Developing without Developers

Software Development Methods for LD Apps

Luke D. Gessler
lgessler.com

Department of Linguistics
Georgetown University

ComputEL-3
February 26, 2019

Outline

Introduction

- Central claim: language documentation (LD) apps will succeed when their developers use software development methods that are adapted to the economic conditions of language documentation

Introduction

- Central claim: language documentation (LD) apps will succeed when their developers use software development methods that are adapted to the economic conditions of language documentation
- Debatable assumptions:
  - Documentation is worthwhile even without revitalization efforts
  - Software can help fieldworkers be more productive
  - (Worth discussing, but out of scope)

Introduction

- Central claim: language documentation (LD) apps will succeed when their developers use software development methods that are adapted to the economic conditions of language documentation
- Debatable assumptions:
  - Documentation is worthwhile even without revitalization efforts
  - Software can help fieldworkers be more productive
  - (Worth discussing, but out of scope)
- Will necessarily critique existing LD apps. Extant apps are very impressive, not saying I could do better

Introduction

- Central claim: language documentation (LD) apps will succeed when their developers use software development methods that are adapted to the economic conditions of language documentation
- Debatable assumptions:
  - Documentation is worthwhile even without revitalization efforts
  - Software can help fieldworkers be more productive
  - (Worth discussing, but out of scope)
- Will necessarily critique existing LD apps. Extant apps are very impressive, not saying I could do better
- Will only consider "traditional" LD workflows

Introduction

- Central claim: language documentation (LD) apps will succeed when their developers use software development methods that are adapted to the economic conditions of language documentation
- Debatable assumptions:
  - Documentation is worthwhile even without revitalization efforts
  - Software can help fieldworkers be more productive
  - (Worth discussing, but out of scope)
- Will necessarily critique existing LD apps. Extant apps are very impressive, not saying I could do better
- Will only consider "traditional" LD workflows
- Roadmap:
  - The state of LD apps
  - Development in industry vs. academia
  - Experiment
  - Conclusion

Outline

LD apps today

- Many fieldworkers use generic "apps" for most of their workflow
  - E.g. Excel, paper, ...
  - No computer assistance in time-intensive tasks like transcription, glossing, concordancing

LD apps today

- Many fieldworkers use generic "apps" for most of their workflow
  - E.g. Excel, paper, ...
  - No computer assistance in time-intensive tasks like transcription, glossing, concordancing
- Others use a combination of linguistic apps
  - E.g. FLEx, ELAN, Praat, ...
  - Not always easy to move data from one into the other
  - Difficult to maintain a source of truth

LD apps today

- Many fieldworkers use generic "apps" for most of their workflow
  - E.g. Excel, paper, ...
  - No computer assistance in time-intensive tasks like transcription, glossing, concordancing
- Others use a combination of linguistic apps
  - E.g. FLEx, ELAN, Praat, ...
  - Not always easy to move data from one into the other
  - Difficult to maintain a source of truth
- Ideal: a comprehensive LD app
  - Built-in support for most tasks
  - No shuffling data around between apps

Why haven't comprehensive LD apps taken over?

- Successful comprehensive apps have been made

Why haven't comprehensive LD apps taken over?

- Successful comprehensive apps have been made
- Still, none has gained very widespread use

Why haven't comprehensive LD apps taken over?

- Successful comprehensive apps have been made
- Still, none has gained very widespread use
- Common complaints:
  - "x is too frustrating/hard to learn" $\rightarrow$ usability issues
  - "I can't use x because it doesn't realize my language's diacritics are making lexical distinctions" $\rightarrow$ missing/incomplete features
  - "I can't use x because I don't use Windows" $\rightarrow$ platform limitations

Why haven't comprehensive LD apps taken over?

- Successful comprehensive apps have been made
- Still, none has gained very widespread use
- Common complaints:
  - "x is too frustrating/hard to learn" $\rightarrow$ usability issues
  - "I can't use x because it doesn't realize my language's diacritics are making lexical distinctions" $\rightarrow$ missing/incomplete features
  - "I can't use x because I don't use Windows" $\rightarrow$ platform limitations
- LD apps are not a new idea—why are these still issues?

