smartFIX: An Adaptive System for Document Analysis and Understanding

Bertin Klein[1], Andreas R. Dengel[1], and Andreas Fordan[2]

[1] German Research Center for Artificial Intelligence (DFKI)
P.O.Box 2080, D-67608 Kaiserslautern, Germany
{dengel,klein}@dfki.de
[2] INSIDERS technologies GmbH
Brüsseler-Str. 1, D-67657 Kaiserslautern, Germany
A.Fordan@insiders-technologies.de

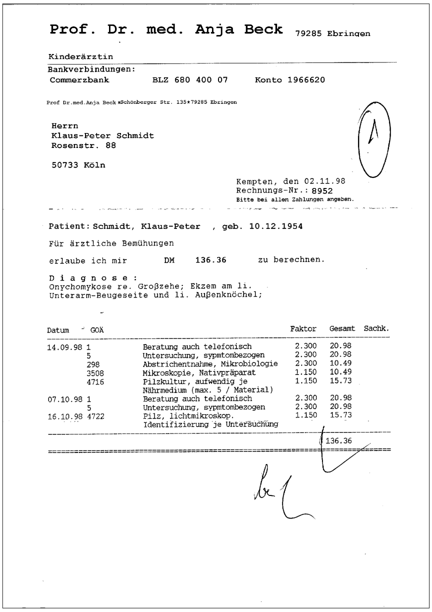
Abstract. The internet is certainly a wide-spread platform for information interchange today and the semantic web actually seems to become more and more real. However, day-to-day work in companies still necessitates the laborious, manual processing of huge amounts of printed documents. This article presents the system *smartFIX*, a document analysis and understanding system developed by the DFKI spin-off insiders. During the research project "adaptive Read", funded by the German ministry for research, BMBF, *smartFIX* was fundamentally developed to a higher maturity level, with a focus on adaptivity. The system is able to extract information from documents – documents ranging from fixed format forms to unstructured letters of many formats. Apart from the architecture, the main components and the system characteristics, we also show some results from the application of *smartFIX* to representative samples of medical bills and prescriptions.

Keywords: Document analysis and understanding, document classification, information extraction, table extraction, extraction optimization, extraction verification, industrial invoice processing

1 Introduction

About 1.2 million printed medical invoices arrive at the 35 German private health insurance companies every day. Those invoices amount to 10% of the German health insurance market and they are actually maintained by printed paper invoices. Figure 1 shows examples of such printed invoices. Until recently the processing of these invoices was done almost completely manually. In addition to the tedious task of initiating every single payment by transcribing a number of data fields from varying locations on the paper documents into a computer, this had the serious disadvantage that only a small number of inconsistent and overpriced invoices were discovered. Conservative estimates predict savings in the range of several hundred million Euros each year if this process could be automated reliably.

A. Dengel et al. (Eds.): Adaptive READ Research Project, LNCS 2956, pp. 166–186, 2004.



Prof. Dr. med. Anja Beck 79285 Ebringen

Kinderärztin

Bankverbindungen:
Commerzbank BLZ 680 400 07 Konto 1966620

Prof Dr.med.Anja Beck *Schönberger Str. 135*79285 Ebringen

Herrn
Klaus-Peter Schmidt
Rosenstr. 88

50733 Köln

Kempten, den 02.11.98
Rechnungs-Nr.: 8952
Bitte bei allen Zahlungen angeben.

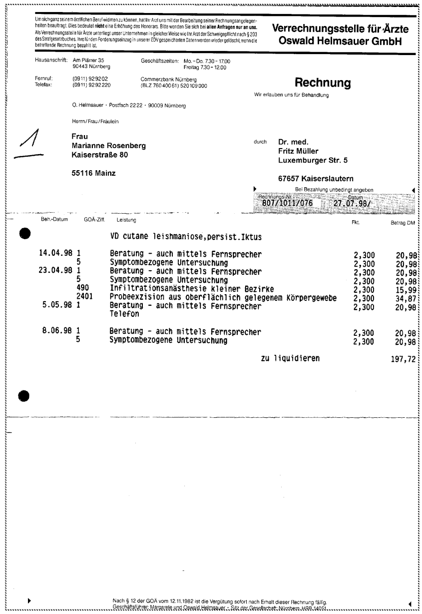
Patient: Schmidt, Klaus-Peter , geb. 10.12.1954

Für ärztliche Bemühungen

erlaube ich mir DM 136.36 zu berechnen.

Diagnose:
Onychomykose re. Großzehe; Ekzem am li.
Unterarm-Beugeseite und li. Außenknöchel;

Datum	GOÄ		Faktor	Gesamt	Sachk.
14.09.98	1	Beratung auch telefonisch	2.300	20.98	
	5	Untersuchung, sypmtombezogen	2.300	20.98	
	298	Abstrichentnahme, Mikrobiologie	2.300	10.49	
	3508	Mikroskopie, Nativpräparat	1.150	10.49	
	4716	Pilzkultur, aufwendig je Nährmedium (max. 5 / Material)	1.150	15.73	
07.10.98	1	Beratung auch telefonisch	2.300	20.98	
	5	Untersuchung, sypmtombezogen	2.300	20.98	
16.10.98	4722	Pilz, lichtmikroskop. Identifizierung je Untersuchung	1.150	15.73	
				136.36	



Um sich ganz seinem ärztlichen Beruf widmen zu können, hat Ihr Arzt uns mit der Bearbeitung seiner Rechnungsangelegenheiten beauftragt. Dies bedeutet **nicht** eine Erhöhung des Honorars. Bitte wenden Sie sich bei **allen Anfragen nur an uns.** Als Verrechnungsstelle für Ärzte unterliegt unser Unternehmen in gleicher Weise wie Ihr Arzt der Schweigepflicht nach § 203 des Strafgesetzbuches. Ihre für den Forderungseinzug in unserer EDV gespeicherten Daten werden wieder gelöscht, wenn die betreffende Rechnung bezahlt ist.

Verrechnungsstelle für Ärzte
Oswald Helmsauer GmbH

Hausanschrift: Am Plärrer 35, 90443 Nürnberg
Geschäftszeiten: Mo.-Do. 7.30 - 17.00, Freitag 7.30 - 12.00
Fernruf: (0911) 929202
Telefax: (0911) 9292220
Commerzbank Nürnberg (BLZ 760 400 61) 520109000

Rechnung
Wir erlauben uns für Behandlung

O. Helmsauer · Postfach 2222 · 90009 Nürnberg

Herrn/Frau/Fräulein

Frau
Marianne Rosenberg
Kaiserstraße 80

55116 Mainz

durch Dr. med.
Fritz Müller
Luxemburger Str. 5

67657 Kaiserslautern

Bei Bezahlung unbedingt angeben
Rechnungs-Nr. 807/1011/076 Datum 27.07.98/

Beh.-Datum	GOÄ-Ziff.	Leistung	Fkt.	Betrag DM
		VD cutane leishmaniose,persist.Iktus		
14.04.98	1	Beratung - auch mittels Fernsprecher	2,300	20,98
	5	Symptombezogene Untersuchung	2,300	20,98
23.04.98	1	Beratung - auch mittels Fernsprecher	2,300	20,98
	5	Symptombezogene Untersuchung	2,300	20,98
	490	Infiltrationsanästhesie kleiner Bezirke	2,300	15,99
	2401	Probeexzision aus oberflächlich gelegenem Körpergewebe	2,300	34,87
5.05.98	1	Beratung - auch mittels Fernsprecher Telefon	2,300	20,98
8.06.98	1	Beratung - auch mittels Fernsprecher	2,300	20,98
	5	Symptombezogene Untersuchung	2,300	20,98
		zu liquidieren		197,72

