SCALING ARCHITECTURE THROUGH FLEXIBLE DEPLOYMENT

NANCY A. NUNES
HAZEL AND JĀD TECHNOLOGIES, INC.
Agenda

- 1:30-3 concepts
- 3-3:30 break
- 3:30-5 hands on
Flexible Architectural Artifacts

- components
- interfaces
- protocols
- execution environment(s)
Implementation Features

- Delivery mechanism
- Thread-safe queue
- Dispatcher
- Serialization
“Work at the highest level of abstraction you can afford.”

–Bjarne Stroustrup
CallBack Pattern

Benefits

- Bridge boundaries
- Interchangeable Components
- Various System Configurations
EXAMPLE SYSTEM: SPEECH ANALYZER

EXAMPLE PROBLEM: CONVERT SPEECH TO TEXT
Use Case: Analyze Speech

- **Goal:** To analyze spoken communication for measurable qualities.

- **Preconditions:** The Setup use case has completed successfully, and the Add Users use case has completed successfully.

- **Triggers:** An analyst initiates a session to analyze streamed or recorded speech which may include speech from one or more speakers.

- **Postconditions:** A speech analysis report is presented to the Analyst.

- **Summary:** An Analyst initiates a speech analysis session, the system acquires speech from one or more Speakers, analyzes the speech for the qualities specified by the analyst and then presents the generated report to the Analyst.
SCENARIO: ANALYZE RECORDED SPEECH

1: AnalyzeRecordedSpeech

1.1: TranscribeAudioFile
  1.1.1: GetAttributes
  1.1.2: OpenAudioFile

opt

1: GetAudioData

2: ReceiveTranscriptionResults

3: ShowMessage

2: CloseAudioFile

Repeatedly send transcription results until the last word is transcribed, or a length or time limit is reached.
STARTUP SEQUENCE

- **SpeechAnalysis**: started at initialization
- **UserInterface**: speech analysis registers for notification of UserInterface events using callbacks

Loop: 1: RegisterForEvent

1: Startup
Speech Analysis
Domain Description

- This domain performs analysis on audio data which contains speech.
- It accepts audio streams or recorded audio files.
- It can analyze the content and delivery of speech to quantify different attributes such as volume, rate, tonal range, grammatical correctness and vocabulary usage.
**SpeechAnalysis Services**

- **AnalyzeRecordedSpeech()**
  
  Analyze the recorded speech contained in the specified location.

- **ReceiveTranscriptionResults()**

  Receive a word of the transcription results.
VoiceToText
Domain Description

- This domain transcribes speech to text.
- It supports multiple spoken languages.
VoiceToText Services

- TranscribeAudioFile()

Transcribe the specified audio file, using the specified language, and delivering individual words in sequence via the transcription callback service.
CLASS DIAGRAM: VOICETOTEXT
UserInterface
Domain Description

- This domain provides the interface to system users including analysts and general users.
- It manages information received from and presented to users.
- It relays user actions to registered listeners.
UserInterface Services

- RegisterForEvent()

  Register a callback to be invoked when a certain event occurs.

- ShowMessage()

  Display this message to the specified user.
CLASS DIAGRAM: USERINTERFACE
Callback Pattern → Flexible Deployment

- Single Transaction
- Single Session
- Sessionless Subscription
Define Callback Profiles

SysProfileAnalyzeRecordedSpeech.java

// System class for SpeechAnalyzer_SystemModel.
// System SpeechAnalyzer_SystemModel IncidentHandle profile AnalyzeRecordedSpeech
// defining incident: SpeechAnalysis.AnalyzeRecordedSpeech (DomainService)
// parameters:
//     file_name (sys_url_type_t)
//     language_type (language_type_e)
//     submitter_id (user_id_t)
//============================================================================ */
package SpeechAnalyzer_SystemModel.Sys;

public interface SysProfileAnalyzeRecordedSpeech
{

    public final static int SYS_PROFILE_PARAM_AnalyzeRecordedSpeech_ZERO_NOT_USED = 0;
    public final static int SYS_PROFILE_PARAM_AnalyzeRecordedSpeech_file_name = 1;
    public final static int SYS_PROFILE_PARAM_AnalyzeRecordedSpeech_language_type = 2;
    public final static int SYS_PROFILE_PARAM_AnalyzeRecordedSpeech_submitter_id = 3;
}
Sys.java