Why haven't comprehensive LD apps taken over?

- Successful comprehensive apps have been made
- Still, none has gained very widespread use
- Common complaints:
  - "x is too frustrating/hard to learn" $\rightarrow$ usability issues
  - "I can't use x because it doesn't realize my language's diacritics are making lexical distinctions" $\rightarrow$ missing/incomplete features
  - "I can't use x because I don't use Windows" $\rightarrow$ platform limitations
- LD apps are not a new idea—why are these still issues?
- One major reason: development methods in LD have been copied from industry

Outline

Developing software in industry vs. academia

- For software development, industry has more of every resource than academia

Developing software in industry vs. academia

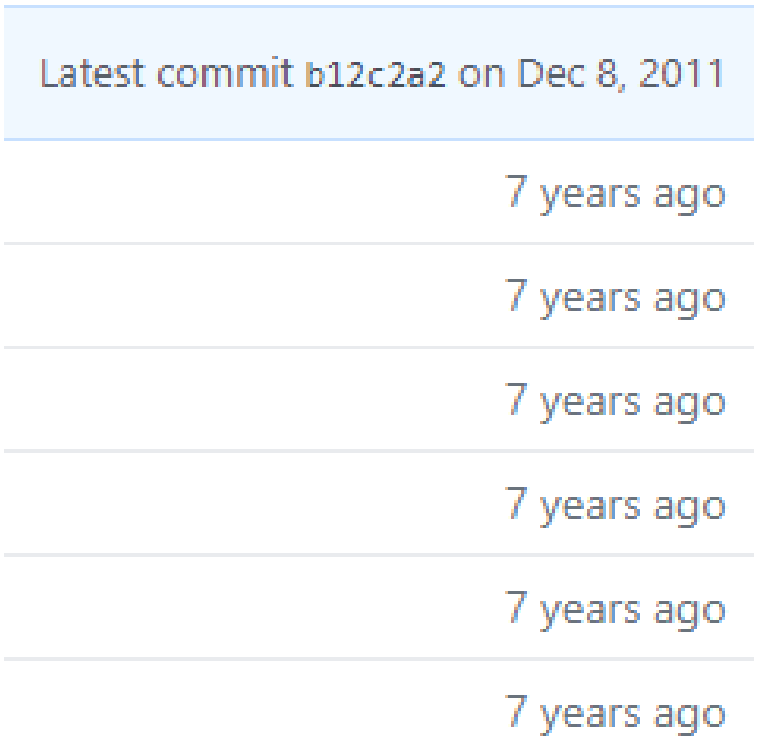
- For software development, industry has more of every resource than academia
- A typical industrial app has:
  - Lots of money
  - Multiple full-time professionals
  - Veteran (5+ yrs) developers
  - Support staff

Developing software in industry vs. academia

- For software development, industry has more of every resource than academia
- A typical industrial app has:
  - Lots of money
  - Multiple full-time professionals
  - Veteran (5+ yrs) developers
  - Support staff
- A typical academic app has:
  - A professor overseeing it (?)
  - Fixed-term funding at best
  - A graduate student or two with moderate programming experience

Developing software in industry vs. academia

- For software development, industry has more of every resource than academia
- A typical industrial app has:
  - Lots of money
  - Multiple full-time professionals
  - Veteran (5+ yrs) developers
  - Support staff
- A typical academic app has:
  - A professor overseeing it (?)
  - Fixed-term funding at best
  - A graduate student or two with moderate programming experience
- ...and the graduate student graduates



Adapting development methods

- The difference in economic conditions is obvious
- Less obvious: development practices from *industry* are suited to *industry*

Adapting development methods

- The difference in economic conditions is obvious
- Less obvious: development practices from *industry* are suited to *industry*
- LD apps would succeed more if methods were aligned with conditions

Adapting development methods

- The difference in economic conditions is obvious
- Less obvious: development practices from *industry* are suited to *industry*
- LD apps would succeed more if methods were aligned with conditions
- Hypothesis: to alleviate the labor shortage, LD app developers need to carefully choose tools (programming language, platform, libraries, etc.) which enhance their productivity
  - As opposed to performance, ease of use, security, correctness, ...

