# 思必驰全双工系统交互特性介绍

[1 全双工定义 2](#_Toc103910286)

[2 整体架构 2](#_Toc103910287)

[3 全双工特性介绍 4](#_Toc103910288)

[3.1 打断 4](#_Toc103910289)

[3.1.1 跳出打断 4](#_Toc103910290)

[3.1.2，打断继续 5](#_Toc103910291)

[3.1.3，场景内打断 5](#_Toc103910292)

[3.2，拒识 6](#_Toc103910293)

[3.2.1，语义对象拒识 6](#_Toc103910294)

[3.3，上下文理解 6](#_Toc103910295)

[3.3.1，意图继承 7](#_Toc103910296)

[3.4，槽位替换 7](#_Toc103910297)

[3.4.1，意图替换 7](#_Toc103910298)

[3.4.2，槽位替换 8](#_Toc103910299)

[3.5，指代消歧 9](#_Toc103910300)

[3.5.1，技能内或技能之间指代消歧 9](#_Toc103910301)

[3.6，全双工语音 9](#_Toc103910302)

[3.6.1，延迟倾听 9](#_Toc103910303)

[3.7，单句多意图 10](#_Toc103910304)

[3.7.1，单句多意图 10](#_Toc103910305)

[3.8，交互 11](#_Toc103910306)

[3.8.1，语音交互 11](#_Toc103910307)

[3.9，人工纠错 11](#_Toc103910308)

[3.9.1，对话纠错 11](#_Toc103910309)

# 1 全双工定义

全双工，它的通信术语定义的就是一个实时的、双向的语音信息的交互，就叫全双工。而我们所熟悉的全双工的最重要的一个应用就是电话，我们以前去打各种各样的电话，双方都可以同时说话，这个就叫全双工。而与之对应的所谓半双工，其实最典型的传统设备就是一个步话机，步话机摁下来的时候就只能一边在说，说完了之后说over，然后对方再把他的步话机的按纽按下去进行对话。这是和全双工对应的半双工。

过去，在人机交互的设计中有一个假设前提：让用户知道是在与机器对话。基于这个前提，产品经理定义产品时，通常设计苛刻的交互条件或者话术。如，频繁使用唤醒词来明确是与机器交互。而用户对人工智能的预期较高，交互方式具有多样性，更加接近人与人的交互方式。为了在有限的条件下，尽可能贴近人与人的交互，研究人员开发了各式交互技术。如，oneshot、唤醒打断等。而使用多种技术，在产品落地中又存在各项技术的相互融合、参数调优等费时、费力的问题，导致落地周期长。在此技术背景上，研发人员从工程的角度，提出了全双工交互的概念。全双工交互只需要在开始交互时唤醒开启。其与半双工单轮交互、多轮交互等不同交互模式的对比，如图1-1所示。

