The Multex generator and its environment: application and development[1]

Christian MATTHIESSEN*, ZENG Licheng*, Marilyn CROSS**, Ichiro KOBAYASHI***, Kazuhiro TERUYA**** & WU Canzhong*

*Macquarie University, Sydney, Australia; **DSTO, Canberra; ***Hosei University, Tokyo, Japan; **** University of New South Wales, Sydney

1. Multex — a multimodal and multilingual generation system

The aim of this paper is to report on generation-oriented research conducted by the Systemic Meaning Modelling Group based at Macquarie University, Sydney, and involving researchers at other institutions in Australia, Germany and Japan. We will describe the core generation system Multex, which is a system for generating multilingual and multimodal presentations, based on the principles of systemic functional theory. We will also describe one of its application environments, the HINTS system. Together Multex and its "environment" constitute a long-term NLP research program based on systemic functional theory. Our work is thus relevant to current efforts in work on generation to link it as a component capability to a more comprehensive system (as in the work on MT in the Pangloss project, in the work on building a generation knowledge source in the PROFILE system [Radev & McKeown, 1997], in the work on generating multimodal healthcare briefings in the MAGIC system [McKeown, Jordan & Allen, 1997], or in the work on linking the KPML generator to an editor's workbench [Bateman & Teich, 1995]).

2. Multex — a generation system in the Penman tradition

2.1 Background: the Penman tradition

Multex is a generator in the Penman tradition. The Penman tradition started in 1980 with the initial development of the Penman generator (Mann & Matthiessen, 1985). The design of Penman is in many respects a systemic functional one (see Matthiessen & Bateman, 1991, for the systemic conception of text generation); that is, the system is based on a theoretical model of language developed within systemic functional linguistics (eg. Halliday 1994) and Multex shares this base with Penman, drawing on central categories that have been important in text generation such as that of the system network. Penman includes:

(i) **a systemic functional generation grammar**, the Nigel grammar, organized around system networks (Matthiessen 1983; cf. also Matthiessen, 1995),

(ii) **an interface** between the grammar and higher-level components (e.g. the knowledge base and the text planner), the chooser & inquiry interface (Matthiessen & Bateman, 1991),

(iii) **a knowledge base** that is organized under the "Upper Model" (Matthiessen & Bateman, 1991; cf. also Halliday & Matthiessen, in press) and

[1] We gratefully acknowledge support for the research reported here from DSTO and the Australian Research Council. Multex is being developed by Zeng Licheng and SysAm by Wu Canzhong within the Systemic Meaning Modelling Group at Macquarie University. HINTS is being developed at DSTO, Canberra, under the direction of Marilyn Cross. We would also like to thank Philip Collier for thorough comments of this paper.

(iv) **a text planner** that is based on RST (Rhetorical Structure Theory, Mann & Thompson, 1988; Hovy, 1993).

The Penman system itself represents the first generation of systems in the Penman tradition; it has been distributed widely and is still being used at various research sites. It has been used for many generation tasks, eg. generation in machine translation within the Pangloss project. Multex can be characterized as a third-generation system within the Penman tradition. The most notable second-generation system is KPML (Komet-Penman MultiLingual), developed by John Bateman and his group, first at GMD/IPSI in Germany and now at Stirling University, Scotland (see Bateman, 1996). KPML includes Penman, but it goes considerably beyond the original Penman system, eg. in its modelling of multilinguality (cf. Bateman et al, 1991). In addition, KPML is equipped with a workbench interface for maintaining and developing its linguistic resources — one of the few or only such tools for generation systems. As a result, KPML is easier to use than Penman.

2.2 Multex as a third-generation system in the Penman tradition

In this section, we will outline some similarities and difference between Penman/KPML and Multex.

