Rough Analysis of the Pre-Workshop Survey

Motivations to Solve the Problem

Achieve Greater Access to Data

- Increased accessibility of annotated data / corpora
- To streamline the process through which "ordinary working linguists" (a term usually used to encompass field linguists with relatively little computational sophistication) can make useful language resources on the basis of data collected in the field.
- To make it easier for the general public (including speaker communities) to make use of language resources created by scholars. (This is of central importance to many endangered language linguists, but less important to me personally.)
- There seems to be a split sometimes between annotating and searching through the annotation, and by keeping the parts more uniform UAT may provide a way to better integrate searching. This is related to my second point, in that flexible UAT would provide different views of the same data to assist in annotating (e.g., sentence-by-sentence, construction-by-construction, annotation inconsistencies, interesting connections between layers, etc.), and such views are likely useful for users who simply want to search.
- Encourging cross disciplinary interaction. Fact that different disciplines is completely different tools really shuts off a lot of data from a lot of people. There's a lot of scope in opening up the kind of tools that we use to broader use and promoting interoperability of tools for collecting data analysing data.
- Enable groups who lack substantial computing expertise and resources to better exploit linguistically annotated data
- The need to embed viewing and annotation of text in a variety of different applications

Avoid Reduplication and Reduce Cost

- Reduplication of work across different projects is, of course, the most obvious issue. Stemming from this, though, seems to be a slight discouragement of developing innovative tools because time needs to be spent on getting the core annotations working and thus cannot be spent figuring out ways to improve interfaces, consistency checking, IAA modules, etc.
- All-in-one solution: The re-use of generic infrastructure for e.g. annotator management, agreement computation, and project workflows.
- To make it easier for people to get started in collecting language resources, and annotating in such a way that they remain useful throughout their life. So we can spend less time developing the same old tools and more time actually studying language.
- More effective use of available funds - better return on investment
- sustained maintenance of tools
- Reduce (or better eliminate) the fragmentation caused by the many tools available
- Concentrate the development effort to a single tool, which will result in more capabilities for the tool, robustness, etc.
- Less bugs, as widespread usage will allow easier detection/bug fixing
- Reduce overall cost/time required to produce linguistic resources
- There will always be a need for annotations
- current segmentation of effort
- a “software waste cycle”, in which tools with substantially overlapping functionality are repeatedly created from scratch by researchers and teams all over the world, causes enormous waste of funding and human resources;
- reduction of marginal cost for developing new systems / lowering the barrier to entry by non-developers
- Reducing the labor currently expended on continuously re-creating annotation tools for different annotation teams.

Ease Annotation

- Facilitation of users' workflow, from coding to analysis
- Improved access to annotation tools
- Ease of starting and completing a wide-variety of annotation tasks
- Classification of annotation tasks (annotation objects and annotation work flows).
- To facilitate workflows where different individuals are adding different kinds of annotations to the same primary linguistic data and where these annotations can be straightforwardly integrated to create richer language resources.
- Ability to use a variety of taggers on CHILDES and TalkBank data
- Ability to analyze web pages in a variety of language for second language learners
- Linkage to systems for automatic segmentation, pause detection, and perhaps ASR

- Efficient expansion of annotated corpora that are free of errors, which can be allowed to grow in scale with relatively easy maintenance of the data.
- Efficient presentation of annotation tasks to human annotators, which take advantage of annotator strengths and speed annotation with automatic pre-processing where possible.
- Distributed annotation: The collaborative annotation tool will be used in a distribution fashion with the only requirement being internet connectivity. Annotators can work at any time and from anywhere, without concerns for data losses and continuous intervention to save the data.
- Unlocking a larger workforce: The main goal of an annotation tool is to generate large annotated corpora. Similar to crowdsourcing platforms, it is possible to generate larger amount of annotated corpora more quickly by making them accessible to larger workforce.
- To make it easier for people to get started in collecting language resources, and annotating in such a way that they remain useful throughout their life. So we can spend less time developing the same old tools and more time actually studying language.
- Chaining processing steps from different providers is very attractive to us
- Easy combinability of existing functionality is the key for the average user, with highly sophisticated functionality available in some tools
- For making it Unified: That annotation of under-resourced languages can become a collaborative enterprise involving members of the language community itself
- there is very little support for crowdsourcing annotations; collaborative, distributed annotation; and semi-automatic annotation involving a “human in the loop”, which are becoming increasingly common.
- the advent of linked data is demanding that consistency among annotation models be achieved, in order to ultimately enable the exploitation of all kinds of annotated data in the Semantic Web.
- Promoting awareness of the entire life-cycle / process associated with the creation of type systems, annotations, corpora etc. from completely human linguistic analysis to automatic computer processing

