Managing Descriptive Metadata with Open XML

Gregory Wiedeman
University Archivist
University at Albany, SUNY
GWiedeman@albany.edu
@GregWiedeman

Why not ArchivesSpace?

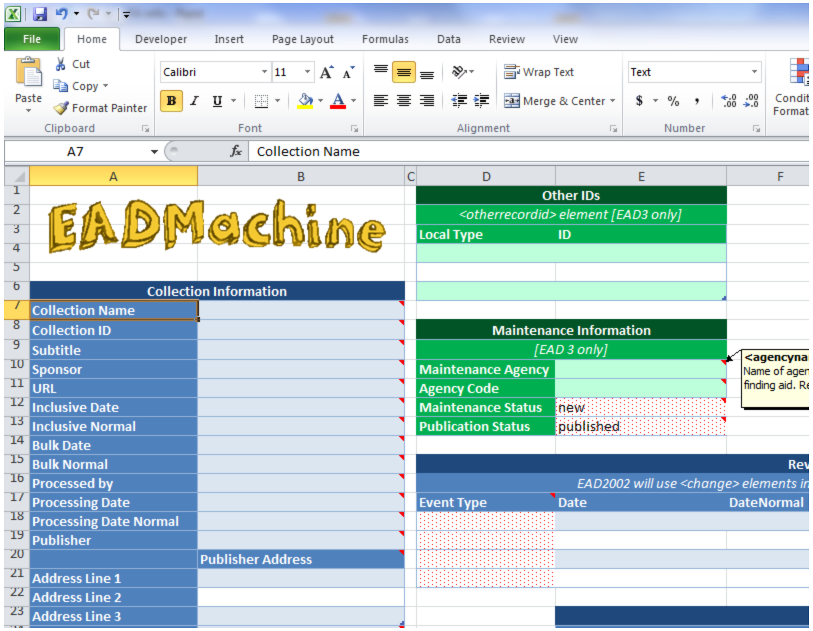
- Legacy unstructured HTML finding aids
- Finishing large EAD conversion project
- Challenging migration of local accession database
- Costly: disproportionate membership fee
  - Little public documentation for automation
- Costly: metadata normalization
- No ArchiveSpace, yet…

Opportunity

- Develop basic metadata infrastructure first, implement more complex tools second
- Modularize metadata management
  - adapt to constant change in tools
- Control over exactly how strict to make metadata controls in the immediate term
- Yet had to address problems developing systems with open XML
  - inadequate data controls

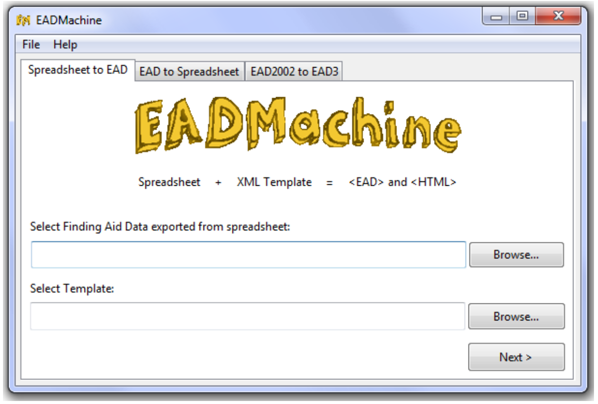
Consistent Creation: EADMachine

- Converts between Excel spreadsheet and complete EAD
- Creates flat HTML access file
- Written in Python, complied to C, runs on any machine without dependencies
- Matches local EAD implementation
- Basic GUI interface
- Works with complex hierarchies up to <c12> (not recommended)
- Compatible with EAD2002 and EAD3



