Generating Summaries of Work Flow Diagrams

Rebecca Passonneau[*†], **Karen Kukich**[*], **Jacques Robin**[‡],
Vasileios Hatzivassiloglou[†], **Larry Lefkowitz**[*], and **Hongyan Jing**[†]

[*]Bellcore
445 South Street
Morristown, NJ 07960
{beck, kukich, wombat}@bellcore.com

[†]Department of Computer Science
Columbia University
New York, NY 10027
{becky, vh, hjing}@cs.columbia.edu

[‡]Departamento de Informática
Universidade Federal de Pernambuco
Caixa Postal 7851, Cidade Universitária
Recife, PE 50732-970, Brazil
jr@di.ufpe.br

Abstract

FLOWDOC is a prototype text generator that summarizes information from work flow graphs in a business re-engineering context. A richer ontology than is typically used allows generalization of input data during content selection, and combination of data during sentence planning.

Keywords: Summarization; Data combining; Content planning; Text generation; Industrial applications.

1 Introduction

FLOWDOC is a prototype application of natural language generation technology to business re-engineering (Rummler & Brache 90). It provides automatic documentation within a software-aided business re-engineering environment under construction at Bellcore. Its input comes from SHOWBIZ (Wittenburg 96), a GUI used in this environment to represent as work flow diagrams both the *Present Mode of Operation* (PMO) of a working group and the *Future Modes of Operation* (FMOs) suggested by re-engineering consultants.

FLOWDOC summarizes the key properties of a SHOWBIZ work flow in a few natural language sentences. Figure 1 shows an input SHOWBIZ PMO work flow for a publishing task, and Figure 2 shows a portion of the corresponding summary produced by FLOWDOC. Figure 1 is typical in having many nodes, each with many attributes, thus illustrating the utility of summarization for presenting key generalizations across nodes.

Re-engineering analysis often results in very complex work flows where numerous nodes are annotated by a large set of features modeling various aspects of the overall task. While SHOWBIZ includes facilities to parse and reduce subflows into a single node, it cannot synthesize key properties that are scattered across distant nodes. By providing an instant textual feedback that abstracts from the complexity of the diagram, FLOWDOC can help a variety of re-engineering subtasks:

- Acquiring and validating the PMO.
- Detecting tasks whose repetitivity is hidden in the complexity of the work flow (such tasks are good candidates for centralization or suppression during re-engineering).
- Helping management choose among alternative FMO proposals.
- Searching related work flows in a repository of past work flows using information retrieval techniques developed for textual documents.

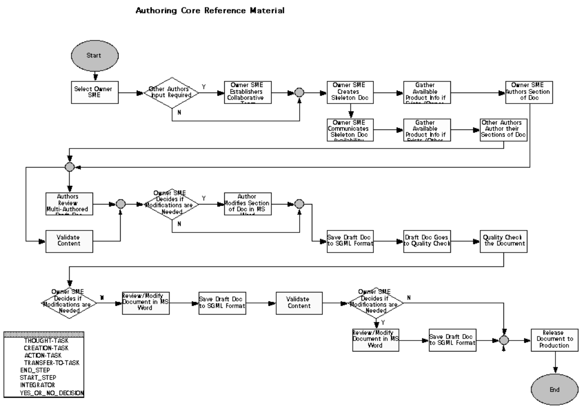

Figure 1: Example input graph: "Authoring Core Reference Materials" work flow.

In addition, using FLOWDOC's (1) well-defined semantics of graph elements, (2) explicit domain ontology, and (3) standardized terminology helps ShowBiz users to draw work flow diagrams in a more rigorous and principled way.

FLOWDOC is based on principles and tools from previous research. In particular, its overall architecture integrates aspects of PLANDOC (Kukich *et al.* 94) and STREAK (Robin 93), and it relies on the FUF/SURGE package (Elhadad 93) for lexicalization and syntactic realization.

The application of summarizing work flows required tackling three open problems:

1. Generating object set descriptions that balance conciseness, accuracy, and homogeneity of detail.
2. Aggregating multiple propositions into complex sentences under a combination of domain, rhetorical, and focus constraints.
3. Employing a partial domain ontology, as reengineering is a meta-domain applied to a variety of application domains, from publishing to assembly line.

In this paper we focus on the first two problems, which pertain to summarization in general, independent of the application domain. Our approach to the third problem, mixing ontology-based componential generation (for work flow entities) with menu-gathered canned text (for specific application domain entities), will be addressed in a future publication.

