines needed for its operation are included in the appendix section.

5.4. Activity Box

To properly implement a demo version of Activity Box, we had to consider the following factors:

- Activity Box is a complex application, with a considerable amount of functionalities that are needed to properly manage every type of reservation. However, since it requires certain training to be used in its full capacity, we

decided to reduce the number of functionalities to the minimum in order to facilitate user navigation

- Non-trained users of the demo version of Activity Box should be able to easily make a reservation as well as cancel any existing one in order to trigger the events that would cause Activity Box to generate calls using Coral CEA services
- Due to the lack of TTS support on the current Coral CEA platform, we decided to reduce the number of operators to two, and the total number of activities managed by the demo to four. By limiting the number of activities, we also reduced the number of audio files that needed to be created for demo purposes. Given that we send call requests when creating a new reservation to both the customer and/or the operator, as well as when cancelling a reservation, that brings the total number of audio files created for the demo to sixteen. The scripted text of each type of audio message is shown in appendix 8.3
- Sensitive information like customer names, credit card information, etc. has been removed or restricted from the demo to avoid privacy issues

Since all the complexities of communicating with Coral CEA are handled by the agent, the modifications required to Activity Box were reduced to a minimum. First, we needed to modify the database routines that process the creation of a new reservation and the cancellation of an existing reservation to insert the required values into the CEAQueue table described in section 5.2. The information inserted includes the type of event (New booking, Cancellation), the name and phone number

of the receiver of the call, the reservation number associated with the call and its priority. The second modification was done to provide a way to the user to verify how the call went through. For this effect, we added visible buttons and links in the application that will display the results of those calls, as shown in figure 4.

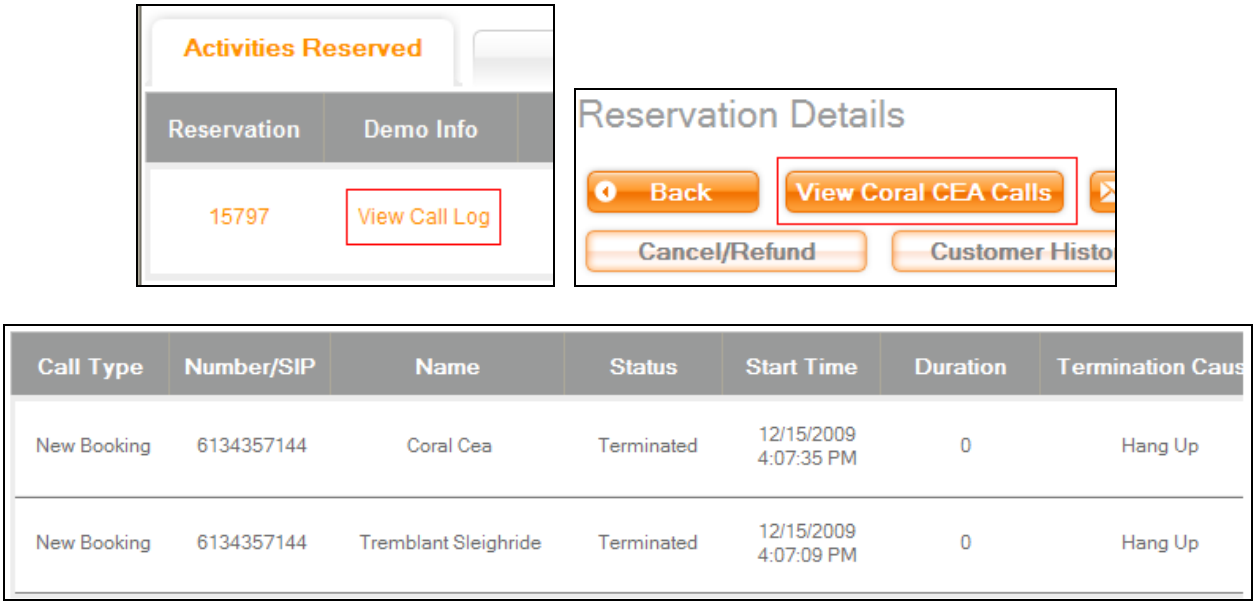

Figure 4. Modifications made to Activity Box

The remaining changes applied in the system were more related to the fact that it was a demo application, so data entry points were added to certain pages to allow users to easily change the phone numbers of the operator or the customer. We also created a default customer to simplify the navigation process of the demo. Besides those points, the actual implementation of Coral CEA feature within Activity Box was considered a very straight forward process.

