

# Automated Testing of VoIP Infrastructure

*Lessons from the field*





## About me

- MSc. in Computer Science
- Bugs are interesting





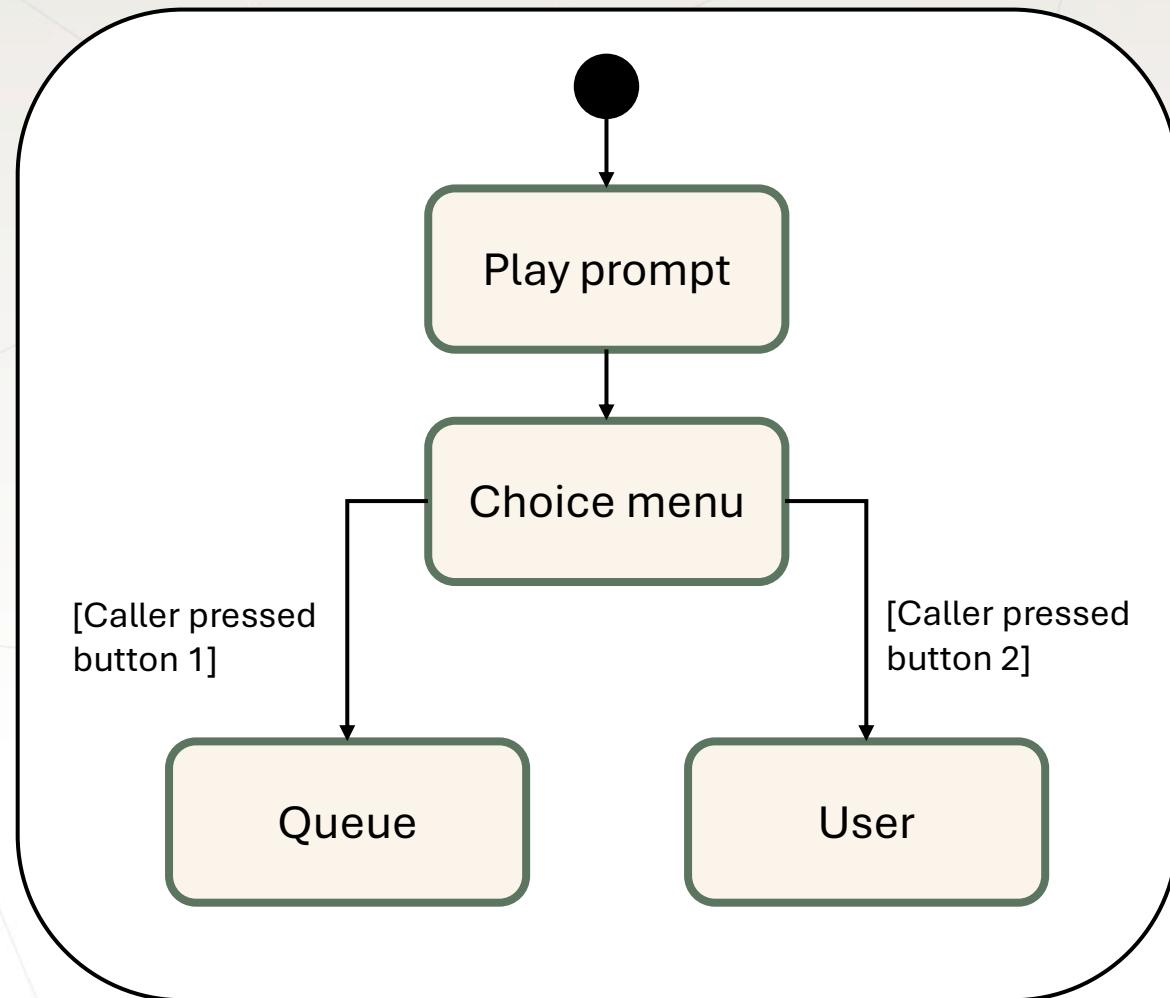
## About me

- Software tester at Vcare connect
  - (Tele-)communications for healthcare
  - Based in Enschede, the Netherlands
- We have our own VoIP infrastructure
- Software correctness is crucial