> The Authoring Core Reference Materials workflow pertains to the Reference Delivery and Automation business process. ... It begins with a product trigger and ends with a handshake with production. It has 24 activities, including 20 tasks and four decisions. ... The SME is the pivotal participant of this workflow. ... The most frequent tasks in this workflow are those of creating, reviewing, and saving documents. ...

Figure 2: Portion of the summary produced by FLOWDOC for the work flow of Figure 1.

2 FlowDoc Architecture

FLOWDOC employs a traditional pipeline architecture, as shown in Figure 3, mostly inspired from PLANDOC (Kukich *et al.* 94). However, unlike PLANDOC and most other generation systems, FLOWDOC makes pervasive use[1] of a declarative ontology of work flow, application domain, and rhetorical concepts. Also unlike PLANDOC, FLOWDOC uses a discourse agenda derived from an ontology of message classes to guide two distinct phases of content selection and sentence planning. These two design choices allow FLOWDOC to incorporate sophisticated summarization strategies for (1) abstracting the input data during content selection, and (2) combining data into complex sentences during sentence planning.

2.1 Ontological Knowledge

A portion of FLOWDOC's ontology, which is implemented in CLOS, is shown in Figure 4. Apart from the traditional application-domain entities and relations, it also includes three other concept classes:

- Work flow entities and relations acting as meta-domain concepts.
- Rhetorical relations (drawn from (Hobbs 85; Polanyi 88)) used by the sentence planner.
- Message classes, clause-sized content structures used by the content selector and linked to the work flow relation classes used by the sentence planner.

[1] In all components except the portable back-end (SURGE) which is used for syntactic processing.

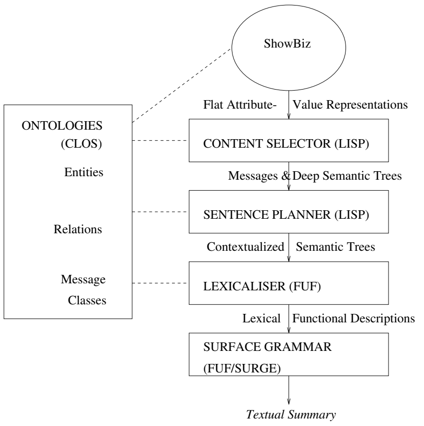

Figure 3: FLOWDOC architecture.

FLOWDOC inherits its work flow ontology from SHOWBIZ which allows users to define application-domain subconcepts of flow node concepts to use when creating a work flow diagram. For example, in our desktop publishing domain, the *procedure* node includes subconcepts such as *creation-task* and *thought-task*.

2.2 Modules and Data Structures

As shown in Figure 3, the input to FLOWDOC is a set of flat attribute-value list representations (FAVRs) describing all the information about the SHOWBIZ diagram. There is one FAVR per node in the work flow diagram.

The content selector module takes FAVRs as input, produces a list of instantiated messages, and then converts each message into a Deep Semantic Tree (DST). The sentence planner takes DSTs as input, restructures them into a more compact form, and imposes local and global coherence constraints on them. Its output is a list of Contextualized Semantic Trees, or CSTs.

A CST represents the semantic content of one or more domain propositions, or messages. Each CST will be realized as a single sentence, whether simple, compound, or complex. Before being realized, a CST must be lexicalized by FLOWDOC's lexicalizer, which is implemented as a FUF grammar following (Elhadad 93). For each CST, the lexicalizer assigns thematic roles to the concepts represented therein, chooses content words for them, and produces one Lexical Functional Description, or LFD.

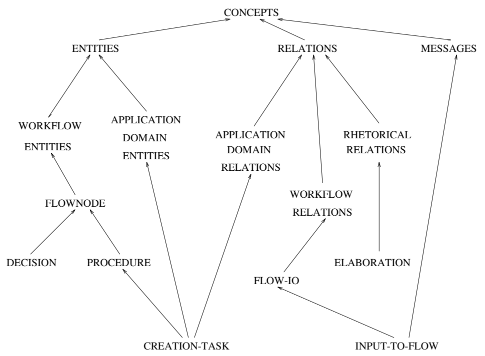

Figure 4: Organization of concepts in FLOWDOC.

Finally, each LFD is passed on to the SURGE grammar. SURGE takes care of mapping thematic roles to syntactic structures, filling in function words, ensuring morphological agreement, linearizing the structure, and outputting a syntactically valid sentence.

