Diogenes at 10 years: milestones and morals

Peter Heslin

Digital Classicist Seminar, London

22 August 2008

1998/9 – Pandora

- Pandora – I have never tried anything else.

1998/9 – Pandora

- Pandora – I have never tried anything else.
- Where's the disk? Slow. Doesn't work on Linux.

1998/9 – Pandora

- Pandora – I have never tried anything else.
- Where's the disk? Slow. Doesn't work on Linux.
- I copied the disks, used grep; it was fast – especially after the first go.

1998/9 – Pandora

- Pandora – I have never tried anything else.
- Where's the disk? Slow. Doesn't work on Linux.
- I copied the disks, used grep; it was fast – especially after the first go.
- Pandora is not foolproof – misses some words! – do people double-check?

1998/9 – Pandora

- Pandora – I have never tried anything else.
- Where's the disk? Slow. Doesn't work on Linux.
- I copied the disks, used grep; it was fast – especially after the first go.
- Pandora is not foolproof – misses some words! – do people double-check?
- Idea: Everyone will flock to a better tool!

1998/9 – Pandora

- Pandora – I have never tried anything else.
- Where's the disk? Slow. Doesn't work on Linux.
- I copied the disks, used grep; it was fast – especially after the first go.
- Pandora is not foolproof – misses some words! – do people double-check?
- Idea: Everyone will flock to a better tool!
- Moral: Classicists are conservative!

1998/9 – Pandora

- Pandora – I have never tried anything else.
- Where's the disk? Slow. Doesn't work on Linux.
- I copied the disks, used grep; it was fast – especially after the first go.
- Pandora is not foolproof – misses some words! – do people double-check?
- Idea: Everyone will flock to a better tool!
- Moral: Classicists are conservative!
- Moral: Visual markup is evil!

First steps

- A script: grep and a bit of scripting: a few lines of context and author.

First steps

- A script: grep and a bit of scripting: a few lines of context and author.
- Perl for poetry: crude work-title and book and line numbers.

First steps

- A script: grep and a bit of scripting: a few lines of context and author.
- Perl for poetry: crude work-title and book and line numbers.
- Rudimentary browser.

First steps

- A script: grep and a bit of scripting: a few lines of context and author.
- Perl for poetry: crude work-title and book and line numbers.
- Rudimentary browser.
- Added beginnings Greek output: Ibycus encoding for Latex.

First steps

- A script: grep and a bit of scripting: a few lines of context and author.
- Perl for poetry: crude work-title and book and line numbers.
- Rudimentary browser.
- Added beginnings Greek output: Ibycus encoding for Latex.
- Started to get curious about the citation info embedded in the text.

Reverse engineering

- Citation info is a state machine and encoded in binary to reduce size.

Reverse engineering

- Citation info is a state machine and encoded in binary to reduce size.
- Originally was decoded in hardware!

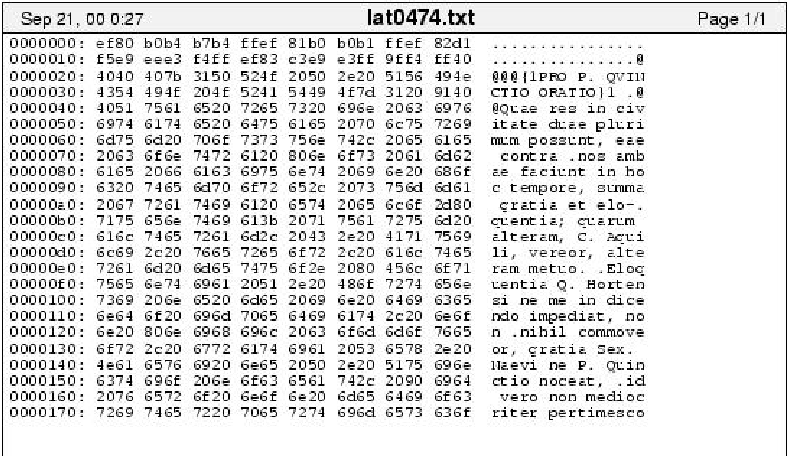
Reverse engineering

- Citation info is a state machine and encoded in binary to reduce size.
- Originally was decoded in hardware!
- Eureka – every 8KB chunk resets the state.