Similarities. Multex inherits major design themes from Penman, which we believe underlie the strength and longevity of Penman and the wide acceptance of KPML within the research community. These design features include:

- **grammar-centric generation.** Instead of using a phrasal lexicon that directly maps concepts to predefined lexical-phrasal templates, both Penman/KPML and Multex have full knowledge of grammar, and build complex lexicogrammatical structures by "executing" the grammar.
- **clean modularity between linguistic resources and the processes** that operate on the resources. Penman has an independent linguistic resource module which includes the Nigel grammar and the upper-model semantic framework. All the processes such as the text generation algorithm are defined based on the operations that are explainable by these linguistic resources. As we will show later, Multex not only inherits this design but also elevates the modularity between the linguistic resources and the NLP processes to a higher degree.
- **systemic functional theory.** Penman, KPML and Multex are based on the same linguistic theory — systemic functional theory. This compatibility makes it relatively easy for the three systems to share resources between them while each system may focus on some particular aspect of the theory.

Differences. In terms of the system architecture and implementation, Multex differs considerably from Penman and KPML. This implementation is completely new. It is designed to be a small and practical content server to dynamically create multilingual and multimodal contents for different applications. To fulfil this design, Multex would need to interoperate with commercial applications, be object-oriented and be able to access mainstream computing resources eg. multimedia packages and databases. Multex, in its current form, can run as a standalone Java application, as a component in a CORBA environment, as a Web application and as a business object in a three tier client-server environment.

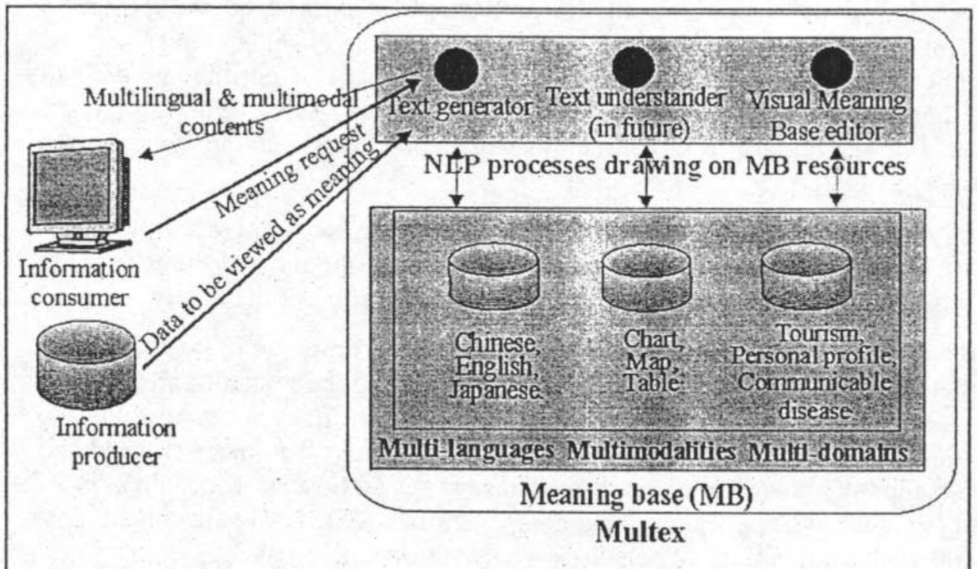
In terms of the text generation technology, Multex differs from the Penman/KPML tradition in the following aspects:

- Multex is designed from the ground up to be a multilingual and multimodal content generator. At present it produces text in English and in limited Chinese and Japanese. It also dynamically constructs labelled maps, charts and tables.
- Multex is discourse-oriented. That is, Multex generates on texts rather than only sentences basis as Penman and KPML do.

- Multex has a higher degree of modularity in the organization of its linguistic resources. It has a self-contained linguistic engine, called the Meaning Base, which manages multilingual lexicogrammar and semantics, multimodal resources and multiple domain models. The meaning base publishes an extensive Application Programming Interfaces (APIs), which NLP processes can use to access and to reason about the vast resources managed in the meaning base.
- Multex uses a completely different generation algorithm from that of Penman/KPML. Chooser-inquiries and system network traversals are absent in Multex. Multex uses interstratal mapping patterns to efficiently draft semantic and lexicogrammatical plans, and it uses constraint-posting and plan-criticising methods to refine, reject and regenerate the drafted plans.
- Multex uses a modular approach to text generation. The generation algorithm consists of a plan controller that controls the execution of a number of small content-creation agents, called Meaning Agents. Each Meaning Agent specialises in generating a particular kind of meaning and modality. Meaning agents for new specific modalities can be easily integrated into this general planning process.
- Multex plans text in two modes: automatic and cooperative. In the cooperative mode, the generation process guides an end-user in their decision-making process, in a fashion similar to wizards in many Windows applications.