6. Recommendations and Conclusions

This final section is further divided in two. First, based on our experience implementing Coral CEA services, we will offer a set of recommendations for Coral CEA and for members of its ecosystem. Finally, we will present the conclusions and the lessons learned from this project.

6.1. Recommendations

6.1.1. For Coral CEA

As a result of our analysis of the current Coral CEA capabilities, we believe that it is completely possible for a commercial application to implement certain basic CEA features. We have found, however, certain limitations that could reduce Coral CEA effectiveness. We believe that the following recommendations based on our experience can mitigate such effects:

- A Text-To-Speech bilingual service is a really important functionality that should be implemented as part of Coral CEA services. We know that Nortel has already offered this service in its Media Processing Server (MPS) platform[3], so Coral CEA can leverage this experience in its own platform. Additionally, we strongly recommend that any TTS implementation should consider at least the two official languages in Canada, English and French
- We also suggest to increase the period for which the status of a call or an audio message is kept in Coral CEA servers to at least one day. This would allow

[3] See http://www.nortel.com/products/04/ivr/collateral/nn103943.pdf

automatic batch procedures on the part of the client to be executed outside peak hours (e.g. at midnight) to query and locally update the status of any requested call

- Coral CEA interfaces could be implemented, at least on the Windows environment, as a Dynamic Link Library or DLL which can be easily referenced by any .Net language. The current method depicted in this report, using the wsdl command, is prone to errors and does not add any value to the service
- The ability to make scheduled calls can be fairly easily provided embedded within the Coral CEA platform with a buffer mechanism like the one we proposed in this project. This would also have the advantage of eliminating the need from client applications to continuously polling Coral CEA servers to find out about the status of any call request. A buffer mechanism would allow client applications to set scheduled updates outside of high demand periods, consequently reducing the load on Coral CEA platform
- Some client applications would not necessarily require a TTS service like Activity Box does, but rely more heavily on audio messages played to their users. Currently, there is no automatic mechanism in place within the Coral CEA sandbox to easily upload audio files to Coral CEA media server. Moreover, the audio files require having an especial format[4] which certainly makes things more complicated for client developers. A web service could be offered by Coral CEA to allow client applications to upload files to the media server,

[4] Linear PCM, 8 bit little-endian unsigned integer, 1 channels, 8000 Hz

encapsulating any format conversion needed to comply with Coral CEA requirements

- Current or potential limitations of the platform, like the number of calls that can be placed simultaneously, or the characteristics of audio files (format, duration) should be clearly indicated on Coral CEA development guides
- Asynchronous method calls for Coral CEA APIs would be of great benefit for clients that have time-sensitive applications. This would allow them to request calls to the platform without having to wait for a response, enabling them to continue with the business process the triggered the request
- During our interviews with Rezact's management team, we identified that the ability to send text messages to cell phones is a much desired feature, because it is perceived as a less intrusive way of communicating information to users with the additional advantage that they can opt to review their messages at a later time. Coal CEA currently does not offer this feature
- Finally, we recommend adding basic user interaction capabilities to the current Coral CEA platform that would allow called parties to provide some useful information back to the platform. A feature of such sort could be the ability of the user to request, by pressing a certain key on the phone, that the message be repeated. A similar mechanism could also be used by clients to reject calls that would indicate to Coral CEA that the user is not the actual intended receiver of the call, as when a wrong phone number is entered in the client application

6.1.2. For Client applications

Based on our particular experience implementing CEA features within Activity Box, we can list the following recommendations for client applications:

- Any interaction with Coral CEA API's should be treated independently and outside of the regular process or event that triggers the call, to prevent scenarios where there is a limited number of available lines or when the application needs to wait for a response. One way to accomplish this would be the utilization of an agent or a Windows service that could perform call requests and also periodically poll Coral CEA servers to update the status of a call or an audio message
- We believe based on our experience that any project pretending to implement Coral CEA features like the ones described on this report should roughly estimate one month of development time, an estimation that obviously depends on the size of the application
- We have found that, once the key elements to communicate with Coral CEA APIs are in place and the buffer mechanism is used, the remaining effort is solely determined by the interactions between the client application and the common table, which is used to place call requests as well as to inquire about the status of a previous request
- Client applications planning to utilize Coral CEA assets should carefully estimate the expected number of events during a period of time that could trigger request for calls, since the number of calls that can be put through by Coral CEA depends largely on the capacity of the deployed infrastructure