3 Content Selection

FLOWDOC initially instantiates messages from a discourse agenda consisting of an ordered list of ten message classes, possibly multiple times for each class. Each class corresponds to one type of high-level information to be included in the summary:

1. Overall task modeled by the work flow.
2. Parent work flow (if any).

3,4. Input and output of the work flow.

5,6. Initial and final sub-tasks in the work flow.

7. Total number of nodes, decision nodes, action nodes, and subflows (if any).

8,9. Salient participants and decisions.

10. Salient actions and their objects.

While instantiating message classes 1-7 is simply a matter of retrieving and counting the corresponding pieces of information from the input graph, producing the messages of types 8-10 involves novel strategies to:

- Identify the *salient* participants, decisions, and actions.
- Choose the description detail level for the objects of each salient action that is the best trade-off between conciseness and accuracy.

3.1 Selecting Interesting Data

FLOWDOC classifies each participant, decision, and action in the input graph as either a salient one, a non-salient one which nevertheless is still worth mentioning, or one that is not even worth mentioning. This determination is based on both the absolute number of occurrences of that concept in the work flow and the ratio of its occurrences relative to the number of occurrences of all concepts from its class (e.g., a particular participant such as *subject-matter expert* is compared to the total number of occurrences of all explicitly mentioned participants in the graph). We currently use fixed thresholds for this classification procedure, but these can be easily changed to adjust the level of detail in the generated summary.

In the example of Figure 1, the input graph contains 24 explicitly mentioned participants, among which the *subject-matter expert* (SME) appears 16 times. Accordingly, "SME" is identified as the sole salient participant in that flow diagram. Similarly, FLOWDOC classifies "Co-authors" and "Editor" as additional (non-salient) participants worth mentioning; "Create", "Review", and "Save" as salient tasks; "Collect", "Transfer", and "Validate" as additional significant actions; and "Whether modifications are needed" as the only salient decision.

3.2 Ontological Generalization

Once the actions to mention in the summary have been selected, FLOWDOC attempts to concisely express the set of objects affected by each of them. Each such object is a concept in the domain ontology. Given the set $C_O = \{O_1, O_2, \ldots, O_N\}$ of objects of a given action and the associated list $(c_1, c_2, \ldots, c_N)$ of their occurrence counts with that action in the work flow, FLOWDOC computes an optimal set of concept generalizations $\{G_1, G_2, \ldots, G_M\}$ such that each generalization replaces a subset of C_O while maintaining a reasonable trade-off between the accuracy, specificity, and verbosity of the description.

We consider as candidate concept generalizations the actual members of C_O and all the concepts in the domain ontology that subsume one or more of them. Each such candidate description is evaluated on its suitability to replace a given subset of C_O using a weighted sum formula trading-off along two antagonistic dimensions:

- **Coverage**, measuring how many of the objects in the subset (proportionally weighed according to their occurrence counts c_i) are actually subsumed by the candidate generalization.
- **Specificity**, defined as the semantic distance between each element of the subset and the candidate generalization.

The semantic distance currently used is simply the number of levels between each object and the generalization in the domain ontology. It could be easily changed to an information-based distance, e.g., along the lines of the metrics proposed in (Resnik 93), who measures semantic distance between two concepts as a function of the lexical probabilities of their common superclasses.

To compute the optimal set of generalizations, FLOWDOC starts by generating all possible partitions of the given set of objects,[2] then locates the best single-term description for each subset in the partition by applying the procedure outlined above to each candidate generalization, and finally combines the single-term description scores in one number. The final score is adjusted by two additional penalties:

- A *verbosity* penalty, penalizing descriptions with more than one generalization.
- A *heterogeneity* penalty, for descriptions that are locally optimal but significantly lower in the ontology (more specific) than the global specificity level.

The global specificity level indicates the appropriate overall level of detail. It is computed by applying the above ontological generalization procedure to the collection of *all* the objects appearing in the input graph, across all actions. It implements the idea of "basic level" descriptions from (Rosch 78) for the application domain modeled by the work flow. For example, while processing a graph which covers documents of many