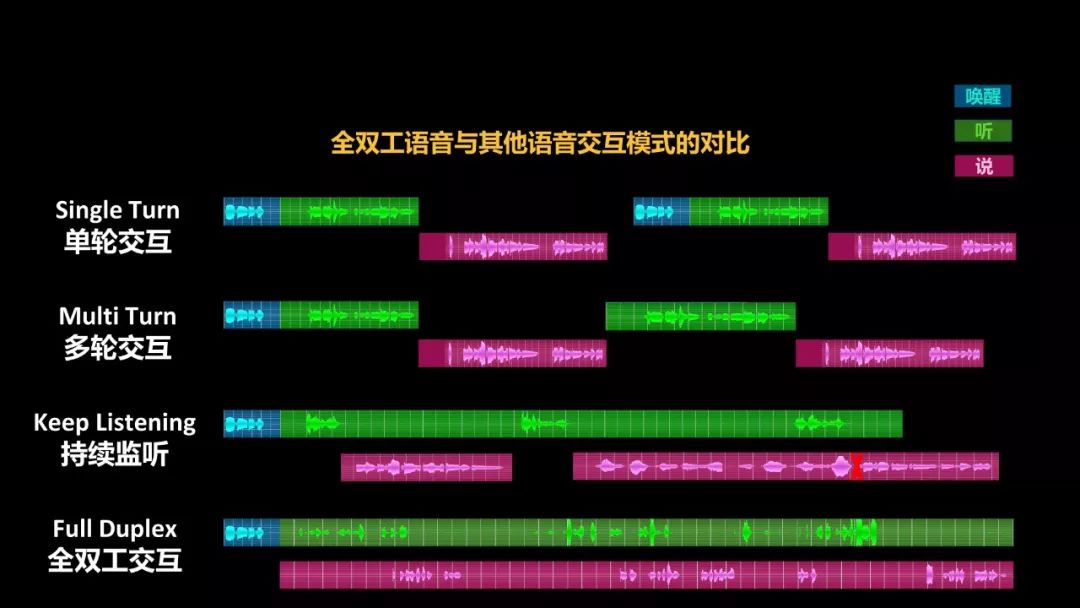


图1-1 全双工与其他语音交互模式对比

# 2 整体架构

当前全双工语音交互的整体系统架构图，如图2-1所示。

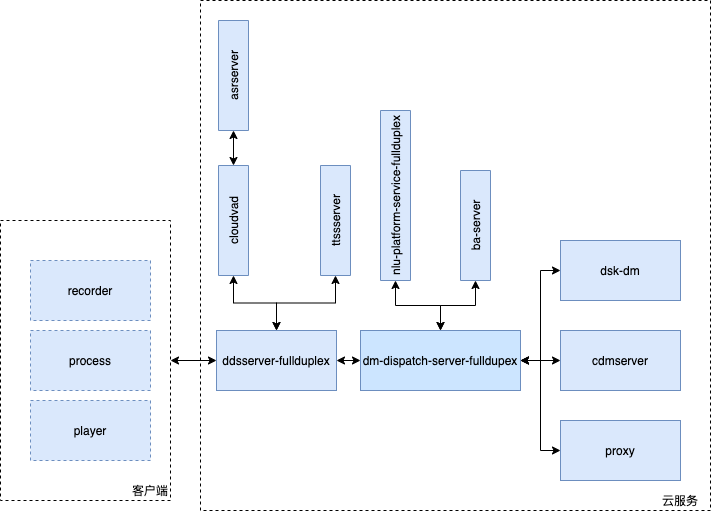


图2-1 全双工语音交互

客户端部分：

* recorder：手机录音模块
* process：处理模块
* player：播放器模块

云服务：

* ddsserver-fullduplex：接入服务
* 收到语音音频后，调用识别服务得到识别结果（包含实时识别结果+最终识别结果）
* 调用中控dm得到对话结果（包含命中的技能信息，对话信息等）
* cloudvad：云端vad服务，切分有效音频
* asrserver：识别服务
* dm-dispatch-server-fullduplex:中控dm
* 接收ddsserver传入的识别结果（包含实时识别结果+最终识别结果）
* 调度语义平台(nlu-platform-service-fullduplex)与知识型平台(baserver)
* 得到任务型语义nbest结果+知识型技能结果后，选出最优技能
* 选出最优技能后，根据技能类型调度技能dm(包含dsk-dm/cdmserver/proxy)
* nlu-platform-service-fullduplex:语义平台
* 完成技能语义分类解析，给出语义nbest结果
* baserver：知识型技能服务（包含）
* ttsserver：在线合成服务
* dsk-dm：通用技能/思必驰高级技能/流程型技能等
* cdmserver：任务型技能
* proxy：外部开发的自定义技能

# 3 全双工特性介绍

## 3.1 打断

### 3.1.1 跳出打断

功能描述:

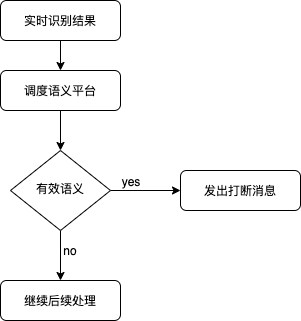
在全场景下,使用通用的有意义的指令打断当前播报

方案:

