음성대화시스템 워크벤취로서의 DialogStudio 개발

정 상 근, 이 청 재, 이 근 배
포항공과대학교 컴퓨터공학과

DialogStudio: A Spoken Dialog System Workbench

Sangkeun Jung, Cheonjae Lee, Gary Geunbae Lee
Department of Computer Science And Enginerring, POSTECH
E-mail: {hugman, lcj80, gblee}@postech.ac.kr

Abstract

Spoken dialog system development includes many laborious and inefficient tasks. Since there are many components such as speech recognizer, language understanding, dialog management and knowledge management in a spoken dialog system, a developer should take an effort to edit corpus and train each model separately. To reduce a cost for editting corpus and training each models, we need more systematic and efficent working environment. For the working environment, we propose DialogStudio as an spoken dialog system workbench.

I. 서론

음성대화시스템은 기술적, 구조적 측면에서 다양한 구성요소들로 이루어져 있다. 음성인식, 언어이해, 대화관리, 지식관리등이 필수적인 구성요소로서 작동을 하게 되며, 각각의 요소들에 대해서 말뭉치의 준비 및 훈련등이 이루어져야 한다. 이러한 작업들에는 시간적, 인적 자원이 많이 소요되는 작업들로 이루어지기 마련이다. 이러한 자원소모적인 작업들을 줄이고 보다 효율적으로 시스템을 개발하기 위한 방법들이 제안되었다. 대화구조 및 언어이해구조의 빠른 설계 및 구축을 위해 CSLU Toolkit[1] 과 SGStudio [2] 이 개발 되었으며, 비전문가가 음성대화시스템을 간단히 구축할 수 있도록 돕는 SUEDE [3] 도 개발되었다.

하지만 위의 방법론들은 대화시스템의 디자인과 개발측면에 초점을 두고 있으며 유지보수 및 음성대화시스템 개발의 모든 작업과정의 효율화에는 부족한 점이 많다. 또한 각 시스템에서 사용하는 음성인식, 언어이해, 대화관리 등이 각 시스템에 의존적이며 제한적이기 때문에 다른 음성대화시스템 개발에 사용하기 어려운 측면이 있다.

본 연구에서는 음성대화시스템의 개발과정에서의 비효율적인 측면들을 해결하고 대부분의 음성대화시스템에서 사용할 수 있는 워크벤취 방법론을 제안한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 음성대화시스템에서의 비효율적인 측면들을 분석하고 3장에서는 음성대화시스템의 워크벤취가 갖춰야 할 필요기능들을 제안하고, 4 장에서는 본 연구에서 제안하는 워크벤취시스템의 구조 및 기능들에 대해 설명하며, 5장에서는 실험을 통해 DialogStudio 의 효율성을 검증하고 6장에서는 본연구의 결론을 내렸다.

II. 음성대화시스템 개발 및 유지과정에서의 비효율 요인

음성대화시스템은 내부적으로 음성인식, 언어이해, 대화관리, 지식관리등의 많은 구성요소로 이루어져 있으며 각자 구성요소들이 유기적으로 연결되어 작동되는 시스템이다. 따라서 시스템의 개발 및 유지의 과정에서 각 구성요소에 대해 각각의 말뭉치 준비 및 훈련의 노력이 필요하게 되며, 구성요소의 숫자에 비례하여 노력이 필요하게 된다.

구성요소의 개수만큼의 각 말뭉치 편집과정, 훈,련기 및 테스터등이 나뉘어져 있을 경우 대화시스템의 유지보수단계에서 대화시스템에 수정을 가하기 위해서는 각 모듈에 대해 훈련 및 테스트를 거쳐야 하는데 이는 매우 비효율적이다. 말뭉치의 편집과정 역시 주어진 문장에 대해 태그를 달거나, 정해진 말뭉치 형태로 재구성 하는 작업인데 이런 편집과정은 상당한 시간적 인적 노력이 필요하게 되며, 각 말뭉치 사이의 유기적인 연관관계를 유지하는 것에도 별도의 노력이 들어가게 된다.

따라서 이러한 대화시스템의 개발 및 유지보수 단계

를 유기적이면서 체계적으로 구성하여 기존의 비효율적인 작업 방식들을 효율적으로 재구성하고, 말뭉치 작업을 손쉬게 하도록 도와주며 각 구성요소의 모델 훈련 및 테스터를 손쉽게 도와주는 워크벤취가 필요하다.