[2] With some performance-imposed constraints, since the number of possible partitions grows exponentially with the number of objects and the number of subsets in the partition.

```
((CONCEPT SALIENT-TASK)
 (ARGS
  ((THEME ((FLOW-ID FLOW-1)
           (CONCEPT PROCESS-FLOWGRAPH)
           (LABEL "Authoring Core Reference Materials")))
   (RHEME ((ELTS
    ((1 ((CONCEPT CREATION-TASK)
         (LABEL "Create")
         (COUNT 4)
         (ARGS ((TO-WHOM-OR-WHAT ((CONCEPT DOCUMENT)
                                  (COVERED-COUNT 3)))))))
     (2 ((CONCEPT THOUGHT-TASK)
         (LABEL "Review")
         (COUNT 3)
         (ARGS ((TO-WHOM-OR-WHAT ((CONCEPT
                                   DRAFT-DOCUMENT-IN-MS-
                                     WORD-FORMAT)
                                  (COVERED-COUNT 3)))))))
    (3 ((CONCEPT ACTION-TASK)
        (LABEL "Save")
        (COUNT 3)
        (ARGS ((TO-WHOM-OR-WHAT ((CONCEPT SGML-DOCUMENT)
                                 (COVERED-COUNT
                                  3)))))))))))))
```

Figure 5: Simplified DST for the salient tasks in the example input graph of Figure 1.

types, FLOWDOC will have a bias in favor of the generic term "Document" rather the too-specific term "Draft document in SGML format"; a trade-off between the heterogeneity penalty and other components of the description score occurs if the latter term looks locally optimal.

3.3 From Messages to DSTs

Each FLOWDOC message is converted to a *Deep Semantic Tree*. Each node of a DST has a single concept attribute. Relational concepts have an attribute representing its *relata*, which can be arguments (ARGS) or elements (ELTS), depending on whether the relata are distinguished in some way. The slot fillers of the messages are mapped to values of ARGS or ELTS. In addition, messages of the same type are combined into a single DST. Figure 5 shows part of the output of this stage for the three salient tasks ("Create", "Review", and "Save") in the graph of Figure 1.

4 Sentence Planning

FLOWDOC's sentence planner converts Deep Semantic Trees (DSTs) to Contextualized Semantic Trees (CSTs). It has three distinct phases, the first two of which pertain to summary generation.

The first phase of *inter-organization* uses ontological knowledge about work flow relations and rhetorical relations to determine whether a pair of input Deep Semantic Trees (DSTs) are sufficiently close semantically to be combined into a single CST. When two DSTs are first combined, the output is cast in a canonical form. As a side effect of inter-organization, what were originally root concepts in the input DSTs can become deeply subordinated in the CST. Such a CST potentially diverges from the original discourse agenda (as specified by the ordered list of message classes).

The second module reconfigures CSTs, when necessary, in order to maintain the discourse agenda. This *intra-organization* module uses the agenda to determine what the root concept and precedence of constituents of a CST should be. The third *node expansion* module enforces local coherence by traversing the tree to add lexical choice constraints at each node. For example, it uses a simplified version of the algorithm from (Passonneau 96) to control definiteness of noun phrases and pronominalization.

4.1 Inter-organization

If two DSTs can be used to instantiate a rhetorical relation, they can be combined. Here we illustrate the instantiation of two types of rhetorical relation: *parallel* and *elaboration*.

Two input DSTs are parallel if the root concepts have an immediate common parent in the ontology, the same tree structure with the same attributes at each node (possibly with different values), and if they are sufficiently shallow (typically a depth of three, but with certain exceptions).The two DSTs in Figure 6 meet these constraints.

In the resulting CST shown in Figure 6, the two input DSTs have become adjoined to a new root concept whose value is *parallel*. The input DSTs are the values of two ELTS of the parallel relation. After lexicalization and grammaticalization, this CST results in the conjoined sentence shown in Figure 6.

Figure 7 shows another pair of input DSTs that meet the same constraints. The root concept values are *total-node-count* and *subset-node-count*, the parent node in the ontology is *node-count*, the feature structures have the same attributes at each node, and the two DSTs are relatively shallow. But before instantiating a parallel relation, the inter-organizer tries to instantiate an *elaboration* relation. The motivation for testing for an

```
DST1:  ((CONCEPT INPUT-TO-FLOW)
        (ARGS ((THEME ((FLOW-ID FLOW-1)
                       (CONCEPT PROCESS-FLOWGRAPH)))
               (RHEME ((CONCEPT INPUT))))))

DST2:  ((CONCEPT OUTPUT-OF-FLOW)
        (ARGS ((THEME ((FLOW-ID FLOW-1)
                       (CONCEPT PROCESS-FLOWGRAPH)))
               (RHEME ((CONCEPT OUTPUT))))))

CST:   ((CONCEPT PARALLEL)
        (ELTS ((1 ((CONCEPT INPUT-TO-FLOW)
                   (ARGS ((THEME ((FLOW-ID FLOW-1)
                                  (CONCEPT
                                  PROCESS-FLOWGRAPH)))
                          (RHEME ((CONCEPT INPUT)))))))
               (2 ((CONCEPT OUTPUT-OF-FLOW)
                   (ARGS ((THEME ((FLOW-ID FLOW-1)
                                  (CONCEPT
                                  PROCESS-FLOWGRAPH)))
                          (RHEME ((CONCEPT OUTPUT))))))))))
```

Sentence 3: *It begins with a product trigger and ends with a handshake with production.*

Figure 6: Combining Parallel DSTs.

elaboration relation first is that this results in more informative output.