Acting on the hypothesis

Hypothesis: to alleviate the labor shortage, LD app developers need to carefully choose tools (programming language, platform, libraries, etc.) which enhance their productivity

1. Prefer libraries over writing new code
   - There's probably a library for it
   - Recent (<5y ago) explosion in availability of high-quality libraries
   - NPM (JavaScript library index) has over 400,000 libraries

Acting on the hypothesis

Hypothesis: to alleviate the labor shortage, LD app developers need to carefully choose tools (programming language, platform, libraries, etc.) which enhance their productivity

1. Prefer libraries over writing new code
   - There's probably a library for it
   - Recent (<5y ago) explosion in availability of high-quality libraries
   - NPM (JavaScript library index) has over 400,000 libraries
2. Choose libraries that minimize programming (development, maintenance)
   - Time spent adjusting for breaking changes is "useless" to users
   - Choose stable languages and libraries

Acting on the hypothesis

Hypothesis: to alleviate the labor shortage, LD app developers need to carefully choose tools (programming language, platform, libraries, etc.) which enhance their productivity

1. Prefer libraries over writing new code
   - There's probably a library for it
   - Recent (<5y ago) explosion in availability of high-quality libraries
   - NPM (JavaScript library index) has over 400,000 libraries
2. Choose libraries that minimize programming (development, maintenance)
   - Time spent adjusting for breaking changes is "useless" to users
   - Choose stable languages and libraries
3. Choose libraries that are especially designed for the challenges of our domain (LD)
   - No internet
   - Multi-user collaboration
   - ...

