Ontology-Based Assessment of Functional Redundancy in Health Information Systems

Alfred Winter[1], Alexander Strübing[1], Lutz Ißler[4], Birgit Brigl[2], and Reinhold Haux[3]

[1] University of Leipzig, Institute of Medical Informatics, Statistics and Epidem., Germany
[2] Dr. Birgit Brigl Krankenhaus-IT Management Beratung, Friedrichsdorf, Germany
[3] Peter L. Reichertz Institute for Medical Informatics of the University of Braunschweig - Institute for Technology and of Hannover Medical School, Germany
[4] Systemantics, Aachen, Germany

Abstract. The paper introduces a formal definition of functional redundancy to determine non-redundant health information system architectures, in order to support information management of, in particular, hospital information systems. We specify an ontology, which is linked to the Three-Layer Graph-Based Meta Model (3LGM^2) and based on enterprise functions and application systems of (health) information systems. A so called functional redundancy rate (FRR) is introduced and elucidated by an example. An algorithm for calculating non redundant health information system architectures is presented. Functional redundancy is a key performance indicator for the quality and efficiency of (health) information systems. With FRR it can now be formally described and quantitatively analyzed. Using 3LGM^2 based models of information systems, the calculation of FRR does not need further efforts.

Keywords: Health information systems, hospital information systems, information management architectural models, functional redundancy.

1 Introduction

Information management for health information systems has become a crucial and significant task, in particular for hospitals but also for 'trans-institutional' regional and national health care settings [1-3]. Assessing the quality and efficiency of health care institutions' information systems is an important field in research and practice of medical informatics [4, 5]. However, there is still a lack of easy to understand and likewise relevant evaluation criteria, which can be accurately defined and thus formally described. Such formal descriptions provide the option to immediately derive these criteria from architectural specifications of health information systems, and so to make them well suitable for the practice of information management. One of those criteria is functional redundancy ([6], pp. 170 and 233). Most information managers may have a certain feeling for redundancy of functional support in the information system they manage. But providing precise information regarding redundancy for decisions concerning the information system's architecture and investments still needs to be solved.

M.R.V. Chaudron (Ed.): MODELS 2008 Workshops, LNCS 5421, pp. 213–226, 2009.

The aim of our research is to introduce a formal definition of functional redundancy and, to calculate a health information system's functional redundancy rate (FRR) (section 3) as well as to outline an algorithm for calculating non redundant health information system architectures (section 5). Before, we need to define an ontology [7] for describing functional redundancy in (health) information systems (section 2). On that basis we want to support information managers to find answers to the following questions:

- What application systems in my information systems can be shut down without loss of functionality?
- Do I have unnecessary costs because different users in my institution use different application systems in order to support the same enterprise function? What are the critical enterprise functions and what application system's usage should be prevented?

Please note that we are using the term information system in a rather comprehensive manner. An institution's (e.g. a hospital's) information system, is that socio-technical subsystem of the institution, which comprises all information processing actions as well as the associated human or technical actors in their respective information processing role [8]. The basic model, introduced in section 2, is closely linked to the Three-Layer Graph-Based Meta Model ($3LGM^2$) [9] serving as a domain ontology for the field of information systems [7].

2 An Ontological Foundation for Assessing Functional Redundancy in Information Systems

Describing and calculating functional redundancy in an information system first of all requires a model of the information system. To guarantee that the assessment of functional redundancy can be applied to the variety of existing modelling techniques, such a model should be based on an ontology for the description of information systems. To our knowledge, such ontology does not exist yet. But we identified two terms that are used in the most common modelling approaches, namely enterprise functions and application systems. Enterprise functions can be considered a directive for human or machine action and a duty arising from an enterprise's mission and goals. For example, "clinical admission", "radiotherapy", or "care planning" may be enterprise functions. Within the computer-supported part of an information system, the tools used to support the execution of enterprise functions can be described as application systems being installations of application software products on computers. Application systems may have a local database to store data and interfaces for communication.

