Interactive Routing

How to create a cloud contact center with NoACD/NoIVR technologies

White Paper

By Nikolay Anisimov

March 15, 2017

Executive Summary

In this whitepaper we present a patent-pending technology for developing contact centers in cloud environments. The keystone of this technology is the creation of a Contact Center Model (CCM) that could be transformed, in a straightforward manner, into executable logic of cloud-based applications. The contact center model is a collection of interoperable models, such as a call model and agent model, but covering all functionalities of contact centers, such as interaction processing, routing, self-service, agent and customer management.

More specifically, we suggest contact center formalism (CCF) and a corresponding programming framework for the creation and execution of contact center applications, comprising all logic of interaction processing. CCF allows us to describe a structure of CCM at an ontological level by defining the main concepts, relationships and behavior of all contact center components, such as customers, their interactions, agents, routing strategies, campaigns, etc. Moreover CCF enables one to specify actual contact center configurations and represent real-time operations in a contact center.

CCF is a formalism built as a combination of finite state machines, labeled property graphs, and dynamic operations. One of the main advantages of CCF is that it is ideal for the representation of relationships between contact center entities such as customer interactions and agents, agents and their skills, customers and their needs. For instance CCF enables one to specify different types of interaction routing, including routing based on agents' statistics, skills, organizational structures, customer properties, and many others.

The detailed description of the technology is contained in the corresponding technical report. We also demonstrate the approach with prototype IntRo implemented in AWS with a Neo4j graph database for model layers and an AWS Lambda service for functional microservices.

Introduction

Virtually all businesses, of all sizes, who deal with customers ought to have a multi-channel customer service. In today's highly competitive business environment, quality of customer service can often become a critical differentiator. Unfortunately, not all businesses can afford dedicated customer services for several reasons.

On-premise contact centers are very expensive as they require special equipment and software as well as fully trained personnel both for creating applications and providing proper maintenance.

Introducing hosted contact centers with cloud computing has improved this situation to some extent, but it is still faced with additional difficulties, such as complexity of contact center software. It turns out that moving existing contact center software to the cloud requires a substantial redesign, predominantly related to providing such infrastructure properties as scalability, reliability and multitenancy. To our knowledge, despite enormous ongoing efforts, none of all current providers have successively coped with these tasks so far. As a result, considerable effort has been focused on these non-functional requirements, at the expense of contact center functionality. Therefore, contact center software is still too expensive and complicated for usage and not readily affordable for the majority of businesses.

At the same time, over the last few years we have witnessed a new long-awaited wave of advanced cloud technologies related to microservice architectures and serverless approaches to the creation of cloud applications, such as Amazon Lambda service. These technologies automatically provide all needed cloud infrastructure, including reliability and scalability, offering an opportunity for developers to focus only on the business logic of their applications.

It may, however, come as a surprise to discover that this approach creates a new problem. The new bottleneck pops up when developers are faced with need to implement the pure logic of contact center applications. A number of questions arise, such as how to arrange a structure of application, how to define an interaction processing and routing logic in particular, how to manage agents and other resources, what programming language to use? To answer these and other questions, we need a new approach to create contact center infrastructure and applications.