Reverse engineering

- Citation info is a state machine and encoded in binary to reduce size.
- Originally was decoded in hardware!
- Eureka – every 8KB chunk resets the state.
- Much richer information and suggested the possibility of robust decoding.

Reverse engineering



Sep 21, 00 0:27 **lat0474.txt** Page 1/1

```
0000000: ef80 b0b4 b7b4 ffef 81b0 b0b1 ffef 82d1  ................
0000010: f5e9 eee3 f4ff ef83 c3e9 e3ff 9ff4 ff40  ...............@
0000020: 4040 407b 3150 524f 2050 2e20 5156 494e  @@@{1PRO P. QVIN
0000030: 4354 494f 204f 5241 5449 4f7d 3120 9140  CTIO ORATIO}1 .@
0000040: 4051 7561 6520 7265 7320 696e 2063 6976  @Quae res in civ
0000050: 6974 6174 6520 6475 6165 2070 6c75 7269  itate duae pluri
0000060: 6d75 6d20 706f 7373 756e 742c 2065 6165  mum possunt, eae
0000070: 2063 6f6e 7472 6120 806e 6f73 2061 6d62   contra .nos amb
0000080: 6165 2066 6163 6975 6e74 2069 6e20 686f  ae faciunt in ho
0000090: 6320 7465 6d70 6f72 652c 2073 756d 6d61  c tempore, summa
00000a0: 2067 7261 7469 6120 6574 2065 6c6f 2d80   gratia et elo-.
00000b0: 7175 656e 7469 613b 2071 7561 7275 6d20  quentia; quarum
00000c0: 616c 7465 7261 6d2c 2043 2e20 4171 7569  alteram, C. Aqui
00000d0: 6c69 2c20 7665 7265 6f72 2c20 616c 7465  li, vereor, alte
00000e0: 7261 6d20 6d65 7475 6f2e 2080 456c 6f71  ram metuo. .Eloq
00000f0: 7565 6e74 6961 2051 2e20 486f 7274 656e  uentia Q. Horten
0000100: 7369 206e 6520 6d65 2069 6e20 6469 6365  si ne me in dice
0000110: 6e64 6f20 696d 7065 6469 6174 2c20 6e6f  ndo impediat, no
0000120: 6e20 806e 6968 696c 2063 6f6d 6d6f 7665  n .nihil commove
0000130: 6f72 2c20 6772 6174 6961 2053 6578 2e20  or, gratia Sex.
0000140: 4e61 6576 6920 6e65 2050 2e20 5175 696e  Naevi ne P. Quin
0000150: 6374 696f 206e 6f63 6561 742c 2090 6964  ctio noceat, .id
0000160: 2076 6572 6f20 6e6f 6e20 6d65 6469 6f63   vero non medioc
0000170: 7269 7465 7220 7065 7274 696d 6573 636f  riter pertimesco
```

Worse is better

- Citation decoding was bolted on top of the existing search functionality.

Worse is better

- Citation decoding was bolted on top of the existing search functionality.
- (Jumping ahead) Someone tried to do it the right way and he sent me the spec later on:

Worse is better

- Citation decoding was bolted on top of the existing search functionality.
- (Jumping ahead) Someone tried to do it the right way and he sent me the spec later on:

“Read in all bytes with the sign bit set. This is the ID for the first record. Call a subroutine to decode the ID data. Now read in all bytes with the sign bit unset. This is the text of the first record. Call a subroutine to process the text. Repeat this process for all records in the block, that is, until the ID data contains the end of block marker.”

Worse is better

- Citation decoding was bolted on top of the existing search functionality.
- (Jumping ahead) Someone tried to do it the right way and he sent me the spec later on:

“Read in all bytes with the sign bit set. This is the ID for the first record. Call a subroutine to decode the ID data. Now read in all bytes with the sign bit unset. This is the text of the first record. Call a subroutine to process the text. Repeat this process for all records in the block, that is, until the ID data contains the end of block marker.”

- This code was slow .....

Worse is better

- This is all wrong for a modern architecture.

Worse is better

- This is all wrong for a modern architecture.
- It may work when you are I/O bound, but not when you are CPU bound.