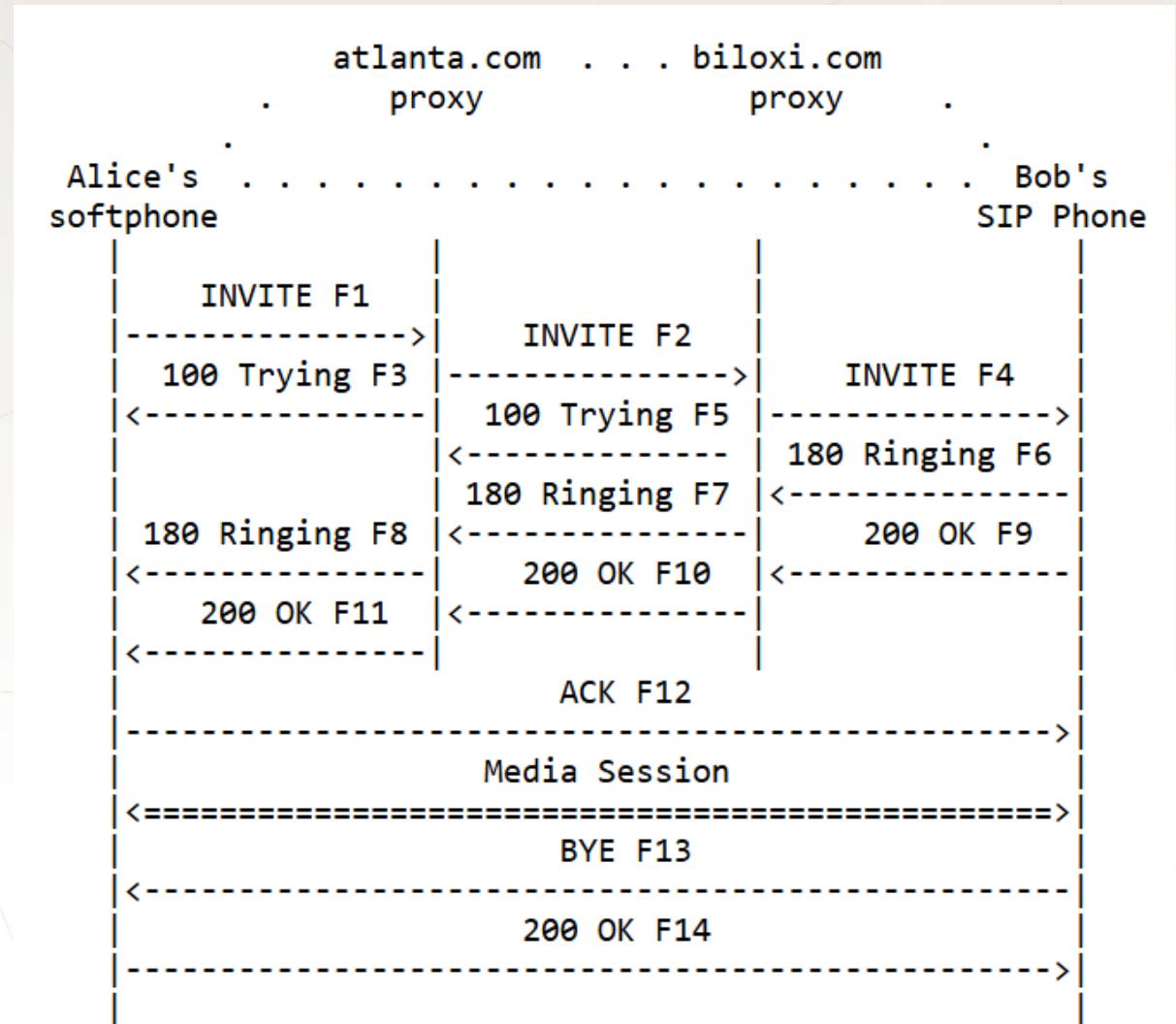
# Functionality

- Call routing
- Dial plans
  - State machine
  - Route call to user
  - Queues
  - Play prompts
  - Choice menus
- Conditional branching