/* define process ids for the system */
/** default process id used in single process system */
public final static int PROCESS_DEFAULT_LABEL = 0;

/** task id for process MAIN task SYS_TASK_ID_MAIN*/
public final static int PROCESS_MAIN_SYS_TASK_ID_MAIN = 0;
public final static int PROCESS_MAIN_SYS_TASK_COUNT = 1;

/** Domain number for SpeechAnalysis.*/
public final static int DOMNUM_SpeechAnalysis = 4;
public final static int DOMNUM_UserInterface = 5;

SpeechAnalysis.java

/**
 * Unique identifier for domain service AnalyzeRecordedSpeech.
 * @see SpeechAnalysis#AnalyzeRecordedSpeech(String, int, int)
 */
   public static final int SVCNUM_AnalyzeRecordedSpeech = 8;

// UserInterface:RegisterForEvent( analyze_stored_svc, USER_EVENT_ANALYZE_STORED_AUDIO);
UserInterface.RegisterForEvent(analyze_stored_svc,
SpeechAnalyzer_SystemModel.Sys.Sys.USER_EVENT_ANALYZE_STORED_AUDIO);
Generic Single Thread Callback Sequence
PfdIncident analyze_stored_svc;
   // IncidentHandle analyze_stored_svc = CREATE IncidentHandle (file_name = "", language_type = LANGUAGE_TYPE_UNKNOWN, submitter_id = -1 ) TO SpeechAnalysis:AnalyzeRecordedSpeech;
   analyze_stored_svc = new PfdIncident(PfdIncident.SERVICE_HANDLE_INCIDENT,
      Sys.PROCESS_DEFAULT_LABEL,
      PfdTask.getTask(Sys.PROCESS_MAIN_SYS_TASK_ID_MAIN),
      Sys.DOMNUM_SpeechAnalysis,
      SpeechAnalysis.SVCNUM_AnalyzeRecordedSpeech, 3, null, null, 0);
SpeechAnalysis.java

analyze_stored_svc.setParameter(new PfdDataContainer((String)""), SYSPROFILEPARAM_AnalyzeRecordedSpeech_file_name);

analyze_stored_svc.setParameter(new PfdDataContainer((int)SpeechAnalyzer_SystemModel.Sys.Sys.LANGUAGE_TYPE.UNKNOWN), SYSPROFILEPARAM_AnalyzeRecordedSpeech_language_type);

analyze_stored_svc.setParameter(new PfdDataContainer((int)-1), SYSPROFILEPARAM_AnalyzeRecordedSpeech_submitter_id);
Make Request
(Response CB)

SpeechAnalysis.java

// UserInterface:RegisterForEvent(analyze_stored_svc,
// USER_EVENT_ANALYZE_STORED_AUDIO);

UserInterface.RegisterForEvent(
   analyze_stored_svc,
   SpeechAnalyzer_SystemModel.Sys.Sys.USER_EVENT_ANALYZE_STORED_AUDIO);
Clone & Set Parameters then Deliver

UserInterface.java

```java
...
cc_notification = (PfdIncident)listener.notification.clone();
...
cc_notification.setParameter(new
PfdDataContainer((String)(((UserInterface_HumanUser)hazel).getnewestStoredAudio())),
SYS_PROFILE_PARAM_AnalyzeRecordedSpeech_file_name);

cc_notification.setParameter(new
PfdDataContainer(((int)(((UserInterface_HumanUser)hazel).getlanguage())),
SYS_PROFILE_PARAM_AnalyzeRecordedSpeech_language_type);

cc_notification.setParameter(new
PfdDataContainer(((int)(((UserInterface_HumanUser)hazel).getuserId())),
SYS_PROFILE_PARAM_AnalyzeRecordedSpeech_submitter_id);

cc_notification.deliver();
```
Receive Response

PfdIncident.java

```java
public void sendLocal()
{
    /* If this is a local service handle, just dispatch it */
    if (type_ == SERVICE_HANDLE_INCIDENT)
    {
        dispatch();
    }

    ...
}

public void dispatch()
{
    PfdTask.router_.routeServiceInvocation (this);
    ...
```
SpeechAnalysis.java