Nach § 12 der GOÄ vom 12.11.1982 ist die Vergütung sofort nach Erhalt dieser Rechnung fällig.
Geschäftsführer: Margarete und Oswald Helmsauer · Sitz der Gesellschaft: Nürnberg HRB 14051

Fig. 1. Examples of medical invoices common in Germany

In 1999, a consortium of German private health insurance companies ran a public benchmark test of systems for Document Analysis and Understanding (DAU) for the respective private insurance sector. The benchmark was won by and proved the suitability of *smartFIX* (smart *F*or *I*nformation e*X*traction). This initial version of *smartFIX* was developed by a spin-off company of the DAU group at DFKI, INSIDERS (www.insiders-technologies.de, founded in 1999 by Prof. Dengel), thinking that the available DAU technology after a decade of focused research was ready to construct a versatile and adaptive DAU system [1, 2, 3]. After the premise of the identification of a viable type of application scenarios, the well-directed research on a feasible combination of available methods and completion with new methods could be successfully accomplished. This background probably explains the clear suitability of *smartFIX* for the project *adaptive Read*, which INSIDERS joined after the benchmark. Funded by the German ministry for research, BMBF, *smartFIX* was significantly extended and raised to a new level of ability to be adapted. The result of this project was further extended to several products for different industries, one e.g., in cooperation with four insurance companies for the analysis of printed medical invoices: *smartFIX healthcare*. In the meantime *smartFIX healthcare* has been established the standard product for the private health insurance sector. *smartFIX* was the result of several man-years investment. The brainpower in *smartFIX healthcare* can be estimated in a larger dimension.

The first two important facts about the target domains, in the following represented with the example of the health insurance domain are:

1. Invoices are more complex than forms.
2. Every invoice is inspected by a human operator anyhow.

Therefore, the DAU task for invoices requires more than the more simple methods that suffice for forms - a challenge for the DAU technology developers in the project. But at the same time, the insurance auditors can be assured, that the economic success does not only start after a distant breakthrough, but every little successfully implemented DAU step reduces the human operators workload right away. Every correctly recognized data item saves typing and can be logically and numerically checked. Diagnoses can be automatically coded into ICD 10 (the international standard code for diagnoses). Actually, even with no recognition results, the efficient user interface of the result viewer facilitates the processing of scanned invoices.

In general, *smartFIX healthcare* is not limited to the domain of medical bills but is applicable also to most kinds of forms and many unstructured documents. In the following, we will describe the major characteristics of the system including architecture, main components as well as some technical aspects. At the end of the paper, we will present run-time results of *smartFIX healthcare* applied to medical bills based on a recent evaluation of a large private health insurance company, which processes several tens of thousands of bills and prescriptions every day.

2 SmartFIX Healthcare

Nowadays, at least since the CommonKADS projects series [4], it is known how indispensable the analysis of the intended application is for successful knowledge-intensive software projects. Then the software has to be adapted in detail to the requirements. Facing the needs of a group of companies, this requirements-centered approach is even more important and challenging, e.g. the number of example documents, which the companies' representatives thought to convey their needs, rapidly grew to much more than 10000. It is important to understand what one can learn from these documents and alone their sheer number is obviously very challenging. So, the original ideas of our DAU researchers underlying *smartFIX* were refined with the needs of the insurance companies.

The insurance companies required from *smartFIX healthcare* (requirements which had certainly not been the primary concern in the ten years work of our DAU researchers):

1. Verified economic advantages (qualitative, quantitative)
2. Reliability (error recovery, error rate, error statistics)
3. Scalability
4. Plug-ability to their workflow (people, tasks, databases, archives)

The actual economic advantages were simply required to be tested. Actually, it turned out we had no problems at all to reach the required measures (see Results section). Generally, this seems to be not a problem also elsewhere: a questioning in German industry, which investigated the results of investments into DAU tools, indicates that companies typically hesitate to spend money first, but are very delighted about the achieved results (and return of investment) later. [12]

Scalability had already been targeted and was mainly available: *smartFIX* is a distributed system, CORBA on networked PCs, which can easily spread its single-

analysis processes over many CPUs and still be controlled from one desktop. One aspect of reliability is achieved with a central transaction memory database. Thus, the only remaining tasks were to extend and adapt *smartFIX* to the insurers workflow requirements and to learn about reliability in terms of stable error-rates.

Extending the most basic principle that the design of DAU systems should not be technology-driven, but explicitly requirements-driven (and thus adapted to the users needs) [4], with our approach we came to the following guiding principles for the design of DAU systems:

1. **Compositionality:** A versatile system has to have a significant spectrum of different basic DAU components; one paradigm alone will almost surely fail to solve a real problem. – Complex methods, which at first glance are very hard to conceive, are obtained from deliberate, small combinations of simple methods. [3]
2. **Practice principle:** There is no way to get around errors. Thus it is important to help users to discover, judge, and either tolerate, or to correct errors. This is especially critical as the user in every day practice is left alone with the control of the system and the responsibility for its results. – The real DAU technology must be powerfully accompanied by technology for logging, tracing, visualizing, interactive testing, statistics.
3. **Epistemological adequacy:** The basic DAU components must be bundled to greater analysis building blocks (we call them "scripts" later in this paper), which are made available to the user. The user has to perceive document characteristics and map those to the scripts. Thus the scripts should be meaningful to the user, easy to memorize and use. Good script metaphors also guide the perception of the document characteristics. Later these scripts might report success or error messages. – This long standing AI principle implies that so-called "syntactic sugar" must not be taken lightly, but can actually make a difference. Beware of software engineers who disregard it.
4. **Constructivism:** The philosophic principle of constructivism says that every knowledgeable agent, human or machine, has a different account of reality and there is no objective truth at all. For a DAU system it means: (a) That every scenario will always require at least one aspect, which is outside of the capabilities of the system at hand. Thus it is important to allow users access to manipulate intermediate results. (b) That two persons at a time (and even often one person at two times) disagree on facts significantly often (cf. TREC human evaluators cross comparison [5]). Thus it is important that the DAU system gives feedback and continuously makes transparent, which information it receives from a user and how this information will be used.

3 System Architecture

The capabilities of *smartFIX healthcare* are diverse, and so is its architecture. The machinery alone to control the executables and configurations according to the needs of its owner – if smartFIX is locked up in a cellar or security zone, spread over several or many analysis computers, connected to company databases – is considerably tricky. However, in the end, and after having learnt many little but important details, e.g. that there are document analysis verifying personnel who cannot press

<Ctrl>+<F12> because they have only one hand: the familiar DAU methodology is still the heart of it all.
The architecture, i.e. the main components, are shown in Figure 2. This section's overview of main components and supporting components is succeeded in the following sections by a sketch of the *DocumentManager* through which the system is instructed and after that an explanation of the most central DAU component, the *Analyzer*. (*Analyzer*s are mainly the component, which is cloned and spread over the CPUs of a networked system implementation.) The *Improver*, though part of the *Analyzer*, is focused thereafter in a separate section and finally also the *Verifier*, the module to check and correct the analysis results.

