Redesigning the Voice Relevance Engine for Scale and Reliability

A Technical Paper prepared for SCTE by

Eileen Bengston
Principal Solutions Architect
Comcast
1700 JFK Boulevard, Philadelphia PA
(267)260-3370
eileen_bengston@cable.comcast.com

Ferhan Ture
Senior Director Machine Learning
Comcast
1325 G Street NW, Washington DC
(202)524-5060
ferhan_ture@cable.comcast.com

Table of Contents

List of Figures

1. Introduction

At SCTE in 2012, a group from Comcast presented the Voice Relevance Engine for Xfinity (VREX). Its goal was to understand human utterances for video search and discovery and to take an appropriate action. Since then, Comcast's voice system expanded to cover numerous platforms, won a technical Emmy, and was used for more than 10 billion voice commands last year alone. After running at scale for 10 years, the voice system is being redesigned to suit the new and growing demands for voice control across a range of products.

The original system as described at SCTE in 2012 [1] was divided into three major areas: automated speech recognition (ASR), natural language processing/natural language understanding (NLP/NLU), and action resolution (AR). These services and the division of labor therein include: first determine what the user said, second understand what the user meant, lastly decide what the system should do to resolve the request.

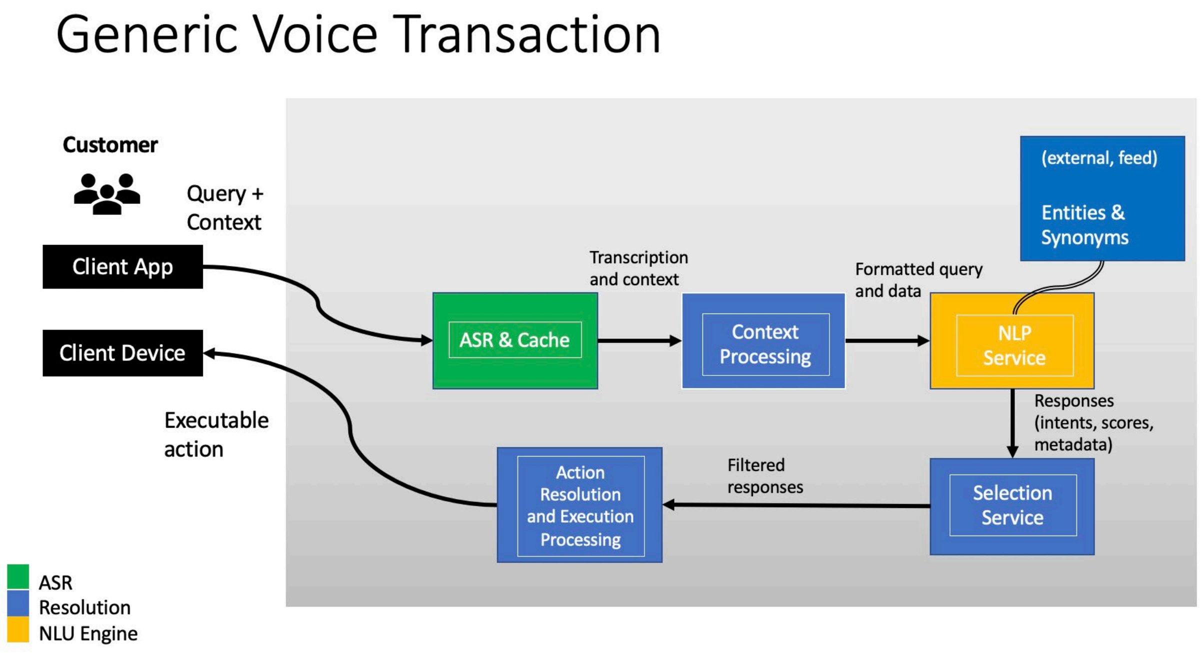


Figure 1 - Basic NLP Process

For the example "show me Adam Sandler movies" the ASR will transcribe the utterance to a text string "show me Adam Sandler movies". Then, the NLP will attempt to ascribe meaning to the transcription. In this case it will canonically break up the string into <show me> an action, <Adam Sandler> a person, and <movies> a content type. In a simple system, Action Resolution involves the system choosing to provide a list of movies starring actor Adam Sandler. The information is provided in a way that can be displayed on the target device.