/**
 * Dispatch indirect service
 * @return FALSE if any errors encountered
 * @param pfdhandle IncidentHandle to be dispatched
 */

public static boolean dispatchService_(PfdIncident pfdhandle)
{
    ... 
    switch (incident_index)
    {
    ... 
    case SpeechAnalysis.SVCNUM_AnalyzeRecordedSpeech:
    {
        String file_name = ((PfdDataContainer)pfdhandle.getParameter
                             (SYS_PROFILE_PARAM_PARAM_AnalyzeRecordedSpeech_file_name)).stringValue();
        int language_type = ((PfdDataContainer)pfdhandle.getParameter
                             (SYS_PROFILE_PARAM_PARAM_AnalyzeRecordedSpeech_language_type)).intValue();
        int submitter_id = ((PfdDataContainer)pfdhandle.getParameter
                             (SYS_PROFILE_PARAM_PARAM_AnalyzeRecordedSpeech_submitter_id)).intValue();
        SpeechAnalysis.AnalyzeRecordedSpeech(file_name, language_type, submitter_id);
        break;
    }
    } 
} ...
MULTIPLE THREADS

THREAD SAFE QUEUES & DELIVERY

Delivery

UserInterface Thread

VoiceToText Thread

Main/Analysis Thread

dispatch
Generic Multithread Callback Sequence

1: new

2: nonlocal_dispatcher
   2.1: new
   2.2: setParameter
   2.3: deliver
      2.3.1: enqueueIncident

3: dequeueNextIncident

4: dispatch
   4.1: request

Finish by doing response, setting parameters, and delivering the response callback.

invoke request service nonlocally, create incident for the request service, and add the response incident as a parameter.
ANALYZE RECORDED SPEECH SEQUENCE DIAGRAM

Multithreaded Deployment
Deliver (nonlocal)

SpeechAnalysis.java

// UserInterface:RegisterForEvent( analyze_stored_svc, USER_EVENT_ANALYZE_STORED_AUDIO);

UserInterface.RegisterForEvent_nonlocalDispatcher(analyze_stored_svc,
SpeechAnalyzer_SystemModel.Sys.Sys.USER_EVENT_ANALYZE_STORED_AUDIO);
Deliver (nonlocal)
Pack Request Service and Payload

UserInterface.java

// Nonlocal handler ---------------------------------------------
public static void RegisterForEvent_nonlocalDispatcher (PfdIncident event_cb, int event_type) {
    PfdIncident handle = new PfdIncident(PfdIncident.SERVICE_HANDLE_INCIDENT, Sys.PROCESS_DEFAULT_LABEL, PfdTask.getTask(Sys.PROCESS_MAIN_TASK_ID_UI), Sys.DOMNUM_UserInterface, UserInterface.SVCNUM_RegisterForEvent, 2, null, null, 0);
    handle.setParameter(new PfdDataContainer(((PfdIncident)event_cb), 1));
    handle.setParameter(new PfdDataContainer(((int)event_type), 2));
    // send asynchronously and keep processing in this task
    handle.deliver();
}
public void deliver()
{
...
    if (PATH_MULTI_TASK)
    {
        PfdTask this_task_id = PfdTask.getCurrentTask();
    ...
    if (targetTask != this_task_id)
    {
        // send to another task
        targetTask.queueExternalIncident(this);
        return;
    }
}
Receive From Queue

PfdTask.java

// Copyright 1995 - 2017 Pathfinder Solutions LLC
// – Used with Permission

// Remove the next event from the queue and return it
public PfdIncident dequeueNextEvent ()
{
    return (this.eventQueue.dequeueNextIncident());
}
Dispatch Service Locally

UserInterface.java`,

```java
/**
 * Dispatch indirect service
 * @return FALSE if any errors encountered
 * @param pfdhandle IncidentHandle to be dispatched
 */
public static boolean dispatchService_(PfdIncident pfdhandle)
{
    ...
    switch (incident_index)
    {
    ...
    case UserInterface.SVCNUM_RegisterForEvent:
    {
        PfdIncident event_cb =
            (PfdIncident)((PfdDataContainer)pfdhandle.getParameter(1)).incHandleValue();
        int event_type = ((PfdDataContainer)pfdhandle.getParameter(2)).intValue();
        UserInterface.RegisterForEvent(event_cb, event_type);
        break;
    }
    ...
}
```