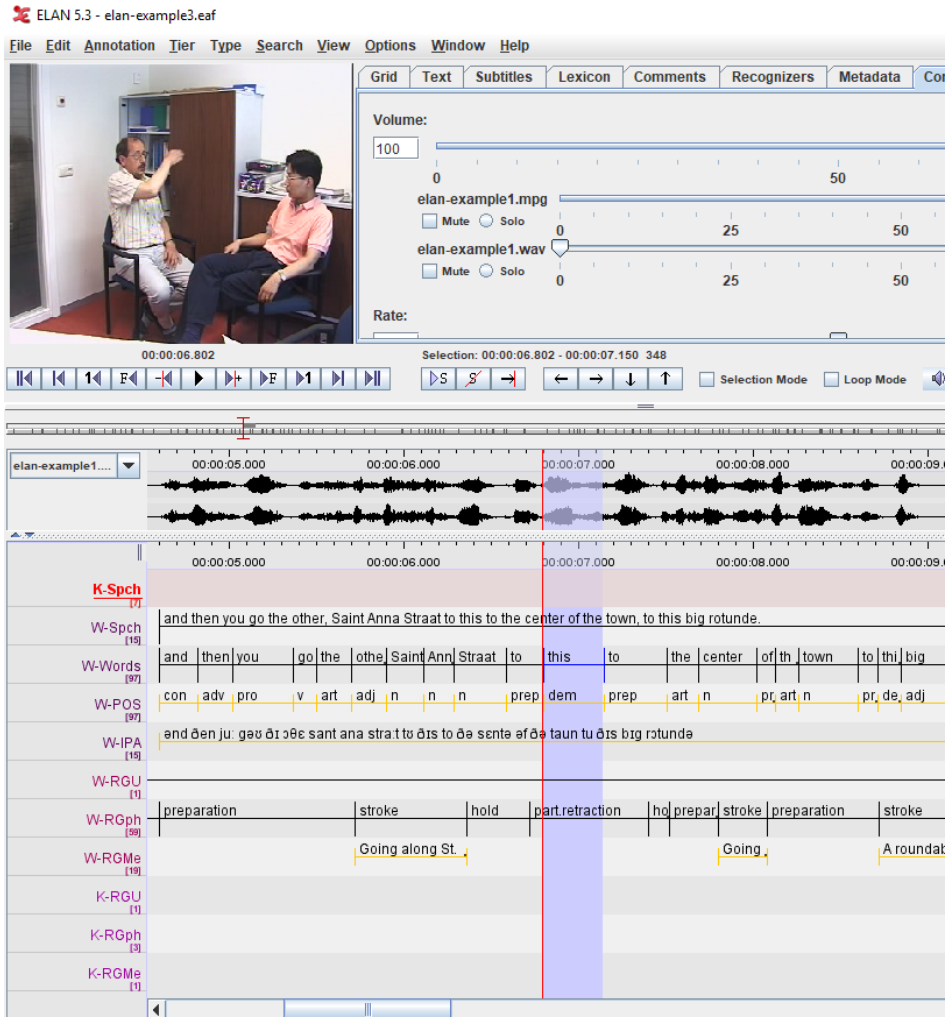
Outline

An informal experiment

- Goal: measure development time of an app, gauge whether careful tool choice helped
- Hypothesis has to do with *programming*, not design
  $\rightarrow$ used an existing design

An informal experiment

- Goal: measure development time of an app, gauge whether careful tool choice helped
- Hypothesis has to do with *programming*, not design
  $\rightarrow$ used an existing design
- Chose to clone a small subset of ELAN
  - Time-aligned annotation
  - Used widely in LD and linguistics
  - Old, proven design
  - Moderately complex



The tools

- Platform: web browser
  - Solves portability, installation problems
  - Browser is stable after a decade of explosive growth

The tools