2.3 An overview of the Multex architecture

Figure 1 presents an overview of the Multex architecture. The role of the information producer is to create or manage data sources that supply useful data to the information consumer. The information consumer is an agent who needs data from the information producer but cannot process the data directly; the information consumer needs to consume the data in a processed form. Multex's function is to process the data into meaningful information in the form of multilingual and multimodal contents.


Multilingual & multimodal contents
Meaning request
Meaning
Data to be viewed as meaning
Information consumer
Information producer
Text generator
Text understander (in future)
Visual Meaning Base editor
NLP processes drawing on MB resources
Chinese, English, Japanese
Chart, Map, Table
Tourism, Personal profile, Communicable disease
Multi-languages
Multimodalities
Multi-domains
Meaning base (MB)
Multex
Figure 1. Multex architecture

Internally, Multex has two major components, a Meaning Base and a set of NLP processes. The Meaning Base contains the following resource modules: multiple linguistic systems (eg. the lexico-grammar and semantics of English, Chinese, Japanese), multiple semiotic modalities (eg. the resources needed for creating charts and maps) , multiple domain models (eg. the knowledge about tourism and about communicable disease).

In addition, the Meaning Base supplies a full range of methods to populate itself with linguistic resources, as well as the methods to access and reason about the resources it maintains. An NLP process is an application or a service that performs some NLP functionality for the information consumer by drawing on the resources in the Meaning Base. Text generator is a NLP process of Multex, another partially implemented NLP process is a visual navigator of the Meaning Base. In future we can add a text understander to Multex as another "Multex-compliant" NLP process.

3. The Meaning Base

Recent advances in Systemic Functional theory attempts to define the entire systemic linguistic model in a relatively small set of theoretical concepts. This set of theoretical concepts, known as the **systemic metalanguage,** outlines the structure of a linguistic system and provides principles and methodologies for modularizing the linguistic resources, for analysing and interpreting language instances with grammar, and for modelling linguistic processes such as understanding and generation (cf. Matthiessen & Nesbitt 1996, Halliday & Matthiessen in press: Section 1.9, for the conception of the metalanguage). Figure 2 shows a simple taxonomy of the systemic metalanguage. The notion of metalanguage can be usefully applied to NLP systems for two reasons: (1) it provides a comprehensive and theory-motivated map of the resources available in a linguistic system. Resource developers can use this map to structure and develop fragments of linguistic resources, and to reason about the properties of the linguistic resources; (2) linguistic processes can be defined with respect to the necessary resources it draws on.

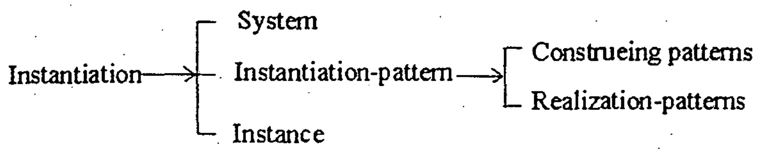


Figure 2. A taxonomy of the systemic metalanguage.

The meaning base is the implementation of the systemic metalanguage plus the linguistic resources maintained by Multex. It is the linguistic engine of Multex. The metalanguage concepts are made operational by being implemented as Java classes. Access to linguistic resources and all reasoning about the linguistic resources are defined as methods in the classes representing the metalanguage concepts. A Java-based Meaning Base Application Programming Interface (MB API), which consists of around 60 metalanguage concepts and over 400 methods, is available for programmers to create NLP processes. In a sense, the systemic metalanguage is the protocol a NLP process talks with the meaning base.