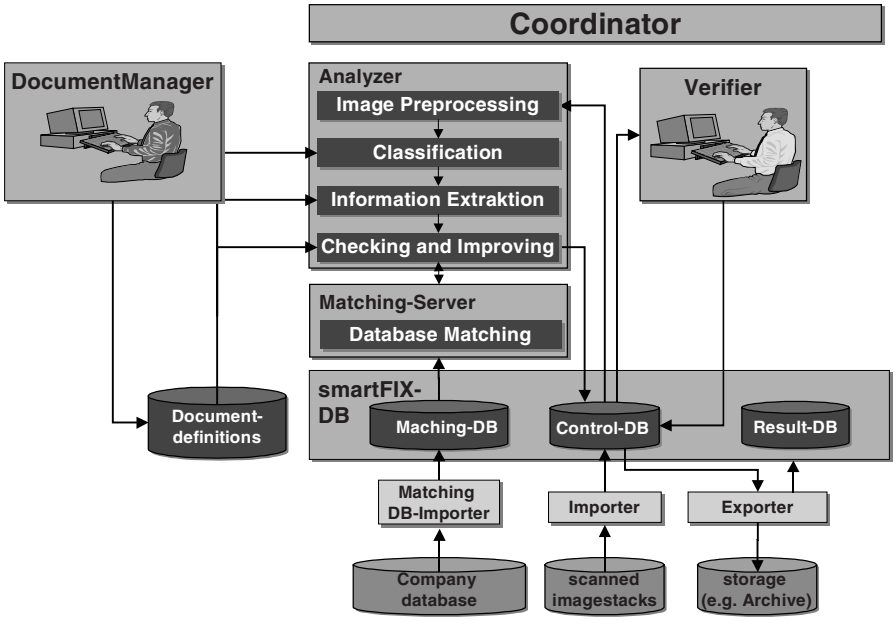


Fig. 2. System architecture of *smartFix*

3.1 Main Modules

In order to apply *smartFIX healthcare* to a new DAU problem, it is necessary to pin down the specific document types and the "what and how" of information extraction. The resulting "document information" –some prefer to call this the "processing knowledge"– is configured with the *DocumentManager*. (Section 4 treats the *DocumentManager* including a screenshot.)

Then the system can be started with the *Coordinator:* a control panel, from which DAU-processes, i.e. *Analyzers*, can be started on a freely-configurable network of PCs (so far we tested with over 60); one *Analyzer* per CPU as a rule of thumb. The *Coordinator* also starts some other processes, first now, two database processes used by the Analyzers: the *Matching-Server* provides very fast retrieval ("matching") on

company knowledge from the *matching data base*. The *Matching Database*, mostly contractual data, is a working copy, which can be updated with a handy tool, the *Matching Database Importer*, from company databases. The *Control Database* is the central working memory of *smartFIX*. It could be regarded the blackboard of the whole architecture.

The *Coordinator* controls an *Importer* process, which transfers document images from a predefined source into the *Control Database*, together with possibly known information. Any idling *analyzer* will check out, process and check in again documents in the *Control Database*. Successful information extraction provided, human agents who are logged on with a *Verifier* process, are prompted the DAU results. With the database it is assured, that not a single bit is lost, even in the event of a power cut. Finished documents are transferred out of the system by an *Exporter,* typically into archive systems.

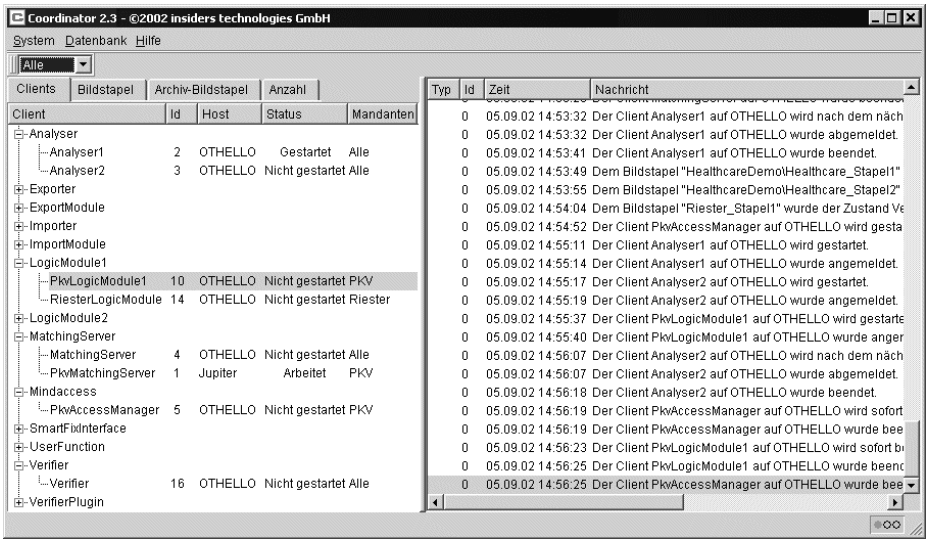


Fig. 3. The *Coordinator* controls the running and idling clients on remote hosts (left) and the status messages (right) of a productive smartFIX healthcare.

3.2 Supporting Components

The main architecture of *smartFIX healthcare* is completed with some more components, helping the user to direct and to understand the system. We briefly list some, but space prohibits to really elaborate on them.

The *Configurator* is a graphical editor for a number of global system values, e.g. a couple of parameters of image pre-processing, and for non-disclosure levels of document classes and user-groups.

The *DCAdmin* (Decision Classifier Administrator) is a graphical interface for the training of a classification component, called “mindaccess”, which exploits machine learning methods to learn semantic similarity from examples (see Section 5.2). The contents of a document very often “tell” its class.

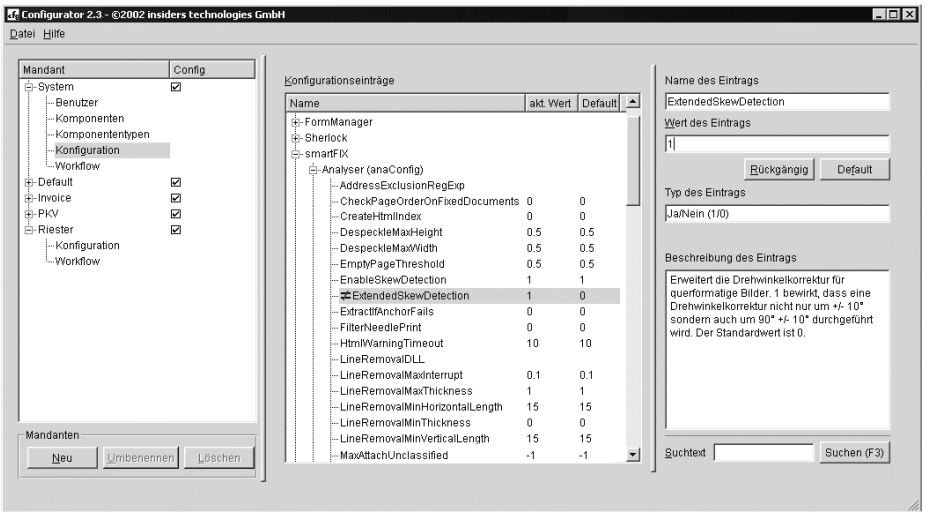


Fig. 4. The *Configurator* allows to create virtual sub-systems (left), to overview values and defaults of their sub-system-wide global system parameters (middle) and to edit them (right). The Figure displays the activation of "ExtendedSkewDetection" by setting it to 1.