Worse is better

- This is all wrong for a modern architecture.
- It may work when you are I/O bound, but not when you are CPU bound.
- Driven by the citation info – this is why a spec should be independent from processing,

Worse is better

- This is all wrong for a modern architecture.
- It may work when you are I/O bound, but not when you are CPU bound.
- Driven by the citation info – this is why a spec should be independent from processing,
- Searching is the more general problem and has had more resources thrown at it: hardware (besides I/O, already at that time 32MB, the size of the largest text, could usually fit in memory) and software (Boyer-Moore).

Worse is better

- This is all wrong for a modern architecture.
- It may work when you are I/O bound, but not when you are CPU bound.
- Driven by the citation info – this is why a spec should be independent from processing,
- Searching is the more general problem and has had more resources thrown at it: hardware (besides I/O, already at that time 32MB, the size of the largest text, could usually fit in memory) and software (Boyer-Moore).
- I think other software may still do it the wrong way.

Worse is better

- This is all wrong for a modern architecture.
- It may work when you are I/O bound, but not when you are CPU bound.
- Driven by the citation info – this is why a spec should be independent from processing,
- Searching is the more general problem and has had more resources thrown at it: hardware (besides I/O, already at that time 32MB, the size of the largest text, could usually fit in memory) and software (Boyer-Moore).
- I think other software may still do it the wrong way.
- Moral: when the corpus is closed, you do better to ignore the spec.

TLG features

- I found some orphan code (C for MSDOS) on the Internet. I was able to figure out odd TLG features from it (which I probably wouldn't have bothered about myself).

TLG features

- I found some orphan code (C for MSDOS) on the Internet. I was able to figure out odd TLG features from it (which I probably wouldn't have bothered about myself).
- Word list Should have ignored it or built my own. I assumed that conforming with the TLG would make my code more robust, but I just demonstrated how badly the TLG dealt with odd cases, then I had to try for bug-for-bug compatibility. This code is a nightmare and I don't understand it anymore.

TLG features

- I found some orphan code (C for MSDOS) on the Internet. I was able to figure out odd TLG features from it (which I probably wouldn't have bothered about myself).
- Word list Should have ignored it or built my own. I assumed that conforming with the TLG would make my code more robust, but I just demonstrated how badly the TLG dealt with odd cases, then I had to try for bug-for-bug compatibility. This code is a nightmare and I don't understand it anymore.
- Time, genre, etc. info. Probably worth doing, but some more useful than others.

TLG features

- I found some orphan code (C for MSDOS) on the Internet. I was able to figure out odd TLG features from it (which I probably wouldn't have bothered about myself).
- Word list Should have ignored it or built my own. I assumed that conforming with the TLG would make my code more robust, but I just demonstrated how badly the TLG dealt with odd cases, then I had to try for bug-for-bug compatibility. This code is a nightmare and I don't understand it anymore.
- Time, genre, etc. info. Probably worth doing, but some more useful than others.
- Moral: don't assume that you need to support all features in the spec – some are obsolete (or nearly so) and speed-up hacks.

The Web! (Sort-of)

- Proabably not going to fly as a command-line tool.

The Web! (Sort-of)

- Proabably not going to fly as a command-line tool.
- First refactoring: module and command-line script separated.

The Web! (Sort-of)

- Proabably not going to fly as a command-line tool.
- First refactoring: module and command-line script separated.
- CGI script. Few browsers supported Unicode adequately if at all and all HTML layout was done using tables (which rendered after the page loaded), so the main method of displaying Greek was as GIFs generated on the fly from LaTeX via PostScript. Probably nobody but me actually understood how to use this.

The Web! (Sort-of)

- Proabably not going to fly as a command-line tool.
- First refactoring: module and command-line script separated.
- CGI script. Few browsers supported Unicode adequately if at all and all HTML layout was done using tables (which rendered after the page loaded), so the main method of displaying Greek was as GIFs generated on the fly from LaTeX via PostScript. Probably nobody but me actually understood how to use this.
- LaTeX output: another bit of code I don't understand anymore.

The Web! (Sort-of)

- Proabably not going to fly as a command-line tool.
- First refactoring: module and command-line script separated.
- CGI script. Few browsers supported Unicode adequately if at all and all HTML layout was done using tables (which rendered after the page loaded), so the main method of displaying Greek was as GIFs generated on the fly from LaTeX via PostScript. Probably nobody but me actually understood how to use this.
- LaTeX output: another bit of code I don't understand anymore.
- First public distribution.