A7 | Collection Name

	A	B	C	D	E	F
1				Other IDs		
2				<otherrecordid> element [EAD3 only]		
3				Local Type	ID	
6	Collection Information					
7	Collection Name					
8	Collection ID			Maintenance Information		
9	Subtitle			[EAD 3 only]		<agencyna Name of ager finding aid. Re
10	Sponsor			Maintenance Agency		
11	URL			Agency Code		
12	Inclusive Date			Maintenance Status	new	
13	Inclusive Normal			Publication Status	published	
14	Bulk Date					
15	Bulk Normal					Rev
16	Processed by			EAD2002 will use <change> elements i		
17	Processing Date			Event Type	Date	DateNormal
18	Processing Date Normal					
19	Publisher					
20		Publisher Address				
21	Address Line 1					
22	Address Line 2					
23	Address Line 3					



Spreadsheet + XML Template = <EAD> and <HTML>

Select Finding Aid Data exported from spreadsheet:

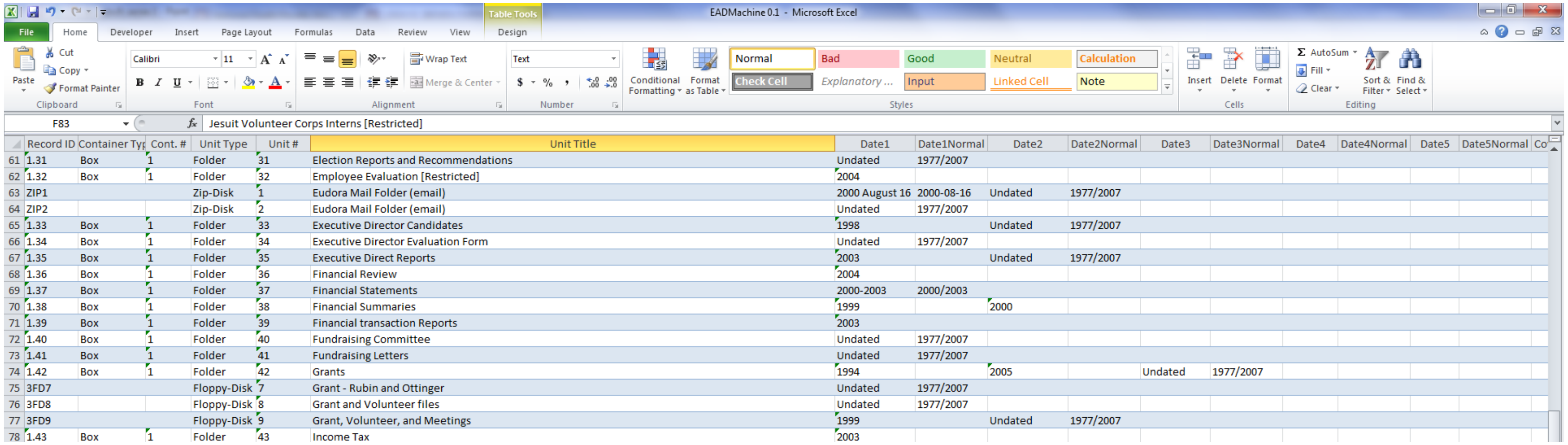
Select Template:

https://github.com/gwiedeman/eadmachine

Consistent Creation: EADMachine

Successes and difficulties

- First large-scale project, lots of bad code
- Long time to develop
- Very easy to implement and use in our specific environment
- Creates standardized EAD



F83 Jesuit Volunteer Corps Interns [Restricted]