III. 음성대화시스템 워크벤취로서의 필요 기능

음성대화시스템의 초기 개발부터 유지보수 단계까지 시스템 개발에 사용할 수 워크벤취에는 다음과 같은 기능들이 필요하다.

3.1 순차적이고 체계적인 작업 순서 구축

대화시스템에 수정을 가하기 위해서는 대단히 많은량의 작업이 필요하다. 각각의 구성요소에 대해서 말뭉치의 편집, 훈련이 이루어져야 하며 각 구성요소에 대한 평가 및 전체 대화시스템에 대한 평가역시 이루어져야 한다. 이렇게 많은 양의 작업을 일련의 체계적인 단계로 나누어서 관리하고, 자동화할 수 있는 부분은 최대한 자동화 함으로서 대화시스템 전체의 개발을 효율적으로 도와줄 수 있는 워크벤취 형태가 되어야 한다.

3.2 손쉽고 효율적인 말뭉치 편집환경

대화시스템 개발에 있어서, 특히 통계적 기법에 기반한 음성인식, 언어이해, 대화관리기법을 사용하는경우라면 말뭉치의 준비 및 편집은 시스템 개발의 개발기간, 품질에 많은 영향을 미친다. 따라서 워크벤취는 개발자에게 몇 번의 마우스 동작만으로도 쉽게 말뭉치에 수정 및 편집을 가할 수 있고 태그를 달 수 있도록 지원해 줄 수 있어야 한다.

3.3 구성요소간의 언어 동기화

음성대화시스템은 필수적으로 음성인식, 언어이해, 대화관리, 지식관리등의 구성요소를 가지게 되며 각 구성요소가 유기적으로 연결되어 동작되어야 한다. 따라서 각 구성요소가 처리할 수 있는 언어가 일치해야 성공적인 대화의 진행을 기대할 수 있다. 예를 들어 EPG 시스템의 경우 매일 매일 새로운 지식으로 바뀌게 되고 이러한 지식 즉 새로운 언어의 추가는 모든 구성요소에 영향을 미치게 된다. 따라서 개발자가 새로운 문장 패턴이나, 지식을 추가하는등의 새로운 언어를 시스템에 학습시키고자 할 때 시스템의 모든 구성요소가 이러한 새로운 언어에 적응할 수 있도록 워크벤취 시스템이 언어동기화를 이루어 주어야 한다.

3.4 구성요소들의 훈련 및 평가 편의성 도모

대화시스템의 각 구성요소들에 대한 훈련 기 및 평가모듈을 하나의 워크벤취 안에 둠으로서 개발자가 말뭉치 편집 및 훈련을 통해 대화시스템에 수정을 쉽게 가하고, 바로 테스트 해볼 수 있도록 도와주어야 한다

3.5 도메인 및 방법론 독립적인 워크벤취

대화시스템 마다 구현하고 있는 도메인은 모두 다르며 각 도메인마다 사용하는 태그나 언어 및 지식구조의 이름 역시 모두 다르다. 또한 음성인식, 언어이해, 대화관리 및 지식관리 등에 사용하는 방법론은 수없이 많으며 각 방법론에 사용되는 말뭉치의 형태 역시 다르다.

음성대화시스템의 워크벤취로서 효율성을 가질 수 있으려면, 도메인 및 구조에 민감하지 않도록 워크벤취 툴이 구성될 수 있어야 하며, 각 방법론에 의존적이지 않도록 워크벤취 디자인이 이루어져야 한다.

IV. DialogStudio의 구조

본 연구에서는 3장에서 언급한 필요기능들을 만족하는 음성대화시스템을 위한 워크벤취로서 DialogStudio를 개발하였다. DialogStudio 는 대화시스템의 초기 디자인부터 개발, 유지보수 단계까지 사용할 수 있도록 각 작업들을 성격적, 시간적으로 구분하여 설계되었다.

성격적으로는 대화시스템이 다루는 언어의 성격에 따라 구분하였다. 즉, 발화의 의미를 다루는 의미구조와 사용자 발화 및 시스템 발화 사이의 관계를 다루는 대화구조, 그리고 발화에 사용되는 도메인 지식에 대한 지식구조로 나누어서 개발자가 대화시스템을 설계하도록 하였다.