Facilitate Archiving

- To facilitate the archiving and long-term preservation of linguistic annotations. (This is something quite important to me, but it falls out more or less naturally from the other issues listed above. So, I've made it a lower priority here.)
- Improve sustainability situation

Improve the Range of Annotations

- Improving the range of annotation layers that can be easily annotated.
- Multi-language situations being researched pose a specific problem to tools designed to support one (specific) language only. It seems much for feasible to mix small components in as needed than to wait for all the necessary tools to be upgraded and made compatible.
- For annotation tooling: To adequately document and describe the thousands of languages of the world

Increase Quality of Results

- Promoting best practices in annotation
- Encourage replicability/comparison/evaluation of results (like IAA)
- we need means to consistently document annotated data and evaluate annotation quality; a well-conceived infrastructure for annotation can support this
- Creating best practices/methods for visualizing annotations and making annotations quickly and easily
- development of global requirements and shared best practices (processes associated with use of UAT);
- How best practices / requirements from both realms should better inform each other and the creation of UAT
- Establishing best practices for making annotation tasks as straightforward and as cognitively light as possible for the annotators.

Obtain Interoperability

- Interoperability of annotations and tooling
- Greater ease of exchanging annotations (even with the current widely adopted standards)
- To allow all kinds of linguistic data to be more easily and opportunistically used across diverse tools.
- To facilitate the interoperation of language resources developed for low-resource languages, especially those created by field linguists working on endangered languages. (This has been an important aim of many working in the area of digital

standards for endangered languages, and it is of interest to me, but lower priority, in personal terms, than the first three points listed above).

- To make it easier for those working in other disciplines to use language data since they would only need to learn one annotation scheme (ideally).
- Interoperability. Annotation resources each have different strengths and weaknesses, as they generally represent different types and levels of detail in their representations. To take advantage of strengths and overcome weaknesses, and to ideally combine independent annotations into a larger, more diverse training corpus, these resources should be interoperable.
- To make sharing a data between different disciplines easier and therefore make resources when they have been created more widely useful.
- Improvement of the decision process for users - trust building will be less difficult - promise of increased interoperability
- ability to process annotations created by other tools
- standardization
- For making it Unified: That the data might interoperate smoothly across a whole ecology of tooling
- Encourage interoperability of linguistic resources
- lack of interoperability
- the development of linguistically-annotated corpora is currently inconsistent; the resulting proliferation of vastly varying annotation models and practices prohibits interoperability and reuse of these resources, which in turn prevents the exploitation of high-quality annotated language data by groups that lack substantial computing expertise and resources; (
- Standardization and interoperability;

<u>Provide Extensibility</u>

- Increased probability of a community process for extending the tool
- Individual tools do not generally account for various possibilities of (future) annotation needs, and UAT would ideally address this by attempting to develop flexible infrastructure which addresses a range of annotation types.
- Enhanced flexibility: A general-purpose annotation tool provides better flexibility, in such a way that any type of annotation layers can be created, depending on the data collection need of the target application.
- Creating a universal set of recommended tools that would be easy to adapt or extend to handle most common (and not-so-common) types of annotation tasks.

What Disciplines Should be Prioritized?

