A Scriptable, Statistical Oracle for a Metadata Extraction System

Kurt J. Maly   Steven J. Zeil   Mohammad Zubair
Ashraf Amrou   Ali Aazhar   Naveen Ratkal
Old Dominion University
Dept. of Computer Science
Norfolk, VA 23529
dtic@cs.odu.edu

Abstract

An oracle is described for dynamic validation of an application (metadata extraction from scanned documents) where a moderate failure rate is acceptable provided that instances of failures during operation can be identified. The oracle combines a variety of deterministic tests and statistical tests based upon characteristics of the document collection on which the system operates. Because this system must adapt to a variety of document collections with different characteristics, a scripting language is developed that binds combinations of tests to the metadata fields expected in a given document collection. The suitability of the oracle is demonstrated by an experiment measuring its ability to mimic human judgments as to which of several alternate outputs for the same document would be preferred.

1. Introduction

The authors have been involved in the design and development of a system to extract metadata information from scanned documents stored in government repositories including collections at DTIC [4], NASA [9], and the GPO [13]. These collections are large (e.g., the DTIC collection contains more than one million documents and adds tens of thousands of new documents each year. The documents are diverse, including scientific articles, slides from presentations, PhD theses, (entire) conference proceedings, promotional brochures, public laws, and acts of Congress. Contributions to each collection come from a wide variety of organizations, each with their own in-house standards for layout and format, so, even among documents of similar kind, the layouts vary widely. The amount of metadata available may vary considerably as well. Many documents have only "conventional" metadata such as title, author names, publishing organization, and date. Others have more specialized metadata including security/release restrictions and waivers, public law numbers, or keywords and index terms. Different collections may also target different metadata fields for storage in their databases.

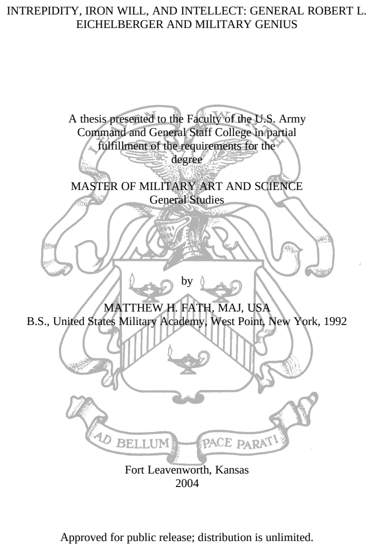
The heterogeneity of these collections poses a challenge to the development of an automated system for metadata extraction. Our approach has been based upon a two-part process [11, 12], in which

- A new document is *classified*, assigning it to a group of documents of similar layout. The goal is to group together documents whose title or other metadata-containing pages would appear similar when viewed (by humans) from several feet away.
- Associated with each class of document layouts is a *template*, a scripted description of how to associate blocks of text in the layout with metadata fields. For example, a template might state that the text set in the largest type font in the top-half of the first page is, in that layout, the document title.

Input to the system consists of scanned documents (e.g., Figure 1) that have been passed though a commercial Optical Character Recognition (OCR) program to produce an XML version of the document text marked with formatting information. The XML version is classified and a template selected. An extraction engine interprets the template, following its scripted instructions to actually locate and extract text strings that make up the values for various metadata fields. Figure 2 shows an output from that extraction engine (for the document from Figure 1).

There are a number of potential sources of error in the system, not all of which can be directly controlled:

- OCR errors, either due to failures of the OCR software or poor quality of the scanned document images,
- Classification errors in which an incorrect template is selected for application to the document,
- Template errors, giving an incorrect description of the location of the desired metadata, and



INTREPIDITY, IRON WILL, AND INTELLECT: GENERAL ROBERT L.
EICHELBERGER AND MILITARY GENIUS

A thesis presented to the Faculty of the U.S. Army
Command and General Staff College in partial
fulfillment of the requirements for the
degree

MASTER OF MILITARY ART AND SCIENCE
General Studies

by

MATTHEW H. FATH, MAJ, USA
B.S., United States Military Academy, West Point, New York, 1992

Fort Leavenworth, Kansas
2004

Approved for public release; distribution is unlimited.