Functional redundancy deals with the adequate relationship between tasks to be done, i.e. enterprise functions, and tools to support these tasks. Using these terms we can model this support relationship by a matrix *SUP*.

Let $\underline{EF}$ be a set of enterprise functions and $\underline{AS}$ a set of application systems.

$$\underline{EF} := \{EF_1, ..., EF_P\}, \quad P > 0 \tag{1}$$

$$\underline{AS} := \{AS_1, ..., AS_N\}, \quad N > 0 \tag{2}$$

The two-dimensional matrix *SUP* describing the relationship between tasks and tools mentioned above is defined as

$$SUP := \left(sup_{p,n}\right)_{p=1\ldots P, n=1\ldots N} \text{ with}$$

$$sup_{p,n} = \begin{cases} 1 \; if \;\; function\; ef_p \; is\; suported\; by\; application\; system\; as_n \\ 0 \; else \end{cases} \tag{3}$$

Example (part 1)
Suppose a set of enterprise functions *EF*:={*A*,*B*,*C*,*D*,*E*,*F*,*G*} where the letters represent enterprise functions as follows: *A* for "clinical admission", *B* for "administrative admission (inpatients)", *C* for "administrative admission (outpatients)", *D* for "radiotherapy", *E* for "decision making", *F* for "patient information", and *G* for "care planning". Additionally, suppose a set of application systems *AS*:={*1*,*2*,*3*,*4*,*5*,*6*,*7*,*8*,*9*} where the numbers represent application systems as follows: *1* for "CareMgmtSys", *2* for "PatientAdministrationSystem(ADT)", *3* for "DepartmentalSystemPsychology", *4* for "DepartmentalSystemRadiotherapy", *5* for "KnowledgeService", *6* for "DiabetesTrainer", *7* for "ClinicalPathwaySys", *8* for "TherapyPlaner", and *9* for "TherapyAdvisor".

Table 1. The matrix *SUP* for *EF* and *AS* is illustrated in figure 1

		application systems n=1,…,9								
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
enterprise functions p=1,…,7	**A**	**1**	0	0	0	0	0	0	0	0
	B	0	**1**	0	0	0	0	0	0	0
	C	**1**	**1**	**1**	**1**	0	0	0	0	0
	D	0	0	0	**1**	0	0	0	0	0
	E	0	0	0	0	**1**	0	**1**	0	0
	F	0	0	0	0	0	**1**	0	0	0
	G	0	0	0	0	**1**	0	**1**	**1**	**1**

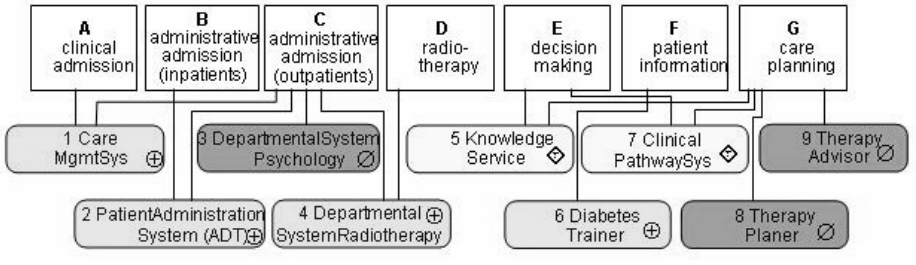

Fig. 1. Matrix *SUP*: rectangles denote enterprise functions, rounded rectangles denote application systems, and connecting lines illustrate a "1" in the respective position of the matrix, i.e. that a certain enterprise function is supported by a certain application system. E.g. enterprise function E "decision making" can be supported by application system *5* "KnowledgeService" or *7* "ClinicalPathwaySys" alternatively. The meaning of the different hatchings and the ⊕, ∅ and ⑨ -signs will be explained in section 4.3.