No Prioritization Needed

- I don't think one can really [prioritize disciplines]. A better approach is probably to "cluster" the various disciplines to see which ones are closer together and which ones are further apart.
- I don't think it makes sense to identify fields and/or disciplines where the tools and procedures should be applicable. I think we need to abstract away from that. What should be identified is what kinds of annotation should be made possible and what kind of work flows are supported.
- If "should have their needs prioritized" means that they should agree on what to forget about, that's difficult from the beginning on. I believe that progress can only be achieved by providing extra value to specific disciplines.
- No single discipline is more identifiable than other, since data markup and annotation of language-related or linguistic information can be a goal within any field. That said, we can measure the linguistic annotation needs of a discipline by asking: how significant a component of the data analytics problem in your field is natural language and text? For humanities and the languages, it is obviously a pressing need, given that is the natural domain of discourse. For the social and behavioral sciences, including economics, annotation can be an enormously critical source of additional metric for trends, sentiment, and other domains. Even in GIS and epidemiology, there are needs involving language annotation.
- I do not feel that any field needs to be prioritized
- I see it not as a matter of priority, but as a matter of classification. The same types of annotation tasks may be required by different subfields, while the same subfield will require different annotation types, depending on the task. In the tutorial we gave at LREC last year, we developed a mini-typology of annotation tasks that classified different types of annotation along several axes, including distinguishing surface spans / attribute annotation from relation annotation, separating annotations with span-specific vs span-dependent label sets, and annotation of overt vs. covert phenomena and elements.

Proffered Prioritizations

- Endangered languages are disappearing rapidly, so a priority for the needs of language documentalists and field linguists would make sense.
- Fields and disciplines which could achieve significant impact (e.g. biomedical QA, legal document analysis) through more effective annotation, corpora creation and automatic processing are obvious targets.
- Documentary lingusitics -- Because there are thousands of languages to document and describe, and the general consensus is that half of them are in danger of dying by the end of the century
- social media/blogs; clincal reports; biomedical literature
- Natural Language Processing; 2. Corpus Linguistics; 3. Semantic Web; 4. Language Documentation; 5. Psycholinguistics; 6. Education
- NLP, biomedical text mining, digital humanities, machine translation, social media mining NLP
- computational linguistics and language technology; 2. corpus linguistics; 3. literary studies; 4. History; 5. political science; 6. sociology
- Linguistics, Sociolinguistics, Language Learning
- Semantic annotation; Annotation with ontologies at the segment level; Sentiment analysis; Argumentation mining
- computational linguistics; corpus linguistics; social sciences (historical, political,...)
- Speaking entirely from the perspective of LDC and our sponsored project needs, I see the greatest need in the area of UAT for semantically annotated and lexical resources.

Problems with Existing Tools

Difficult to Learn

- too-steep learning curve, inadequate documentation/tutorials available
- Steep learning curve
- Opacity of functionality and ease of adoption

Difficulty Running or Installing

- Don't run on particular operating systems (e.g., OSX)
- Often what they do is so simple that we just reimplement them in CLAN.
- Difficulty accessing and running the tools. Some annotation tools that I've worked with require annotators, who may not be familiar with a Unix environment, to launch their tools within a Unix environment by giving a command. Although they are trained on this process, this can be a frustrating barrier to annotators who may have difficulty navigating a Unix environment and giving the correct command.
- Difficulties installing and setting up tools, and finding out how they work and what they do.
- Installation complexity
- tool requires computing environment we can't easily support (e.g. special server, security concerns)

Inadequate Importing, Exporting, or Conversion

- Difficulties in going from the annotation tool to data formatted for publications. Linguistics publications tend to require data in a visually appealing interlinear-glossed format. Most of the tools that I use don't make it easier to explore such a format. So, to make a publication, one has to do a lot of the work by hand. Sometimes, rather than annotate properly, I just start with the publication-ready format.
- Only partial support for conversion between annotation formats
- Lack of a standardized data interchange format that would allow the new tool to interoperate nicely with the other tools I was using
- data output format is too inflexible and/or doesn't meet our requirements
- "Academic code" and built-in platform assumptions
- lack of means to input/output annotations in a desired format
- no ability to read in or perform pre-annotations
- tools don't always handle schemas with a large number of types in a graceful manner
- mapping between type systems created by different tools / schemas
- They mostly failed to interoperate with CHAT format.