Linguistic resources are specified in a formalism called the Meaning Base Modelling Language (MBML). When the meaning base is being loaded, the linguistic resources are compiled, optimised and stored as objects in the meaning base. Space does not permit us to provide the details of MBML, but we will present some examples.

```
FieldType {
        ID(communicable-disease-outbreak)
        isa(violent-social-event)
        // define slots
        slot(disease :type disease)
        slot(cause :type animal)
        slot(range :type place)
        slot(cases :type human :unify-with victim)
        slot(fatality :type human :unify-with victim)
        slot(medical-investigate :type investigate)
        slot(trend :type disease-outbreak-profile)

    ConstrueStrategy Brief-report() {
        ideationObj { ID(?general-report) type(addition)
                        slots(:nuclear ?report-incidence :satellite *.trend) }
        ideationObj { ID(?report-incidence) type(addition)
                        slots(:nuclear ?report-outbreak :satellite *.medical-investigate)}
        ... ...
    }
    ConstrueStrategy Detailed-report(report-source, confidence-level) {
          ... ... // semantic structure for detailed report
    }
  // other construe strategies
}
```

Figure 3. Definition of the FieldType *communicable-disease-outbreak.*

This example defines a domain concept called *communicable-disease-outbreak.* In addition to specifying ISA relations and slots, one can define any number of construe strategies for a domain concept.

A construe strategy is in fact a set of parameterized semantic objects that construe a given domain situation as meaning in a specific communicative context. Let's consider another example:

```
RealizeAs {
    realizing(cause-effect)
    register(communicable-disease-report)
    Language(ENGLISH)
    //   meaning pattern
    IdeationObj { ID(?causation) type(cause-effect) slots(:cause ?cause :effect ?effect) }
    IdeationObj { ID(?cause) type(figure) slots(:time-loc ?Time :space-loc ?Place) }
    //   wording pattern
    LexgrmrObj {
            ID(?relational)
            type(relational relational-cause)
            slots(:identified ?x :identifier ?y :time-loc ?PP1  :space-loc ?PP2)
            map(?cause ?x) map(?effect ?y) map(?causation ?relational)
            map(?time ?pp1) map(?place ?pp2)
    }
}
```

Figure 4. Definition of a Realization pattern for the concept *cause-effect*.

Figure 4 defines an interstratal mapping pattern. It encodes the following linguistic knowledge: for the register *communicable disease report*, the conjunctive relation *cause-effect* at the semantic level can be realized lexicogrammatically as "*X causes Y*" in English, the temporal and spatial circumstance of the cause event should be realized as the *Time-loc* and *Space-loc* functions of the "*X causes Y*" clause.

```
LexgrmrType {
  Language(English)
  ID(outbreak)
  ISA(nominalize-neutral-material)
  FromAbove(break-out)
  slot(thing :value $word("outbreak"))
  slot(qualifier :map-from actor) }
```

Figure 5. Definition of the lexicogrammatical type *outbreak*.

Figure 5 defines a lexicogrammatical system *outbreak*. The `FromAbove` clause indicates that this system prototypically realizes the semantic concept *break-out*. The `:map-from actor` term specifies that the *Qualifier* function is mapped from the *Actor* slot of the *break-out* event (i.e. *what breaks out*).

The meaning base has many other important features, eg. management of multilingual and multimodal resources. In addition, each metalanguage concept is associated with a visualizer (although this is only partially implemented), which enables a meaning base visualization tool to be easily constructed.

4. The Text Planner

4.1 The text planning architecture

The Multex text planner is structured into two layers: the plan control layer and the meaning agent layer. The plan control layer implements a general-purpose constraint-based planner, and the meaning agent layer maintains a number of processes specializing in creating certain kinds of meaning. This architecture is inspired by the work on meta-planning (Hayes-Roth & Hayes-Roth 1978, Stefik 1981).

The plan control layer consists of the processes for: (1) creating goals to be solved by the meaning agents; (2) spawning, scheduling and starting meaning agents; (3) introspecting on the local plans generated by the meaning agents. Plan introspection includes deciding when the planning should stop, assimilating local plans into a global plan, posting constraints entailed by sub-plans to the global plan. Here ***plan*** and ***sub-plan*** refer to a set of partially specified linguistic objects generated by meaning agents.

The meaning agent layer consists of a number of meaning agents. A meaning agent is a self-contained process that creates a specific kind of meaning in the form of semantic and lexicogrammatical objects by instantiating resources in the meaning base. Table 4.1 summarises the meaning agents available in Multex as well the the meaning base resources they rely on.

Table 4.1 Meaning agents defined in Multex.