	Record ID	Container Typ	Cont. #	Unit Type	Unit #	Unit Title	Date1	Date1Normal	Date2	Date2Normal	Date3	Date3Normal	Date4	Date4Normal	Date5	Date5Normal	Co
61	1.31	Box	1	Folder	31	Election Reports and Recommendations	Undated	1977/2007									
62	1.32	Box	1	Folder	32	Employee Evaluation [Restricted]	2004										
63	ZIP1			Zip-Disk	1	Eudora Mail Folder (email)	2000 August 16	2000-08-16	Undated	1977/2007							
64	ZIP2			Zip-Disk	2	Eudora Mail Folder (email)	Undated	1977/2007									
65	1.33	Box	1	Folder	33	Executive Director Candidates	1998		Undated	1977/2007							
66	1.34	Box	1	Folder	34	Executive Director Evaluation Form	Undated	1977/2007									
67	1.35	Box	1	Folder	35	Executive Direct Reports	2003		Undated	1977/2007							
68	1.36	Box	1	Folder	36	Financial Review	2004										
69	1.37	Box	1	Folder	37	Financial Statements	2000-2003	2000/2003									
70	1.38	Box	1	Folder	38	Financial Summaries	1999		2000								
71	1.39	Box	1	Folder	39	Financial transaction Reports	2003										
72	1.40	Box	1	Folder	40	Fundraising Committee	Undated	1977/2007									
73	1.41	Box	1	Folder	41	Fundraising Letters	Undated	1977/2007									
74	1.42	Box	1	Folder	42	Grants	1994		2005		Undated	1977/2007					
75	3FD7			Floppy-Disk	7	Grant - Rubin and Ottinger	Undated	1977/2007									
76	3FD8			Floppy-Disk	8	Grant and Volunteer files	Undated	1977/2007									
77	3FD9			Floppy-Disk	9	Grant, Volunteer, and Meetings	1999		Undated	1977/2007							
78	1.43	Box	1	Folder	43	Income Tax	2003										

https://github.com/gwiedeman/eadmachine

Strict Control: EADValidator

- Python rule-based validation tool
- .EXE file reads all EAD XML files in directory and produces Bootstrap HTML report
- Architecture designed also for automated processes
- Mandates many DACS rules
- 300+ Detailed Rules:
  - 183 at collection-level
  - 34 at series-level
  - 47 at file-level
  - 25 at item-level
  - 12 for each @normal date
- Does one thing, easy to develop, ~20 hours
- Not all data is standardized but have a documented set of what is standardized