The above model is simple and linear. But what happens when there are several domains to cover? (A domain is an area of knowledge like video, customer experience, or home.) More information is needed to determine the appropriate resolution. Is the user interacting with a television? Are they walking in the front door? Are they talking to their car? Context as to when and where the utterance occurred affects what the domain is for processing the request and what the action resolution will be. There may be several domains that could be applicable, so there can be multiple responses. Selection then must occur based on some quantification of how likely it is that the result is the expected solution. The Action Resolution phase increases in complexity as well because there is more to resolving the action. If the user is in their car speaking to the GPS, the Adam Sandler query should generate an error saying that it cannot find that road. But if the user is speaking to their television, the action resolution should result in a list of movies starring actor Adam Sandler being displayed on the television.

In practice, the VREX system grew organically to support a significant amount of complexity. The redesign assessed the current requirements for the system, those expected in the near future, and the complexity in deployment and support of the voice system. The voice recognition system was redesigned to improve the development, deployment, and servicing of the individual components of the system. These changes will be discussed as they affect each portion of the system.

2. Current Voice Relevancy System

Over time the number of domains, the number of languages, and the breadth of possible uttered commands has increased dramatically. So too, has the relative complexity of the system. The initial design supported speaking into a phone app to choose content to watch on a television set. The use case at design time was that a mobile device was paired to a television and all commands were for tuning content on the television. That model has grown to support not only tuning content but most television controls across smart televisions, voice remotes, hands-free devices, and mobile devices. It now covers many languages in multiple countries. In addition to video, the Voice System processes customer experience, home, sports, and other domain requests. Currently the model looks like this:

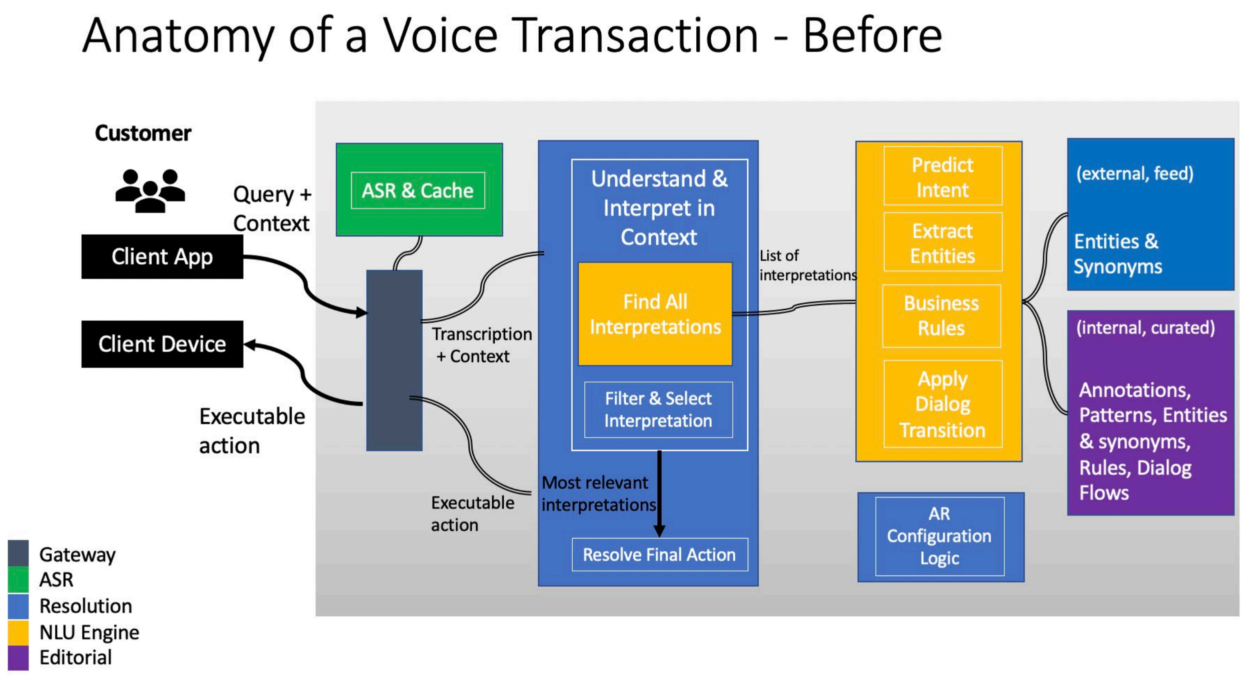


Figure 2 - VREX Voice Transaction (Before)

This drawing is a simplification of the model but includes several parts not included in the earlier simple block diagram. Specifically, it addresses the training of the model including the input of external feeds and the editorial, both of which are a significant piece in managing and training the NLU model. It also includes the configuration necessary to interpret the context of various domains.