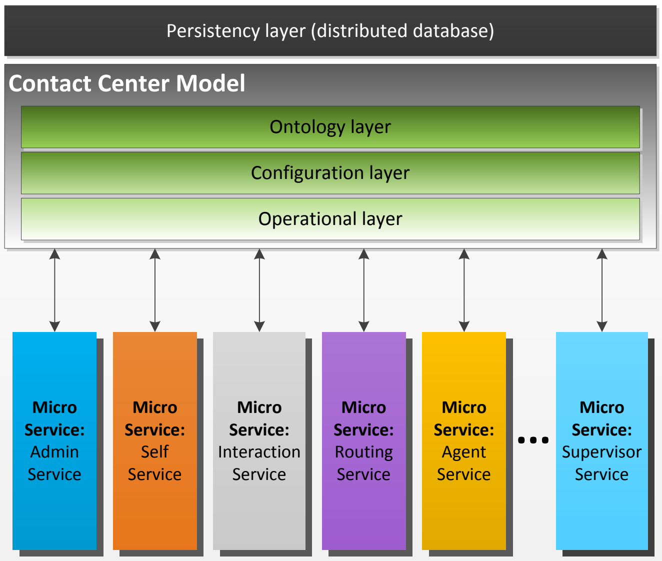

Persistency layer (distributed database)
Contact Center Model
Ontology layer
Configuration layer
Operational layer
Micro Service: Admin Service
Micro Service: Self Service
Micro Service: Interaction Service
Micro Service: Routing Service
Micro Service: Agent Service
...
Micro Service: Supervisor Service

Figure 1: Microservice architecture of contact center

In this paper we suggest a new model-driven approach to the creation of contact center software. The approach supposes the creation of well-defined models for main components, and processes of contact centers that could be straightforwardly transformed into executable application logic for different cloud platforms.

General picture

The high-level view of the contact center environment is presented in Figure 1. A contact center infrastructure is built as a set of microservices that work with a single contact center model (CCM). CCM is a central component containing all information about a contact center structure, its static configuration, and operational context.

Each microservice implements a self-contained portion of contact center functionality. For example, an interaction service is responsible for receiving customer interactions and further communication with customers via these interactions. Another example of a microservice is an agent service that is responsible for manipulations with agents and their statuses.

What is a contact center model?

CCM is the component containing a strict definition of an entire contact center structure and behavior, in abstract terms, independent to any particular implementation environment. CCM could be perceived as an expansion and generalization of the well-known call model and agent model towards capturing all other functionalities of contact centers. For instance, CCM should cover such important functionalities as interaction routing, self-service, dialog management, campaign management, reporting, etc.

> Metaphorically, contact center ontology is a genotype of a contact center.

CCM comprises three layers – operational, configuration and ontological. An operational layer is a layer where such entities as customer interactions, agents, and campaigns live and evolve in real-time. The main elements here are entities and relationships between entities which reflect actual relationships between customers, customer interactions, agents and other resources of the contact center.

A configuration layer contains static information about the contact center infrastructure, including the particular configuration of agents, their skills and other properties, knowledge base, etc.

An ontological level comprises contact center ontology (CCO), an ontological model that contains high-level knowledge about the structure (schema) of the contact center. More specifically, CCO contains knowledge about the main concepts of the contact center (e.g. interaction, agent, skill, campaign, knowledge article) and relationships between them. CCO also contains inference rules for specifying the evolution of the model in time. For example, routing of customer interactions is expressed as inference rules that create new matching relationships between these interactions and appropriate agents based on interaction and agents' properties.

Metaphorically, CCO is a genotype of a contact center. It contains all information about the structure of contact center entities, their relationships, and how the contact center will evolve in time. The main advantage of CCO is that it is not only a declarative specification, but at the same time it is executable. CCO enables the automation of the process of creation configuration and operation.

CCF: Main formalism

As main contact center formalism (CCF) for representation of CCM models of all layers, we use a combination of a labelled property graph (LPG) with final state machines (FSM). LPG is the model of a directed graph where each node and edge is equipped with properties and labels. Nodes specify entities of a contact center and edges represent the relationship between entities. LPG is a powerful and simple formalism with a sound, intuitive meaning thanks to good graphical representation. At the same time, it is a completely strict formal model treatable by computer programs. LPG is supported by several graph-oriented NoSQL databases. The main advantage of LPG is explicit support of the notion of relationship. A relationship is a key concept in the contact center model. Indeed, the principal relationships are relationships between customer interactions and agents, as well as other resources in a contact center. Establishing these relationships is a key mission of contact centers. To accomplish a mission, other relationships are used, such as relations between agents and their skills, customers and their needs and capabilities, contact center resources, and other metadata.

Some properties of nodes and relationships of LPG could be states of state machines associated with them. These state machines govern the dynamic evolution of the system.

What are novelties?