- Platform: web browser
  - Solves portability, installation problems
  - Browser is stable after a decade of explosive growth
- Programming language: ClojureScript
  - A Lisp-family language with elegant but practical abstractions
  - Very stable: code from 2009 runs unaltered
  - Compiles to JavaScript

The tools

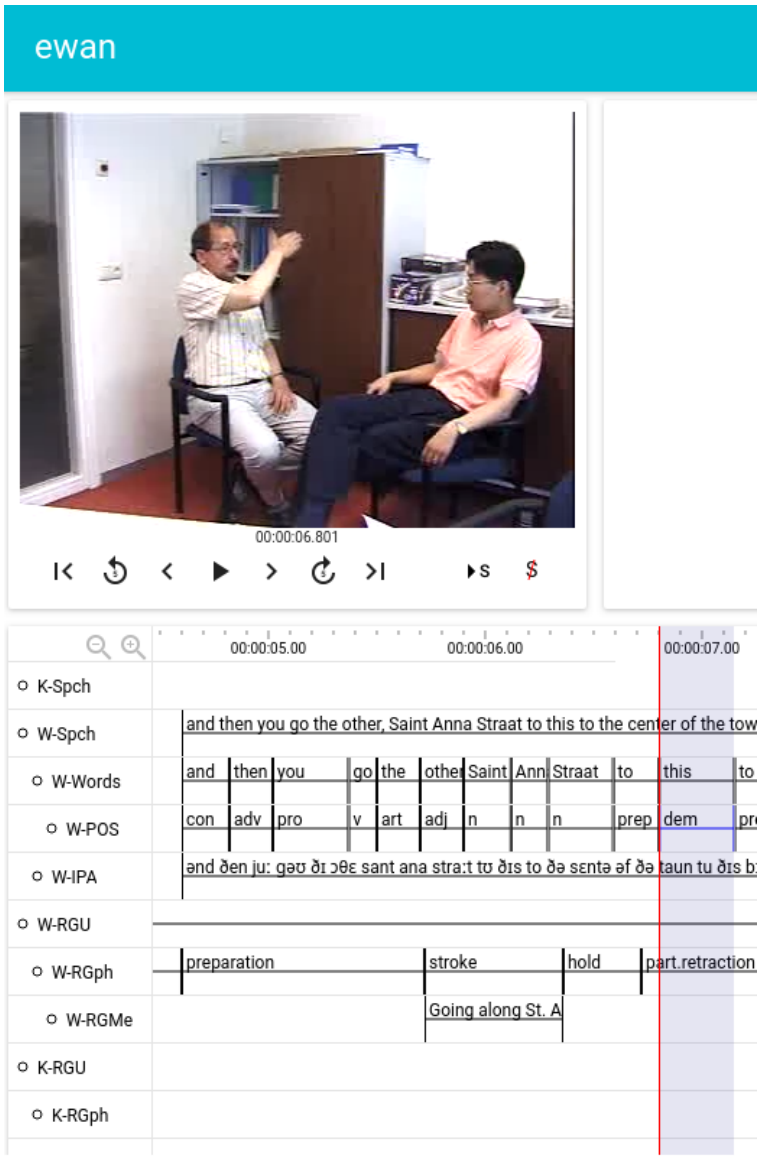
- Platform: web browser
  - Solves portability, installation problems
  - Browser is stable after a decade of explosive growth
- Programming language: ClojureScript
  - A Lisp-family language with elegant but practical abstractions
  - Very stable: code from 2009 runs unaltered
  - Compiles to JavaScript
- Database: PouchDB
  - Allows the app to be used offline even though it's in a web browser