시간적으로는 대화시스템 전체의 구조를 설계하는 Desgin Step, 말뭉치의 편집을 통해 시스템의 수정을 가하는 Annotation Step, 각 구성요소들 간의 언어동기화를 이루어주는 Synchronization Step, 각 구성요소의 훈련을 통해 모델을 생성하는 Training Step, 각 구성요소의 개별 테스트 및 전체 시스템의 구동을 해보는 Running Step 으로 나뉘어져 설계되었다.

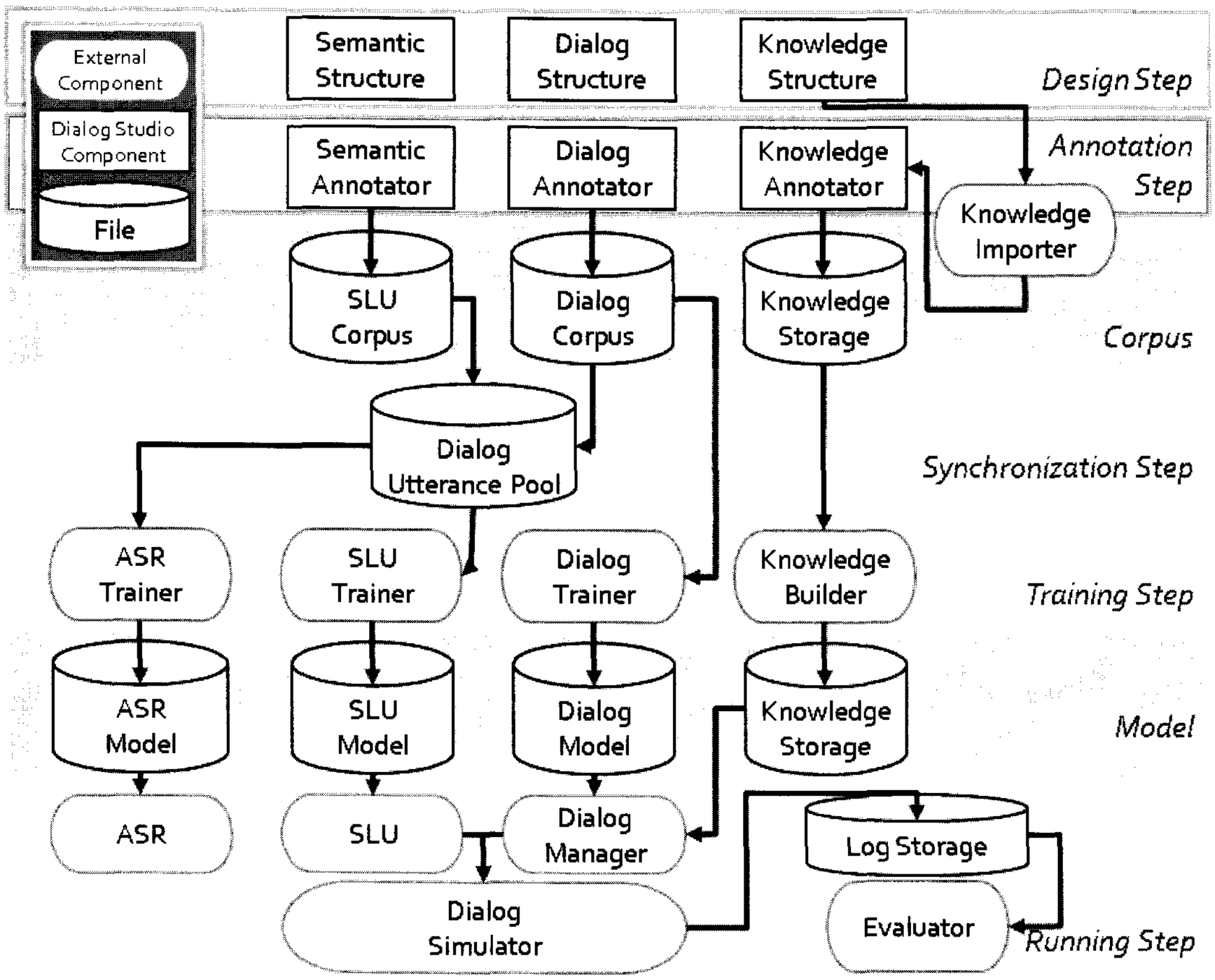
DialogStudio 의 전체 구조는 그림 1과 같다.

4.1 Design Step

DialogStudo 에서는 DialogStudio Design Language(DSL) 라는 간단한 언어형태를 제공한다. 개발자는 DSL을 통해서 DialogStudio 의 전체적인 GUI 환경 및 작업방식을 통제할 수 있도록 구현되었다.