3 A Measure for Functional Redundancy

Functional redundancy is a characteristic of information systems which should be addressed by information management. In order to reduce complexity of the information system it is interesting to know which application systems could be omitted without loss of functionality, i.e. without hindering the execution of any enterprise function. Before we define a measure for functional redundancy we want to explain, how we can detect redundant support of enterprise functions by application systems.

3.1 Redundant Support of Enterprise Functions

For every enterprise function $ef \in \underline{EF}$ we can easily calculate

$$isup_p := \sum_{n=1}^{N} sup_{p,n} \tag{4}$$

For every p, $isup_p$ denotes the number of application systems actually supporting the individual enterprise function ef_p; we call it its individual degree of support by application systems. Every $isup_p > 1$ may be an indicator that some application systems are dispensable, with $isup_p - 1$ indicating the number of possibly superfluous systems. However, this number needs a careful investigation because some of the apparently superfluous application systems may be necessary for other enterprise functions. Obviously, measuring functional redundancy in a way, which is supportive for information management, needs a measure which takes these interrelationships into account.

Example (part 2)
Continuing part 1 of our example we can easily calculate the $isup_p$ as shown in table 2.

Table 2. $isup_p$

p	ef_p	$isup_p$
1	A	1
2	B	1
3	C	4
4	D	1
5	E	2
6	F	1
7	G	4

The value of $isup_3 = 4$ indicates that perhaps there are three superfluous application systems supporting *C* (“administrative admission (outpatients)”). But detailed analysis shows that the application systems *1* (“CareMgmtSys”), *2* (“PatientAdministrationSystem(ADT)”) and *4* (“DepartmentalSystemRadiotherapy”) cannot be omitted, because they are needed for the functions *A* (“clinical admission”), *B* (“administrative admission (inpatients)”) and *D* (“radiotherapy”). However, application system *3*

("DepartmentalSystemPsychology") is a good candidate for being removed from the information system because function *C* as the only function it supports is also supported by application systems *1*, *2*, and *4*. A measure of redundancy should therefore correctly indicate that considering the enterprise function *C* ("administrative admission (outpatients)") only one application system could be omitted (namely DepartmentalSystemPsychology).

3.2 A Measure for Functional Redundancy for Information Management

We now want to introduce a measure for functional redundancy for information management as a key performance indicator, denoting the percentage of application systems in a given information system, which could actually be shut down and omitted without loss of support of any enterprise function. First we have to check, whether particular application systems can be omitted or not, given $\underline{EF}$, $\underline{AS}$ and *SUP*. With the notions introduced earlier, the challenge is to calculate a minimal subset $\underline{AS}^{\min} \subseteq \underline{AS}$ of application systems which guarantees that all functions are supported and that there are no superfluous application systems in use. Each set $\underline{AS}^{\min}$ we call a "minimal functionally non-redundant set of application systems". In general, there is more than one such set $\underline{AS}^{\min}$ for a given information system, i. e. there is more than one way to cut down the functional redundancy in an information system. In the real setting of the information system of the Leipzig University Medical Center we actually found several hundreds of minimal functionally non-redundant sets of application systems.

Let us describe any subset $\underline{AS'} \subseteq \underline{AS}$ of application systems being actually in use by a vector $\overrightarrow{USE}$, indicating whether application systems are member of the subset $\underline{AS'}$ or not.

$$\overrightarrow{USE} := (use_n)_{n=1...N} \qquad \text{with} \qquad use_n = \begin{cases} 1 \; if \; as_n \in \underline{AS'} \\ 0 \; else \end{cases} \tag{5}$$

Hence $\underline{AS}^{\min}$ can be described by $\overrightarrow{USE}^{\min} := \left(use_n^{\min}\right)_{n=1...N}$ and $use_n^{\min} \in \{0,1\}$. Given what application systems are in use, i.e. given the respective vector *USE*, we can calculate the individual degree of support for all enterprise functions as well as:

$$\overrightarrow{ISUP}^{T} = SUP * \overrightarrow{USE} \qquad \text{with} \qquad \overrightarrow{ISUP} = \left(isup_p\right)_{p=1...P} \tag{6}$$