The model-driven approach to the creation of contact centers has numerical advantages. Below we consider the most impressive ones. First we explain why we characterize the technology as "NoACD" and "NoIVR".

Routing: Why NoACD?

Clearly this term NoACD relates to interaction routing. Routing logic is defined at the CCO level as a collection of routing strategies. It is a part of a contact center application and common for all interactions and agents related to the application.

In short NoACD stands for "Not only ACD". In greater detail, this technology allows you to use not only standard types of routing such as skill-based and ACD-based routing, but many others. Moreover it provides an opportunity to create your own custom types of routing, taking into consideration your specific business needs.

Models of routing comprise models of customer interactions and agents, as well as models of establishing matching relationships between them. All known types of interaction routing, such as routing based on agent skills, statistics, cost and values, are naturally specified by routing models. Moreover, other more sophisticated types of routing could also be easily implemented. For example, the combination of different routing types in one strategy, as well as using escalation between strategies, allows for the creation of sophisticated and optimal interaction processing within the contact center.

Figure 2 illustrates the use of LPG for representing a simple skill-based routing type.

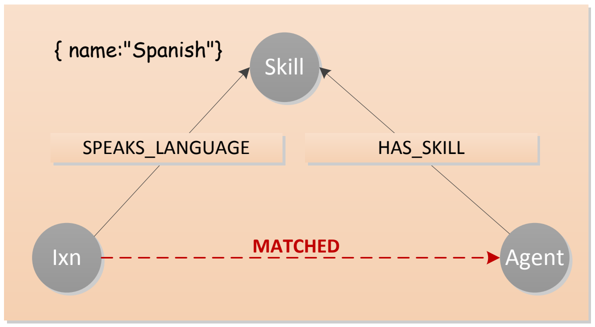


Figure 2: Simple skill-based routing

There are three instances - interaction, agent and skill types. The skill instance represents a configuration entity with the name "Spanish". The agent has the skill corresponding to their ability to speak Spanish. This fact is specified by the relationship HAS_SKILL. On the other side, an interaction represents a customer who can speak Spanish. This fact is specified by the relationship SPEAKS_LANGUAGE. A skill-based routing strategy finds that there is a common language and creates the relationship MATCHED between the interaction and the agent. The routing inference rule is a routing logic and is a part of CCO.

Dialogs: Why NoIVR?

Again, in short, NoIVR could be expanded as "Not only IVR". However, in a broader sense, it means that IVR functionality will be changed dramatically.

Routing of customer interactions require a certain amount of metadata about these interactions and the customers. Collecting this missing metadata could be accomplished by carrying on conversations with customers. These conversations are generated on-the-fly and executed with the aid of dialog management models. The system determines what data is needed for routing and generates and executes a corresponding dialog with a customer.

In the above example, in Figure 2, the interaction has the relationship SPEAKS_LANGUAGE. However, initially, this relationship does not exist and needs to be created as a result of a conversation with a customer. In accordance with CCO, the system creates a dynamic dialog and executes it with the aid of the corresponding media channel. When the language is determined, the corresponding relationship is created in the model and the routing service is triggered.

The main advantage of this approach is that only that data which is relevant for routing is collected.

Another advantage is that dialog is media independent. A generated dialog is extracted from CCM and expressed in an abstract and media-independent format and is transformed into executable dialog with a media-specific format. For example, if the media type is voice, or text chat, then an abstract dialog is converted into a sequence of questions and answers (aka slots). Actual rendering of executable dialog is performed within a corresponding media channel.

Conversation with a customer is also used in self-service with the aid of a self-service model that describes a knowledge base and the interactive navigation within the knowledge base.

True omnichannel

Notice CCM is media agnostic and could be successively used in omnichannel contact centers. Indeed, from a technical perspective, an omnichannel contact center supposes a uniform processing of interactions for all media channels.

The concept of the omnichannel contact center emerged in response to the need to reduce the complexity of interaction processing, making them uniform for all channels.

Media independence in interaction processing enables a flexible usage of media channels, including a parallel media channel, and a smooth transition between channels. The latter enables you to start processing a customer interaction through one media channel, and complete it through another one.