언어구조, 대화구조 및 지식구조를 DSL에 선언을 하고 각 구조가 말뭉치 편집 GUI에 어떤식으로 표현되는지 선언을 함으로서 각 도메인이나 시스템에 따라 각각 다른형태의 말뭉치들이 DialogStudio 안에 표현이 되고 편집이 가능하도록 하였다. 또한 각 구성요소들의 통제 및 인자들을 DSL 에 추가하여 각 모듈에 전달할 수 있도록 하였다.

그림 1 DialogStudio 의 전체구조



4.2 Annotation Step

말뭉치의 편집의 편의성과 GUI 디자인은 매우 밀접한 관계를 가진다. 예를 들어 Tree 형태의 말뭉치 구조라면, Tree 형태의 GUI를 그대로 구현할 때 매우 강력한 편집환경이 될 것이다. 그러나 말뭉치 구조는 개발자, 도메인, 시스템에 따라 모두 다르기 때문에 GUI Design 을 임의의 모든 말뭉치 구조에 맞추는 것은 불가능하다. 따라서 말뭉치의 구조와 GUI 환경사이의 절충점을 찾아야하는데, DialogStudio 에서는 테이블형태의 Grid GUI 환경을 통해서 그 절충점을 찾도록 하였다. GUI 의 모습은 Relation DB 혹은 MS Excel 의 편집환경과 유사한 환경으로 구축되며 테이블의 헤더 혹은 attribute들은 DSL의 언어구조, 대화구조 및 지식구조등을 통해서 선언된 이름들이다. 또한 사용자의 말뭉치 편집의 편의성을 도모하기 위해서 컨텍스트 메뉴를 통한 간단한 단어 태깅, 찾기 바꾸기 등을 제공한다.

4.3 Synchronization Step

새로운 지식 및 문장패턴 혹은 대화패턴이 추가됨에 따라서 각 구성요소들 모두가 이러한 새로운 언어에 적응이 될 수 있도록 언어동기화 단계를 수행하도록 하였다. DUP(Dialog Utterance Pool)이라는 중간단계의 동기화 결과물을 말하는데, 개발자가 기존의 말뭉치와 새로운 지식의 그리고 새로운 문장패턴등을 입력으로 주게되면, 같은 의미를 가지는 비슷한 형태의 문장을 통계적으로 만들어 내고 새로운 지식을 적응시키켜 DUP를 만들어 낸다. DUP 의 형태는 언어이해 모듈의 말뭉치 형식을 따르도록 하였으며 새롭게 만들어진 DUP 를 음성인식의 발음사전 및 언어모델생성 그리고 언어이해모듈의 훈련의 입력물로 사용되게 된다.

표 1 Baseline 및 DialogStudio의 개발환경 효율성 비교

	Baseline		DialogStudio	
	Keyboard	Mouse	Keyboard	Mouse
훈련과정	143	0	0	5
언어이해 말뭉치 편집	43.3	2.6	0	8.6

4.4 Training Step

말뭉치 편집 및 언어동기화의 결과물을 이용해서 음성인식의 발음사전, 언어모델 생성, 언어이해 모델 생

성, 대화관리 모델 생성, 지식 DB 생성등의 훈련기들을 모두 한곳에 모아서 개발자가 마우스 클릭 몇 번으로 쉽게 모든 모든 모듈의 훈련이 가능하도록 하였다.

도메인 및 방법론에 비의존적인 워크벤취 시스템을 만들기 위해서 DialogStudio 에서는 도메인 및 방법론에 의존적인 모듈들은 시스템 외부에 위치하도록 설계하였다. 특히 Training Step에서 사용되는 각 훈련모듈은 항상 도메인, 방법론 의존적이기 때문에 DialogStudio 외부에 위치하도록 하였으며 각 훈련기와는 통신을 통해서 데이터를 주고 받도록 설계하였다. 즉, 각 훈련기를 개발하는 개발자가 기존의 훈련시스템에 DialogStudio에서 제공하는 통신 Protocol 에 맞추어서 동작하도록 수정을 가해주면 기존의 훈련기를 DialogStudio 의 GUI의 환경에서 실행시키고 그 결과물을 다시 DialogStudio에서 확인할 수 있도록 설계하였다.