As stated above, we want that despite of some application systems being not in use, every function is supported by at least one application system. We introduce a vector $\vec{e}$ of length *P* containing only "1":

$$\vec{e} = \left(e_p\right)_{p=1...P} \qquad \text{with} \qquad e_p := 1, \; p = 1...P \tag{7}$$

Now we can state the first postulation:

(P1) For every vector $\overrightarrow{USE}$ which is as a candidate for being considered as a possible reduced set of application systems, the following constraint holds:

$$SUP * \overrightarrow{USE} \geq \vec{e}^{T} \tag{8}$$

Second we want to have as few application systems in use as possible. We introduce a vector $\vec{c}$ of length *N* containing only "-1":

$$\vec{c} = (c_n)_{n=1...N} \quad \text{with} \quad c_n := -1, n = 1...N \tag{9}$$

This leads to the second postulation:

(P2)
$$\vec{c}^{T} * \overrightarrow{USE} \rightarrow \max \tag{10}$$

Since *SUP* is a matrix of zeroes and ones, we have a pure 0-1 linear programming problem. This problem is well known in literature as the "set covering problem" [10]. Corresponding to our statement that there will be more than one "minimal functionally non-redundant set of application systems" there are also different solutions for the set covering problem. The simplest algorithm, known as "brute-force", checks all combinations of application systems for postulations (P1) and (P2). Of course this would need too much computing resources for realistic information systems with several tenths of application systems. Moreover, set covering is an NP-complete problem generally, which, roughly, means that the complexity of any algorithm will be in the order of an exponential function of N. In section 0 we will briefly sketch an algorithm which manages the situation of usual information systems quite well and we will report on the application of this algorithm in Leipzig in section 6. So we can assume here that we actually can find a solution for the problem. The solution is the set $\underline{USE}^{\min}$ of all vectors $\overrightarrow{USE}_k^{\min} := (use_{k,n}^{\min})_{n=1...N}$, for which (P1) and (P2) hold, is defined as

$$\underline{USE}^{\min} := \{\overrightarrow{USE}_1^{\min}, ..., \overrightarrow{USE}_K^{\min}\} \tag{11}$$

This corresponds with the set

$$\underline{AS}^{\min} := \{\underline{AS}_1^{\min}, ..., \underline{AS}_K^{\min}\} \tag{12}$$

of minimal functionally non-redundant sets of application systems $\underline{AS}_k^{\min}$. In the sense of the set covering problem we could say every $\underline{AS}_k^{\min}$ covers $\underline{EF}$. Because of (P2), all those sets $\underline{AS}_k^{\min}$ are of the same cardinality

$$M := \left|\underline{AS}_k^{\min}\right| \tag{13}$$

We can now define the key performance indicator *Functional Redundancy Rate (FRR)* as a measure for functional redundancy in an information system, which can be used for information management:

$$FRR := \frac{N - M}{N} \tag{14}$$

FRR can be interpreted as the percentage of application systems which could be removed from the information system without loss of functionality.

Example (part 3):
Since the given information system in part 1 of the example is quite small, we can immediately identify two minimal functionally non-redundant configurations:

$\underline{AS}_1^{\min} = \{1,2,4,5,6\}$ and $\underline{AS}_2^{\min} = \{1,2,4,6,7\}$ which correspond to the vectors $\overrightarrow{USE}_1^{\min} = (1,1,0,1,1,1,0,0,0)$ and $\overrightarrow{USE}_2^{\min} = (1,1,0,1,0,1,1,0,0)$.

For $\overrightarrow{USE}_1^{\min}$ as one of the two minimal solutions in our example holds: $SUP * \overrightarrow{USE}_1^{\min} = \left(\overrightarrow{ISUP}_1^{\min}\right)^T = (1,1,3,1,1,1,1)$ (see table 3).