1，dm-dispatch-server-fullduplex收到实时识别结果后，调度语义平台（nlu-platform-service-fullduplex），如果可以分类解析成功，此时dm-dispatch-server-fullduplex给出打断信号。

2，客户端收到打断信号后，打断当前合成音播放（备注，打断信号同样可以是有效对话回复）

流程图：



### 3.1.2，打断继续

功能描述：

在跳出打断后，完成当前的任务，继续回到前一个任务

方案：

* 打断继续功能是产品高级配置开启，可以指定产品，指定领域开启，建议只在指定领域开启打断继续
* 对于开启打断继续的产品，次轮请求语义平台时，携带之前未结束上下文技能的技能信息给到语义平台，语义平台给出之前未结束上下文技能的语义解析结果（包含弱说法）+ 其他技能语义结果（强说法召回）的语义nbest
* 对话中控选出最优技能（优先上一个未结束上下文的技能）
* 选出最优技能后，调度技能dm得到对话结果，对于技能dm内，所有未结束上下文技能的上下文都不清除

### 3.1.3，场景内打断

功能描述：

在特定场景下，在TTS播报结束前，用户提前说该场景下的某些指令，可立即打断TTS播报，并进入对应领域。

方案：

同3.1.1方案

## 3.2，拒识

### 3.2.1，语义对象拒识

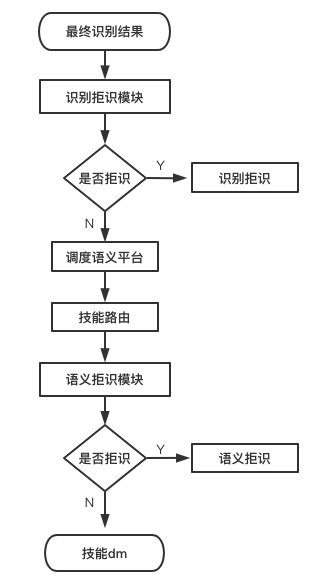
功能描述：

识别用户是对VPA或对其他人说话，识别无关语句和意图，不做出干扰反馈。

方案：

* 识别拒识，给到对话中控dm-dispatch-server-fullduplex时，会包含识别相关信息，经过识别拒识模块，可以拒识背景噪声。
* 语义拒识，中控dm-dispatch-server-fullduplex完成技能路由，选出最优技能后，经过语义拒识模块，可以拒识跟产品无关的说法。

流程图：



## 3.3，上下文理解

### 3.3.1，意图继承

功能描述：

在单人对话中，结合上文槽位补充下文槽位信息，识别意图。

场景示例：

U:播放刘德华的冰雨

A: 已为您找到刘德华的冰雨…

U:放个三生三世

A:已为您找到三生三世十里桃花的主题曲（避免进入视频领域）

U:我要看成长的烦恼

U:第四季

U:车窗关闭—>关一个车窗

U:全部关闭—>补充上下文为车窗

方案：

* 当前对话中控dm-dispatch-server-fullduplex做技能路由时，参考产品中技能优先级、语义nbest置信度/知识型技能置信度、跨技能上下文（上一轮命中的技能信息）等多个维度选出最优技能，上述场景涉及到技能路由优先上一轮技能的维度
* 技能dm(cdmserver/dsk-dm/proxy)内对于多轮的技能具备 不同语义槽累加的功能。

## 3.4，槽位替换

### 3.4.1，意图替换

场景示例：

U:放个三生三世

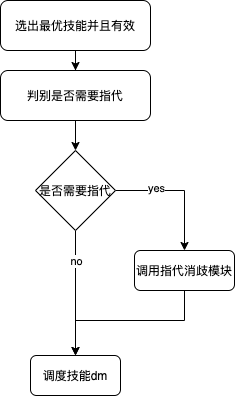
A:已为您找到三生三世十里桃花的主题曲

U:我说的是放电视剧

方案：

* 以上需要对于次轮说法整句标注 指代功能
* 对话中控dm-dispatch-server-fullduplex拿到指代标签后，调用指代消歧模块，完成跨技能指代消歧，将之前的语义槽流转到当前轮语义解析中。