Media channels are connected to interaction processing through relevant microservices, such as an interaction microservice.

How it works

Now creation of a contact center is easy. First we should define ontology of a contact center. More strictly, we will need to specify main concepts such as customer interactions, agents, their properties and relationships, routing logic, etc. To accomplish this task, we could use a corresponding framework with an administrator desktop. The predefined set of out-of-box ontology elements could help in performing this task.

Then we should create a configuration of our contact center. For example, if a CCO defines a concept of agents' skills, we will need to specify a concrete set of skills for our contact centers. Other contact centers will have their own set of agents' skills. The configuration could also be defined with the aid of the framework and administrator desktop.

Configuration also involves connecting to media channels.

When ontology and configuration is defined, the system is ready for operation. Operation of a contact center is where a dynamic part of CCM comes to life. Customer interactions enter the system and modify it. Agents change their statuses and also change the system. Execution of dialogs and routing of interactions and assignment of them to agents are also a part of contact center operation.

Born in cloud

There are different ways to access the cloud. A contact center created on premise could be moved to the cloud, but with considerable difficulty (if not impossible) as it requires serious architectural redesign. Even if a contact center were born in the cloud, it does not mean that it will survive in the cloud as it requires complex system architecture with cloud-specific features such as scalability, reliability, and multi-tenancy. In our approach, however, a contact center is initially conceived in the cloud and gains its cloud birth-marks from the beginning. All non-functional and cloud-oriented features are the responsibility of a particular cloud infrastructure, leaving us a creation of pure contact center application logic. However, this logic is embodied in the form of the contact center model.

Prototype

This document contains the description of a prototype called IntRo. The prototype contains CCM with a sophisticated skill-based routing and simple pull-based interaction assignment. The model layers are implemented as a graph database Neo4j that is deployed in the Amazon Web Services (AWS) cloud.

Figure 3 demonstrates a CCM graph of a simple contact center application for a pull-based interaction assignment. The graph contains elements of all tree model layers. The ontological layer comprises classes IXN, AGENTS, and MATCHED along with their properties. The configuration layer contains an agent instance (node in green). The operational layer has five interaction instances (nodes in gray). All interactions and the agent are connected by relationships MATCHED, saying that the agent could pull one of several of these interactions for handling.

Agent, interaction and routing services are implemented as an AWS Lambda service with API Gateway. IntRo comprises such media channels as SMS and voice from Twilio, web form and a web chat.

The prototype also uses a universal desktop for agent, supervisor and administrator. The snapshot of the user desktop is presented in Figure 4.

Conclusion

In this whitepaper, we have presented a new technology for the creation of contact centers in a cloud environment. The technology comprises the following components:

- Formal model based on CCF to specify contact center ontology and configuration, including a set of out-of-box models.
- Framework for creation of CCMs for concrete contact centers.
- Universal desktop enabling access to contact center activities by users in different roles, such as agents, supervisors, and administrators.
- Execution environment governing operation of CCM in real time, comprising model layers and microservices.

Based on this technology, you can define your own contact center model that fits your business needs, select or create an optimal routing strategy, specify a configuration of your contact center, and deploy it in an executable cloud environment.

The detailed description of the technology is contained in the corresponding technical report and demonstrated in the prototype IntRo.