Of course, this drawing also excludes several things. There is a significant amount of logic needed for routing, authenticating, and gathering context prior to context processing. While it shows a box for the gateway into the system it does not include the additional external calls to gather context and other needed information. And it does not include any of the testing and observability tools that aid the release and monitoring of the system. But importantly, this drawing adds a level of detail to context processing, NLP, selection, and action resolution. The structure of dealing with a voice command is far from linear. There are loops back to the decision engine with duplication of some common features. This level of interconnectedness, complexity, rigidity, organic system growth, and escalating support costs led to the redesign of the Voice Relevancy System.

3. Redesigning Voice Relevancy System

The goal in redesigning VREX was to improve the development, deployment, and servicing of the components that make up the voice recognition system. This does not mean that the model was simplified, in fact, the drawing below shows there are more components involved. But it allows better separation of work and modularity thus improving flexibility and decoupling and leading to a system that is more reliable and easier to maintain.

3.1. External Interfaces

The gateway for the VREX system had the single purpose of getting queries into the system and returning executable actions, but it grew organically over time and was expanded to cover several disparate cases. The gateway was responsible for directing voice queries to the ASR, gathering metadata from various points in the system (simultaneously to reduce delay), and passing ASR results, context, and metadata to the interpretation and resolution services. It was then responsible for passing the resultant resolution action back to the calling device.

3.1.1. Queries

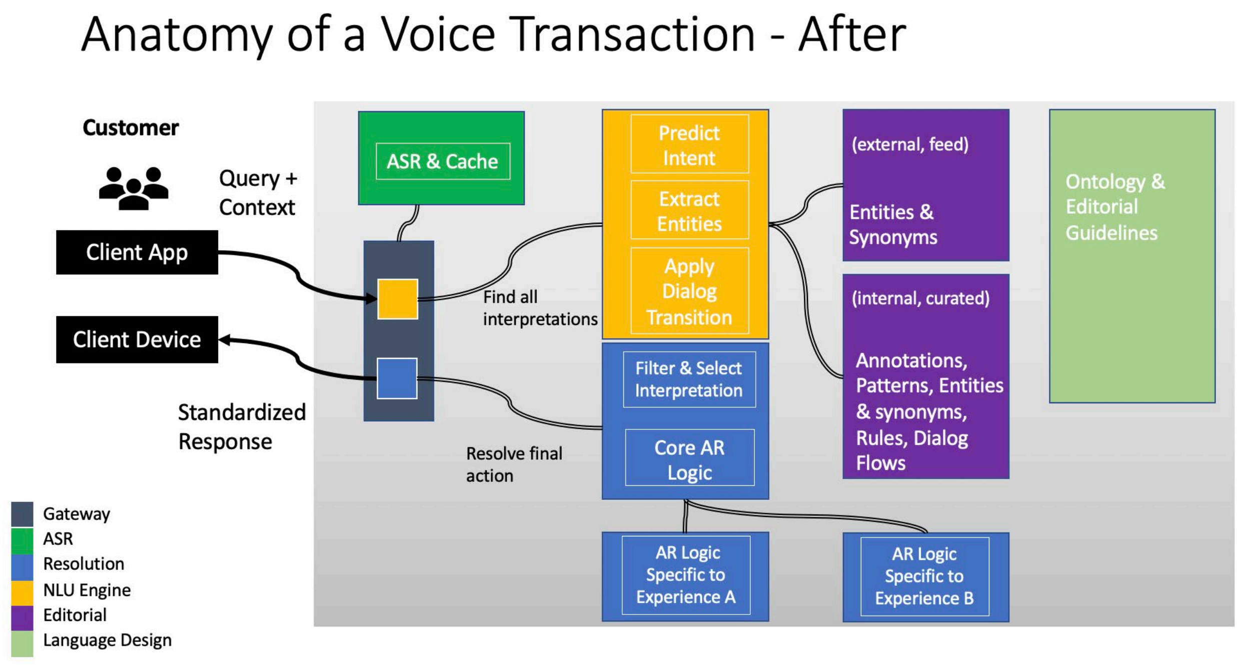


Figure 3 - VREX Voice Transaction (After)