Lack of Documentation or Support

- Support for the tool
- For syntactic dependency taggers, the problem has been the poor documentation of the range of tags, bad interoperability, etc.
- some tools lack enough documentation to know exactly what to do
- Problems with maintenance of tools, finding tools that look useful but that nobody is taking care of
- Lack of documentation, tutorials, examples that were adequate to figure out the tool within the timeframe I was willing to spend on evaluating it
- Lack of documentation, at best, a text file. It is difficult to even get the tool running.
- Lack of support, people leave academia and suddenly there is no one to ask about how things work.
- lack of consistency of formats and scheme design;

Missing Functionality

- Limited functionality: no “coding scheme”
- Often the tool only provides a subset of the features desired.
- Lack of support for languages written where diacritic marks are used to encode tonal contrasts. This is a concern that is somewhat technical but, at its heart, relates to how some transcription systems are based around a model where "accents" encode tonal contrasts, but no tool that I know of is aware of this, which causes problems for automated parsing and searching. (I can give more details if you'd like. This is a particular problem for Africanists.)

- For morphological analysis, there are not really any tools out there that get to the level we need. The Xerox/PARC FST framework has computational problems and is not open enough for general use.
- Lack of easy search functionality for "homegrown" types of searches, i.e., searches that are specific to our particular needs. For example, we wanted to find mismatches between annotation layers that might have helped reveal something about the annotation scheme we would have wanted to revise.
- Visualization is thus sometimes a challenge
- while dedicated video and image annotation tools exist, their support for text annotation is poor.
- Many lack support for collaborative annotation over the web, which is now a key requirement for the projects we work on
- Lack of sufficient flexibility in supporting different annotation flows and mixing with automatic bootstrapping and adjudication methods
- Many tools handle video now, but don't give it a centlra role
- many tools target either single researchers or languages with an established writing system with an orthography, both are not the case for our research.
- The tool was missing key functionality I was looking for
- lack of workflow support (inability to manage users & assignments, track progress and perform other management functions)
- lack of some key functionality
- lack of means for quality control and assessment
- can't be embedded in other applications - is a standalone
- not able to do instance-level annotation as well as document- or corpus-level annotations
- cannot create shortcuts for users (e.g., mark all instances of "pain" as Symptom - don't make me re-annotate the same words over and over."
- Lack of necessary functionality for the task at hand.
- Lack of “coverage” e.g. inability to handle relations between either overt or covert entities.

Poor User Interface

- Bad user interface: you want coding to be efficient because it is very time-consuming (and boring!)
- My main beef, including on tools I wrote, is usually the ease of use of the interface.
- Tediousness of performing some tasks (For example, the Alembic Workbench required a long sequence of click to add a relation.)
- General usability. Most tools produced by scholars have clunky user interfaces and, in particular, they don't follow the standards used by professionally-made software. This requires learning new interfaces and workflows for each of these tools, and I choose carefully when it is worth learning them.
- Usability
- Annotator fatigue
- poor design makes things cumbersome or error-prone or unnecessarily tedious for annotators
- too many clicks - burdensome to use
- Uncomfortable, difficult, or needlessly silly interface.
- Too much functionality, too much configurability (MATE workbench)
- Complexity and lack of transparency. Some annotation tools benefit the annotation process by proving built-in reference tools. While this is advantageous, it can also create an interface that is intimidating to annotators, especially when the functionality of the tool is not totally transparent to the annotator
- Poor workflow design, i.e. sequences of actions required to create an annotation were too long and/or seemingly unmotivated.

Problems with Extensibility

- I found integrating rhetorical relations difficult in ANVIL.
- Lack of customization of data formats and annotation categories
- Rigid restrictions on annotation types - e.g., not (easily) allowing for two separate part-of-speech fields.
- Many tools were annotation task specific and/or closed source, which made them hard to modify and customise to new corpus annotation needs/tasks
- Inability to adapt tools to established practices in sub-fields
- Customisation of annotation schemas

- too difficult/time-consuming to modify compared to building our own custom tool
- "Academic code" and built-in platform assumptions
- difficult to customize
- Workflows that allow for partial annotation or annotation of one "layer" or component of a problem have been hard to develop.