# Session Initiation Protocol (SIP)

- Based on HTTP
- Messaging back-and-forth to initiate a call



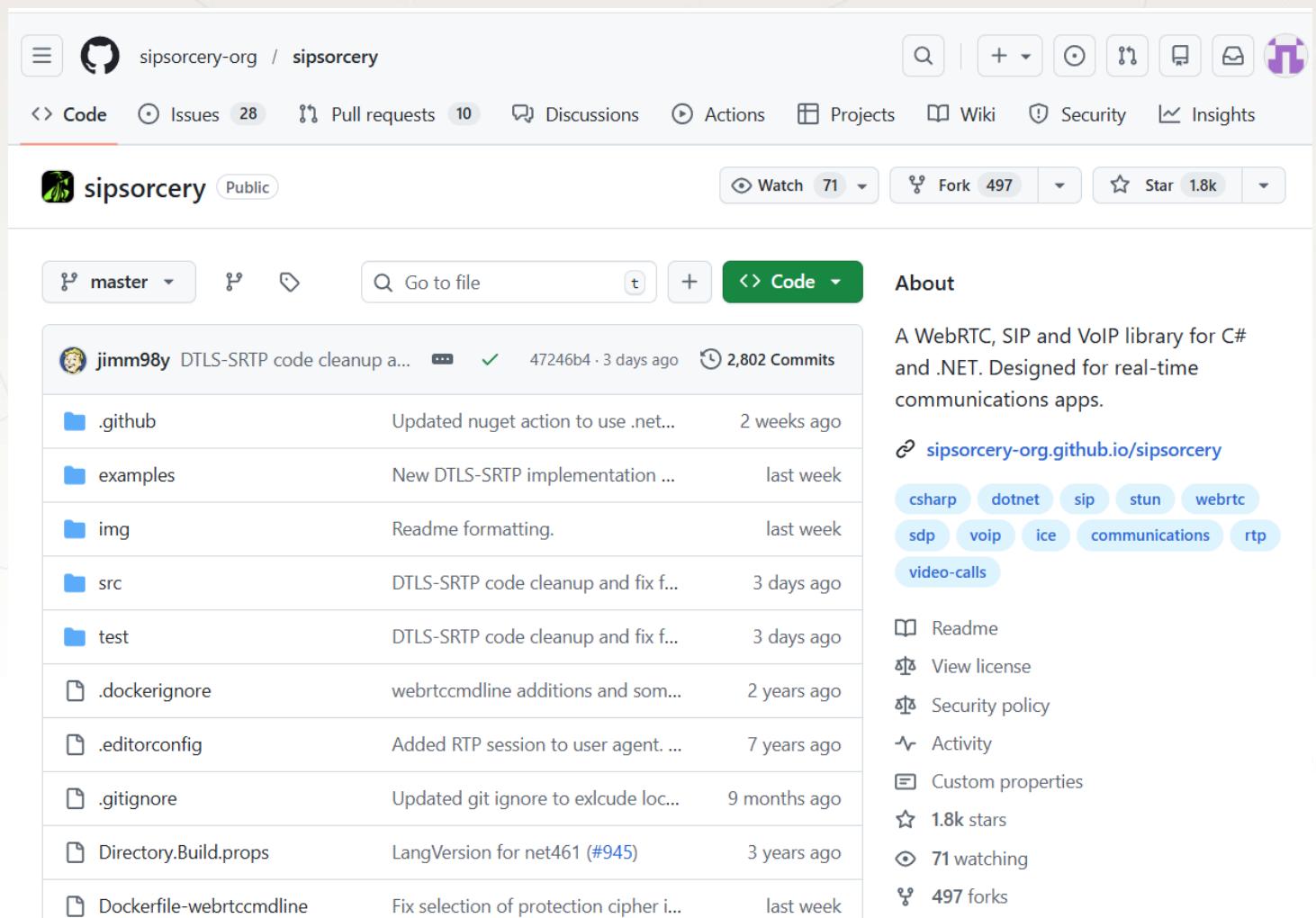


# Basic test setup

- Written in .NET
- ~1000 system / integration tests for VoIP calling
- Test pattern
  - Arrange
  - Act
  - Assert

# SIPSorcery

- Use SIPSorcery to make VoIP calls
- Written in .NET
- Open-source
- Customizable out-of-the-box
- Active development





```
// Create a SIP user agent and media session
var userAgent = new SIPUserAgent(new SIPTransport(), null);
var voipMediaSession = new VoIPMediaSession();

// Start a call to DESTINATION
var callResponse = await userAgent.Call(DESTINATION, null, null, voipMediaSession);

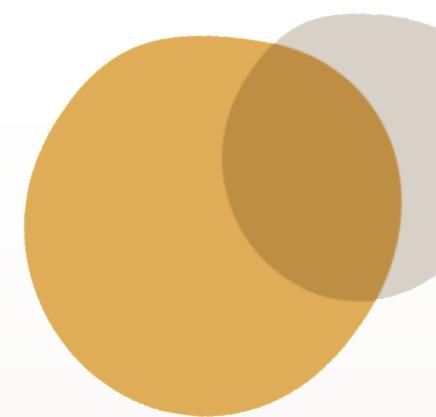
// Send DTMF tone '3'
await userAgent.SendDtmf(0x03);

// Hang up the call
userAgent.Hangup();
```