Table 3. *Vector* $\overrightarrow{ISUP}_1^{\min}$

p	ef_p	$isup_p$
1	A	1
2	B	1
3	C	3
4	D	1
5	E	1
6	F	1
7	G	1

Thus (P1) holds for $\overrightarrow{ISUP}_1^{\min}$. In the same way (P1) can be shown to hold for $\overrightarrow{ISUP}_2^{\min}$ as well.

FRR is only dependent on *N* and on *M*, being the number of application components and the cardinality of all minimal functionally non-redundant sets of application systems, respectively. With *N*=9 and *M*=5, we get:

$$FRR = \frac{9-5}{9} = 0,44 \tag{15}$$

Hence 44% of the application systems in our example could be removed.

4 Using the Functional Redundancy Rate and Minimal Non-redundant Sets of Application Systems to Support Information Management

This approach may be supportive for information management in different ways:

4.1 Benchmarking Information Systems

The Functional Redundancy Rate *FRR* may be used as a quality indicator, which supports benchmarking of information systems. Since besides the set of application systems the set of enterprise functions is one of the two input variables of *FRR*, it is obvious, that the structure (especially the granularity of the functions modelled) as well as the cardinality of this set will influence the result. Thus *FRR* depends on the individual way of modelling the enterprise functions in an institution and *FRR*s of different information systems may be incomparable. Moreover the *FRR*s derived by different models of different modellers of the same information system may differ as well. This problem can be overcome by using the same set of enterprise functions for models of those information systems which shall be benchmarked and compared. An appropriate set of enterprise functions for hospitals has recently been published as a reference model in [11]. Using this as a basis for *FRR* calculation can make information systems comparable with respect to their *FRR*. But of course complete modelling is needed anyway.

Example (part 4)
The Functional Redundancy Rate of 44%, which has been calculated in part 3 of the example, indicates that according to the model 44% of the application systems in this information system – 4 out of 9 – are superfluous. This is an indicator, that – given the model is sound and complete – information management in the respective hospital may not have been performed very systematically.

4.2 Reducing Operational Costs

Even if a particular application system cannot be shut down and omitted, it may cause unnecessary operational costs. Let $\underline{AK}_k^{\min}$ be a minimal set of used application systems and $isup_p$ the related individual degree of support by application systems in $\underline{AK}_k^{\min}$ for every enterprise function $ef_p \in \underline{EF}$. If for some p holds $isup_p^{\min} > 1$, this indicates, that users can use different application systems as support for the enterprise function ef_p Information managers should check, whether this option is really favored; since it may cause additional costs e.g. for customizing the different application systems the same way, providing catalogues of terms and diagnoses redundantly, additional training courses, and so on.

Example (part 5)
As can be seen in the vector $\overrightarrow{ISUP}_1^{\min}$ in part 3 of the example, users having to perform "administrative admission (outpatients)" (function *C*) have the option to choose between 3 application systems. Information management should check, whether it is appropriate to allow employees to choose between application systems *1* "CareMgmtSys", *2* "PatientAdministrationSystem(ADT)", and *4* "DepartmentalSystemRadiotherapy", if they have to admit outpatients; because this option causes additional expenses e.g. for training. Information management could decide that only the "PatientAdministrationSystem(ADT)" has to be used for the admission of

outpatients and could block the respective modules of the "CareMgmtSys" and "DepartmentalSystemRadiotherapy".

4.3 Shut Down of Superfluous Application Systems

Realizing the *FRR* of the information system the responsible chief information officer (CIO) will ask what application systems actually are superfluous and can be omitted. Using the concepts introduced before we can calculate the subset of those application systems, which are superfluous and can be omitted anyway. Other way round those application systems can be found, which by no means should be deleted. The calculation of the latter can simply be based on the matrix *SUP* (see formula (3)). An application system as_n cannot be deleted, exactly if there is an enterprise function ef_p such that as_n is the only application system supporting this function. Let us collect these application systems in the set:

$$\underline{AS}^{+} := \left\{ as_n \in \underline{AS} \mid \left(\exists ef_p \in \underline{EF} : (sup_{p,n} = 1) \wedge \left(\forall m \neq n : sup_{p,m} = 0 \right) \right) \right\} \tag{16}$$