Meaning Agents	Function	Meaning Base resources accessed
Construe	construe domain situations as meanings (ie. semantic objects)	FieldTypes, Construing Strategies, domain facts, domain examples, IdeationTypes.
VisualConstrue	the same as Construe, except that it uses a visual tool to interact with end users.	The same as above.
Realize	realize meanings by generating lexicogrammatical objects.	IdeationTypes, Realization patterns, lexicogrammatical types.
VisualMapMaker	realize meanings in the map modality, using a visual map editor.	IdeationTypes, Realization patterns, semiotic systems of map.
VisualChartMaker	realize meanings in the chart modality, using a visual chart editor. (partially implemented)	IdeationTypes, Realization patterns, semiotic systems of chart.
VisualTableMaker	realize meanings in the table modality, using a visual table editor. (not implemented)	IdeationTypes, Realization patterns, semiotic systems of table.
ReferExpressionMaker	create referring expressions. (not implemented)	FieldTypes, Construe Strategies, domain facts, domain examples, IdeationTypes.

In fact, the meaning agents **Construe** and **Realize** alone suffice for the whole text generation process, because from a theoretical point of view, text generation consists of exactly two steps: mapping contextual situation to meaning and mapping meaning to wording. The first step is carried out by the Construe agent and the second step, by the Realize agent. The rest of the meaning agents provide additional functionality that is designed for specific applications.

Moreover, a meaning agent has to implement a protocol, called meaning agent protocol, in order to be administered by the plan control layer. The protocol includes methods for determining whether goal has achieved, for inferring more goals to achieve, for searching the meaning base for appropriate resources and for turning them into a plan, etc.

4.2 An example of text planning

Here we will give a brief example of text planning in Multex. The input to the text generator is provided as a meaning request, which is passed from the information consumer either as a stream, or as an object. Figure 6 shows a meaning request in the form of a stream.

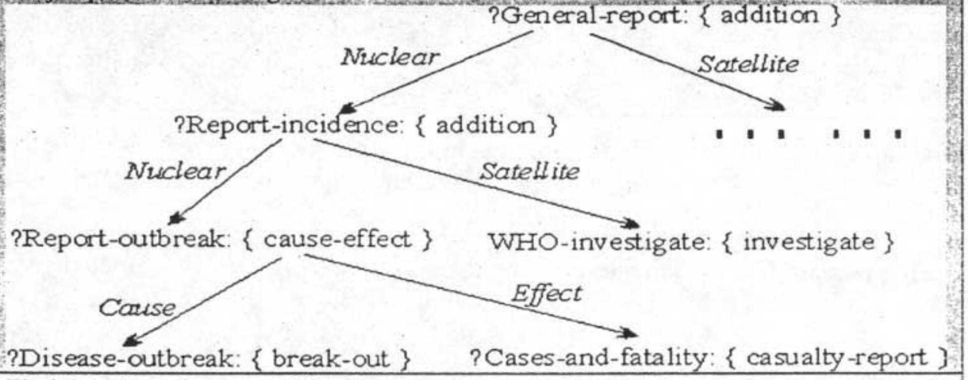
```
MeaningRequest {
    MeaningService(brief-report)
    register(CommunicableDiseaseReport)
    Topics(diarrhoea-outbreak)
    Template {
      ID(diarrhoea-outbreak)
      type(communicable-disease-outbreak)
      slots(  :disease     <ebola-disease non-specific>
              :cause       <rat plural>
              :time-loc    outbreak-time
              :space-loc   outbreak-place
              :cases       human-1
              ... ...)
    }
```

```
template { ID(outbreak-time)  type(in-the-time) slot(anchor april-1995)}
     . // other specification of contextual objects.
```

Figure 6. An example of meaning request.

The space limit does not permit us to give a detailed trace of the text planning process. We can only list some salient points in the generation process in Table 2.

Table 2: Critical steps in the text planning process.