Figure 1. Title Page from a Sample Input Document

```
<?xml version="1.0"?>
<metadata>
  <UnclassifiedTitle>Thesis Title:
    Intrepidity, Iron Will, and
    Intellect: General Robert L.
    Eichelberger and Military Genius
  </UnclassifiedTitle>
  <PersonalAuthor>
    Name of Candidate: Major Matthew H. Fath
  </PersonalAuthor>
  <ReportDate>
    Accepted this 18th day of June 2004 by:
  </ReportDate>
  <approvedby>Approved by:
  Thesis Committee Chair Jack D. Kem, Ph.D.,
  Member Mr. Charles S. Soby, M.B.A. ,
  Member Lieutenant Colonel John A. Suprin,
  M.A.
  </approvedby>
  <acceptedby>
     Robert F. Baumann, Ph.D.
  </acceptedby>
</metadata>
```

Figure 2. Sample Metadata Record (including mistakes)

- Internal faults in the extraction engine that interprets the templates.

The automated extractor is intended to replace a labor-intensive human process. As such, it could be economically viable even with failure rates on the order of 20-30% of documents having incorrectly extracted fields, *provided that* these incorrect outputs can be identified during program operation and referred for human corrective action.

There is a compelling need, then, for *dynamic validation* to identify unacceptable outputs, which leads directly to the question of how to devise an automated oracle that serves as the heart of this validator.

2. Related Research

The problem of selecting an appropriate oracle for determining the correctness of a program output is well known in testing literature and practice. Testing oracles have traditionally run the gamut from human inspection of printed outputs to automated checks for conformance with a formal statement of expected values.

Oracles for programs that, like our metadata extractor, attempt to mimic a human activity are notoriously problematic [14]. In meta-data extraction, for example, even expert humans may differ on what is the best description of a given document. Typically, such applications are validated by allowing human experts to rate outputs as reasonable or unreasonable, e.g., [6]. Such an oracle makes sense only prior to software deployment, however. It would be self-defeating to require continuous human inspection of outputs during dynamic validation of a deployed system.

The terms *dynamic verification* and *dynamic validation* have been coined to describe the provision of run-time checks as part of the running software to monitor the correctness of a calculation [2, 5]. The most common approach to dynamic verification and/or validation is based upon programmed assertions [3, 10, 15], which can provide valuable checks on the internal structure and consistency of a computation. Assertion checking is not a viable option for the metadata extractor, partly because of the lack of a precise specification of desired behavior and also because it is questionable whether internal computation state can be useful in contexts where the selection of the appropriate computation (template) is one of the primary sources of uncertainty.

In such circumstances, it seems reasonable to exploit heuristic [7] or statistical [8] techniques for examining outputs. Most prior work in this area has focused on validation prior to deployment and operation, but the applications of these techniques to dynamic validation seems a particularly suitable combination.

We have therefore devised an oracle based primarily on a combination of simple statistical tests, with a scripting language describing how those tests should be combined in various situations (such as when working with different document collections with different structural and statistical characteristics).

3. An Oracle for Dynamic Validation

In this section we present our validation techniques that we successfully applied to legacy collections from DTIC and NASA. The main idea behind our statistical validation techniques is measuring a relevant property, such as the length of the title (a metadata field), and comparing it to values known to be typical of documents already in that collection. The comparison results are quantified and normalized into a confidence value for the value extracted for that metadata field. Confidence values range from a minimum of 0.0 indicating an almost certainly incorrect value to 1.0 indicating an almost certainly correct value.

3.1. Validation Specifications and Scripts

The metadata record extracted from a document will typically consist of many fields, which vary both from one collection to another and for different kinds of documents within a collection. Each field may be subject to several different validation tests. Which tests are most appropriate will vary both by field and by collection. Additionally, as will be noted later, dynamic validation can be invoked at different stages within the overall system, and the preferred tests and the means of combining those tests into an overall confidence score may depend upon when the validator is being invoked.

The selection of the appropriate tests and the combination of the test results is therefore a non-trivial activity requiring substantial flexibility. We have addressed this by creating a special-purpose language (embedded in XML) for describing the selection of tests for a field and the computation of combined confidence values over multiple tests.

Figure 3 shows a the basic skeleton of a validation specification (this one for the DTIC collection) indicating that the overall confidence for a collection of metadata will be computed as an average of confidences computed for four different metadata fields. In addition to averages (possibly weighted), confidences may be combined by operators for weighted sums, minima, and maxima. Operators also allow rescaling of confidences via arbitrary piecewise linear functions.