4.5 Running Step

Running Step 에서는 각 구성요소의 개별적인 실행 및 대화시스템 전체의 실행을 해 볼 수 있도록 설계되었다. 또한 대화시뮬레이션 시스템도 이곳에 위치할 수 있도록 하였다. 각각의 실행모듈 및 대화시뮬레이션 또한 훈련기와 마찬가지로 외부모듈로 두어서 DialogStudo 와는 통신을 통해서 실행이 가능하도록 구현되었다.

V. DialogStudio의 효율성 평가

워크벤취 시스템으로서 성능의 기준은 기존의 대화시스템 개발방식에 비해 워크벤취를 사용하였을 때 소요되는 노력이 얼마나 줄었는가로 살펴 볼 수 있을 것이다.

본 연구에서는 DialogStudio 와의 비교대상으로서의 Baseline은 어떠한 GUI 및 자동편집 기능을 사용하지 않은 상태에서의 대화시스템 개발환경과 비교를 하였다. 본 실험에 사용된 대화시스템의 도메인은 EPG 시스템이며, 통계적 언어이해, 예제기반의 대화관리, Relation DB 형태의 지식관리기법을 사용하였다.

Baseline 에서의 작업 효율성과 DialogStudio 에서의 작업효율성 비교를 위해 키보드 타이핑 횟수와 마우스 클릭 횟수를 비교하였다. 평가 측면으로서 언어이해의 말뭉치에 대해서 한 발화에 대한 화행, 주행 및 슬랏의 태깅을 하는데 있어서 몇 번의 키보드 타이핑과 마우스 클릭이 이루어졌는지 100문장에 대해서 평균적으로 산출을 하였다. 또한 발음사전 훈련기, 언어모델 훈련기, 언어이해 훈련기, 대화모델 훈련기 및 지식구축기를 모두 구동시키는데 baseline 시스템과 DialogStudio 에서의 키보드 타이핑과 마우스 클릭횟수를 비교하였다. 표 1 에서 보다시피, 키보드 타이핑 횟수 및 마우스 클릭 횟수측면에서 DialogStudio를 사용하는 것이 훨씬 적은 수로 나타났다. 이는 DialogStudio 를 사용하는 것이 기존의 워크벤취를 사용하지 않은 개발환경보다 훨씬 효율적이고 빠르게 개발할 수 있다는 것을 의미한다.

VI. 결 론

본 연구에서는 음성대화시스템의 개발 및 유지보수 측면에서의 비효율적인 측면을 다방면으로 분석하고, 이러한 작업들을 효율적으로 만들기 위한 워크벤취 시스템의 필요기능들을 제안하였으며, 각 필요기능을 구현 한 음성대화시스템의 워크벤취로서 DialogStudio를 개발하였다. DialogStudio 는 다루는 언어의 성격적으로 의미구조, 대화구조 및 지식구조로 구분하여 DialogStudio Design Language 를 통해 DialogStudio의 GUI 및 작동방식에 적용되도록 하였으며, 시간적인 측면으로 Design Step, Annotation Step, Synchronization Step, Training Step 및 Running Step 으로 나누어 작업을 순차적이고 유기적으로 배치함으로서 대화관리시스템의 개발 및 유지가 체계적으로 이루어지도록 작업을 재배치 및 자동화 하였다. 워크벤취를 사용하지 않은 시스템을 Baseline 으로 삼아 DialogStudio 와의 말뭉치 편집 및 훈련과정의 효율성을 비교해본 결과 DialogStudio 를 사용하는 것이 훨씬 효율적인 개발환경을 구축할 수 있다는 것을 확인할 수 있었다.

VII. 감사의 글

본 논문은 정보통신부 사업인 "신성장동력산업용 대용량/대화형 분산/내장처리 음성인터페이스 기술 개발" 과제에 의해 지원되었습니다.

참고문헌

[1] CSLU Toolkit. http://cslu.cse.ogi.edu/toolkit/

[2] Wang, Y and Alex Acero. SGStudio: Rapid Semantic Grammar Development for Spoken Language Understanding. Proceedings of the Eurospeech Conference. Lisbon, Portugal. September, 2005.

[3] Anoop, K. S., Scott R. K., Chen., J., Landay, J. A., Chen, C., "SUEDE: Iterative, Informal Prototyping for Speech Interfaces." Video poster in Extended Abstracts of Human Factors in Computing Systems: CHI 2001, Seattle, WA, March 31-April 5, 2001, pp. 203-204.