Plan control	Meaning Agents layer	Descriptions of planning
spawn		create the first meaning agent: Construe(diarrhoea-outbreak.brief-report())
	Construe(diarrhoea-outbreak. brief-report())	start the Construe Meaning agent. This results in instantiating the Construe Strategy *brief-report* shown in figure 3. A semantic network as shown below is created.  ?General-report: { addition } Nuclear Satellite ?Report-incidence: { addition } . . . . . . Nuclear Satellite ?Report-outbreak: { cause-effect } WHO-investigate: { investigate } Cause Effect ?Disease-outbreak: { break-out } ?Cases-and-fatality: { casualty-report }
Introspect		Find the object ?cases-and-fatality to be a domain concept that needs to be construed as meaning as well.
......		execute all Construe meaning agents to create a complete semantic network.
spawn		spawn a **Realize** meaning agent for each semantic object in the semantic network. The text planner performs a topological sorting on the semantic network so that the less dependent nodes get realized first, eg. the decision for realizing *?Report-outbreak* is made earlier than the decision for realizing *?Disease-outbreak* and *?cases-and-fatality*
......		executing a series of Realize meaning agents.
	Realize(?Report-outbreak)	The realization pattern in Figure 4 is instantiated. An "*X causes Y*" clause is added to the partially generated text.
Introspect		The planner infers that, in order to fit into the *X causes Y* clause, *?Disease-outbreak* has to be realized as a nominal group. It then posts this constraint to the overall text plan being maintained.
	......	realizing other semantic objects.
	Realize(?Disease-outbreak)	Although disease outbreak is prototypically realized as a clause in English eg. "*disease breaks out*", this realization is in conflict with the constraint already imposed on ?Disease-outbreak, which requires the object to be realized as a nominal group. Multex then searches the meaning base for a solution that compromises this constraint. As a result, it selects and instantiates the resource shown in Figure 5 and creates the nominal group: *an outbreak of ebola-disease*.
	...	realizing the rest of the semantic objects accordingly.

Multex finally generates the following passage from the meaning request in Figure 6:

"An outbreak of ebola disease, which was caused by rat, in Kikwit, Zaire has led to 189 cases and 59 deaths in April 1995. The world health organization investigated the disease on 10 May 1995. Incidence of ebola disease increased in 1995."

Multex's generation is robust. For example, all the slots in Figure 6, except the disease slot, can be totally or partially omitted, and Multex can still produce coherent text. If all optional slots are missing, Multex generates the text *there is an outbreak of disease.*

5. Multex working with production applications: HINTS

Multex has been designed to be able to work together with other NLP systems in an integrated system capable of various "information processing" tasks in addition to generation. One such integrated system is the HINTS system currently being developed by DSTO, Canberra, with contributions by the Systemic Meaning Modelling Group at Macquarie University and by the team working on the Fact Extractor at DSTO, Adelaide. (HINTS may be compared with MAGIC, a system capable of generating multimodal healthcare briefings [McKeown, Jordan & Allen, 1997]; but whereas MAGIC is intended to produce multimodal briefings about particular patients for "time-pressured caregivers", HINTS is a resource for health officers who monitor communicable diseases around the world based on collected documents.

HINTS is a system developed to process information concerning communicable diseases. It has been designed for health officers of various kinds to help them cope with the fast flow of information and the daunting demand for regular reports and briefings of various kinds. HINTS integrates a number of systems that it can make demands on for different kinds of information processing services. From the point of view of Multex, HINTS constitutes an information production, for which Multex provides a service in the form of multimodal communicable disease reports. In addition, Multex provides a resource that is used by other components of the HINTS system — the meaning base.

Let us describe HINTS first in terms of the general work flow and then discuss its significance for the Multex generator. Users interact with HINTS through friendly GUIs; they have all been designed jointly with representative users. A user will start a HINTS session by retrieving documents according to a certain retrieval template — at present, this is just a collection of key words. For example, the user might want to retrieve all documents that are concerned with (outbreaks of) *Ebola* in a certain region over a certain period of time. These documents are retrieved either from an existing collection of documents or from on-line sources via the Internet.

Once the relevant documents have been retrieved, the user can ask HINTS for a summary. The summarizer that HINTS calls upon at present operates at fairly low levels of abstraction; it relies on aspects of the layout of a document (eg. the subject header of e-mail messages), on paragraph initial placement, on conjunctive markers such as *in summary,* and the like. It does not engage in any lexicogrammatical or semantic processing of the texts.