6.2. Conclusions

- Coral CEA platform has key technological elements needed to create a successful business ecosystem. We believe that Coral CEA value proposition can be greatly increased by offering complementary services, like the Text-To-Speech service
- It is relatively easy for commercial applications to implement CEA features using Coral CEA APIs. Certain limitations at the deployment level can be overcome with some development effort until they can be resolved by the platform itself
- A key factor for a successful implementation of Coral CEA services is to keep the client application as isolated as possible from any interaction with Coral CEA APIs. This can be done by using the agent proposed in this project, but other mechanisms could be found according to particular needs and scenarios
- Coral CEA, as a keystone entity and a platform leader, provides a set of services that the entire ecosystem, including Rezact, can take advantage of. It is by leveraging Coral CEA platform expertise that Rezact could increase its market size and consequently Activity Box perceived value
- Rezact, acting as a niche player, could explore new market segments, face problems that are currently not evident for other members of the community, and, in the process of solving those issues, it can create innovative solutions that can benefit the entire ecosystem. The agent proposed in this project, as well as the recommendations listed earlier are clear examples of such a successful symbiotic relationship

7. References

- Cusumano, M., Gawer A. 2002. The Elements of Platform Leadership. MIT Sloan Management Review, Spring: 51-58
- Iansiti, M. & Levien, R. 2004. Strategy as Ecology. Harvard Business Review, March: 68-78
- Iansiti, M. & Levien, R. 2004. The Keystone Advantage. Harvard Business School Press
- Luthje, C & Herstatt, C. 2004. The Lead User Method: An Outline of Empirical Findings and Issues for Future Research. R&D Management, 34(5): 553-68
- Moore, J.F. 1993. Predators and Prey: A New Ecology of Competition. Harvard Business Review, May/June: 75-86
- Von Hippel, E. 1986. Lead Users: A Source of Novel Product Concepts. Management Science, 32 (7): 791-806
- Von Hippel, E. 2005. Democratizing Innovation. Cambridge, MA. MIT Press