All the different system modules adhere to one message format which can be visualized by *LogView*. This tool allows to log onto some module which runs somewhere in the system, on some of the possibly many host machines. It allows for a very efficient overview as well as debugging of the functioning of modules, due to its structural approach. It is equally used by our system developers and end-users.

The *Reporter* allows to query, display and print statistical data from the result database, like the number of pages, documents, and data fields read, the time needed for classification, extraction, the calculated reliabilities and also finally the actual reliability, i.e. the manual corrections.

The *Monitor* watches the system state by scanning all messages and can be configured either to mail, or to be queried by an SNMP management application or to send a so-called "trap"-message via SNMP if the system does not run stable.

The *StatisticTool* allows to run automatic regression tests, necessitating of course a set of ground truth documents. It is used to check and quantify the effect of system-changes (implementation or configuration). When the system configuration is changed one uses the *StatisticTool* to assure that the effect is positive and no degradations are unconsciously entered.

The complicated information for spotting and extracting tables is handled by a *TableTool,* which follows the approach we proposed in [10]. Amongst other things, the freely configurable workflow logic allows intermediate processing results to be externalized with a *user exit* process. The format is configurable, mostly XML. The intermediate results can thus be checked and changed with external software and then be reintegrated into regular processing. The workflow also allows for tricky DAU processes, where, for example, documents are analyzed to a first level and in a second run from the first level to a second level.

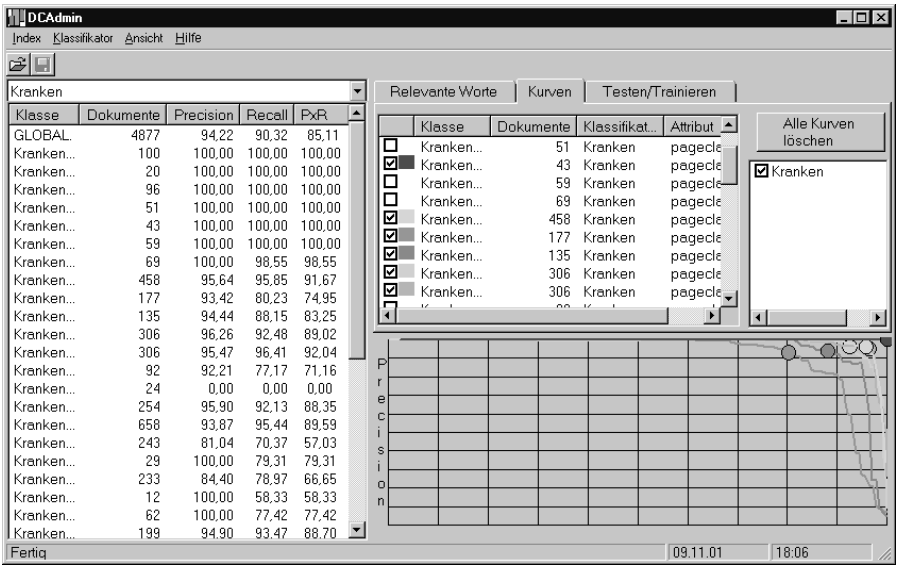


Fig. 5. The *DCAdmin* helps to supervise the learning of semantics of document content in order to classify future documents with significantly similar content.

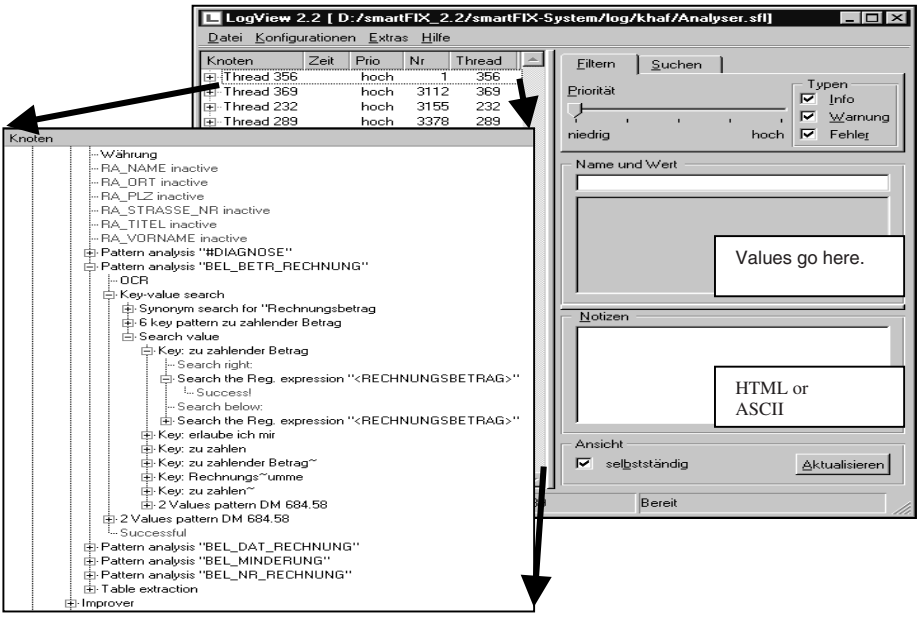


Fig. 6. *LogView* visualizes the messages of *smartFIX healthcare* modules. With a priority-slider messages can be suppressed depending on their priority. Messages can comprise values, e.g. an SQL statement, which caused an error, and can be augmented with arbitrary HTML or ASCII text, which is occasionally useful e.g. for messages from table extraction.

4 DocumentManager

Prior to going operative, *smartFIX* has to be programmed. The *DocumentManager* is a graphical editor with which the user defines document classes to distinguish, qualifies all necessary reference patterns, the analysis "recipe" and lets them be stored in a knowledge base of analysis configurations, simply called "document definitions". The *DocumentManager* is composed of five windows shown in Figure 3. On the right is a user-chosen sample document image, on the left:

- The directory of the different document classes and their aggregations
- The classification features
- The labels and the types of logical objects capturing the information to be searched
- So-called *Label Object Groups* (*LOG*), information on relations between objects

In order to configure *smartFIX*, example documents are loaded. In a first step, the user defines and links the corresponding document classes. He or she selects a region in the image where to search for a logical object of interest. This region of interest (ROI) is drawn with the mouse or selected to be the entire page wherever it is not sensible to constrain the region of search. To name the logical object, the user may either select the label from the existing list in the window or can add a new label. In addition, the user has to define the ROI type, which determines the type of analysis script which will be applied for the extraction of the contained information. *SmartFIX* offers various scripts. The most significant are:

- ADDRESS bases on a grammar of typical German addresses.
- CHECK BOX
- BAR CODE
- SEARCH PATTERN addresses a *key string* (including synonyms and acronyms from a configurable thesaurus) to which a corresponding *value string* has to be found. For example (see Fig. 4), the value of the logical object `<bill no.>` (in German `<rechnungs-nr.>`) is a numerical expression located within a certain relative distance in the image.
- ASSOCIATION skims with a whole database of known facts to find occurrences of them in the document.
- TABLE allows the extraction of tables and their logical structure, including many tricky features, e.g. interleaved columns.
- TEXT addresses objects which can be described with regular expressions and occur within free text, like "my insurance number is BG/1495-H".