Unstable, Slow, or Buggy

- Stability and reliability of the tool.
- Speed. Ideally, the annotation tool should work as quickly as the annotator. Some web-based tools and those accessed via a secure shell can become very slow outside of the ideal connectivity contexts.
- Tools with bugs. We cannot take for granted that a tool simply saves data properly, instead of crashing or saving data with small errors.
- Some tools are not well-suited for large data files,
- tool is buggy
- Bugs. The more complicated the annotation task and the corresponding tool, the more likely one finds bugs in the way the tool functions.

Desired Capabilities

Import/Export Capabilities

- good export functionality is a must: easy to export such that the dada can be used in a publication
- JSON needs to be supported for handling tweets
- Both inline and standoff markup needs to be supported
- Reading any text format and XML would be a necessary capability.
- importing and exporting data and annotations
- transferring between formats

Media to Support

- core ability to manipulate time-aligned data (heavy data, such as multi-HD video streams)
- ability to annotate data that cannot easily be moved to different places because of bandwith limitations or legal reasons
- Support for images and, even possibly, video annotation, when mixed with text: social media and other digital content is no longer just textual and while dedicated video and image annotation tools exist, their support for text annotation is poor. Ideally, we want something unified and capable of doing all those to at least some degree.
- It is expected that in many fields the use of and availability of multimedia recordings will increase

Supported Schemes

- should provide tools to do syntactic and semantic role annotation in many different languages
- Providing information on coreference should also be supported by UAT. Coreference information can span documents, and therefore a tool that allows users to visualize coreference chains, and connect entity mentions, can be difficult to develop.
- semantic annotations need to be enriched with causal/temporal chains that go beyond a single clause or sentence.
- I think it will be important for a UAT to support some sort of Linked Data friendly format (e.g., RDF)
- UAT should support a tier-based, multilevel, hierarchical annotation scheme. And probably a syntax or treebank type of scheme as well.
- The file format should be XML based.
- linking transcripts to audio
- Support for relevant linguistic annotation standards, such as ISO/TC 37/SC 4
- Support as many possible import/export formats as possible.
- it is clear that annotation schemes is not a closed set, so it is important that UAT be defined in terms of extensible schemes that can represent new schemes (such as RDF and the Linked Data framework).

Workflows

- Creation of dependency parses - a web-based, fast, user-friendly tool to create dependency parses for a variety of languages would be very useful, providing syntactic information and semantic dependencies for languages that may not be suited to a phrase structure analysis.
- I believe that it would be good to support more "distributed" annotation, in an "assembly line" process where people with different skills perform different kinds of annotation.
- work flows should be customizable
- there needs to be support for crowdsourcing and distributed (collaborative) annotation and annotation involving “human-in-the-loop”.
- Multi-role support, including user groups, access privileges, annotator training, quality control, and corresponding user interfaces.
- Shared, efficient data storage to store and access text corpora and annotations.
- Support for automatic pre-annotation services and their configuration, to help achieve time and cost savings.
- Flexible workflow engine to model complex annotation methodologies and interactions.
- I think it would be fine if an UAT just does the annotation part of the workflow. Collection of raw data and (possibly) conversion to usable formats, project management and metadata creation, uploading to an archiving / long term storage system: those parts might best be dealt with by other tools. Where possible smooth integration with tools that perform these other tasks should be provided.
- Distributed annotation approaches

- A workflow that incorporated interactive annotation in a way that was scalable to larger corpora would be good to investigate
- Every succesful annotation project has been iterative. [support for this is a must.]
- possibly interactions with Mturk
- we need a very flexible and powerful workflow system that allows us to arbitrarily combine manual and automatic annotation stages, to manage data sets and user roles, and to prioritize the various components
- supporting multiple annotators
- adjudication
- assignment of annotators to annotation tasks
- ability for annotators to work online and offline
- A UAT should be as versatile as possible: offer a set of workflows, but not impose them, allowing the annotation within a custom workflow
- We can enumerate annotation tasks. Workflow is the general problem of how we orchestrate the flow in which the output of one task is the input to the next. What is needed is not a set of workflows that are supported, but a system for orchestrating workflow among tasks