The calculation of superfluous application systems is more difficult. Of course all those application systems supporting no enterprise function can be omitted. Furthermore already matrix *SUP* provides valuable information concerning possible replacement of one application system by a different one. If we define two application systems functionally equivalent if they support the same set of enterprise functions, *SUP* can be used to determine those application systems which are mutually equivalent.

$$equal(as_x, as_y) = true \Leftrightarrow \forall p : sup_{p,x} = sup_{p,y} \tag{17}$$

Doing so every application system could be replaced by one of its equivalents. Thus first decisions can be made, what application systems should be shut down. But there may be more superfluous application systems, which can be found by using the set $\underline{AS}^{\min} := \{\underline{AS}_1^{\min}, ..., \underline{AS}_K^{\min}\}$ as defined in (12) resp. calculated before. Let us define the set

$$\underline{AS}^{-} := \bigcap_{k=1}^{K} \left(\underline{AS} \setminus \underline{AS}_k^{\min} \right) \tag{18}$$

of those application systems, which have been found as *not* needed in *all* minimal sets $\underline{AS}_k^{\min}$. Thus the application systems in $\underline{AS}^{-}$ can be omitted anyway.

$$\underline{AS}^{?} := (\underline{AS} \setminus \underline{AS}^{-} \setminus \underline{AS}^{+}) \tag{19}$$

Finally the set $\underline{AS}^{?}$ in (19) contains application systems which are not clearly marked as needed or not. But we can use the equivalence relation mentioned before to group the members of $\underline{AS}^{?}$ into equivalence classes. Based on this we have to decide for every equivalence class, what member of this class should be used; the rest of the class can be omitted.

Example (part 6)
Using *SUP* of part 1 of the example immediately results in $\underline{AS}^{+} := \{1,2,4,6\}$; these application systems are marked with ⊕ in figure 1 and must not be omitted. As stated in part 3 of the example, $\underline{AS}^{\min} = \{\{1,2,3,5,6\},\{1,2,4,6,7\}\}$. Thus $\underline{AS}^{-} = \{3,8,9\}$, which means, that the application systems marked with ∅ in figure 1 should be omitted anyway. The application systems marked with ⑨ in figure 1 belong to the set $\underline{AS}^{?} = \{5,7\}$. Since both support the same set of functions, they belong to the same equivalence class and on of them can be selected to support function E. Given *Clinical-PathwaySys* is a personal favourite of the hospital's medical director the CIO maybe decides for *ClinicalPathwaySys* and consequently shuts down the *KnowledgeService.* Finally the CIO can reduce the information system according to figure 2:

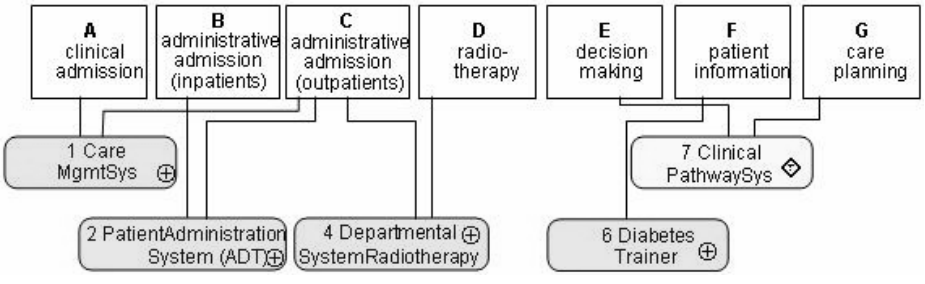

Fig. 2. Reduced information system (caption see figure 1)

4.4 Exploiting Potentials of Application Systems and Reducing Heterogeneity

In section 2 we defined the matrix *SUP* for modeling the support of enterprise functions by application systems. But there may be cases, that particular application software products could support more enterprise functions than the actual implementation, i.e. the application system, does. If a modeler adjusts the matrix *SUP* in a way, that it maps an enterprise function not only to application systems, which actually support this enterprise function, but also to those application systems, which *could* do so, usually more potentially superfluous application systems may be identified.