Within the outer `average` tag are rules for individual fields. Figure 3 shows the details for two of these, showing that the title field is validated using a combination of length and dictionary statistical tests and that the report date field is validated using a deterministic format test.

```
<?xml version="1.0"?>
<val:validate collection="dtic"
 xmlns:val="jelly:edu.odu.cs.dtic...">
   <val:average>
     <val:field
        name="UnclassifiedTitle">
         ...tests for this field...
     </val:field>
     <val:field name="ReportDate">
         ...tests for this field...
    </val:field>
     <val:field name="PersonalAuthor">
         ...tests for this field...
     </val:field>
     <val:field name="CorporateAuthor">
         ...tests for this field...
     </val:field>
  </val:average>
</val:validate>
```

Figure 3. Top Level of a Validation Specification

The validation specification is combined with a given metadata record to form an executable script. The output of the script is the metadata record (possibly re-ordered) with attributes attached to each field indicating the confidence value for that field and to the record root indicating the overall confidence value for the output as a whole. Figure 5 shows how the metadata record of Figure 2 might have been marked up by the script produced from the specification of Figures 3 and 4.

In addition to marking up each field with a confidence value, a field with confidence less than 0.5 is marked with an warning message explaining the primary reason for the poor confidence rating. Output records containing such warnings will be referred to human operators for inspection and, if necessary, correction.

By using an XML-based language for the specification and the derived scripts, we avoid the need for a customized parser. The semantics of the validation scripting language is implemented in Jelly, a framework for defining executable XML languages[1]. The modular structure of a implementation, combined with the lack of a custom parser, means that addition of new operators to the specification and scripting languages is a nearly trivial task.

3.2. Reference Models

Most of the tests supported by our validator (detailed below in Section 3.3) are based upon a point statistic com-

```
   ⋮
  <val:field
     name="UnclassifiedTitle">
     <val:average>
        <val:dictionary/>
        <val:length/>
     </val:average>
  </val:field>
  <val:field name="ReportDate">
     <val:dateFormat/>
 </val:field>
   ⋮
```

Figure 4. Field Rules in a Validation Specification

Field	Avg.	Std. Dev.
UnclassifiedTitle	9.91	4.78
Abstract	114.32	58.02
PersonalAuthor	2.75	0.52
CorporateAuthor	6.99	2.29

Table 1. Field Length (in words), DTIC collection

```
<?xml version="1.0"?>
<metadata confidence="0.460"
   warning="ReportDate field does not match
              required pattern">
  <UnclassifiedTitle confidence="0.979">
    Thesis Title: Intrepidity, Iron
    Will, and Intellect: General Robert
     L. Eichelberger and Military Genius
  </UnclassifiedTitle>
  <PersonalAuthor confidence="0.4"
    warning="PersonalAuthor: unusual number
        of words">
    Name of Candidate: Major Matthew H. Fath
  </PersonalAuthor>
  <ReportDate confidence="0.0"
    warning="ReportDate field does not
      match required pattern">
    Accepted this 18th day of June 2004 by:
  </ReportDate>
  <approvedby warning="unvalidated">Approved
    by: Thesis Committee Chair Jack D. Kem,
     Ph.D., Member Mr. Charles S. Soby,
     M.B.A., Member Lieutenant Colonel
        John A. Suprin, M.A.
  </approvedby>
  <acceptedby warning="unvalidated">
    Robert F. Baumann, Ph.D.
  </acceptedby>
</metadata>
```

Figure 5. Sample Output from the Validator

puted for a metadata field value extracted by the program and compared to models of that field. The model information is obtained from metadata records previously produced by the human staff for documents already in the collections.

For the DTIC collection, for example, the average and standard deviation of the field lengths are computed for titles, author fields, organization names, and abstracts for approximately 800,000 (unclassified) documents that were made available to us. Table 1 shows the length properties for the DTIC fields we are currently validating.

For the titles and abstracts, the average and standard deviation of the percentage of words in an English dictionary are computed for the same 800,000 documents. Table 2 shows the recognition rates for these DTIC fields.

For author and organization names, which should feature a more specialized but recurring vocabulary, phrase dictionaries are constructed for all phrases of length 1-4 words over a randomly selected set of 600,000 of those metadata records. Then the remaining 200,000 are used to compute the average and standard deviation of the percentage of phrases in each field that are recorded in the phrase dictionaries (an approximation of the percentage of phrases that, in new documents, can be expected to be recurrences of previously encountered phrases). Table 3 shows the recurrence percentages for these DTIC fields. It shows that, particularly in the corporate author field, a document will rarely introduce a word or even a phrase that has not been encountered before, suggesting that almost any output with novel content in that field would be suspect.