The tools

- Platform: web browser
  - Solves portability, installation problems
  - Browser is stable after a decade of explosive growth
- Programming language: ClojureScript
  - A Lisp-family language with elegant but practical abstractions
  - Very stable: code from 2009 runs unaltered
  - Compiles to JavaScript
- Database: PouchDB
  - Allows the app to be used offline even though it's in a web browser
- Interface: Material-UI (React)
  - Expansive, high-quality collection of off-the-shelf, mobile-compatible UI components
- See paper or talk to me for more details

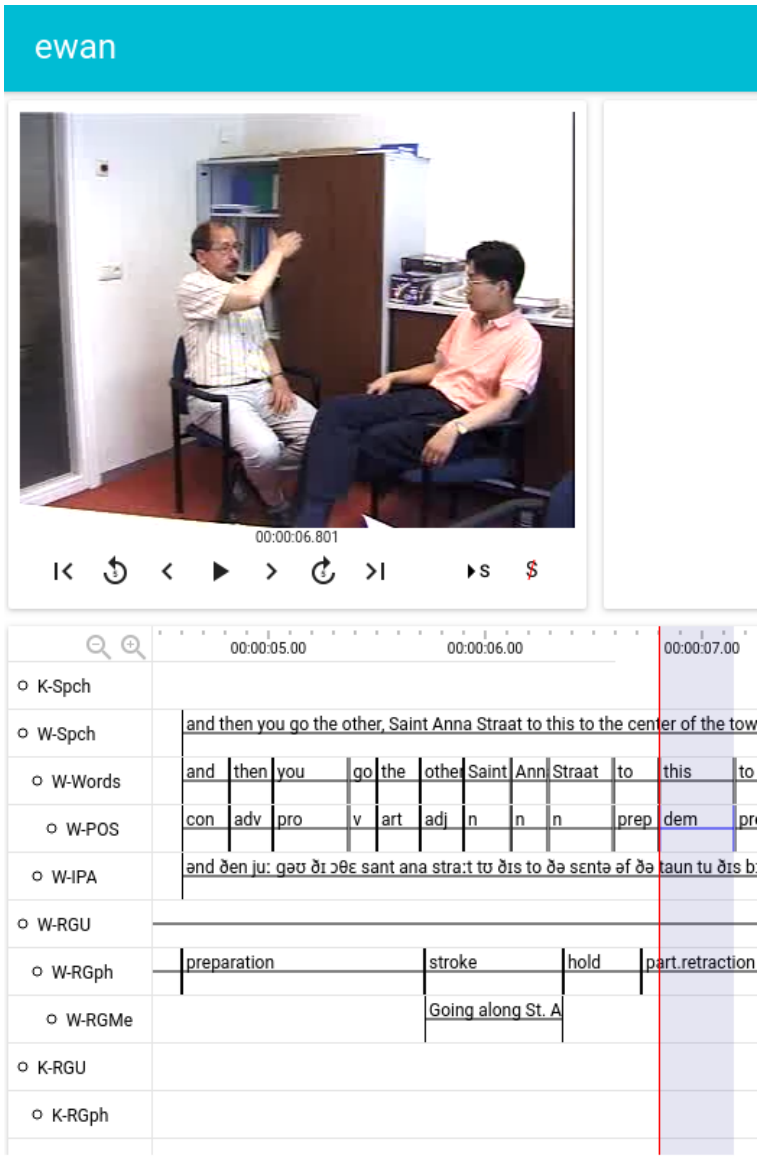
Results

- Check it out!
  https://lgessler.com/ewan/



Results

- Check it out!
  https://lgessler.com/ewan/
- Created a view-only subset of ELAN (not for real use!)
  - Development completed in only 3 40-hour weeks
  - Offline support achieved
  - Sync with other users provided for free by the database
  - Real-time collaboration greatly facilitated by the database chosen



Results (contd.)

- Check it out! https://lgessler.com/ewan/
- Comparing performance with how it would have gone with "standard" tools:
  - Offline support took almost no work
    - No browser-based LD apps offer this, possible but difficult to implement in a traditional architecture

Results (contd.)

- Check it out! https://lgessler.com/ewan/
- Comparing performance with how it would have gone with "standard" tools:
  - Offline support took almost no work
    - No browser-based LD apps offer this, possible but difficult to implement in a traditional architecture
  - Data sync provided for free by the database
    - A few LD apps offer sync, users are wary of it, reported to be buggy

Results (contd.)

- Check it out! https://lgessler.com/ewan/
- Comparing performance with how it would have gone with "standard" tools:
  - Offline support took almost no work
    - No browser-based LD apps offer this, possible but difficult to implement in a traditional architecture
  - Data sync provided for free by the database
    - A few LD apps offer sync, users are wary of it, reported to be buggy
  - Real-time collaboration greatly facilitated by database
    - No LD apps offer real-time collaboration

Results (contd.)

- Check it out! https://lgessler.com/ewan/
- Comparing performance with how it would have gone with "standard" tools:
  - Offline support took almost no work
    - No browser-based LD apps offer this, possible but difficult to implement in a traditional architecture
  - Data sync provided for free by the database
    - A few LD apps offer sync, users are wary of it, reported to be buggy
  - Real-time collaboration greatly facilitated by database
    - No LD apps offer real-time collaboration
  - Little expected maintenance thanks to very stable PL

Results (contd.)

- Check it out! https://lgessler.com/ewan/
- Comparing performance with how it would have gone with "standard" tools:
  - Offline support took almost no work
    - No browser-based LD apps offer this, possible but difficult to implement in a traditional architecture
  - Data sync provided for free by the database
    - A few LD apps offer sync, users are wary of it, reported to be buggy
  - Real-time collaboration greatly facilitated by database
    - No LD apps offer real-time collaboration
  - Little expected maintenance thanks to very stable PL
  - UI library provided almost all components needed, only a few needed to be created from scratch
    - Non-browser platforms often require much more work, even LD apps on the browser tend not to use as many libraries as they could
- "Experiment" is obviously subjective and limited

Results (contd.)

- Check it out! https://lgessler.com/ewan/
- Comparing performance with how it would have gone with "standard" tools:
  - Offline support took almost no work
    - No browser-based LD apps offer this, possible but difficult to implement in a traditional architecture
  - Data sync provided for free by the database
    - A few LD apps offer sync, users are wary of it, reported to be buggy
  - Real-time collaboration greatly facilitated by database
    - No LD apps offer real-time collaboration
  - Little expected maintenance thanks to very stable PL
  - UI library provided almost all components needed, only a few needed to be created from scratch
    - Non-browser platforms often require much more work, even LD apps on the browser tend not to use as many libraries as they could
- "Experiment" is obviously subjective and limited
- That said, hypothesis seemed corroborated: providing similar features with standard tools would have taken much longer

Outline

Conclusion

- Changing software development methods is key to progress in making comprehensive LD apps succeed because LD apps have so little labor available to them
- LD app developers need to optimize for productivity in their tools
- My three proposed commandments:
  1. Prefer libraries over writing new code
  2. Choose libraries that minimize programming
  3. Choose libraries that are especially designed for the challenges of LD