8. Appendix Section

8.1. Coral CEA Agent Source Code

The code below is a sample of the main routine performed by the Coral CEA Agent we have implemented on this project. The remaining routines are included as part of this project as an attachment.

```
Private Sub LocalQueryEvent(ByVal state As Object)
  Try
    If Date.Now.Hour >= intStartHour
    And Date.Now.Hour < intEndHour And blnStillExecuting = False Then
      Dim db As Database = DatabaseFactory.CreateDatabase()
      Dim dbPendingCallList As DbCommand = db.GetStoredProcCommand("cea_CallsToMake_list")
      Dim dbUpdateCallInfo As DbCommand = db.GetStoredProcCommand("cea_CallInfo_upd")

      Dim dtCallsToMake As DataTable

      db.AddInParameter(dbUpdateCallInfo, "RowId", DbType.Int32)
      db.AddInParameter(dbUpdateCallInfo, "Attempts", DbType.Int32)
      db.AddInParameter(dbUpdateCallInfo, "CallSessionIdentifier", DbType.String)
      db.AddInParameter(dbUpdateCallInfo, "CallParticipantStatus", DbType.Int32)
      db.AddInParameter(dbUpdateCallInfo, "LastCallFaultCode", DbType.String)
      db.AddInParameter(dbUpdateCallInfo, "AudioStatus", DbType.Int32)
      db.AddInParameter(dbUpdateCallInfo, "AudioIdentifier", DbType.String)
      db.AddInParameter(dbUpdateCallInfo, "CallParticipantTerminationCause", DbType.Int32)
      db.AddInParameter(dbUpdateCallInfo, "CallParticipantStartTime", DbType.String)
      db.AddInParameter(dbUpdateCallInfo, "CallParticipantDuration", DbType.Int32)

      ' We indicate the maximum nmber of attemps we want to filter
      db.AddInParameter(dbPendingCallList, "Attempts", DbType.Int32, intAttempts)

      dtCallsToMake = db.ExecuteDataSet(dbPendingCallList).Tables(0)
      ' Note: connection was closed by ExecuteDataSet method call

      If dtCallsToMake.Rows.Count > 0 Then
        blnStillExecuting = True
        Dim strParticipants(1) As String

        For i As Integer = 0 To dtCallsToMake.Rows.Count - 1
          Dim makeCallArguments As makeCallSession = New makeCallSession()
          strParticipants(0) = encodeURLSafe(strParticipantPrefix &
                               dtCallsToMake.Rows(i)("CallParticipant") & strParticipantSuffix)
          makeCallArguments.callParticipants = strParticipants

          db.SetParameterValue(dbUpdateCallInfo, "RowId", dtCallsToMake.Rows(i)("RowId"))
          db.SetParameterValue(dbUpdateCallInfo, "Attempts",
                                                 dtCallsToMake.Rows(i)("Attempts") + 1)
```

```
Dim objCallResponse As New makeCallSessionResponse
Try
   ' Initialize the call status to prevent being requested again
   db.SetParameterValue(dbUpdateCallInfo, "CallParticipantStatus",
                                          CallParticipantStatus.CallParticipantInitial)
   db.ExecuteNonQuery(dbUpdateCallInfo)

   ' Make the call
   objCallResponse = getThirdPartyCallService.makeCallSession(makeCallArguments)

   ' Obtain the call session identifier from CoralCEA
   db.SetParameterValue(dbUpdateCallInfo, "CallSessionIdentifier",
                                                              objCallResponse.result)

   Dim getCallInformationArguments As New getCallParticipantInformation()
   getCallInformationArguments.callSessionIdentifier = objCallResponse.result
   getCallInformationArguments.callParticipant = strParticipants(0)

   Dim objCallInfo As New getCallParticipantInformationResponse
   ' Request the status of the call
   objCallInfo =
getThirdPartyCallService().getCallParticipantInformation(getCallInformationArguments)

   ' Update the participant status
   db.SetParameterValue(dbUpdateCallInfo, "CallParticipantStatus",
                                                 objCallInfo.result.callParticipantStatus)
   db.ExecuteNonQuery(dbUpdateCallInfo)

   db.SetParameterValue(dbUpdateCallInfo, "Attempts", System.DBNull.Value)

   Dim blnRequestNotSent As Boolean = True
   Dim strAudioCorrelator As String = ""
   Do While objCallInfo.result.callParticipantStatus <>
                                 CallParticipantStatus.CallParticipantTerminated
      If blnRequestNotSent And objCallInfo.result.callParticipantStatus =
                                 CallParticipantStatus.CallParticipantConnected Then
         Dim objAudioResponse As New playAudioMessageResponse
         Dim objAudioMessage As New playAudioMessage()
         objAudioMessage.callSessionIdentifier = objCallResponse.result
         objAudioMessage.audioUrl = strAudioURLPrefix &
                                 dtCallsToMake.Rows(i)("AudioURL") & strAudioURLSuffix
         Try
            db.SetParameterValue(dbUpdateCallInfo, "CallParticipantStatus",
                                                 objCallInfo.result.callParticipantStatus)
            db.ExecuteNonQuery(dbUpdateCallInfo)

            objAudioResponse =
                        AudioCallPlayMediaService.playAudioMessage(objAudioMessage)
            db.SetParameterValue(dbUpdateCallInfo, "AudioIdentifier",
                                                            objAudioResponse.result)
            strAudioCorrelator = objAudioResponse.result
            If blnCreateLog Then
               myLog.WriteEntry("LocalQueryEvent. Audio Request:" & strAudioURLPrefix
                                    & dtCallsToMake.Rows(i)("AudioURL") &
                                    strAudioURLSuffix,
```

```
                                                    EventLogEntryType.Information)
            End If
         Catch ex As Exception
            myLog.WriteEntry("LocalQueryEvent: At
                        AudioCallPlayMediaService.playAudioMessage(objAudiMessage).
                        Message:" & ex.Message
                        & ". Stack Trace:" & ex.StackTrace, EventLogEntryType.Error)
         Finally
            blnRequestNotSent = False
         End Try
      End If
      objCallInfo =
getThirdPartyCallService().getCallParticipantInformation(getCallInformationArguments)
   Loop

   db.SetParameterValue(dbUpdateCallInfo, "CallParticipantTerminationCause",
                              objCallInfo.result.callParticipantTerminationCause)
   db.SetParameterValue(dbUpdateCallInfo, "CallParticipantStartTime",
                              objCallInfo.result.callParticipantStartTime.ToString)
   db.SetParameterValue(dbUpdateCallInfo, "CallParticipantDuration",
                              objCallInfo.result.callParticipantDuration)
   db.SetParameterValue(dbUpdateCallInfo, "CallParticipantStatus",
                              objCallInfo.result.callParticipantStatus)
   db.ExecuteNonQuery(dbUpdateCallInfo)

   If strAudioCorrelator <> "" Then
      Dim objAudioMessageStatus As New getMessageStatus()
      Dim objAudioInfo() As MediaMessageStatus
      objAudioMessageStatus.correlator = strAudioCorrelator 'Audio Correlator
      objAudioInfo =
                AudioCallPlayMediaService.getMessageStatus(objAudioMessageStatus)
      db.SetParameterValue(dbUpdateCallInfo, "AudioStatus", objAudioInfo(0).status)
      ' Update the status of the audio
      db.ExecuteNonQuery(dbUpdateCallInfo)
   End If

Catch ex As Exception
   db.SetParameterValue(dbUpdateCallInfo, "CallParticipantStatus",
                              CallParticipantStatus.CallParticipantTerminated)
   db.SetParameterValue(dbUpdateCallInfo, "LastCallFaultCode", Left(ex.Message, 50))
   db.ExecuteNonQuery(dbUpdateCallInfo)
   myLog.WriteEntry("LocalQueryEvent: " & ex.Message & ". Stack Trace:" &
                        ex.StackTrace, EventLogEntryType.Error)
End Try

'Clears all query parameters
db.SetParameterValue(dbUpdateCallInfo, "RowId", System.DBNull.Value)
db.SetParameterValue(dbUpdateCallInfo, "Attempts", System.DBNull.Value)
db.SetParameterValue(dbUpdateCallInfo, "CallSessionIdentifier", System.DBNull.Value)
db.SetParameterValue(dbUpdateCallInfo, "CallParticipantStatus", System.DBNull.Value)
db.SetParameterValue(dbUpdateCallInfo, "LastCallFaultCode", System.DBNull.Value)
db.SetParameterValue(dbUpdateCallInfo, "AudioStatus", System.DBNull.Value)
db.SetParameterValue(dbUpdateCallInfo, "AudioIdentifier", System.DBNull.Value)
db.SetParameterValue(dbUpdateCallInfo, "CallParticipantTerminationCause",
                                                            System.DBNull.Value)
```

```
                db.SetParameterValue(dbUpdateCallInfo, "CallParticipantStartTime",
                                                                    System.DBNull.Value)
                db.SetParameterValue(dbUpdateCallInfo, "CallParticipantDuration",
                                                                    System.DBNull.Value)

            Next
          End If
          dtCallsToMake = Nothing
          blnStillExecuting = False
        End If

    Catch ex As Exception
      blnStillExecuting = False
      myLog.WriteEntry("LocalQueryEvent: " & ex.Message & ". Stack Trace:" & ex.StackTrace,
                                EventLogEntryType.Error)
    End Try
  End Sub
```