As the client base using VREX expanded, so did the interaction patterns. Rather than just speaking into a mobile device, there was the voice remote push-to-talk, hands-free devices, and text-based chatbots. Some of these clients need ASR, NLU, and AR. But others need only NLU or ASR. The voice system was designed with a single input that gathered all information from multiple sources for every query. It allowed the client to specify the use of ASR, NLU, and AR, but it followed the same pathways regardless. Systems that used the older http requests were on a different, and older, version of the API and the teams had to keep multiple distinct APIs, and the stacks that support them up to date or else allow the older APIs to lag in functionality. It directed transcriptions to a dispatcher that did the domain selection based on the additional context, and answer selection based on the results from the NLU engine. This meant that every query entered the system, gathered all the external information, and traversed the same path through dispatch and interpretation regardless of what the query really required.

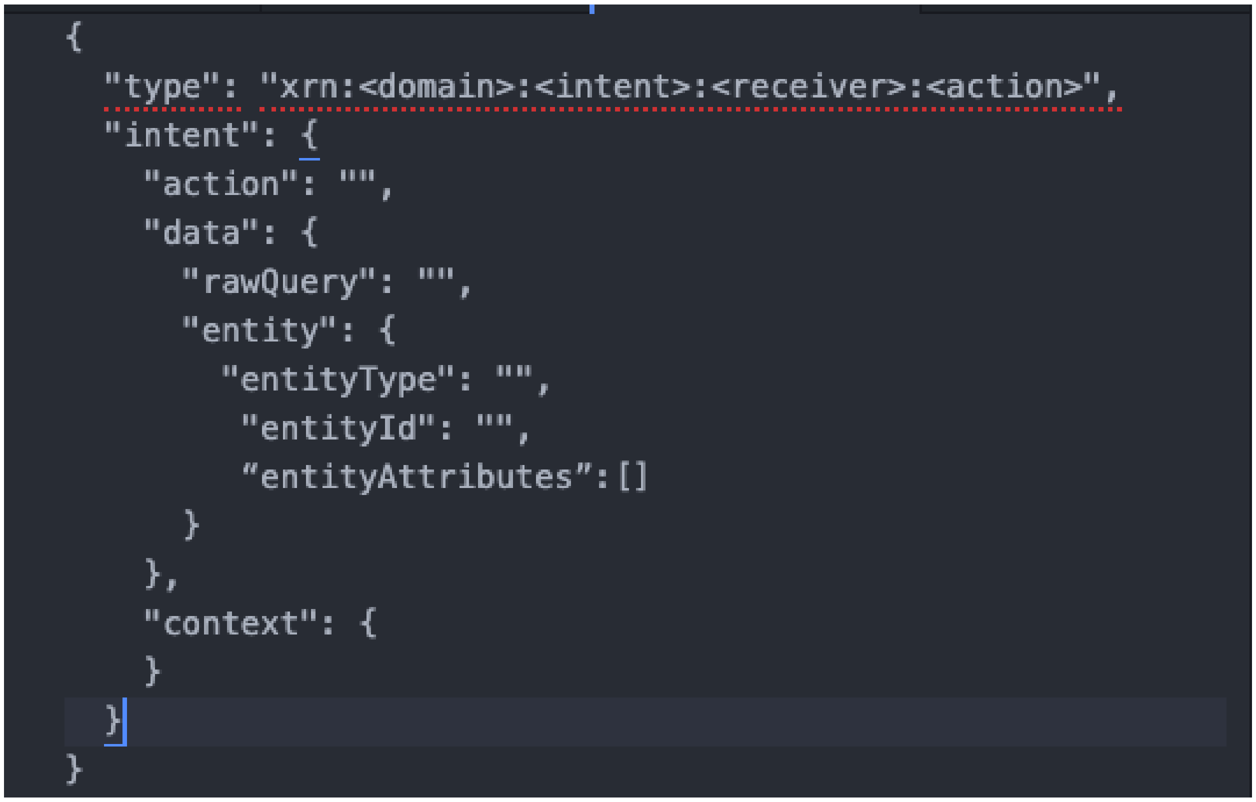
This web socket request is still used for voice commands, especially for session-based, multiple message communications. Commands requiring only NLU may prefer to use a lighter weight http call. Most importantly, the gateway can direct the request and so it can be passed where it is needed. If no interpretation is required, then that portion of the system can be skipped entirely.