https://github.com/UAlbanyArchives/EADValidator

Strict Control: EADValidator

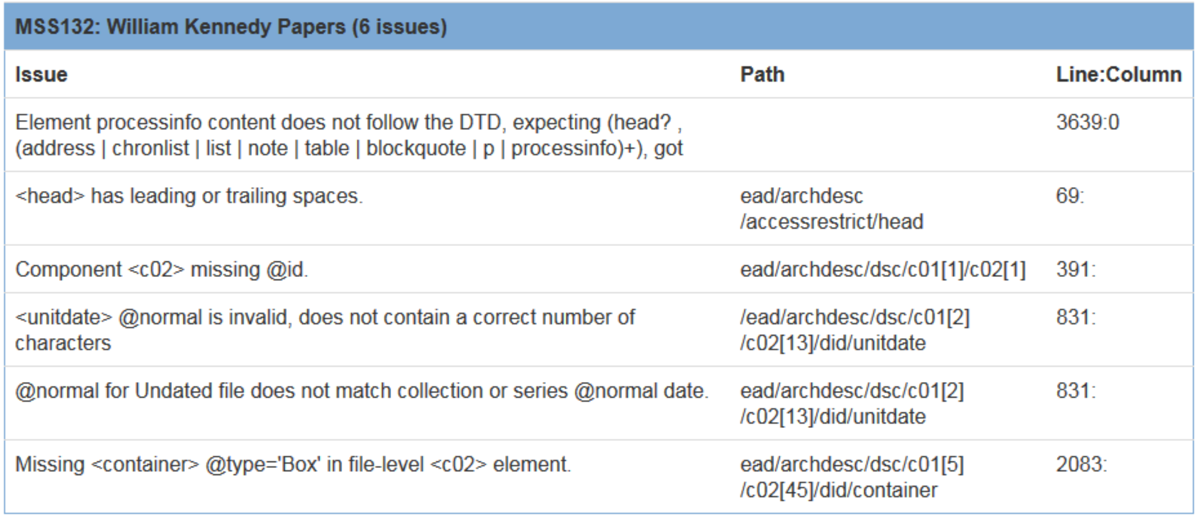
Legacy <physdesc>

- <extent> is controlled

  <extent @unit=”cubic ft.”>23.5</extent>

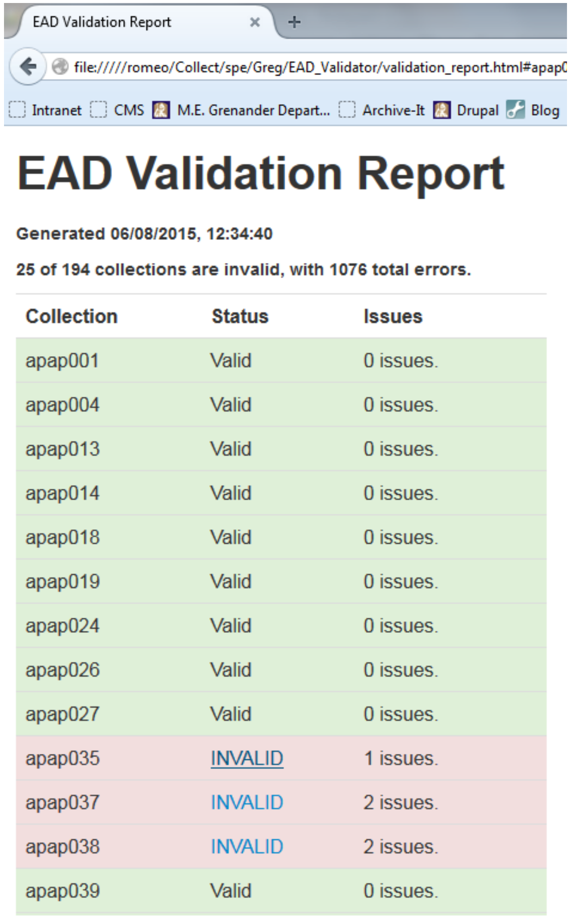
- <physfacet> is uncontrolled

  <physfacet>29 folders and 1 giraffe</physfacet>



MSS132: William Kennedy Papers (6 issues)

Issue	Path	Line:Column
Element processinfo content does not follow the DTD, expecting (head? , (address | chronlist | list | note | table | blockquote | p | processinfo)+), got		3639:0
<head> has leading or trailing spaces.	ead/archdesc /accessrestrict/head	69:
Component <c02> missing @id.	ead/archdesc/dsc/c01[1]/c02[1]	391:
<unitdate> @normal is invalid, does not contain a correct number of characters	/ead/archdesc/dsc/c01[2] /c02[13]/did/unitdate	831:
@normal for Undated file does not match collection or series @normal date.	ead/archdesc/dsc/c01[2] /c02[13]/did/unitdate	831:
Missing <container> @type='Box' in file-level <c02> element.	ead/archdesc/dsc/c01[5] /c02[45]/did/container	2083:



EAD Validation Report

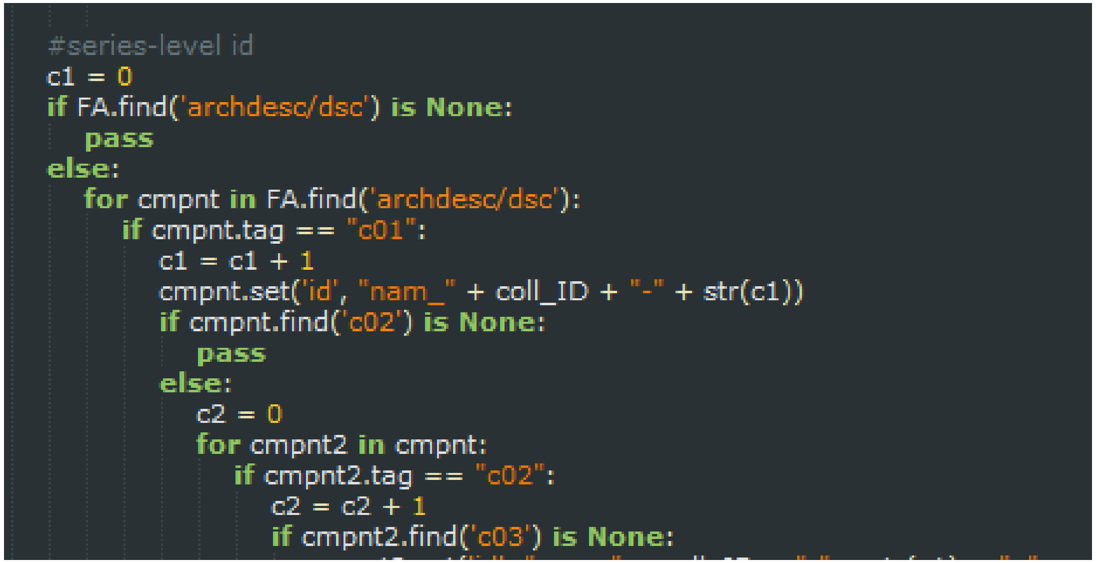
Generated 06/08/2015, 12:34:40

25 of 194 collections are invalid, with 1076 total errors.

Collection	Status	Issues
apap001	Valid	0 issues.
apap004	Valid	0 issues.
apap013	Valid	0 issues.
apap014	Valid	0 issues.
apap018	Valid	0 issues.
apap019	Valid	0 issues.
apap024	Valid	0 issues.
apap026	Valid	0 issues.
apap027	Valid	0 issues.
apap035	INVALID	1 issues.
apap037	INVALID	2 issues.
apap038	INVALID	2 issues.
apap039	Valid	0 issues.

Unique Identification

- Simple script to insert ids based on collection ids and context in hierarchy
  - independent of containers
  - nam_ua629-1_132
  - nam_apap101-1.2_49



```
#series-level id
c1 = 0
if FA.find('archdesc/dsc') is None:
    pass
else:
    for cmpnt in FA.find('archdesc/dsc'):
        if cmpnt.tag == "c01":
            c1 = c1 + 1
            cmpnt.set('id', "nam_" + coll_ID + "-" + str(c1))
            if cmpnt.find('c02') is None:
                pass
            else:
                c2 = 0
                for cmpnt2 in cmpnt:
                    if cmpnt2.tag == "c02":
                        c2 = c2 + 1
                        if cmpnt2.find('c03') is None:
```

Automated Records: AutoUpload

AutoUpload.py

- Automatically uploads PDF scans based on ID in filename
- Archivists reviews scans for restrictions, etc. and copies to upload folder
- Automatically updates EAD

https://github.com/UAlbanyArchives/AutoUpload

1. Detects new file
2. Creates log
3. Logs original finding aid
4. Bags preservation copy
5. Uploads access copy
6. Copies finding aid to working directory
7. Inserts <dao>
8. Logs both original and modified record
9. Validates finding aid
10. Writes finding aid
11. converts to HTML
12. Any errors freezes process, dumps to error folder, sends email

Automated Records: AutoUpload

AutoUpload.py

- Enables mass digitization based on use
- Simple to initially develop, 20-25 hours, more time for testing
- Further potential
  - Automated requests from finding aids
  - Automated post to twitter?

https://github.com/UAlbanyArchives/AutoUpload

Metadata Infrastructure

- Modular system based on simple functional needs
- Strict controls enable automation
- Can later implement larger tools
  - New access system in development
  - Need to adopt preservation system, new accession system.
  - Can easily adapt to automated description of born-digital records

Gregory Wiedeman
University Archivist
University at Albany, SUNY
Gwiedeman@albany.edu

@GregWiedeman
https://github.com/gwiedeman
https://github.com/UAlbanyArchives