# Challenges

- Realistic phone behaviour
- Audio correctness
- Time-sensitive testing





# Realistic phone behaviour

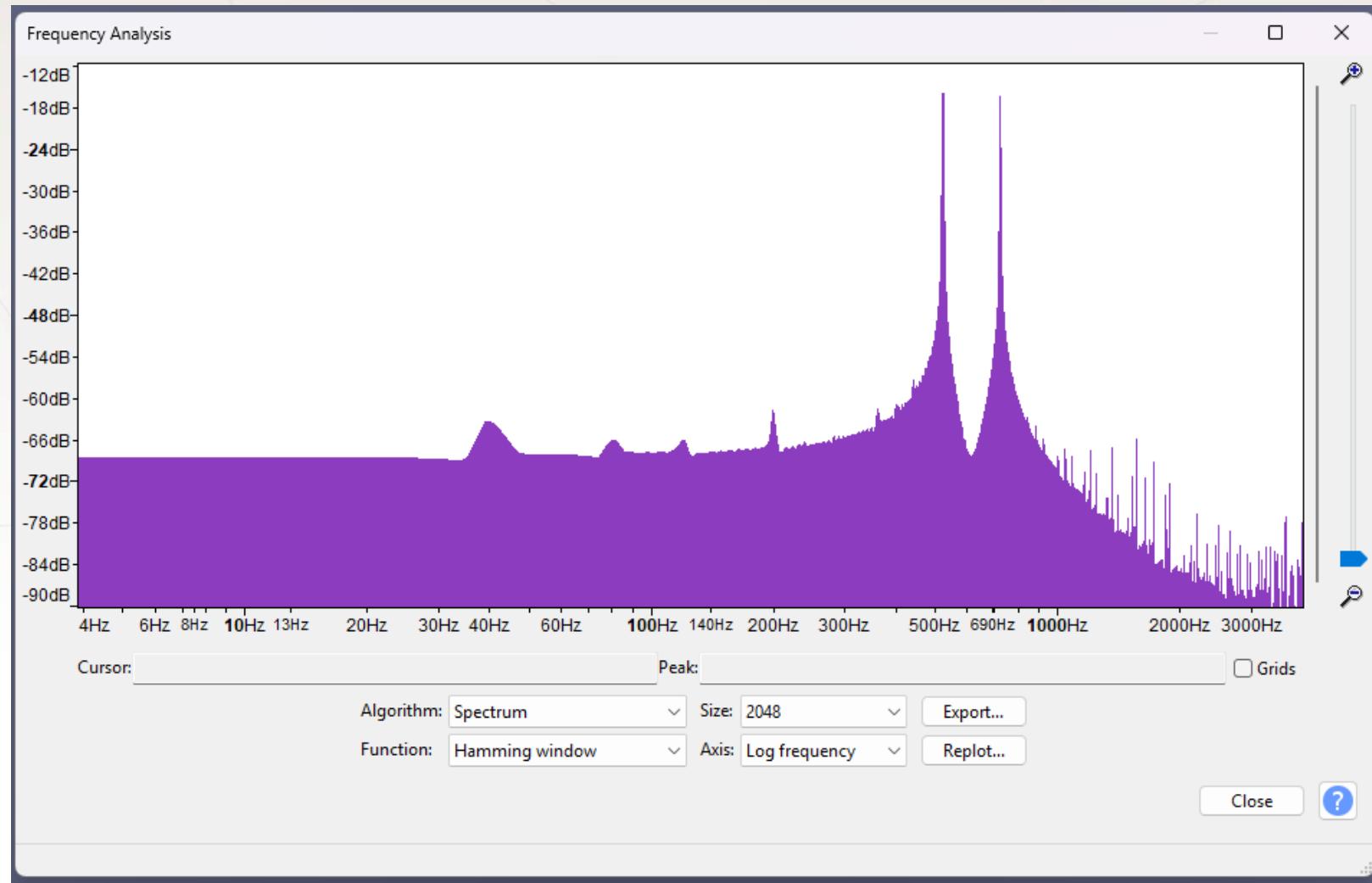
- Permanently plugged into the wall
  - Reuse same SIP Sorcery phones across tests
  - Wipe history and ignore call messages from previous test
- Custom header overrides
  - Possible as SIP Sorcery is customizable and open-source
- Unique bugs found due to differences
  - Sending DTMF too fast



## Audio correctness

- Each phone plays a tone in a different frequency
  - Built-in in SIPSorcery, but had to extent it slightly
- Store every audio sample SIPSorcery receives
- Asserting correctness
  - Frequency analysis
  - Fast Fourier Transform with Hamming window
  - Restrictions

## Challenges



Source: Screenshot from Audacity's frequency analysis tool (<https://github.com/audacity/audacity>)



# Time-sensitive testing

- Phone actions are not instant
- But you can subscribe to call events

```
// Subscribe to the incoming call event
userAgent.OnIncomingCall += IncomingCallHandler;
```

```
void IncomingCallHandler(SIPUserAgent userAgent, SIPRequest request)
{
    // Handle incoming call
}
```



# Time-sensitive testing

- Wait till a certain event happened
  - Timeout hit → test invalid
  - Event found → continue test
- Timeout lengths
- Timeout multiplier for slower target machines



# Summary

- Using SIPSorcery to test a VoIP backend
- Challenges
  - Realistic phone behaviour
  - Audio correctness
  - Time-sensitive testing