Field	Avg.	Std. Dev.
UnclassifiedTitle	88%	13%
Abstract	94%	5.0%

Table 2. Dictionary Detection (% of recognized words), DTIC collection

Similar models are computed for the NASA collection, although the available set of metadata is smaller, comprised of about 10,000 records.

The DTIC and NASA collections include many more metadata fields than have been mentioned here. As the extraction system moves toward final deployment, similar models will be constructed for the remaining fields.

Field	Phrase Length	Avg.	Std. Dev.
PersonalAuthor	1	97%	11%
	2	83%	32%
	3	71%	45%
CorporateAuthor	1	100%	2.0%
	2	99%	6.0%
	3	99%	10%
	4	98%	13%

Table 3. Phrase Dictionary Hit Percentage, DTIC collection

3.3. Basic Validation Tests

Our validation tests can be categorized into two main categories: *deterministic tests* and *statistical tests*. The former include pattern matching using regular expressions and date format. The latter include the length, vocabulary, and dictionary tests described below. The pattern matching and regular expressions are straightforward but are applicable only to a relatively small fraction of metadata fields that have restricted formats. The probabilistic tests are described next.

3.3.1 Length Test

The length test compares the length of the metadata field value to an average length previously calculated for this metadata field from known correct metadata. If the value is significantly longer or shorter than typical length, it receives low confidence value. The closer the length to average, the higher the confidence score it receives.

3.3.2 Vocabulary Test

The vocabulary test is suitable for metadata fields that tend to have a specialized vocabulary, such as author and organization names. Phrase dictionaries are constructed for these metadata fields using preexisting correct metadata. A phrase dictionary contains all the sub-sequences of tokens for each metadata entry. For example, if the string is “Kurt John Maly” then the phrase dictionary would contain the following sequences: Kurt, John, Maly, Kurt John, John Maly, Kurt John Maly. Then, when validating an extracted metadata value, it is tokenized into phrases and the corresponding phrase dictionary is consulted to count how many of these phrases exist in the phrase dictionary. The percentage of recurring phrases for this extracted value is compared to the average recurrence rate of phrases for that metadata field as determined from the preexisting correct metadata.

3.3.3 Dictionary Test

The dictionary test is suitable for metadata fields that do not have a specialized vocabulary, but rather are composed of free text, e.g., titles or abstracts.

When validating an extracted metadata value using the dictionary test, the extracted value is tokenized into words and an English language dictionary is consulted to count how many of these words exist in the dictionary. The percentage of recognized words is compared to the average percentage of recognized words for correct values of that metadata field in the preexisting metadata.

3.4. Normalization

Each of the statistical tests entails, as noted earlier, the computation of a point statistic (x) for an extracted metadata field. In each case, the associated model for that field supplies an average ($\bar{x}$) and standard deviation (s) for that statistic. This permits the computation of the standard score $z = (x - \bar{x})/s$ for that point statistic.

Lacking information as to the precise distribution of each statistic, it would be difficult to map these standard scores directly into a probability. Such an effort would probably be misspent, in any event, because, as noted below, we wish to preserve the flexibility of combining test results in a variety of different ways.

Consequently, the standard scores are mapped onto a confidence value in the range 0.0-1.0 via a piecewise linear function that is metadata field-specific and collection-specific (and that can also be overridden in a validation specification as described below).

The shape of the normalizer function can be tuned to adjust the strictness of the tests and to guarantee the appropriate interpretation of the standard score. For example, the normalizer for the dictionary and vocabulary tests are typically monotonically increasing functions because higher-than-average values for these statistics imply increased confidence. On the other hand, the normalizer for the length test would typically peak at the average, as both lower-than-average and higher-than-average values may be indicative of untrustworthy output.

4. Evaluating The Oracle

To determine if the oracle described here would actually be useful, it was applied to the output of a pre-release version of the metadata extraction system on a number of documents.

Determining whether the validator’s confidence values were “reasonable” was somewhat problematic as the exact confidence values could not be predicted or even specified to high precision for realistic outputs. Validation of expert systems, knowledge bases, and other A.I. applications frequently work by comparing the recommendations of the software to that of a human expert, e.g., [6].