The Web! (Sort-of)

- Proabably not going to fly as a command-line tool.
- First refactoring: module and command-line script separated.
- CGI script. Few browsers supported Unicode adequately if at all and all HTML layout was done using tables (which rendered after the page loaded), so the main method of displaying Greek was as GIFs generated on the fly from LaTeX via PostScript. Probably nobody but me actually understood how to use this.
- LaTeX output: another bit of code I don't understand anymore.
- First public distribution.
- Moral: Most CLassics dept. do not have good tech. support.

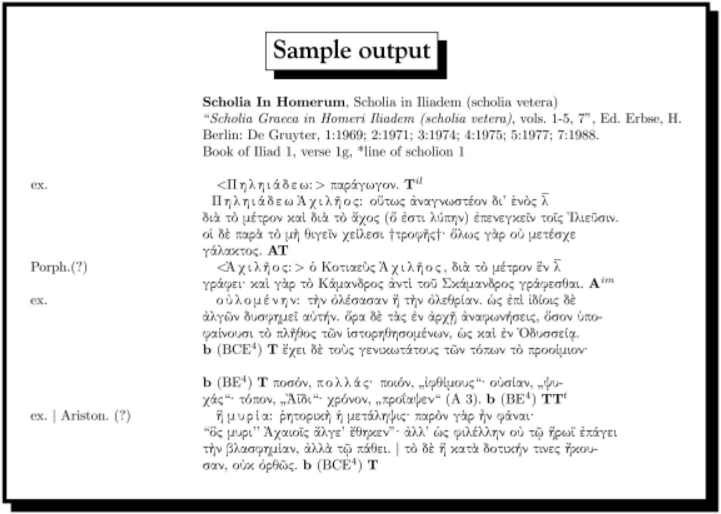
The Web! (Sort-of)

- Proabably not going to fly as a command-line tool.
- First refactoring: module and command-line script separated.
- CGI script. Few browsers supported Unicode adequately if at all and all HTML layout was done using tables (which rendered after the page loaded), so the main method of displaying Greek was as GIFs generated on the fly from LaTeX via PostScript. Probably nobody but me actually understood how to use this.
- LaTeX output: another bit of code I don't understand anymore.
- First public distribution.
- Moral: Most CLassics dept. do not have good tech. support.
- Moral: Do not assume people will install other software.

The Web! (Sort-of)

- Proabably not going to fly as a command-line tool.
- First refactoring: module and command-line script separated.
- CGI script. Few browsers supported Unicode adequately if at all and all HTML layout was done using tables (which rendered after the page loaded), so the main method of displaying Greek was as GIFs generated on the fly from LaTeX via PostScript. Probably nobody but me actually understood how to use this.
- LaTeX output: another bit of code I don't understand anymore.
- First public distribution.
- Moral: Most CLassics dept. do not have good tech. support.
- Moral: Do not assume people will install other software.
- Moral: Web was a good choice of interface.

The Web! (Sort-of)



Sample output

Scholia In Homerum, Scholia in Iliadem (scholia vetera)
"*Scholia Graeca in Homeri Iliadem (scholia vetera)*, vols. 1-5, 7", Ed. Erbse, H.
Berlin: De Gruyter, 1:1969; 2:1971; 3:1974; 4:1975; 5:1977; 7:1988.
Book of Iliad 1, verse 1g, *line of scholion 1

ex. <Πηληιάδεω:> παράγωγον. **T**^{il}
Πηληιάδεω Ἀχιλῆος: οὕτως ἀναγνωστέον δι' ἑνὸς λ̄
διὰ τὸ μέτρον καὶ διὰ τὸ ἄχος (ὅ ἐστι λύπην) ἐπενεγκεῖν τοῖς Ἰλιεῦσιν.
οἱ δὲ παρὰ τὸ μὴ θιγεῖν χείλεσι †τροφῆς†· ὅλως γὰρ οὐ μετέσχε
γάλακτος. **AT**
Porph.(?) <Ἀχιλῆος:> ὁ Κοτιαεὺς Ἀχιλῆος, διὰ τὸ μέτρον ἓν λ̄
γράφει· καὶ γὰρ τὸ Κάμανδρος ἀντὶ τοῦ Σκάμανδρος γράφεσθαι. **A**^{im}
ex. οὐλομένην: τὴν ὀλέσασαν ἢ τὴν ὀλεθρίαν. ὡς ἐπὶ ἰδίοις δὲ
ἀλγῶν δυσφημεῖ αὐτήν. ὅρα δὲ τὰς ἐν ἀρχῇ ἀναφωνήσεις, ὅσον ὑποφαίνουσι
τὸ πλῆθος τῶν ἱστορηθησομένων, ὡς καὶ ἐν Ὀδυσσείᾳ.
b (BCE^4) **T** ἔχει δὲ τοὺς γενικωτάτους τῶν τόπων τὸ προοίμιον·