Example (part 7)
An analysis of the application software product, underlying application system "CareMgmtSys", may turn out that by a proper installation this application system could also support function "patient information". In this case, the "DiabetesTrainer" could be omitted, too.

5 An Algorithm for Calculating Minimal Non-redundant Sets of Application Systems

Our approach is mainly based on the set $\underline{USE}^{\min} := \{USE_1^{\min},...,USE_K^{\min}\}$ of minimal vectors defining minimal non-redundant sets of application systems. But up to now

we did not elucidate how to compute this set. As mentioned before, there are algorithms presented in literature, to solve the set covering problem. Far from starting a new discussion on optimal solutions for set covering problems in general we want to show the feasibility, i.e. the computability of the set $\underline{USE}^{\min}$ in real settings within acceptable time. According to our experiences there may be hundreds of application systems and enterprise functions in those settings. Applying set covering solving algorithms immediately would result in inacceptable computing efforts. But we have made also the following experiences:

E1. Most of the functions will be supported by exactly one application system. The corresponding application systems are members of $\underline{AS}^{+}$.

E2. Due to incomplete models there will be more or less application systems supporting none of the enterprise functions: $\underline{AS}^{0}$.

E3. There will be more or less application systems supporting only enterprise functions which are already supported by one of the application systems in $\underline{AS}^{+}$: $\underline{AS}^{-}$.

Thus we can reduce the set of application systems to $\underline{AS}^{reduced} := \left(\underline{AS} \setminus \underline{AS}^{+} \setminus \underline{AS}^{0} \setminus \underline{AS}^{-}\right)$. This set can be further reduced by using the equivalence relation (17) introduced in 0 and calculating the respective equivalence classes. Based on this we collect one (arbitrary) element from each class into the set $\underline{AS}^{equi}$. Now we can use $\underline{AS}^{equi}$ in place of $\underline{AS}$ to solve the set covering problem by one of the algorithms well known in literature delivering $\underline{AS}^{equi\min} := \{\underline{AS}_1^{equi\min}, ..., \underline{AS}_L^{equi\min}\}$ [10]. Based on this calculation and according to (17) we can calculate

$$\underline{AS}^{equi-} := \bigcap_{l=1}^{L}\left(\underline{AS}^{equi} \setminus \underline{AS}_l^{equi\min}\right) \tag{20}$$

If we take into account that every member of $\underline{AS}^{equi-}$ in fact is a place holder for an equivalence class, we can also derive $\underline{AS}^{?}$ as defined in 0.

6 Using the Functional Redundancy Rate and Minimal Non-redundant Sets of Application Systems at Leipzig University Medical Center

We implemented an algorithm in JAVA solving this set covering problem, which first reduces the set of application systems to be examined to $\underline{AS}^{equi}$ as described. $\underline{AS}^{equi}$ is explored using decision trees and a backtracking algorithm [12]. Using this algorithm we explored the 3LGM² model of the information system of Leipzig University Medical Center [13]. See table 3 for the results of the assessment.