In that spirit, we opted to take advantage of the multi-template structure of the extractor. For a group of documents that had been previously assigned to document classes by manual inspection, the template for that manually assigned class and for the largest other classes were all applied to each document. The resulting multiple sets of extracted metadata were then passed through the validator. The purpose of this exercise is to see if the validator's score for the best quality output among the alternatives corresponds to the prior classification performed by human inspection of the documents.

The base set of documents for this experiment was a sample of 2000 drawn from among nearly 60,000 new documents submitted to DTIC over the prior two years. These two thousand were chosen at random, subject to the limitation that they were unclassified documents released for public distribution.

Approximately half of these documents contain an RDP, a special form in which metadata fields are enumerated and filled in. Such forms are easily recognized and are highly distinctive. As such, it was felt that these would not offer an informative exercise of the validator.

Among the remaining documents with no such forms, the metadata extractor must find pages that contain metadata and locate the metadata field values as instructed by the template. In the course of development of the extractor, these documents had been examined by humans who grouped them by layout and chose the most common layouts for an initial set of ten templates. These templates covered 696 documents of the original set of 2000, and these documents were employed as the basis of this study.

The distribution of these documents among the human-assigned classes can be seen in Table 4.

The document classes and their templates have been given arbitrary names by the template authors, usually based upon characteristics observed in the first few documents of that class. For example, the "eagle" class consists of degree theses for U.S. Military Academy at West Point and is named for the characteristic background image seen in Figure /refeagle-page. As a group, these ten templates include clusters of very similar document layouts (often differing only by the number and placement of blocks of author names and addresses) although there is little inter-clusters similarity, with documents ranging from degree theses to corporate reports and technical articles.

The validation specification of Figures 3 and 4 formed the starting point for the specification employed in this experiment. It computed the average confidence in the fields UnclassifiedTitle, PersonalAuthor, CorporateAuthor, and ReportDate. This was judged to be somewhat inappropriate for the evaluation technique proposed for this experiment however. The use of an overall average would unfairly penalize templates that had successfully extracted additional fields at marginally lower confidences. For example, if the rand and eagle templates each extracted the same title field at a confidence of 1.0, but the rand template also extracted a personal author with confidence 0.95, a simple average over the extracted fields would favor the title template, but human observers would favor the rand template. An adjustment was therefore made by replacing the overall average by a simple sum, and applying a rescaling to each field so that confidence values of less than 0.5 were rescaled to small negative values. In this way, fields extracted with unacceptably low confidence would not be treated as positive factors in the overall sum (i.e., extracting a bad value would not improve the overall score).

Allowances were also made for the fact that the extractor design called for a post-processing component between the extractor and the validator, which was designed to clean the extracted metadata according to collection-specific rules. For example, DTIC removes all titles and military ranks from author names, a fact which has a direct impact on both length statistics for author fields and on the contents of the phrase dictionaries constructed from existing examples of that metadata. The post-processor had not been implemented at the time of this experiment, but was simulated using simple editor scripts prior to feeding metadata to the validator.

Table 4 shows the results of applying the adjusted specification to the template outputs for 646 documents. For approximately 91% of these documents, the validator gave the highest confidence score to the same document template as had been anticipated by the human evaluator.

Interestingly, the oracle agreed with the human choice even in cases where the extractor did a poor job on some fields. For example, Figure 5 shows that the extractor incorrectly included the phrase "Name of Candidate:" as part of an author name, but because the other templates either made this same mistake or failed to extract the author name at all, the oracle still preferred the eagle class overall.

Examination of the individual field scores suggests that the validator did exactly what it was intended to do, flagging field values that we would not wish to pass into a database without intermediate clean-up. In the majority of cases where the validator chose a class different from the initial human choice, the extracted metadata was of uniformly poor quality from all templates, so that the overall score for the metadata set would have guaranteed the referral of that document for human inspection.