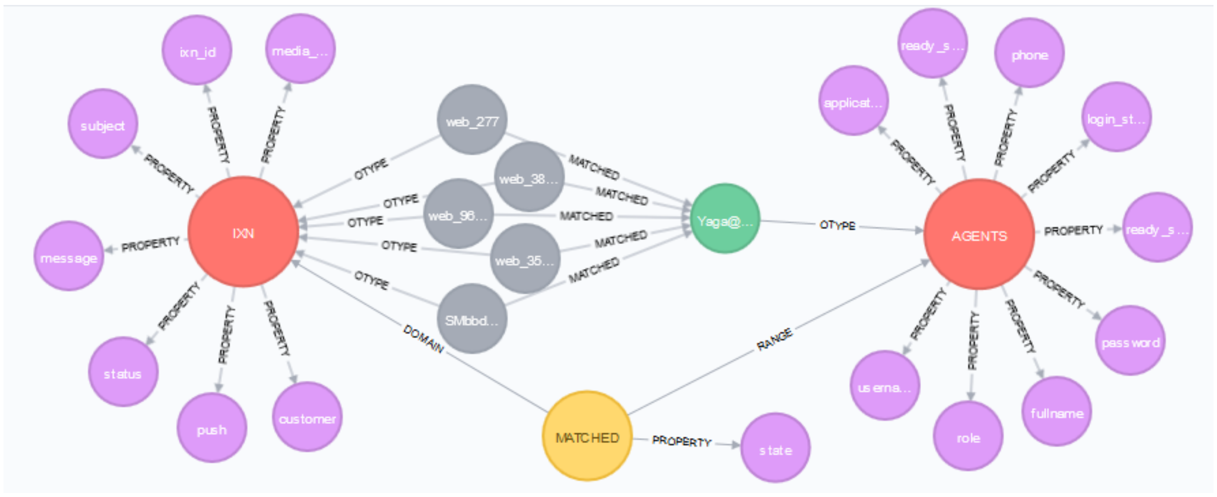
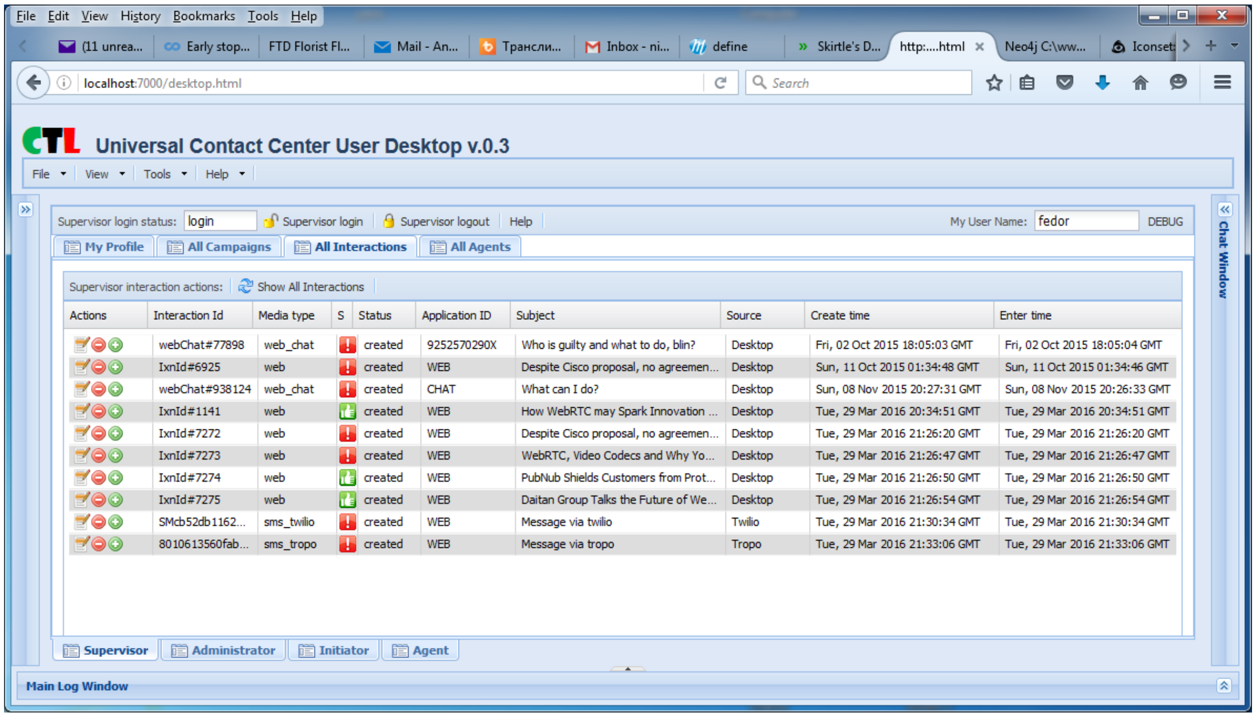