All logical objects, which are defined, are related to ranges of values, e.g. lexical knowledge, regular expressions, or numerical constraints, depending on their type. For example, columns of a table are linked to sets of values, i.e. the logical object `<product>`, which represents a table column name, can be linked with all valid product names in a data base.

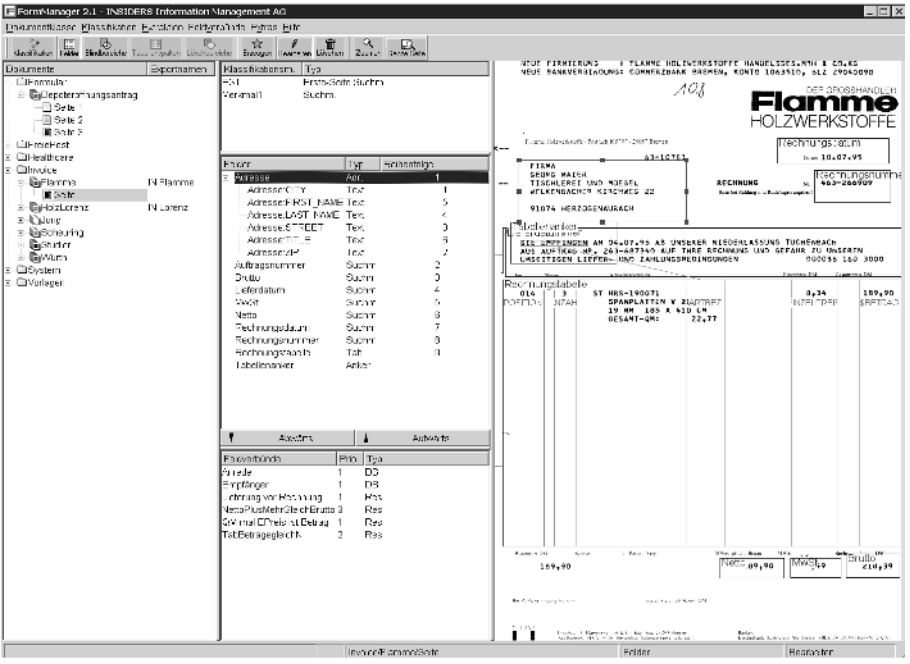


Fig. 7. The DocumentManager

After the definition of logical objects and corresponding ROIs, it is possible to support the analysis further with *LOG*s. A *LOG* relates a named set of scripts which depend on each other, e.g. a LOG `<SERVICE SUBJECT>` could be composed of `<insurance number>`, `<person>`, and `<address>`, while the latter two again consist of `<first name>`, `<last name>` as well as `<zip code>`, `<city>`, and `<street name>`.

Finally, all information is stored in a knowledge base of document definitions to be used later for the analysis of documents.

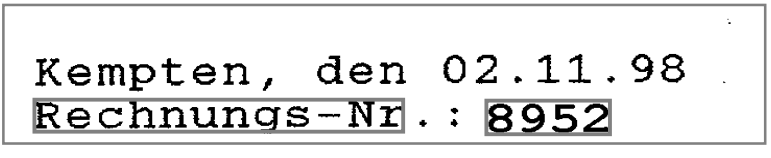


Fig. 8. Typical constellation of numerical data close to a meaningful keyword.

5 Analyzer

A number of *Analyzer*s actually accomplish the real DAU processing. Every running *Analyzer* polls the brain of the system, the *control data base,* for documents to process. After finishing with a document it is labeled and put back into the *control data*

base. The label is like the address of the next necessary processing stage, e.g. *Verifier*, and the labels flexibly implement the internal workflow.

For one page the analysis progresses in roughly four stages:

1. Image pre-processing (Section 5.1),
2. Classification (Section 5.2),
3. Information extraction (Section 5.3),
4. Improvement (Section 6).

OCR is called lazily, i.e. it is called for segments only when their character content is needed. As a matter of fact, the four phases are not always clearly distinguished. E.g. after the classification it is possible to have found that the image must be scaled. Newly, for classification, all methods out of the information extraction toolbox are usable. The first three steps are explained in the remainder of this section. Improvement, because of its probable novelty to the reader, is the topic of the next section.

5.1 Image Pre-processing

Image pre-processing consists of five optional steps. Each step can be individually switched on and off, tuned with parameters, and be required for the whole page or arbitrary regions.

1. Removal of graphical lines (vertical and horizontal)
2. Optimization of matrix printed characters
3. Despeckle
4. Deskew (±10° landscape and portrait)
5. Correction of upside-down

The pre-processed images are stored for the later visualization of the extraction results.

5.2 Classification

The classification runs in predefined phases and searches the following features until a class is determined:

1. Layout similarity [7, 8]
2. Recognized tables [6]
3. User-defined or machine-learned patterns (including Not-patterns)
4. Machine-learned semantic similarity [11]
5. Document size
6. Barcodes, signatures, vertically oriented patterns

The known classes and their features are configured using the *DocumentManager*. The classes can be structured decision-tree-alike, so that the classification can at least narrow down the alternatives whenever no single class can be determined. The classes can be set up so that non-classified documents are sent to a *Verifier* for manual classification, or run through a default information extraction first.

The classification also has to deal with the collation of pages to documents. The most reliable method, interspersing different documents with easy-to-recognize dividing pages at scanning, is sometimes not possible. Therefore *smartFIX healthcare* provides a collation-machinery, which is not really sophisticated but confusing to explain without using many examples.

The classification can utilize if document pages are entered into the system accompanied with pre-information, in the form of attributes, which perhaps codes the source of documents like scanner type, scanning location, personnel or the like. One important consequence of this is that it allows for one aspect of SAP-compatibility: documents from different sections of a company can be attributed as such, thus distinguished in the system and thus be processed on one and the same hardware. So two or more systems can share one hardware.

The class of a document fixes its further path through the system. Different classes are often handled totally different, including their export out of the system to different target systems.

5.3 Information Extraction

Information extraction is performed according to the "scripts" which –with the *DocumentManager*– were individually configured for the respective document class. Scripts have a type and a name. According to their type, scripts perform a specific action referring to the page at hand, after which they provide a value under their name. Where this is important, scripts can be assigned priorities to constrain the sequence in which they are run. Simple extraction scripts search the image of the page. The complex extraction scripts use complicated user-provided and built-in knowledge. The data scripts are simply providing access to logical page data. These three categories of script types are explained in the following.

Simple Extraction Scripts Types

1. "Text" and "Combination": extract characters in a specified area. The combination script sends the extracted characters later to have it split into components with the support of database lookups. It is possible to have the area initially cleared from form preprints.
2. "Checkbox": two different algorithms can be chosen, to determine whether a checkbox area is ticked.
3. "Pattern search": either on the whole page or in an area, a regular expression is either searched directly, or searched in the configurable surroundings (direction and distance) of a prior regular expression match or thesaurus synonyms match.
4. "Address": a specific script for the extraction of German addresses, split into standard pieces. A regular expression may be used to exclude strings from the search space.

5. "Anchor": does a pattern search and after matching it triggers other linked scripts, which then run in an area relative to the position of the match.
6. "Format": currently simply measures and evaluates to A4, A5, or A6.