4.2 Enforcing Global Coherence

The canonical form produced by instantiating an *elaboration* relation with the *total-node-count* and *subset-node-count* DSTs shown in Figure 7 would have the elaboration concept as its root, analogous to the inter-organization illustrated in Figure 6. A sentence that would correspond to this CST is "*The 24 nodes of the graph break down into 4 decision nodes and 20 procedure nodes*." However, this sentence is not generated because the CST is re-configured to fit the current discourse agenda. Here we briefly discuss the conditions for intra-organizing a CST.

The discourse agenda is a stack of the message classes, with agenda items popped after CSTs are produced by the sentence planner. At the time of the inter-organization illustrated in Figure 6, the top two items of the discourse agenda are *input-to-flow* and *output-of-flow*. Rules for comparing the agenda to the CSTs specify that a parallel structure with ELTS X_1 and X_2 meets the same agenda as a sequence of CSTs X_1 followed by X_2. Thus the CST shown at the bottom of Figure 6 meets the current agenda, and no intra-organization is performed.

The CST in Figure 7 results from intra-organization, due to a conflict between the discourse agenda and the structure of the canonical inter-organized CST. After the inter-organization step that takes the DSTs in Figure 7 as input, the top two items of the discourse agenda are *total-node-count* and *subset-node-count*. However, the root concept of the canonical CST is *elaboration*, and its theme and expansion arguments are the *total-node-count* and *subset-node-count* DSTs, respectively. The *total-node-count* concept is semantically more basic, and must become the root in order to meet the agenda. So the CST is restructured by elevating the input theme argument (the *total-node-count* DST) to the root, while attaching the expansion argument (the *subset-node-count* DST) at a subordinate position in the tree, in the constituent that expresses cardinality of nodes.

```
DST1:  ((CONCEPT TOTAL-NODE-COUNT)
        (ARGS ((THEME ((FLOW-ID FLOW-1)
                       (CONCEPT FLOWGRAPH)))
               (RHEME ((CONCEPT CARDINALITY)
                       (ARGS ((THEME ((CONCEPT FLOWNODE)))
                              (VALUE ((CONCEPT CARDINAL)
                                      (CARDINAL 24)))))))))))

DST2:  ((CONCEPT SUBSET-NODE-COUNT)
        (ARGS ((THEME ((FLOW-ID FLOW-1)
                       (CONCEPT FLOWGRAPH)))
               (RHEME
                ((CONCEPT PARALLEL)
                 (ELTS
                  ((1 ((CONCEPT CARDINALITY)
                       (ARGS ((THEME ((CONCEPT PROCEDURE)))
                              (VALUE ((CONCEPT CARDINAL)
                                      (CARDINAL 20)))))))
                   (2 ((CONCEPT CARDINALITY)
                       (ARGS ((THEME ((CONCEPT DECISION)))
                              (VALUE
                               ((CONCEPT CARDINAL)
                                (CARDINAL 4))))))))))))))

CST:   ((CONCEPT TOTAL-NODE-COUNT)
        (ARGS ((THEME ((FLOW-ID FLOW-1)
                       (CONCEPT FLOWGRAPH)))
               (RHEME ((CONCEPT ELABORATION)
                       (ARGS
                        ((THEME
                          ((CONCEPT CARDINALITY)
                           (ARGS
                            ((THEME ((CONCEPT FLOWNODE)))
                             (VALUE ((CONCEPT CARDINAL)
                                     (CARDINAL 24)))))))
                         (EXPANSION
                          ((CONCEPT SUBSET-NODE-COUNT)
                           (ARGS ((THEME ((FLOW-ID FLOW-1)
                                          ...))
                                  (RHEME (...)))))))))))))
```

Sentence 4: *It has 24 activities, including 20 tasks and 4 decisions.*

Figure 7: Inter-, then Intra-organization.