Figure 3: CCM graph of simple contact center application (Neo4j snapshot)



Actions	Interaction Id	Media type	S	Status	Application ID	Subject	Source	Create time	Enter time
	webChat#77898	web_chat	❗	created	9252570290X	Who is guilty and what to do, blin?	Desktop	Fri, 02 Oct 2015 18:05:03 GMT	Fri, 02 Oct 2015 18:05:04 GMT
	IxnId#6925	web	❗	created	WEB	Despite Cisco proposal, no agreemen...	Desktop	Sun, 11 Oct 2015 01:34:48 GMT	Sun, 11 Oct 2015 01:34:46 GMT
	webChat#938124	web_chat	❗	created	CHAT	What can I do?	Desktop	Sun, 08 Nov 2015 20:27:31 GMT	Sun, 08 Nov 2015 20:26:33 GMT
	IxnId#1141	web	👍	created	WEB	How WebRTC may Spark Innovation ...	Desktop	Tue, 29 Mar 2016 20:34:51 GMT	Tue, 29 Mar 2016 20:34:51 GMT
	IxnId#7272	web	❗	created	WEB	Despite Cisco proposal, no agreemen...	Desktop	Tue, 29 Mar 2016 21:26:20 GMT	Tue, 29 Mar 2016 21:26:20 GMT
	IxnId#7273	web	❗	created	WEB	WebRTC, Video Codecs and Why Yo...	Desktop	Tue, 29 Mar 2016 21:26:47 GMT	Tue, 29 Mar 2016 21:26:47 GMT
	IxnId#7274	web	👍	created	WEB	PubNub Shields Customers from Prot...	Desktop	Tue, 29 Mar 2016 21:26:50 GMT	Tue, 29 Mar 2016 21:26:50 GMT
	IxnId#7275	web	👍	created	WEB	Daitan Group Talks the Future of We...	Desktop	Tue, 29 Mar 2016 21:26:54 GMT	Tue, 29 Mar 2016 21:26:54 GMT
	SMcb52db1162...	sms_twilio	❗	created	WEB	Message via twilio	Twilio	Tue, 29 Mar 2016 21:30:34 GMT	Tue, 29 Mar 2016 21:30:34 GMT
	8010613560fab...	sms_tropo	❗	created	WEB	Message via tropo	Tropo	Tue, 29 Mar 2016 21:33:06 GMT	Tue, 29 Mar 2016 21:33:06 GMT

Figure 4: IntRo universal desktop (supervisor view)

About author

Nikolay Anisimov, Ph.D., is a computer scientist with a strong academic background. He is an industry veteran with twenty years' experience in contact center technologies, is the author of numerous patents, technical and research papers, articles in industry journals and whitepapers. Nikolay has worked for Genesys Telecommunication Labs, Alcatel-Lucent, Front Range Solutions, Five9, Aspect Software, and Bright Pattern, Inc. He is a co-founder of Contact Technology Labs, Inc.
You can contact Nikolay at Email: anisimov@computer.org
His LinkedIn profile: https://www.linkedin.com/in/nikolayanisimov

About Contact Technology Labs.

Contact Technology Labs, Inc. is a privately held, California-based company that performs advanced research in the area of contact center technologies and related fields. The main goal of the research is the development of a contact center of a new generation based on a model-driven approach and fully suitable for deployment in a cloud computing environment. The company also does consulting work to solve various practical problems related (but not limited) to contact center modeling and simulations, interaction routing, algorithms, outbound dialing, self-service, cloud computing, artificial intelligence, machine learning, natural language processing, and data science.
For more about Contact Technology Labs, visit: http://www.contacttechnologylabs.com

Address of this paper:
http://fiztech-usa.net/anisimov/papers/Interactive%20Routing-WP.pdf