8.2. Coral CEA Agent Database Procedures

The two procedures listed below are used to obtain the list of calls request that are still pending and to update the status of an existing call.

```
CREATE PROCEDURE dbo.cea_CallsToMake_list (
        @Attempts       int = -1
)
AS
SET NOCOUNT ON

SELECT CEAQueue.RowId,
        CEAQueue.Attempts,
        CEAQueue.CallParticipant,
        CEAQueue.AudioURL
FROM    CEAQueue
WHERE (@Attempts = -1 OR CEAQueue.Attempts < @Attempts)
        AND (CEAQueue.CallParticipantStatus = 99 OR -- Pending to call
                        (CEAQueue.CallParticipantStatus = 2 -- Call Terminated
                        -- If caller or called party could not be reached for some reason, let's try again
                                AND CEAQueue.CallParticipantTerminationCause <> 3)
                )
        -- Filter only requests for which certain period of time has elapsed (15 minutes by default)
        AND CEAQueue.NextAttemptUTC < GETUTCDATE()
ORDER BY CEAQueue.CallPriority DESC



CREATE PROCEDURE dbo.cea_CallInfo_upd (
        @RowId                                  int,
        @CallSessionIdentifier                  varchar(50) = NULL,
        @CallParticipantStatus                  int = NULL,
        @Attempts                               int = NULL,
        @LastCallFaultCode                      varchar(50) = NULL,
        @CallParticipantTerminationCause        int = NULL,
        @AudioIdentifier                        varchar(50) = NULL,
        @AudioStatus                            int = NULL,
        @CallParticipantStartTime               varchar(100) = NULL,
        @CallParticipantDuration                int = -1
)
AS
SET NOCOUNT ON

UPDATE CEAQueue
SET     CEAQueue.CallSessionIdentifier = CASE WHEN @CallSessionIdentifier IS NULL THEN
                                        CEAQueue.CallSessionIdentifier ELSE @CallSessionIdentifier
                                        END,
        CEAQueue.CallParticipantStatus = CASE WHEN @CallParticipantStatus IS NULL THEN
                                        CEAQueue.CallParticipantStatus ELSE @CallParticipantStatus
                                        END,
```

```
        CEAQueue.Attempts = CASE WHEN @Attempts IS NULL THEN
                                    CEAQueue.Attempts ELSE @Attempts
                                    END,
        CEAQueue.LastCallFaultCode = CASE WHEN @LastCallFaultCode IS NULL THEN
                                    CEAQueue.LastCallFaultCode ELSE @LastCallFaultCode
                                    END,
        CEAQueue.CallParticipantTerminationCause = CASE WHEN @CallParticipantTerminationCause IS
                                                                        NULL THEN
                                    CEAQueue.CallParticipantTerminationCause ELSE
                                    @CallParticipantTerminationCause
                                    END,
        CEAQueue.AudioIdentifier = CASE WHEN @AudioIdentifier IS NULL THEN
                                    CEAQueue.AudioIdentifier ELSE @AudioIdentifier
                                    END,
        CEAQueue.AudioStatus = CASE WHEN @AudioStatus IS NULL THEN
                                    CEAQueue.AudioStatus ELSE @AudioStatus
                                    END,
        CEAQueue.CallParticipantStartTime = CASE WHEN @CallParticipantStartTime IS NULL THEN
                                    CEAQueue.CallParticipantStartTime ELSE
                                                            @CallParticipantStartTime
                                    END,
        CEAQueue.CallParticipantDuration = CASE WHEN @CallParticipantDuration IS NULL OR
                                                        @CallParticipantDuration = -1 THEN
                                    CEAQueue.CallParticipantDuration ELSE 0
                                    END
FROM    CEAQueue
WHERE   CEAQueue.RowId = @RowId
```