流程图：



### 3.4.2，槽位替换

场景示例：

U:换成去{南京}的吧

方案：

* 技能dm(cdmserver/dsk-dm/proxy)内，状态跟踪模块，同一个技能（领域）多轮交互时，相同语义槽相互覆盖

流程图：



* norm：归一化,完成部分匹配等功能，sys.电话选择只匹配当前页，sys.用户选择匹配所有页
* state：状态跟踪,根据当前语义解析的结果及上一轮系统的回复对对话状态进行更新，主要包括两方面的更新
  + 用户输入
  + 查询结果
* policy： 对话决策,主要工作是根据用户的输入做出合理的决策
* nlg ：根据用户定制的输入,生成自然语言反馈给用户

## 3.5，指代消歧

### 3.5.1，技能内或技能之间指代消歧

功能描述：

多轮对话交互中，结合上下文识别指代词的意义，形成更自然的语音交互流程。

场景示例：

U:我要听王菲的歌

U:再放首她的歌

方案：

* 次轮说指代说法时，语义会出指代标签，同时会告知指代消歧的目标语义槽
* 对话中控dm-dispatch-server-fullduplex拿到指代标签后，调用指代消歧模块，完成跨技能指代消歧，将之前的语义槽流转到当前轮语义解析中

流程图：

同3.4.1流程图

## 3.6，全双工语音

### 3.6.1，延迟倾听

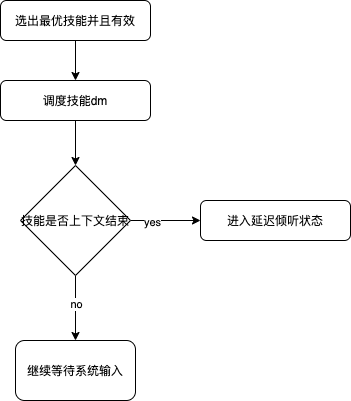
功能介绍：

全双工交互中，某个技能（领域）上下文结束时，进入延迟倾听状态，此时设备可以在一定时间范围内继续交互，备注：延迟倾听时长可以产品配置

方案：

* 全双工交互中，某个技能（领域）上下文结束时，对话中控dm-dispatch-server-fullduplex将是否继续监听的标志设成false，这样客户端继续收音，继续交互，延迟倾听时长内无有效交互，此时对话中控推送 会话超时提醒给到客户端，客户端结束交互。

流程图：



## 3.7，单句多意图

### 3.7.1，单句多意图

功能介绍：

识别一句话中多个意图的语义，并做出相应反馈。

方案：

语义出一句话多意图解析后，如果是跨域的一句话多意图，对话中控dm-dispatch-server-fullduplex依次发起多个技能dm调用，得到多个对话结果，一次性返回给调用端

## 3.8，交互

### 3.8.1，语音交互

功能介绍：

支持用户二次确认

方案：

技能dm内多轮逻辑

流程图：

参考3.4.2流程图

## 3.9，人工纠错

### 3.9.1，对话纠错

语音识别是任务型对话系统交互任务唤醒后的首要环节，也是后续模块处理的基础。然而在语音交互过程中，说话人的口音、环境噪音，或者语句内容本身的同音多词等因素，会造成语音识别错误。语音对话系统中，识别错误会逐级传递放大，并累积至后续链路，最终造成对话任务的失败。纠错系统旨在对错误的语音识别结果进行鲁棒理解和修正，给业务后续搜索提供更准确的输入，以及改善识别展示结果。

场景示例：

U:拨打123456789

U:不是6789，是9876

方案：

* 启动对话纠错后，如果遇到上述pattern，会替换上一轮识别文本，替换成 "拨打123459876" 重新送语义平台/知识型平台，经过技能路由后，后再送技能dm