Being more intentional in what the gateway does, and does not do, allows better use and routing of the services in the system. The client is now responsible for providing whatever additional context is required. The gateway's sole responsibility now is to maintain the contracts for routing information to the right service. It matches the usage of the product with the service and verifies the type of input and output expected. More calls go through the gateway, but it is skinnier and faster. It includes the API to client services, authentication logic, and configurable logic for dispatching messages based on the experience. It can send audio and context to the speech engine, transcriptions, metadata, and context to the NLU engine, and logical forms to the resolution service.

3.1.2. Responses

A common way of communicating intents back to the client regardless of the experience in question was introduced to decouple the voice system from the experiences they support. The response messages were standardized because prior to the redesign, each client experience could determine what and how they wanted to receive responses from the voice system. This meant that the logical forms created by the NLU had to be translated for each client experience. Many used deeplinks directly to the resultant pages with the result that any resolution often had to be custom designed to support a specific client and changed when the client UI changed. By defining a standard interface, similar, but different from the logical forms created by the NLU system, the entire VREX system has a known language through which it communicates user intents and was separated from client-side changes.

In this case the action resolution provided general intents that include all the information that a client may need to determine the appropriate UI experience to present to the user. It is not prescriptive of the specific page to render, so the same structure can be used by multiple experiences and does not change if there are changes to the UI flow. A result of this is that some of the information is supplied in multiple formats, to allow this flexibility. For example, the rawQuery and the entity could be the same information presented differently for use by different clients. General pseudocode structure of an intent response is below.



```
{
  "type": "xrn:<domain>:<intent>:<receiver>:<action>",
  "intent": {
    "action": "",
    "data": {
      "rawQuery": "",
      "entity": {
        "entityType": "",
         "entityId": "",
         "entityAttributes":[]
      }
    },
    "context": {
    }
  }
}
```

Figure 4 - Example Intent Pseudocode

Thin adaptation layers can then be created at the interface where needed. The VREX system is insulated from changes on the client side and the results are more deterministic because the patterns of their resolution and communication are the same.

3.2. NLU and AR

The system was designed for a single client and domain which meant that many of the services were tightly coupled. The speech recognition, NLU, and action resolution are distinct functions and yet, in understanding and interpreting, and doing so within the context of the request, those lines were blurred. This coupling caused cascading changes in the system. For example, when upgrading the ENLP (elastic NLP) which does the pattern matching NLP, the upgrade drove changes into the selection and resolution portion of the system. This was compounded when multiple experiences were supported as the coupling existed for every experience built. Thus, reestablishing the division between the responsibilities of the gateway (as explained above), the NLU, and the filtering and action resolution was needed.

As seen in the 'Before' VREX transaction (Figure 2), NLU related functions for finding interpretations was actually part of the resolution services. Everything that was needed to understand, interpret, and resolve actions for the domain was in one place. But this meant there was tight coupling between the interpretation and resolution services, and the NLU engine to find all the interpretations and resolve them. It also meant that everything had to pass through this interpretation even if action resolution was not required

Re-establishing clear boundaries between services within the VREX system meant separating those functions. The NLU logic exists separate from the interpretation and action resolution logic. The gateway contains the per experience information needed to dispatch the queries appropriately. The NLU receives the query text, context, and any metadata it needs from the gateway, and passes back logical forms. The interpretation and resolution services receive logical forms and select answers, determine resolution, and return standardized responses on a per domain basis. The two services, once again, focus on their core responsibilities and communicate only through the defined interfaces in the gateway.

At the same time, the layers of the system were restructured to separate the experience business rules from the core functionality meaning that the individual pieces that need to be built for a specific experience are thinner, allowing for better reuse. Core logic is that which is needed for turning any voice command into an action. Business logic is the configuration and function that is needed to create a meaningful action in each environment.