MULTIPLE PROCESSES

CREATE CALLBACK & DELIVER + SERIALIZATION & SOCKETS
DEPLOYMENT DIAGRAM - MULTIPLE PROCESSES

UserInterface Thread

Main/Analysis Thread

Analysis Process

VTT Process

VoiceToText Thread

ipc
MULTIPROCESS

ANALYZE RECORDED SPEECH
Generic Multiprocess Sequence

1: new

2: nonlocal_dispatcher
   2.1: new
   2.2: setParameter
   2.3: deliver
     2.3.1: insertToBuffer
     2.3.1: sendMessageToProcess

Deliver across process boundary, so serialize the incident

Deserialize the incident and put on target task queue

3: getMessage
   3.1: extractFromBuffer
   3.2: enqueueIncident

4: dequeueNextIncident
   4.1: dispatch
   4.1.1: request
Multiple Process Identifiers

Sys.java

/* define process ids for the system */

/** process id for process MAIN */
public final static int PROCESS_LABEL_MAIN = 0;
/** process id for process VTT */
public final static int PROCESS_LABEL_VTT = 1;
Construct Callback

SpeechAnalysis.java

// IncidentHandle analyze_stored_svc = CREATE IncidentHandle (file_name = "", language_type = LANGUAGE_TYPE_UNKNOWN, submitter_id = -1 ) TO SpeechAnalysis:AnalyzeRecordedSpeech;

analyze_stored_svc = new PfdIncident(
    PfdIncident.SERVICE_HANDLE_INCIDENT,
    Sys.PROCESS_LABEL_MAIN, PfdTask.getTask(Sys.PROCESS_MAIN_TASK_ID_SA),
    Sys.DOMNUM_SpeechAnalysis, SpeechAnalysis.SVCNUM_AnalyzeRecordedSpeech, 3, null, null, 0);
public void deliver()
{
    if (PATH_MULTI_TASK || PATH_MULTI_PROC)
    {
        if (PATH_MULTI_PROC)
        {
            if (this.processorId() != SW_PROCESS_DEFAULT_LABEL &&
                this.processorId() != PfdProcess.getProcessLabel())
            {
                // Non-local (interprocess) - send off and return
                PfdProcess.sendIncidentInterProcess(this);
                // Don't fall through into local processor processing below
                return;
            }
        }
    }
    ...
}
public void internalInsertToBuffer(SockMessageOut msg, boolean is_interprocess)
{
    int param_index;
    int provided_param_count = 0;   // Copyright 1995 - 2017 Pathfinder Solutions LLC
    // – Used with Permission

    msgBase.put_int_in_buffer(msg, this.type_);
    msgBase.put_int_in_buffer (msg, this.processorId());
    msgBase.put_int_in_buffer (msg, this.domainIndex());
    msgBase.put_int_in_buffer (msg, this.incidentIndex());

    ...   //put number of provided parameters in buffer
    msgBase.put_int_in_buffer (msg, this.parameterCount());

    // Now provide the parameters: one based index for parameters(0 reserved for ret value)
    for (param_index = 0; param_index < this.parameterCount_ + 1; param_index++)
    {
        PfdDataContainer param = getParameter(param_index);
        if (param != null)
        {
            msgBase.put_int_in_buffer (msg, param_index);
            param.insertToBuffer(msg);
        }
    }
}
Send / Receive *Packed* Callback

User Interface Thread

Main Thread

Analysis Process

ipc

recv

Voice To Text Thread

send

VTT Process
private PfdIncident internalExtractFromBuffer(SockMessageIn msg, boolean is_interprocess) {
    
    /* Get the processor id, task id, domain index, service index and number of parameters */

    /* Pull out the provided parameters */
    for (param_index = 0; param_index < provided_param_count.getInt(); param_index++)
    {
        /* Get parameter "name" index */
        if (msgBase.get_int_from_buffer(msg, param_name_index))
        {
            param = PfdDataContainer.extractFromBuffer(msg);
            retval.setParameter(param, param_name_index.getInt());
        }
    }
    
    return retval;
}
Enqueue Callback

PfdSocketRouter.java

```
if (type == SockMessage.sock_msg_type_t.SW_SOCK_MSG_INTERPROCESS_INCIDENT_HANDLE.getNum()) {
    PfdIncident incident = PfdIncident.EMPTY_SERVICE_HANDLE;
    incident = (PfdIncident)incident.extractFromBuffer (msg);
    if (incident != null) {
        // route incident to task
        incident.getTask().queueExternalIncident(incident);
    }
}
```
NANCY A. NUNES

hazel and jād technologies, inc
nancy.nunes.sw.architecture@gmail.com