Value Script Types. With the choice of one of the following scripts it is possible to access page information: "Document class": Evaluates to the result of the classification. "Document ID": evaluates to the unique identifier of the document (which often allows access to further related external archive data). "Task ID": evaluates to the task which the document belongs to.

A "free value script" does nothing, but can be set later with a value from a database lookup. This might be helpful as a variable for further database lookups, for automatic checks, or as an additional information for human Verifiers.

Complex Extraction Script Types. "Association": skims for occurrences of patterns from a possibly huge database. Several fields of the database can be mixed. Prior successful scripts can be used to restrict the search space of database field combinations. This is a powerful tool e.g. to find exceptionally formatted senders' addresses, if they are only known in a customer database.

"Fixed table": extracts a table according to user-defined layout rules.

"Free table": is a comprehensive Analyzer in its own right. According to a complex knowledge configuration with the *TableTool*, this script extracts tables of different layouts, even spanning more than one page, rather intelligently. Extracts also table-related data like sum total. [6]

6 Improver

The improving module, based on the Symbolic AI technique of constraint solving, checks consistency, searches optima, and fills in deducible information. The main idea is that the results of all the single scripts, which are fuzzy and unsafe, can be constrained step by step to more specific and reliable results, with local auxiliary conditions and interdependency conditions. In detail the improving module is described in [9].

The improving module employs a unique formalism to cope with the manifold alternatives generated by almost any component used in DAU. As far as we know, there are no comparable systematic DAU optimizations yet, and the *smartFIX healthcare* system is the first to integrate database and other constraints, and to profit from the theory of constraint solving. Optical Character Recognition (OCR) is still the slowest and weakest processing step in our application domain. As a rule, 1-10% of machine font characters are not read correctly; even more in cases of bad scan quality. Most OCR implementations return lists of alternatives to express ambiguity, but for the assessment of the quality of the alternatives, it can be hardly relied on the quality values returned by the OCR. A much safer approach is to exploit some context knowledge to exclude inadequate interpretations. E.g. the results of two scripts like for a name and a contract number are rather unsafe after simple extraction. But their

combination most often allows the determination of the one right name and contract number.

The execution time for the constraint solving process e.g. of one medical invoice including a table with a couple of cells rarely exceeds one second. The error rate for fields that are marked *safe* is not allowed to exceed 1/1000. On average, roughly 70-90% fields can be presented *safe* to the user.

Context knowledge adds a flexible power of modeling to *smartFIX healthcare*, because its specification is substantially faster than a re-implemention of the document analysis for a certain domain. Moreover, the same knowledge serves as a correction mechanism later in the *Verifier*, so that after the typing of missing information, deducible information is provided by the system.

6.1 Constraint Solving for Exploiting Context

Context knowledge in document analysis can be denoted as constraints, i.e. as relations among the text fields. The variables in DA are the regions of interest (ROI). The fuzzy set of OCR interpretations can be seen as the natural domain for fields. In this way, document understanding becomes a matter of constraint solving and optimization.

A search pattern that identifies the invoice date, for instance, will generate several potential locations on the page. Constraints can eliminate inconsistent date alternatives like those before the date of the last medical treatment. Another practice example is the *hospital invoice* where *smartFIX healthcare* is supposed to find the hospital address. Different hospitals have different typical rates for certain services. Ambiguous address matches can exploit the special rates help identify the correct address and vice versa.

The constraints that the improving module exploits are the user defined *Labeled Object Groups* (LOG), user-provided auxiliary conditions and it also finishes the "Association" scripts, heavily using the *matching server* process. LOGs relate different scripts (i.e. their values) together, by a reference to databases. Also the "Combination" scripts, introduced before, are processed in this module. In their case, the difference is only, that name and number and perhaps more data are concatenated in one string, possibly permutated. The "Association" scripts skim a whole region or entire page whether any of the facts from a whole database trigger an association and where, e.g. whether and where on a page one or more out of perhaps 10000 known company names can be found. With the local auxiliary conditions the system estimates levels of reliability for every single piece of extracted information, which is used in the *Verifier* as colored backgrounds green, light blue, blue, to direct the attention of human operators.

Auxiliary conditions need not only be monadic operators, but can also relate two or more script values analytically, then resembling LOGs. E.g. all the fields in a summation column sum up to the sum total below the column. Available predicates and functions address arithmetics, dates, checkboxes, strings. Some of the auxiliary conditions can generate correction hypotheses or fill empty fields (e.g. a wrong or empty OCR result in a summation column) others only work as a filter. Via CORBA or COM users can also call user-implemented condition functions (returning a string).

There are a number of important differences to classical constraint solving.

- There is no discrete truth model for the constraints. We have to deal with probabilities. Even a perfect match of OCR and database does not give 100% accuracy as we never know about the correctness of the OCR.
- In DA, we are not focused on the question of consistency. Classical logic will return *false* or *no* in case of inconsistency. But the typical application in DA does not know a thing like an inconsistent document. Rather, inconsistent constraints are omitted such that the remaining values can be proposed to the user.
- The result of the DA is supposed to be corrected by verifying personnel. In order to accelerate the correction, the system returns a *rating* for any field which is one of (a) safe, (b) proposal, or (c) OCR result. The user is not supposed to check the safe values (although he can).
- There are at least two kinds of *truthes*: (a) the truth from the viewpoint of the author of the document, and (b) the so-called ground truth, i.e. the mathematical arithmetics, respectively the database.

6.2 Relating OCR Values with Constraints

The domain of the variables is the *fuzzy* set of OCR interpretations. Matching it to some database values or confirmation by another constraint, changes it from fuzzy to *exact*, which means its reduction to a finite table of possible values (exact values are more tractable). This implies that results may vary depending on the order of evaluation. However, the performance gain is too big to be wasted and *smartFIX healthcare* allows for a priority setting to give control of order to the user if necessary.

Any constraint has access to the document fields, which are considered the variables here. Alternative values are assigned a quality. This quality is computed as the average Levenshtein similarity of the new alternative values and their associated OCR graphs. Lines are only kept if they are better than configurable thresholds.

A database constraint takes the fuzzy or exact alternatives from its variables and builds a database query. As this query may be fuzzy, this cannot be achieved with SQL. The wanted result consists of those DB lines that are the most similar to the OCR values. The *smartFIX healthcare* system thus comprises a complex tool called *Matching-Server* which exploits Levenshtein editing distance, Trigram search, and other methods to efficiently scan large databases for fuzzy matches. For *Association* fields, the locations are initially unknown, so that the task of the *Matching-Server* there is to find location as well as contents. A realistic size for an address database is roughly 100 million entries on a standard PC with 1 GB RAM attached.