Misc Functionality

- the tool should be designed to work on all languages, especially low-resource ones, from the start, rather than a more usual approach where a tool is designed to work on well-known languages first, with other languages only supported after
- Open source: An open source annotation tool can be extended with new functionalities, and is thus subject to a collaborative (programming) process. This flexibility makes a tool more attractive for people that conduct and oversee annotation projects.
- anotator analysis (IAA, agreement with adjudicator, speed)
- built-in logging and commenting
- It should include a web-based version.
- Annotation tools (or composers of multiple tools) should facilitate different layers of annotation and abstraction, in a easy to use and simple to understand fashion. This layered annotation should be easily integrated or composed with other levels and domains of annotation.
- The domain user or expert (e.g., clinical or legal texts) should have a gentle learning curve and an intuitive experience for annotating the material in text that they are most familiar with.
- An annotation tool should be integratable on top of any document format (doc, pdf, txt), and any basic application program; e.g., MS Word, Adobe, Excel, iPhoto, Latex, etc.
- Keeping the learning curve for tools very low [from the point of view of the annotator] is critically important.
- UAT should be modular in design and implementation.

How to Implement the Solution?

Build Links and Reuse Existing

- In my opinion the most promising avenue is to build bridges between tools, either in the form of additional software or in the form of tool extensions and/or in the form of interface and file format conventions that people agree to. I am not speaking of an all encompassing super file format though. We tried this once by using Annotation Graphs and it turned out that a file format always has limitations wrt. a tool which have to be fixed with special notation which in turn makes a unifying format meaningless
- I want methods for linking existing tools.
- I think that we should focus on creating a web-based application that allows users to easily access a variety of different annotation tools that can be used in a modular fashion to efficiently complete the desired steps of annotation with consistent input/output formats that ensure interoperability of annotated corpora.
- Maximal compatibility with existing major tools is going to be one of the biggest problems. This should perhaps be the primary concern when designing a new tool.
- Starting from scratch always has the danger of repeating all the mistakes that existing tools have already made and fixed.
- Also, how can one claim to unify annotation tooling while creating yet another tool? Will it be better than the old ones? And most of all: when will it be ready for use?
- It should, wherever possible, re-use existing tools and workslow engines.
- I probably lean towards trying to leverage as much as possible
- I think it is important to build on what we have now that is working well. Previous efforts to build the one true tool have not worked but there are many tools that have a solid following and support base. Establishing new tools is hard.
- My feeling is that we need to make use of the emerging interoperability standards and look at how to adopt existing tools with a good following to work with them and so work with each other.
- Building on existing tools has the advantage of potentially being more cost effective because of the re-use of existing implementations. But adopting existing code might appear to be hard and more time consuming than anticipated.
- Existing best practices must be exploited and reused.
- Ideally, code that works as expected (and is expandable) must be reused as much as possible.
- I would suggest leveraging an existing generic infrastructure (e.g., the LAPPS framework), providing support for annotation with existing annotation infrastructures such as GATE and UIMA, and, for other purposes, building on and/or adapting existing tools such as LDC's WebAnn and Darmstadt's WebAnno. This would include customized interfaces for different users/applications, and even the ability for a user to specify and control the features of the interface.
- I strongly suggest that any such tool should at first be an integration or modification of current tools.
- The best of present tools should be taken as a "must have" for any redesign into a new functional specification for a new tool.
- There is something to be said though for selecting a popular tool and tinker with it till it better fits UAT principles.