5 Conclusion

Work by (Gulla & Willumsen 93) addresses explanation generation in the same application domain. However, FLOWDOC innovates by its pervasive use of a rich ontology for summarization tasks. It generates object sets that explicitly trade-off between conciseness and accuracy while maintaining a balanced level of detail, something that has not been addressed in previous research. While

FN (Reiter 91) also takes into account these three factors, it generates only descriptions of *individual* objects. In addition, FN does not feature a separate lexical choice component, but instead folds the lexicon into the domain ontology, which prevents considering syntactic and interlexical constraints when generating descriptions. On the other hand, EPICURE (Dale 92) does generate set descriptions, but it does not possess a domain hierarchy for generalization, focusing on entirely different issues such as domain fluidity and discourse structure.

The functions of the sentence planner include what (Shaw 95) and (Dalianis & Hovy 93) refer to as aggregation. Our inter-organization is closer to the aggregation rules proposed in (Dalianis & Hovy 93), using trees as the input representation and a declarative representation of RST-like relations. However, their aggregation operations of subject and predicate grouping operate only on identical elements with the goal of eliminating redundancy. FLOWDOC inter-organizes trees based on semantic relatedness of nodes, not identity. Both (Shaw 95) and (Dalianis & Hovy 93) use operators whose consequences have multiple effects beyond aggregation, including control of ellipsis, insertion of specific lexical items, and so on. In our approach, tree combination is independent of coherence or lexical choice constraints.

References

(Dale 92) R. Dale. *Generating Referring Expressions*. ACL-MIT Press Series in Natural Language Processing, Cambridge, Massachusetts, 1992.

(Dalianis & Hovy 93) H. Dalianis and E. H. Hovy. Aggregation in Natural Language Generation. In *Proceedings of the Fourth European Workshop on Natural Language Generation*, Pisa, Italy, 1993.

(Elhadad 93) M. Elhadad. *Using Argumentation to Control Lexical Choice: a Unification-based Implementation*. PhD thesis, Computer Science Department, Columbia University, New York, 1993.

(Gulla & Willumsen 93) J. A. Gulla and G. Willumsen. Using Explanations to Improve the Validation of Executable Models. In *Proceedings of the 5th International Conference on Advanced Information Systems Engineering (CAiSE'93)*, pages 118–142, Paris, June 1993. Springer-Verlag.

(Hobbs 85) J. R. Hobbs. On the Coherence and Structure of Discourse. Technical Report CSLI-85-37, CSLI, 1985.

(Kukich *et al.* 94) K. Kukich, K. McKeown, J. Shaw, J. Robin, N. Morgan, and J. Phillips. User-needs Analysis and Design Methodology for an Automated Document Generator. In A. Zampolli, N. Calzolari, and M. Palmer, editors, *Current Issues in Computational Linguistics: In Honour of Don Walker*. Kluwer Academic Press, Boston, 1994.

(Passonneau 96) R. J. Passonneau. Using Centering to Relax Gricean Informational Constraints on Discourse Anaphoric Noun Phrases. *Language and Speech*, 1996. Forthcoming.

(Polanyi 88) L. Polanyi. A Formal Model of Discourse Structure. *Journal of Pragmatics*, **12**:601–638, 1988.

(Reiter 91) E. B. Reiter. A New Model for Lexical Choice for Open-class Words. *Computational Intelligence*, **7**, December 1991.

(Resnik 93) P. Resnik. Semantic Classes and Syntactic Ambiguity. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 278–283, Plainsboro, New Jersey, March 1993. Morgan Kaufmann, San Francisco, California.

(Robin 93) J. Robin. A Revision-based Generation Architecture for Reporting Facts in their Historical Context. In H. Horacek and M. Zock, editors, *New Concepts in Natural Language Generation: Planning, Realization, and Systems*. Frances Pinter, London and New York, 1993.

(Rosch 78) E. Rosch. Principles of Categorization. In E. Rosch and B. Lloyd, editors, *Cognition and Categorization*, pages 27–48. Lawrence Erlbaum, Hillsdale, New Jersey, 1978.

(Rummler & Brache 90) G. A. Rummler and A. P. Brache. *Improving Performance*. Jossey-Bass Publishers, San Francisco, 1990.

(Shaw 95) J. Shaw. Conciseness through Aggregation in Text Generation. In *Proceedings of the 33rd ACL (Student Session)*, pages 329–331, 1995.

(Wittenburg 96) K. Wittenburg. Relational Grammars: Theory and Practice in a Visual Language Interface for Process Modeling. In *Workshop on Theory of Visual Languages*, Gubbio, Italy, 1996.