A successful database constraint chooses one or more table lines and writes the results back into the table of alternatives. Exact variables provide a *hard* selection like the WHERE clause in a SQL SELECT statement. Consequently, all result lines satisfy this selection. In contrast, fuzzy variables provide a *soft* selection. Thus they may be changed to a considerable degree if the whole line matches well. Empty variables do not give any selection. Rather, all possible results are returned. A typical query would be the following:

Field	Value	State
Last name	`M6yor'	fuzzy
First name	`J0n'	fuzzy
Account no.	`'	fuzzy
City	`London'	exact

Note that the values are OCR results in the case of fuzzy fields while exact values may have another origin. After the query, the table of alternatives could look as follows:

Quality	First name	Last name	Account no.	City
0.8	John	Mayor	12345678	London
0.7	Joe	Major	87654321	London
0.5	John	Maiden	18181818	London

Arithmetic constraints are potentially infinite relations, requiring a different approach. First, the exact variables are regarded, by running the algorithm for any possible combination of exact input values. (This calculation possibly explodes and thus the system has configurable limit, above which it gives up.) Second, if the constraint is satisfied an alternative line is generated. Otherwise, it is tried to correct one of the fuzzy variables. Often one can create sensible proposals for the variables, written into the table of alternatives.

A special case is an *assignment*. This is an equation with one empty fuzzy variable. Assignments are a perfect fit, but will never be rated *safe*.

As an example, suppose that the OCR has read 4.8 as a value for x and 2.3 for y and there is a constraint `[x] = [y] * 2`. The solver would then create two alternative solutions for this constraint:

Quality	X	y
0.833	4.6	2.3
0.833	4.8	2.4

Recall the example alternative table given in the last section. A constraint like `[Account no.] ≥ 20000000` would eliminate the records `John Mayor' and `John Maiden'. So would `SUBSTR([Account no.], 0, 1) = '8'`. So would `[First name] = 'Joe'`. Suppose that there is no OCR input for field X. Then the constraint `[X] = [First name]` would generate the following alternative table:

Quality	X	First name
1	John	John
1	Joe	Joe

It should be mentioned that *smartFIX healthcare* offers a full range of constraints like =, <, >, ≥, ≤, plus linear arithmetics, and a number of constraints for checkboxes, for strings, and for dates. A user-defined function is available which can be implemented in a CORBA object. Also, some extra-logical predicates like IF-THEN-ELSE are available.

6.3 Propagation and Rating

Once the constraints have generated their alternative tables, they start a classical propagation. This ensures eventually that every constraint on a variable *x* will contain exactly the same set of values for this variable. The only exception is an empty alternative table. Empty tables do not propagate to prevent the whole system from being emptied and possible proposals from being deleted.

After the propagation the labeling stage begins. Obviously, this is a critical step w.r.t. optimization. We use heuristics to find an order to process the constraints. As

opposed to regular labeling, we select one alternative line per constraint rather than one value per variable, since we administer the domains in the constraints rather than in the variables. For any constraint we select the regionally best alternative, assign the results to the variables, and then propagate.

The final step is the rating of the fields. Recall that alternative lines contain a quality rating. One of the lines has been selected after labeling. If its rating exceeds a `success' threshold, the constraint is considered *successful*. If there is no other alternative line within a `distance' threshold, it is considered *unique*. If at least one constraint over a field is successful and unique, the field is rated *safe*.

In classical CSP, constraints are usually seen conjunctively i.e. connected by *and*. This is the expected behavior w.r.t. the ground truth, too. *And* logic implies that the basic assumption is true (i.e. *safe*) and constraints can contradict (respectively weaken) this rating.

This is anticipated by our *confidence* that what we read is really what the author wrote (or intended to write). Pure OCR gives a very low confidence state, in fact the least in our three-step rating scheme. Only when a constraint has confirmed the OCR result, the rating becomes stronger. This corresponds to a classical *or* logic.

Both logics *and* and *or* are justified and thus we decided to combine them. In classical logics, there is no `combination' of *and* and *or*, but it has proven a very useful heuristics. We use the *or* scheme during the evaluation and only as a final step, we introduce an aspect of *and*. This is achieved by making sure that no unsuccessful constraint is rated completely *safe*, i.e. if all fields of an unsuccessful constraint are rated safe, we reset the ratings.

7 Verifier

Verifiers automatically poll the result database for documents after analysis, in the status to be verified. The original page is displayed, also, zoomed**,** the source area of an extracted data item, and the extraction result. Focus jumps to the next data item, as soon as the first was acknowledged or corrected. Colored areas can be configured for document classes to support the eyes finding zones on pages (like the field in the background in Figure 9, right below the address, the area to typically find a textual diagnosis), “safe” estimated result data are automatically coded green, the results less safe are coded light blue and blue. Manually typed data is immediately used to fill up and cross check deducible data, perhaps insurance number and first name, after the correction of a date of birth. Thus, the *Verifier* allows to check and correct the extraction results one after the other rather efficiently.

Verifiers are a crucial part of *smartFIX healthcare*, as they serve an intrinsically critical task of DAU: the re-integration of computed DAU results into the world of business processes and humans, where these results represent monetary values. Tasks that a) affect values and b) are delegated to somebody else, in this case even a mainly dumb system, rely very much on a significantly high chance to find errors wherever they might origin from.

Verifiers can have different priorities and rights assigned and can log the user-actions.

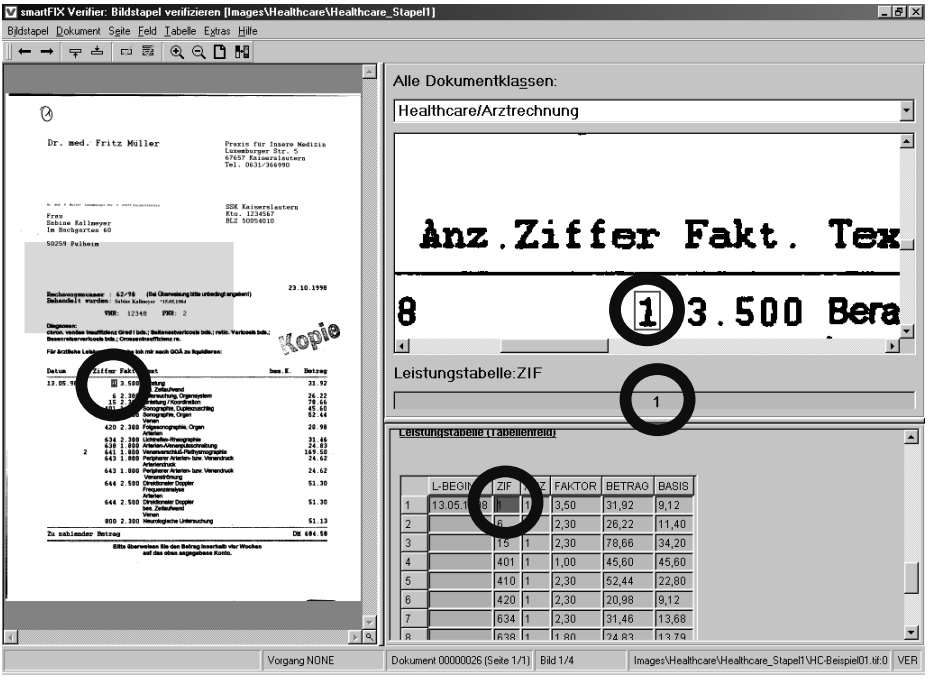


Fig. 9. The *Verifier*: four synchronized views on the same data for efficient checking

8 Results

smartFIX healthcare was installed at the insurances' sites and connected to their workflow successfully, i.e. adapted to the specific requirements at hand. It was configured to distinguish about 50 different types of mostly unstructured documents, like hospital bills, dentist bills, medicament bills etc., all coming with data which trigger and control processes. Most of the processes are directly involving money transfers.