b (BE^4) **T** ποσόν, πολλάς· ποιόν, „ἰφθίμους“· οὐσίαν, „ψυχάς“·
τόπον, „Ἄϊδι“· χρόνον, „προΐαψεν“ (A 3). **b** (BE^4) **TT**^{t}
ex. | Ariston. (?) ἣ μυρία: ῥητορικὴ ἡ μετάληψις· παρὸν γὰρ ἦν φάναι·
"ὃς μυρι'' Ἀχαιοῖς ἄλγε' ἔθηκεν"· ἀλλ' ὡς φιλέλλην οὐ τῷ ἥρωϊ ἐπάγει
τὴν βλασφημίαν, ἀλλὰ τῷ πάθει. | τὸ δὲ ἣ κατὰ δοτικὴν τινες ἤκουσαν,
οὐκ ὀρθῶς. **b** (BCE^4) **T**

2000 – The Web! (Really)

- Added other encodings, including Unicode – why not?

2000 – The Web! (Really)

- Added other encodings, including Unicode – why not?
- Actually a bad idea. Should have gone with Unicode and binned everything else. An all-purpose Greek encoding converter should have been a separate application.

2000 – The Web! (Really)

- Added other encodings, including Unicode – why not?
- Actually a bad idea. Should have gone with Unicode and binned everything else. An all-purpose Greek encoding converter should have been a separate application.
- Just HTML and Unicode: only two things to interfere with each other.

2000 – The Web! (Really)

- Added other encodings, including Unicode – why not?
- Actually a bad idea. Should have gone with Unicode and binned everything else. An all-purpose Greek encoding converter should have been a separate application.
- Just HTML and Unicode: only two things to interfere with each other.
- Added HTML output. CSS still was not there yet in browsers, so I ignored it.

2000 – The Web! (Really)

- Added other encodings, including Unicode – why not?
- Actually a bad idea. Should have gone with Unicode and binned everything else. An all-purpose Greek encoding converter should have been a separate application.
- Just HTML and Unicode: only two things to interfere with each other.
- Added HTML output. CSS still was not there yet in browsers, so I ignored it.
- Moral: embrace emerging standards even if they are unsupported.

2000 – The Web! (Really)

- Added other encodings, including Unicode – why not?
- Actually a bad idea. Should have gone with Unicode and binned everything else. An all-purpose Greek encoding converter should have been a separate application.
- Just HTML and Unicode: only two things to interfere with each other.
- Added HTML output. CSS still was not there yet in browsers, so I ignored it.
- Moral: embrace emerging standards even if they are unsupported.
- Moral: an application should do one thing well.

New functionality

- Multiple patterns and subsets of texts for searching.

New functionality

- Multiple patterns and subsets of texts for searching.
- Support for DDP disk – popular demand! Still buggy?

New functionality

- Multiple patterns and subsets of texts for searching.
- Support for DDP disk – popular demand! Still buggy?
- Second refactoring: object-oriented module, documentation.

New functionality

- Multiple patterns and subsets of texts for searching.
- Support for DDP disk – popular demand! Still buggy?
- Second refactoring: object-oriented module, documentation.
- Moral: adding core functionality is good, and so is documentation.

2001/2 – Documentation

- It was at this point that I was sent documentation on binary citation info.

2001/2 – Documentation

- It was at this point that I was sent documentation on binary citation info.
- I fixed the browser to take advantage of some extra speed-up indexing which actually slowed it down.

2001/2 – Documentation

- It was at this point that I was sent documentation on binary citation info.
- I fixed the browser to take advantage of some extra speed-up indexing which actually slowed it down.
- I re-wrote the stopping code in the browser, which made it marginally more correct and substantially slower.