The user can also ask HINTS to extract "facts" from the collection of documents. HINTS uses the Fact Extractor (FE) developed by Peter Wallis and his team (e.g., Wallis & Chase, 1997). FE operates with a set of templates for extracting information about communicable diseases. These templates include dates, locations, cases and disease outbreaks. They consist of slots or roles that have to be filled by FE with values extracted from the collection of documents. They are all derived from Multex's meaning base and are represented within FE by means of regular expressions. Once FE has extracted the relevant values, it fills in "forms" based on the templates and if the user wants to generate reports based on the information extracted, a meaning request is generated from the templates and passed over to Multex.

Multex then construes the information in terms of its domain model of communicable diseases. Since the templates used by FE are derived from the Multex meaning base, all the information they provide can be classified according to existing domain types. However, Multex will have to draw on domain knowledge to expand the information to the point where it can support generation. Once Multex has processed the information it receives from FE, it starts the incremental generation process sketched above. This will include not only decisions controlling Multex's generation process but also opportunities for the user to include quoted material from any of the documents that have been retrieved and to add his/her own text. The latter option means that users can add information that embodies a fair amount of interpretation. In the register of communicable disease reports, this information has an interpersonal orientation: either it represents the user's expert evaluation of the information produced automatically by Multex ('how common?, how likely?') or it represents the user's recommendation (actions that should be taken by health authorities based on the information produced by Multex). This is a case where the prototypical **Construe** meaning agent is not adequate, a **VisualConstrue** meaning agent is hence supplied to meet the additional demand of HINTS. When the user is satisfied that the generation process has finished, Multex produces a document in HTML format and it is handed over to a browser for display.

As this brief description indicates, HINTS is an interesting, information-rich environment for exploring multimodal generation. In particular, it is worth noting that Multex receives information that has been extracted from written documents, but it produces presentations that may include charts and labelled maps. For example, Multex is able to retrieve a relevant map from the meaning base based on the spatial information in the meaning request and then label some hot spots on the map with the text fragments it produces. It is also worth noting that in the HINTS environment, the process of generation is very much a collaborative effort. The user exercises control over information sources and s/he can make decisions during the incremental generation process. This means that in this environment Multex functions as a writer's tool; and it can be used in preparing regular briefings or web pages. Further, although Multex's multilingual capability is not presently deployed in HINTS, Multex is able to generate the multimodal reports in languages other than English. For example, users could extract information from English documents and use Multex to generate a multimodal report in Chinese. This capability can be of considerable value, as is demonstrated by the TREE project (Somers et al, 1997): the TREE system can search the Internet for job ads and then summarize these in various languages.

6. Conclusion

In this paper, we have attempted to give a sense our "contextualized" approach to text generation. By **contextualized**, we mean that we are taking an ecological approach to text generation where it is located in relation to application environments and is linked to an environment of support tools. This move towards contextualized text generation capabilities resonates with the field in general and reflects the growing maturity of the "art" of text generation. Having discussed the two kinds of context — application and development — and having presented the architecture of Multex, we can now be more precise about the way in which they relate to the core Multex system than we were able to be in our introduction. Both application and development systems relate to resources within Multex; but at present they relate to different levels within Multex. HINTS and similar systems relate to the highest levels of resources — to context.

Application relates to the highest level of organization in the first instance — to context. As we have noted, this is the level where Multex can work with a complete model of its operational environment. Context is thus the appropriate level to capture generalizations about systems of different kinds — systems such as the Fact Extractor of HINTS or image interpretation systems in other environments.

We have only been able to provide a brief sketch, leaving out many operational details. In conclusion, we would just like to add something about developments in the near future. We will add spoken output to Multex by linking it to a speech synthesizer; this work will be based on the flexible design of the expression level that we noted earlier and it will be guided by previous work by John Bateman and Elke Teich linking KPML to a speech synthesizer (Teich et al, 1997). We will also develop an interface system that is capable of translating KPML specifications into Multex specifications and vice versa. Further, we plan to integrate a development and reference workbench that we are building, SysAm (for Systemic Amanuensis) and Multex so as to provide a homogeneous generation and development environment of the kind currently offered by KPML.