Manually Assigned Class		Number of documents Oracle Preferred									
Class	# docs	**alr_1**	**alr_2**	**au**	**crs_1**	**crs_2**	**crs_3**	**eagle**	**nps_thesis**	**nsrp**	**rand**
alr_1	42	36	6	0	0	0	0	0	0	0	0
alr_2	12	6	6	0	0	0	0	0	0	0	0
au	86	0	0	86	0	0	0	0	0	0	0
crs_1	37	0	0	0	33	0	1	0	0	2	1
crs_2	15	1	0	0	0	12	1	0	0	0	1
crs_3	54	0	1	3	2	1	43	0	0	0	4
eagle	47	0	0	0	0	0	0	47	0	0	0
nps_thesis	232	1	10	0	0	0	0	0	211	0	10
nsrp	109	0	1	0	0	0	0	0	0	106	2
rand	12	0	1	0	0	0	0	0	0	0	11
Total:	646						# matched:	591			

Table 4. Oracle's Choices (Highest Confidence) versus Human's

5. Conclusions

We have described a system for dynamic validation of a system for extracting metadata from scanned documents. Because this is not an application where provision of an exact oracle is possible, we developed an oracle that combines statistical and deterministic tests. Most of the tests are statistical, based upon models of priviously generated outputs from a manual process.

Flexibility and adaptability of the validator was achieved by developing a customized scripting language describing how the various tests should be combined to obtain an overall measure of confidence in a program output.

We believe that a similar statistical dynamic validation process could be useful in a variety of applications where the specification of exact oracles is impossible or impractical. Although this application may have been unusually fortunate in the amount of prior good-quality outputs available, in other circumstances the statistics could be accumulated during the early stages of deployed operations by starting with a high degree of human inspection of program outputs, then relaxing the degree of direct human oversight as the body of human-approved outputs is accumulated.

The suitability of the validator was demonstrated by an experiment measuring its ability to mimic human judgments as to which of several alternate outputs for the same document would be preferred. It performed better than had actually been anticipated, agreeing with human judgments on approximately 91% of the documents examined and behaving reasonably on the majority of the remaining cases.

Curiously, this is a more consistent performance than we had been able to obtain from the various vision-inspired algorithms for classifying document layouts prior to performing the actual metadata extraction. We are currently considering the possibility of exploiting the validator as a kind of "post hoc" classification, applying all available templates to each document and selecting the one that scores best under validation.

6. Acknowledgments

This work was supported by grants from the Defense Technical Information Center and the National Aeronautics and Space Administration.

The authors would like to thank Paul Flynn and Li Zhou for their contributions to the development of the metadata extraction system.

References

[1] Apache Software Foundation. Jelly: Executable XML. http://jakarta.apache.org/commons/jelly/, 2006.

[2] W. T. Chen, J. P. Ho, and C. H. Wen. Dynamic validation of programs using assertion checking facilities. In *Proc. COMPSAC 78*, pages 533–538, Nov. 1978.

[3] L. A. Clarke and D. S. Rosenblum. A historical perspective on runtime assertion checking in software development. *SIGSOFT Softw. Eng. Notes*, 31(3):25–37, 2006.

[4] Defense Technical Information Center. Public Scientific and Technical Information Network. http://stinet.dtic.mil/str/index.html, 2007.

[5] R. S. Fabry. Dynamic verification of operating system decisions. *Commun. ACM*, 16(11):659–668, 1973.

[6] C. Hernandez, J. J. Sancho, M. A. Belmonte, C. Sierra, and F. Sanz. Validation of the medical expert system RENOIR. *Comput. Biomed. Res.*, 27(6):456–471, 1994.

[7] D. Hoffman. Heuristic test oracles. *Software Testing and Quality Enginerring*, 1(2):29–32, 1999.

[8] J. Mayer. Towards a reliable statistical oracle and its applications. In *Proceedings of 22nd GI-TAV Meeting*, pages 21–27. Softwaretechnik-Trends, 2005.

[9] National Aeronautics and Space Administration. NASA Technical Reports Server. `http://ntrs.nasa.gov/search.jsp`, 2007.

[10] D. S. Rosenblum. A practical approach to programming with assertions. *IEEE Trans. Softw. Eng.*, 21(1):19–31, 1995.

[11] J. Tang. *Template-based Metadata Extraction for Heterogeneous Collections*. PhD thesis, Old Dominion University, 2006.

[12] J. Tang, K. Maly, S. Zeil, and M. Zubair. Automated Building of OAI Compliant Repository from Legacy Collection. In *Proceedings of the 10th International Conference on Electronic Publishing (ELPUB)*, June 2006.

[13] U.S. Government Printing Office. A Strategic Vision for the 21st Century. Technical report, 2004.

[14] E. J. Weyuker. On testing nontestable programs. *The Computer Journal*, 25(4), 1982.

[15] S. S. Yau and R. C. Cheung. Design of self-checking software. In *Proceedings of the international conference on Reliable software*, pages 450–455, New York, NY, USA, 1975. ACM Press.