Difficulty in Reuse

- Extending an existing tool can be quite painful given the evolutionary growth of most tools and the ensuing chaos in the code base
- I generally don't ever favor a "build from scratch" approach, but that doesn't mean it's not the right approach here. I don't think any existing tool is good enough to form the basis for UAT.
- Design and build from scratch has many advantages; start with a clean slate with no legacy code that might not be flawless. It might be the most expensive option, re-implementing functionality that has already been implemented by others many times
- Especially when attempting to build on multiple tools the question of supported platform(s) and applied programming language(s) becomes manifest.
- With all kinds of OS dependencies, some fresh start might make life much easier.
- While it is often a good idea to re-use code, sometimes it is more practical to start over.
- I always like the idea of building on what has already been done, but one reason there are so many tools is that existing tools aren't usually easily adapted to meet the needs of another project.

No One Tool is Possible

- Best to avoid something like GATE by focusing on interoperability.

- I think we need to view UAT not as an ideal tool to be developed, but as an ecology of tools that interoperate over common formats and schemes.
- I suspect that building an all-encompassing single tool from scratch would be doomed from the start. Clearly we have a variety of groups that are willing to create annotation tools, the problem lies in the lack of an incentive to work together and to create tools that work well with each other. Thus it would most likely be a better idea to somehow coordinate these groups. Perhaps towards a unified API of some sort?
- I do not believe there can be a UNIVERSAL tool, and it is difficult to address this question without narrowing the types and usages of the annotations that are potentially being produced.
- I do not think there is any likelihood of a single annotation tool for all tasks and domains, it should be the case that all major annotation tools or environments should be interoperable, both in functionality and in content produced.

Misc

- My hunch would be the hybrid approach. It is important to incorporate formats and and other elements of existing workflows, whether embodied in a tool or not. But I think the core would be implemented from scratch.
- What I've always thought would be a good approach, though admittedly I am not a developer, is one where the core of a UAT would be a kind of "engine" (adapted from the notion of a browser engine) designed to perform basic operations over a standardized annotation model (e.g., Annotation Graphs, just to pick one) such as "add annotation", "delete annotation", "modify annotation", with different projects building new tools on top of this engine.
- More importantly, maybe, is the decision on what kind of devices the UAT should run and, related to that, whether it should be an online or offline tool (or both). It will not always be possible to port a desktop application to an app for tablet or phone (or other new devices of the future). The decision on this might determine whether it is at all possible to build on existing tools, or at least on which ones of them.
- Re-engineering with maintainability in mind might be a good solution. This of course does mean that everything needs to be reinvented.
- WebLicht gives online access to annotation tools, eliminating the need to download and install the tools. It uses a building-block approach, which allows users to mix-and-match tools for different annotation layers. This approach is made possible by using a common data exchange format, so that one tool can process the output of another tool
- As I mentioned, I think we need a set of tools, specialized for different (general) types of annotation tasks - cf. the mini-taxonomy of annotation tasks.

How to Manage/Fund/Organize the Solution?

Funding

- If it is possible to acquire funds for a longer period of time to guarantee development, maintenance and support for a longer period of time, that should be preferred, of course, but I don't know of such funds.
- At least with DFG, the German Science Foundation, it has become possible to get support for creating "research infrastructure". Having such an initial grant to set up a project in a way that it can be maintained by the user community seems one way to go.
- I would imagine funding coming from two sources: -core infrastructure and sustained maintenance via NSF CRI or similar programs -customized "modules" for particular annotation tasks via individual project funding Any group who wanted to access the core infrastructure & services could be required to contribute funding to customize the module(s) that would support their particular task(s).
- subscription services can bring in some amount of support. The problem with this is how to bill, and how to differentiate users. Ideally, a platform or tool should be as freely available as possible, so as to encourage adoption and use by as many people as possible. Silver and Gold levels of support might be considered for corporate sponsors, once they see a monetizable payoff.
- I would propose to NSF (or whoever), a project that would work on a UAT as one component and also develop a sustainability model as another component. This latter component would involve workshops, expert consultation, etc., and, by the end of, say, two years, I'd hope for a proposal which could be tested. In other words, I'd treat sustainability as a research question like any other. I'd also ask around for what disciplines have pulled this sort of thing off, probably starting with biology, since my impression is that they have tools which have been supported for a while (though this is helped by the fact that they have a lot of money).
- By building a tool that will attract a community that understands how to extend it. If the community needs the tool, it may provide resources to help sustain the tool.