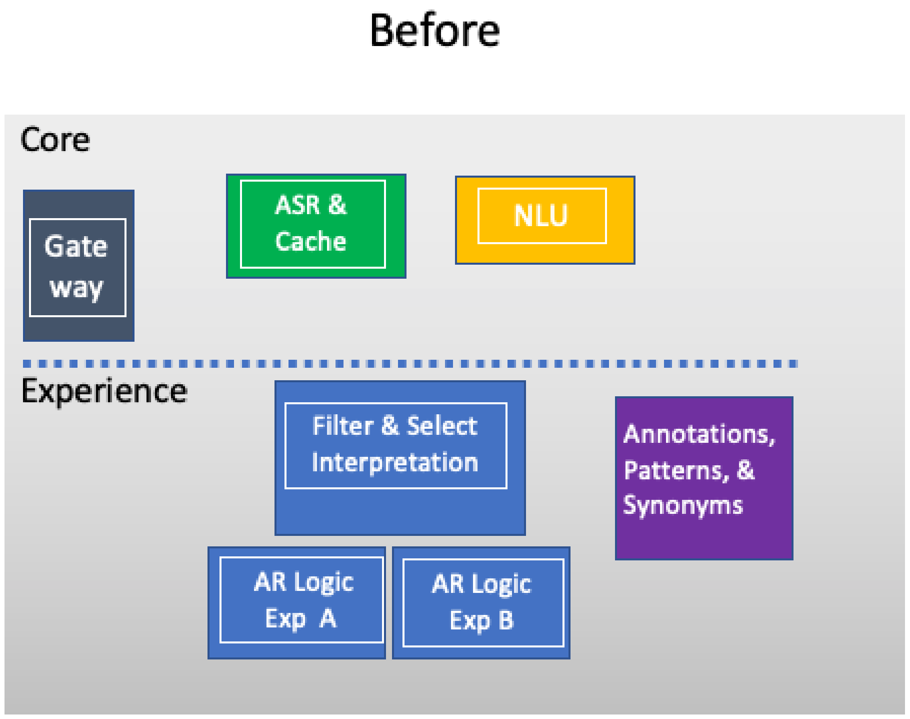


Figure 5 - Core vs Experience Layers (Before)

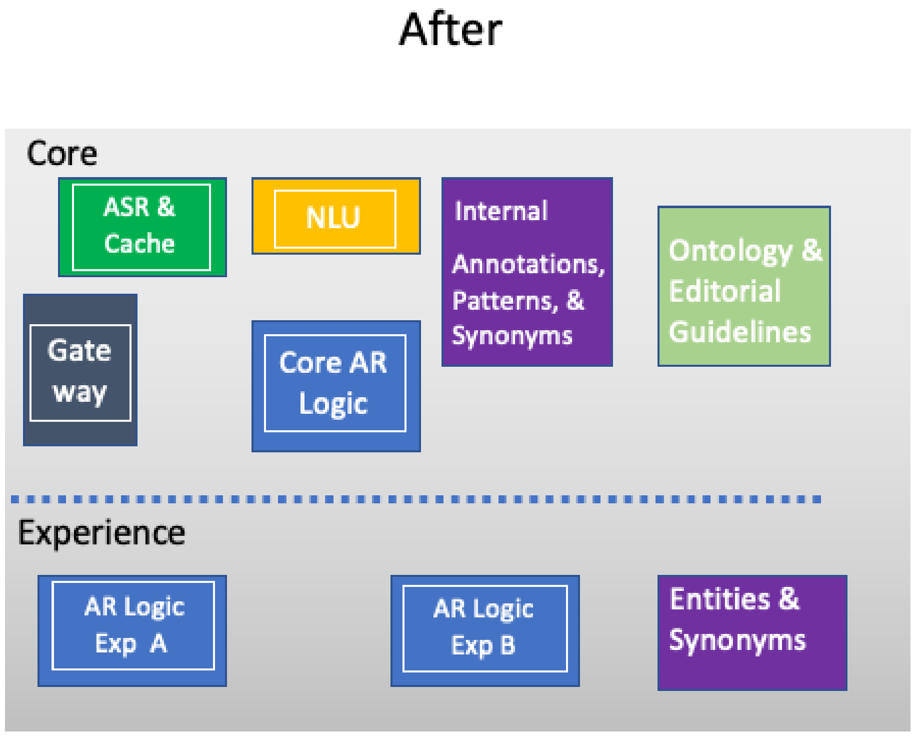


Figure 6 - Core vs Experience Layers (After)

In the above drawings, before the redesign only the ASR and the NLU services were core functionality, and even they had some configuration within the services that required redeployment when changed. Most of the training data input, interpretation, and resolution logic was linked to the specific experience, so each new client or environment involved significant design and deployment resources.