2001/2 – Documentation

- It was at this point that I was sent documentation on binary citation info.
- I fixed the browser to take advantage of some extra speed-up indexing which actually slowed it down.
- I re-wrote the stopping code in the browser, which made it marginally more correct and substantially slower.
- Moral: someone else's optimization might not be your own.

2001/2 – Documentation

- It was at this point that I was sent documentation on binary citation info.
- I fixed the browser to take advantage of some extra speed-up indexing which actually slowed it down.
- I re-wrote the stopping code in the browser, which made it marginally more correct and substantially slower.
- Moral: someone else's optimization might not be your own.
- Moral: it's better to be fast and 98% correct than slow and 99% correct.

New features

- Links to Perseus morphological tools.

New features

- Links to Perseus morphological tools.
- Coptic support.

New features

- Links to Perseus morphological tools.
- Coptic support.
- Added a dedicated HTTP server. initially to allow running on Windows, but was a faster and cleaner solution all around.

2002 – Version 1.0

- Basic features complete, but ...

2002 – Version 1.0

- Basic features complete, but ...
- Hard to install.

2005 – OS X

- Increasing number of non-technical OS X users.

2005 – OS X

- Increasing number of non-technical OS X users.
- Added CPAN directory. Ease of installation and protecting against 3rd-party changes.

2005 – OS X

- Increasing number of non-technical OS X users.
- Added CPAN directory. Ease of installation and protecting against 3rd-party changes.
- First step in that direction. Moving away from a Unix mentality.

2005 – OS X

- Increasing number of non-technical OS X users.
- Added CPAN directory. Ease of installation and protecting against 3rd-party changes.
- First step in that direction. Moving away from a Unix mentality.
- Still, many users could not fathom starting the server then the client.

2005 – OS X

- Increasing number of non-technical OS X users.
- Added CPAN directory. Ease of installation and protecting against 3rd-party changes.
- First step in that direction. Moving away from a Unix mentality.
- Still, many users could not fathom starting the server then the client.
- Moral: for non-technical users a big, simple bundle is better than having to install many tools.

2006 – Version 2.0

- Big attempt to make things simpler.

2006 – Version 2.0

- Big attempt to make things simpler.
- Idea was right: to have client and server in the same package.

2006 – Version 2.0

- Big attempt to make things simpler.
- Idea was right: to have client and server in the same package.
- Browser in Objective-C using the WebKit framework in OS X, and in Visual Basic, using the .NET framework on Windows.

2006 – Version 2.0

- Big attempt to make things simpler.
- Idea was right: to have client and server in the same package.
- Browser in Objective-C using the WebKit framework in OS X, and in Visual Basic, using the .NET framework on Windows.
- A nightmare: buggy and unmaintainable.

2006 – Version 2.0

- Big attempt to make things simpler.
- Idea was right: to have client and server in the same package.
- Browser in Objective-C using the WebKit framework in OS X, and in Visual Basic, using the .NET framework on Windows.
- A nightmare: buggy and unmaintainable.
- Never released into the wild – too hard to cut a release.

2006 – Version 2.0

- Big attempt to make things simpler.
- Idea was right: to have client and server in the same package.
- Browser in Objective-C using the WebKit framework in OS X, and in Visual Basic, using the .NET framework on Windows.
- A nightmare: buggy and unmaintainable.
- Never released into the wild – too hard to cut a release.
- Moral: lots of platform-specific code is not the same as cross-platform code.

Version 2.1

- More code refactoring: sub-modules.

Version 2.1

- More code refactoring: sub-modules.
- CGI re-written: was spaghetti, removed on-the-fly LaTeX generation.

Version 2.1

- More code refactoring: sub-modules.
- CGI re-written: was spaghetti, removed on-the-fly LaTeX generation.
- New features: marginal line numbering, user-defined corpora.

Version 2.1

- More code refactoring: sub-modules.
- CGI re-written: was spaghetti, removed on-the-fly LaTeX generation.
- New features: marginal line numbering, user-defined corpora.
- Never released: release process too complex.

Version 2.1

- More code refactoring: sub-modules.
- CGI re-written: was spaghetti, removed on-the-fly LaTeX generation.
- New features: marginal line numbering, user-defined corpora.
- Never released: release process too complex.
- Moral: Unless you have slaves, make sure you can cut a release on one machine.

2007 – Version 3.0

- New platform: Mozilla.