8.3. Audio Messages

For demo purposes, we created 4 different types of audio messages, whose text is shown in the table below.

Event Type	Receiver	Message
New Booking	Customer	Hi. This is an automated message from the Activity Box reservation system and Coral CEA. Your [ActivityName] activity has been successfully reserved and is now confirmed. We hope you have a great time. Thanks for your booking.
New Booking	Operator	Hi. This is an automated message from the Activity Box reservation system and Coral CEA. You have received a new booking for the [ActivityName] activity. Check your e-mail or fax for the booking confirmation and further details and refer to your arrivals report in Activity Box
Cancellation	Customer	Hi. This is an automated message from the Activity Box reservation system and Coral CEA. Your [ActivityName] activity has unfortunately been cancelled by the Operator. Please contact your original booking agent to reschedule or obtain a refund. You will find their contact details on your original booking confirmation. We are sorry for any inconvenience
Cancellation	Operator	Hi. This is an automated message from the Activity Box reservation system and Coral CEA. A booking for the [ActivityName] activity has been just cancelled by the customer. Check your e-mail or fax for the cancellation confirmation and further details

Table 3. Audio Messages

8.4. Useful Links

This section lists some web resources that were useful during the development phase of this project. These links are up to date as of December 2009.

- ActivityBox Demo
  - Demo portal
    http://coraldemo.activitybox.ca
  - Professional voice talent provided by Angelique Papadopoulos
    http://www.greatbritishvoice.com
- Text-To-Speech (TTS)
  - Media Processing Server (MPS) TTS Integration
    http://www.nortel.com/products/04/ivr/collateral/nn103943.pdf
  - AT&T Labs Natural Voices® Text-to-Speech Demo
    http://www2.research.att.com/~ttsweb/tts/demo.php
- Windows Services in .NET
  - Creating a Windows Service in .NET
    http://www.developer.com/article.php/2173801
  - Creating a .NET Windows Service: Three Different Approaches
    http://en.csharp-online.net/Creating_a_.NET_Windows_Service
  - Creating an Extensible Windows Service
    http://www.15seconds.com/issue/021007.htm
- Web Services:
  - Web Services Description Language Tool (Wsdl.exe)
    http://msdn.microsoft.com/en-us/library/7h3ystb6(VS.80).aspx
  - Web References
    http://msdn.microsoft.com/en-us/library/tydxdyw9(VS.80).aspx
  - How to: Call a Web Service
    http://msdn.microsoft.com/en-us/library/6h0yh8f9(VS.80).aspx

- Database:
  - MSSQL notifications (to avoid periodic queries to the database):
    http://msdn.microsoft.com/en-us/library/a52dhwx7%28VS.80%29.aspx
  - Enterprise Library 3.1 - May 2007
    http://msdn.microsoft.com/en-us/library/aa480453.aspx