Information items for which *smartFIX healthcare* was configured comprise: (a) insurance number, (b) name of patient, (c) date, (d) service period, (e) diagnosis, (f) medical services table, (g) duration of treatment, (h) name and address of doctor, altogether a number of 107. Out of this base set, about 20 are to be found on one type of document on average.

For an evaluation, we considered a representative stack of 525 documents as they arrived at one of the largest health insurance companies in Germany. The test comprised of classification into classes reimbursement form (RF), prescription (P), dentists bills (D), hospital bills (H), bills from medical doctors (M), and others (O) while the latter ones had to be sub-classified into another 20 classes. The following Table 1 shows the classification results:

Table 1. Classification results of *smartFIX healthcare.*

RF	95%	D	100%	M	92%
P	96%	H	94%	O	73%

The classification rate over all classes was 92%; the human workload is thus reduced by 92%. The remaining rejects are sent to personnel at a Verifier. Another key measure, the systems error rate was less than 0.1%.

After classification, the instructed ROIs were searched and their information content extracted. For the evaluation, two aspects were important: The extraction rates as well as savings in time. The extraction rate for all fields (a) to (h) was 81 % and saved an average of 64% of time compared to manual input. (Other numbers from a couple of other customers and scenarios mostly even tend towards roughly ¾ of savings of time.)

Special add-on modules (included in *smartFIX healthcare*) can recalculate the figures on the invoice, to check its correctness. This implies a check of the single service fees, allowed multiplications with so-called factors, and the sums and their final addition on the invoice. Finally, *smartFIX healthcare* can check whether the service positions on each invoice adhere to the German scale of charges and fees, the so-called GOÄ.

This result is just one example presented to the reader. *smartFIX* runs at more than a dozen of customers sites, with very different configurations. A huge number of very different documents were classified and their information extracted to date. Note for example, that at one company the classification rate went down to 30 % last year. They were happy. Without changing the system configuration, they strained the system with a variety of new and complicated input documents, which could of course not be classified; and which should not be classified. And this is what *smartFIX* did not do, correctly. Our statistics are not important to us, the customers satisfaction is.

9 Summary

We have given an overview of a commercial document analysis system, *smartFIX,* which is the result of the exploitation of research results from the DFKI, their non-trivial combination and additional research by the spin-off company INSIDERS. The system is able to process stacks of mixed-format documents including forms, invoices, letters, all of which may contain machine-written or hand-printed information. The system is installed at about two dozens of customers analyzing several hundred thousands of document pages every day. The purpose of *smartFIX* is in the distribution (classification) of scanned incoming mail and the extraction of information relevant for the user or customer. *smartFIX* has some special features. It processes unsorted incoming mail of any format and independent of the degree of structure or preprints. Furthermore it is not limited to a small number of sorted and separated document types. *smartFIX* classifies all documents and therefore is independent of pre-sorting. Images are combined automatically into single- or multi-page documents. Documents from different sections of a company can be attributed as such, distinguished in the system and thus be processed on one and the same hardware. So two or more systems can be run on one hardware.

A special feature of *smartFIX* is its dedication to what we call *adaptivity*. *Adaptivity* aims to reduce the (human) effort to maintain applications and to configure the system for new applications. Thus the requirements of adaptivity are a dedicated interfacing machinery to both computer/hardware environments and human/business environments. Not only are modules required to technically connect the system to fixed computer infrastructures. The system was also re-constructed more modular, so as to allow for a greater variety of possible configurations. And last, adaptation requires to help the human instructor in his task to assess the scenario and find the best system configuration to adhere to it. Thus, we have developed learning tools and domain assessment tools. Learning tools aid, because it is simpler to just provide examples, than determine the right features and explicate them to the system. Domain assessment tools aid determination of relevant features, by allowing the evaluation of e.g. the average distance between two consecutive words on 1000 sample documents. During the research project "adaptive Read", funded by the German ministry for research, bmbf, *smartFIX* was fundamentally furnished with the respective adaptivity-features.

Recently, we have been working on some improvements and additional features of *smartFIX*. The first addresses the restructuring of the system into even smaller modules which then allow for more flexible mixing, which in turn allows to serve new and more complex application scenarios.

The second addresses a new paradigm for the *DocumentManager* with full access to all *Analyzer* features, so that the effect of every single configuration-information can be immediately tried out interactively. This will look and feel like a software engineering environment with full debugging support.

References

1. A. Dengel, R. Bleisinger, R. Hoch, F. Hönes, M. Malburg and F. Fein, OfficeMAID — A System for Automatic Mail Analysis, Interpretation and Delivery, Proceedings DAS94, Int'l Association for Pattern Recognition Workshop on Document Analysis Systems, Kaiserslautern (Oct. 1994), pp. 253-276.
2. S. Baumann, M. Ben Hadj Ali, A. Dengel, T. Jäger, M. Malburg, A. Weigel, C. Wenzel, Message Extraction from Printed Documents A Complete Solution. In: Proc. of the 4.th International Conference on Document Analysis and Recognition (ICDAR), Ulm, Germany, 1997.
3. A. Dengel and K. Hinkelmann, The Specialist Board – a technology workbench for document analysis and understanding. In M. M. Tanik, F.B. Bastani, D. Gibson, and P.J. Fielding, editors, Integrated Design and Process Technology – IDPT96, Proc. of the 2nd World Conference, Austin, TX, USA, 1996.
4. G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde, and B. Wielinga. Knowledge Engineering and Management – The CommonKADS Methodology. The MIT Press, Cambridge, Massachusetts, London, England, 1999.
5. http://trec.nist.gov/
6. B. Klein, S. Gökkus, T. Kieninger, A. Dengel, Three Approaches to "Industrial" Table Spotting. In: Proc. of the 6.th International Conference on Document Analysis and Recognition (ICDAR), Seattle, USA, 2001.

7. A. Dengel and F. Dubiel, Computer Understanding of Document Structure, International Journal of Imaging Systems & Technology (IJIST), Special Issue on Document Analysis & Recognition, Vol. 7, No. 4, 1996, pp. 271-278.
8. F. Dubiel and A. Dengel, FormClas — OCR-Free Classification of Forms, in: J.J. Hull, S. Liebowitz (eds.) Document Analysis Systems II, World Scientific Publishing Co. Inc., Singapore, 1998, pp. 189-208.
9. A. Fordan, Constraint Solving over OCR Graphs. In: Web-knowledge management and decision support. 14th International Conference on Applications of Prolog (INAP), Tokyo, Japan, 2001, Revised papers. LNAI series, Springer 2003.
10. T. Kieninger and A. Dengel, A Paper-to-HTML Table Converting System, Proceedings DAS98, Int'l Association for Pattern Recognition Workshop on Document Analysis Systems, Nagano, Japan, Nov. 1998, pp. 356-365.
11. M. Junker and A. Dengel, Preventing overfitting in learning text patterns for document categorization, ICAPR2001, 2nd Intern´l Conference on Advances in Pattern Recognition, Rio de Janeiro, Brazil, March 2001.
12. C. Altenhofen, M. Stanišic-Petrovic, M. Junker, T. Kieninger, H. Hofmann, Werkzeugeinsatz in der Dokumentenverwaltung (German), in: Computerworld Schweiz, Nr. 15/2002, April 2002, S. 6-11,
http://www.kodok.de/german/literat/artikel/index_artikel.html