2007 – Version 3.0

- New platform: Mozilla.
- Custom browser (XUL and Javascript), starts server automatically.

2007 – Version 3.0

- New platform: Mozilla.
- Custom browser (XUL and Javascript), starts server automatically.
- All binaries are downloaded from mozilla.org.

2007 – Version 3.0

- New platform: Mozilla.
- Custom browser (XUL and Javascript), starts server automatically.
- All binaries are downloaded from mozilla.org.
- Then my code is added and it is repackaged.

2007 – Version 3.0

- New platform: Mozilla.
- Custom browser (XUL and Javascript), starts server automatically.
- All binaries are downloaded from mozilla.org.
- Then my code is added and it is repackaged.
- Releases for all 3 platforms done via one script on my Linux box.

2007 – Version 3.0

- New platform: Mozilla.
- Custom browser (XUL and Javascript), starts server automatically.
- All binaries are downloaded from mozilla.org.
- Then my code is added and it is repackaged.
- Releases for all 3 platforms done via one script on my Linux box.
- Moral: let someone else do the porting work!

Version 3.1

- Perseus morphological data and dictionaries.

Version 3.1

- Perseus morphological data and dictionaries.
- No Perseus code – extensive preprocessing of data.

Version 3.1

- Perseus morphological data and dictionaries.
- No Perseus code – extensive preprocessing of data.
- Big download, but who cares?

Version 3.1

- Perseus morphological data and dictionaries.
- No Perseus code – extensive preprocessing of data.
- Big download, but who cares?
- Web 2.0 features.

Version 3.1

- Perseus morphological data and dictionaries.
- No Perseus code – extensive preprocessing of data.
- Big download, but who cares?
- Web 2.0 features.
- Unicode input.

Version 3.1

- Perseus morphological data and dictionaries.
- No Perseus code – extensive preprocessing of data.
- Big download, but who cares?
- Web 2.0 features.
- Unicode input.
- Moral: it's easier to distribute data than binaries.

Present

- Back-burner – fixing major bugs.

Present

- Back-burner – fixing major bugs.
- Collecting feature requests.

Present

- Back-burner – fixing major bugs.
- Collecting feature requests.
- New features were not allowed to be submitted for the RAE.

Present

- Back-burner – fixing major bugs.
- Collecting feature requests.
- New features were not allowed to be submitted for the RAE.
- Moral: do fig-leaf publications and get grants.

Future – Problems

- General public does not have the CD-ROMS.

Future – Problems

- General public does not have the CD-ROMS.
- Perhaps they will disappear.

Future – Problems

- General public does not have the CD-ROMS.
- Perhaps they will disappear.
- Can't expand the corpus.

Future – Solutions

- Output TEI-conformant XML.

Future – Solutions

- Output TEI-conformant XML.
- Have Diogenes spit out XML and have client convert to XHTML via XSLT on the fly?

Future – Solutions

- Output TEI-conformant XML.
- Have Diogenes spit out XML and have client convert to XHTML via XSLT on the fly?
- Or make Diogenes convert in bulk and then write a new app. to work on the XML?

Future – Solutions

- Output TEI-conformant XML.
- Have Diogenes spit out XML and have client convert to XHTML via XSLT on the fly?
- Or make Diogenes convert in bulk and then write a new app. to work on the XML?
- Perseus stuff is easy to port to a new application.

Future – Solutions

- Output TEI-conformant XML.
- Have Diogenes spit out XML and have client convert to XHTML via XSLT on the fly?
- Or make Diogenes convert in bulk and then write a new app. to work on the XML?
- Perseus stuff is easy to port to a new application.
- PhiloLogic?

Future – Solutions

- Output TEI-conformant XML.
- Have Diogenes spit out XML and have client convert to XHTML via XSLT on the fly?
- Or make Diogenes convert in bulk and then write a new app. to work on the XML?
- Perseus stuff is easy to port to a new application.
- PhiloLogic?
- Diogenes has the advantage of a nice user interface and easy installation.

Future – Solutions

- Output TEI-conformant XML.
- Have Diogenes spit out XML and have client convert to XHTML via XSLT on the fly?
- Or make Diogenes convert in bulk and then write a new app. to work on the XML?
- Perseus stuff is easy to port to a new application.
- PhiloLogic?
- Diogenes has the advantage of a nice user interface and easy installation.
- Licensing issues?