With the redesign, the core filtering interpretation and selection, and the ontology and editorial guidelines, became reusable core functions. This separability of the custom logic from the core logic increases the complexity of the model but greatly reduces the unique work for development and deployment of a specific experience. This then allowed the separation of understanding a query from filtering and selecting the resolution action. This is important because understanding the queries depends on learning processes in the NLU engine and its prediction of the intents based on the appropriate rules. These change only when the model information - the feed, the annotations, or the feedback - change. Filtering and resolution depend on several factors. The first is the fundamental logic for selection and filtering. Understanding what 'play' means is near universal. But how to 'play' depends very much on the experience in question. The goal here was to, as much as possible, separate that which is universal, the meaning of an utterance, from how to address that user intent in each environment.

Reduced complexity and interdependency resulted. Once most of the specific configuration and logic was pushed as far to the edge of the system as possible, the surface area that needs customization for new experiences was reduced. The entire process of supporting a new function, a new client, or a new experience has a much shorter development time.

3.3. Annotations, Synonyms, Ontology, and Testing

To improve the consistency of the system, a set of ontology and guidelines were specified to simplify the patterns for the editorial work that is used to enrich the dataset. Fallback patterns from optimal behavior to acceptable behavior improve both the flexibility and the resiliency of the system. Finally, each service was shifted to use to a common logging, tracing, and debugging system.

As a system grows it gathers a lot of information. The ontology redesign found several ways of representing similar concepts that were being supported simultaneously. The team took the time to define an ontology and guidelines for the editorial additions that enrich the dataset. This involved combing through the existing rules to define reusable patterns, removing duplication, and designing a process of intro, request, update for creating patterns and entering synonyms. This makes for fewer, more consistent patterns, and makes the job of an editor easier because most additions will fit into a series of well-defined patterns. The ontology changes included using labels that create flexibility for ranking importance of certain structured utterances.

With more commonality in patterns and with the separation of the experience specific logic, and its push to the edge of the system, some behavior can be defined on an experience basis without affecting the core logic.

Additionally, while not expressly mentioned previously, any system is going to have a considerable amount of logging and debugging code. Because of the modular nature of the VREX system a lot of this was done by each individual team in the way that best suited them. But this made it disjointed and harder to debug the voice system end-to-end. The re-architecture was an opportunity to take the best and most useful practices across teams and consolidate them into one structured process including our own testing system and ELK (Elasticsearch and Kibana) cluster for common observability. Note that Comcast collects, stores, and uses all data in accordance with its disclosures to users and applicable laws.

4. Conclusion

VREX was presented exactly 10 years ago at SCTE as a novel way to perform voice resolution for the video space at scale. A large-scale system can be expected to have experienced a lot of organic growth and change over a decade. As systems grow organically to suit changing needs, they often take on unexpected forms.

The Comcast team determined its needs as greater flexibility, faster development time for new features, faster deployment time for new customers, and easier and more consistent support. They then redesigned the system, re-defining the APIs, service boundaries, patterns, and ontology. This fundamentally changed how the services can be called and how they respond. It changed service boundaries to separate core functionality from experience driven code, made configuration handling common, and reduced the scope of the experience layers. Finally, they simplified the patterns and ontology that is used to train the model and make decisions. This redesign, while still in the process of being implemented, has made the development and support of the VREX system faster, and more scalable, by reducing the scope of work needed to support an experience, increasing the reusable code, and isolating the code in the system that needs to be modified closer to the edge of the system.

Abbreviations

API	Application Programming Interface
AR	Action resolution
ASR	Automatic Speech Recognition
BIO	Beginning-inside-outside
ELK	Elasticsearch and Kibana
ENLP	Elastic Natural Language Processor
HTTP	Hypertext Transport Protocol
NLP/NLU	Natural Language Processing/Natural Language Understanding
SCTE	Society of Cable Telecommunications Engineers
UI	User Interface
VREX	Voice Relevance Engine for Xfinity

Bibliography & References

[1] V-REX – Voice Relevancy Engine For Xfinity, Stefan Deichmann, Oliver Jojic, Akash Nagle, Scot Zola, Tom Des Jardins, Robert Rubinoff, Amit Bagga, Comcast Labs, SCTE 2012.