Implementation

- having an excellent method of distribution of code & versions is also critical to getting it going.
- To me, simplicity and ease of use is often the thing that puts me in the good mood with respect to a tool.
- A shift in focus to developing reusable, library-based code with task-specific modules and customizations would greatly improve the overall quality and cohesiveness of annotation systems to the benefit of the entire research community.
- Development of such an industrial strength toolkit will only be possible with sustained support for a robust linguistic annotation infrastructure that can marry the "science of annotation" with well understood principles of software engineering.
- We shouldn't imagine that we can start from scratch and impose "the one true UAT". A hybrid approach seems the most likely to succeed.
- The most important management strategy is to develop the most compelling and universal capabilities first, so that early adoption is widespread.
- To be successful, there needs to be an application for UAT from early on.

Open Source

- After a startup period, all code and other materials should be available to everybody and contributions by people outside of the core should be welcomed.
- My first inclination would be to suggest an open source development strategy. However this doesn't always result in production level code.
- I think that an open-source code base promotes active development.

Organization

- I think it would be simpler to maintain one site that houses and chains together different modules for annotation, which could be independently maintained. If one particular annotation tool is not well maintained, then such a site would offer users other competing tools that are being maintained. That being said, it would also be important for independent tool creators to provide living guidelines on the shared site that would communicate developments and updates.
- For most approaches, it seems rather feasible to maintain peripheral functionality pluggable in some way. So the core functionality is more of a problem. None of the current systems that I know of have been developed in a way that it could be maintained by a group should the main developer no longer be available. So that certainly has to change, and definitely means quite some overhead.

- The secret to sustainability to be build a starfish rather than a spider, as in "The Starfish and the Spider: The Unstoppable Power of Leaderless Organizations"
- We should conceive of UAT as an ecology of interoperating tools, data repositories, and automated services builds on the starfish model and allows for the whole to thrive, even as the individual members wax and wane. The Open Language Archives Community offers an example in which a small amount of grant funding helped to build the infrastructure for interoperation, and now 15 years later there are 56 participating institutions each of whom are funding their own participation.
- Perhaps, by creating a working group that would oversee the development / integration of different types of tools, maintaining a common project website and code repository.

Partnerships

- I guess there must be at least two bigger funded projects, one on the US side and one on the EU side, that run in parallel and are coordinated with each other.
- join forces with existing iniatives (e.g. LDC, Linguist List, SIL) and research infrastructures.
- Getting buy-in from different developers seems critical (which probably means leveraging some currently existing tools & formats).
- It might make sense to partner with industry on current product development like clinical coding modules (aka Nuance/IBM). Development based around common research/product goals ensures sustainability over the long run.
- If UAT is perceived as the best tool for building / managing large scale resources (like those disseminated by the Linguistic Data Consortium), then seeking partnerships with large-scale resource creators would be another strategic move to consider.

Review

- I think it is important to look at what is working now.
- We need a good survey of the current status: what is available, what are its advantages/disadvantages.

Other

- It would be great if, apart from content and engineering, we could discuss the potentials for project proposals. It may also make sense to discuss this both for the whole group and for the US and EU separately (since funding structures are quite different, I suppose).
- As I have no idea whether I am the complete outlier with my approach to annotation, I am not sure what to say. A good idea might be to start [the workshop] with what you had in mind, and then have a discussion at the end of the first day about priorities for the second.
- The workshop should accomplish the following things:
    1. Everyone in attendance agrees on what the problem is. This requires a statement of unanimous support for what our needs are.
    2. Everyone buys into one or two avenues or directions for how to solve the problem(s) mentioned in 1. This may entail some breakout sessions, to really get to what the architecture for a UAT would be, if that's what we want/need as a community.
    3. Solicit the help at the highest levels possible from members of the workshop to make the chances for success as great as possible.
    4. People should be excited by the prospects for what is being proposed.