7. References

Bateman, John A., Christian Matthiessen, Keizo Nanri & Licheng Zeng. 1991. The rapid prototyping of natural language generation components: an application of functional typology. *Proceedings of the 12th international conference on artificial intelligence*, Sydney, 24-30 August 1991. Sydney. San Mateo, CA: Morgan Kaufman.

Bateman, John A. & Elke Teich. 1995. Selective information presentation in an integrated publication system: an application of genre-driven text generation. *Information Processing and Management* 31.5: 753-768.

Bateman, John A. 1996. KPML Development Environment: multilingual linguistic resource development and sentence generation. Release 0.9, March 1996. IPSI/ GMD, 15 Dolivostrasse, Darmstadt, Germany.

Halliday, M.A.K. 1994. *An introduction to functional grammar.* London: Edward Arnold. Second. 2nd revised edition.

Halliday, M.A.K. & C. Matthiessen, in press. *Construing experience through meaning: a language-based approach to cognition.* London: Cassell.

Hayes-Roth, B. and Hayes-Roth, F. (1978). "A Cognitive Model of Planning", In Allen, Hendler and Tate (eds) (1990). *Readings in Planning.* 245-262. San Mateo, CA: Morgan Kaufmann Publishers, Inc.

Hovy, Eduard H. 1993. Automated Discourse Generation Using Discourse Structure Relations. *Artificial Intelligence* 63(1-2) Special Issue on Natural Language Processing (341-386).

Kittredge, R. 1987. The significance of sublanguage for automatic translation. S. Nirenburg (ed.), *Machine translation: theoretical and methodological issues.* Cambridge: Cambridge University Press.

McKeown, Kathleen R., Desmond A. Jordan & Barry A. Allen. 1997. Language generation for multimedia healthcare briefings. In *Proceedings of the Fifth Conference on Applied Natural Language Processing,* 31 March - 3 April 1997, Washington Marriott Hotel, Washington, DC, USA. Association for Computational Linguistics. pp. 277-82.

Mann, William C. & Christian M.I.M. Matthiessen.. 1985. Demonstration of the Nigel Text Generation Computer Program. Benson & Greaves (ed.), *Systemic Functional Approaches to Discourse.* Norwood: Ablex.

Mann, William C. and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text* 8.3: 243-281.

Matthiessen, Christian M.I.M. 1983. Systemic grammar in computation: the Nigel case. The First Annual Conference of the European Chapter of the Association for Computational Linguistics. Pisa.

Matthiessen, Christian M.I.M. 1995. *Lexicogrammatical cartography: English systems.* Tokyo: International Language Sciences Publishers.

Matthiessen, Christian M.I.M & Christopher Nesbitt. 1996. On the idea of theory-neutral descriptions. Hasan, R., C. Cloran & D.G. Butt (eds.). *Functional descriptions: theory in practice.* Amsterdam: Benjamins..

Matthiessen, Christian M.I.M. & John A. Bateman. 1991. *Systemic linguistics and text generation: experiences from Japanese and English.* London: Frances Pinter.

Radev, Dragomir R. & Kathleen R. McKeown. 1997. Building a generation knowledge source using internet-accessible newswire. In *Proceedings of the Fifth Conference on Applied Natural Language Processing,* 31 March - 3 April 1997, Washington Marriott Hotel, Washington, DC, USA. Association for Computational Linguistics. pp. 221-8.

Somers, Harold, Bill Black, Joakim Nivre, Torbjörn Lager, Annrosa Multari, Luca Gilardoni, Jeremy Ellman & Alex Rogers. 1997. Multilingual generation and summarization of job adverts: the TREE project. In *Proceedings of the Fifth Conference on Applied Natural Language Processing,* 31 March - 3 April 1997, Washington Marriott Hotel, Washington, DC, USA. Association for Computational Linguistics. pp. 269-76.

Stefik, M. (1981). "Planning and Meta-Planning (MOLGEN: Part 2)", *Artificial Intelligence* (16) 141-170.

Teich, Elke, E. Hagen, B. Grote & J.A. Bateman. 1997. From communicative context to speech: integrating dialogue processing, speech production and natural language generation. Speech Communication 21.1-2: 73-99.

Wallis, Peter & Greg Chase. 1997. An Information Extraction System. In *1997 Australasian Natural Language Processing Summer Workshop,* Macquarie University, February 1997.