Table 4. Analysis of the 3LGM² model of the information system of Leipzig University Medical Center

Functional Redundancy Rate (FRR)	25%
Number of application systems (N=$\lvert\underline{AS}\rvert$)	123
Number of application systems exclusively supporting functions ($\underline{AS}^{+}$)	86
Number M of needed application systems	92
Number of application systems supporting only functions which are already supported by one of the application systems in $\underline{AS}^{+}$ ($\underline{AS}^{-}$)	25
Number of equivalence classes	6
Number of application systems supporting no function ($\underline{AS}^{0}$)	0
Number of application systems to be examined by set covering solving algorithms	5
No. of redundant application systems found by set covering solving algorithms	4
Computing time (on a usual PC):	< 1 sec

The resulting *FRR* 25%, taken for itself, indicates that a quarter of the application systems could be removed without loss of functionality. The sets $\underline{AS}^{?}, \underline{AS}^{-}, \underline{AS}^{+}$ uncovered some interesting aspects regarding the model contents, e.g.:

- Since we modeled not only application systems of Leipzig University Medical Center but also of some hospitals in the neighbourhood, the algorithm suggested to omit the ADT-systems of these hospitals because the ADT-System of Leipzig University Medical Center would cover the functionality sufficiently.
- Two application systems supporting classification of diagnoses and procedures have been found as being superfluous. They can be omitted since a new system has been introduced some time before.

As shown in table 3 we have had quite small computing time; this is due to only 5 application systems to be examined by time-consuming algorithms.

7 Discussion

We have introduced the *Functional Redundancy Rate (FRR)* as a new key performance indicator for information systems. Even if redundancy of functions has been discussed in medical informatics in the context of functional integrity (e.g. [6, 14]) we could not find any formal and quantitative approach for computing a related key performance indicator before. Moreover it was surprising, that the set covering problem being discussed since many years turned out to be such a well suited formal description of the problem.

But the *FRR* and its use depends strongly on the solution of an NP-complete problem. Because of the NP-hardness, there is no way but to accept a possibly high running time of the algorithm. But this might not be a problem since a calculation time of a weekend or two would be acceptable for gaining a saving of several thousands of Euro. We proposed an algorithm to better manage the situation. Of course we could

not proof the computability of *FRR* in all settings using our algorithm. But we could show its computability in a realistic setting giving reason to assume its usefulness in similar settings as well.

Based on the algorithm the possible questions of information managers cited in the introduction can be answered. The sets $\underline{AS}^+$ and $\underline{AS}^-$ deliver the application systems being crucial respectively being obsolete. Additionally the *Functional Redundancy Rate (FRR)* supports benchmarking between different information systems.

Besides the complexity of the underlying set covering problem there is the additional problem of collecting all enterprise functions, all application systems and all their relationships for the calculation of *FRR*. Of course these efforts don't pay for only calculating the *FRR*. But if information management has a thorough description of the information system at its disposal, perhaps by having used the 3LGM² tool [15] the calculation of *FRR* does not need any further efforts.

The *FRR* for functional redundancy is only one key performance indicator for quality of information systems. Especially data redundancy is one more extremely relevant problem. Future research has to examine this and its relationships with functional redundancy as well and hopefully can result in considerable steps towards a sound and complete theory of quality of information systems. Dealing with quality criteria like functional or data redundancy makes evident, that a distinct ontological basis is needed independently from modelling approaches used. We need a common, unified ontology for describing information systems – not only in health care. We consider 3LGM² to be a proposal for first steps in this direction.

Acknowledgements

Thanks to Ernst Schuster. He helped us to find the proper formulation for our optimization problem. Parts of work have been supported by grants of the Deutsche Forschungsgemeinschaft (DFG).

References

1. Kuhn, K.A., Giuse, D.A., Lapao, L., Wurst, S.H.: Expanding the scope of health information systems - from hospitals to regional networks, to national infrastructures, and beyond. Methods Inf. Med. 46(4), 500–502 (2007)
2. Lorenzi, N.M., Riley, R.T.: Managing technological change: organizational aspects of health informatics. Springer, New York (2004)
3. Haux, R.: Individualization, globalization and health-about sustainable information technologies and the aim of medical informatics. Int. J. Med. Inform. 75, 795–808 (2006)
4. Ammenwerth, E., Aarts, J., Berghold, A., Beuscart-Zephir, M., Brender, J., Burkle, T., et al.: Declaration of Innsbruck. Results from the European Science Foundation Sponsored Workshop on Systematic Evaluation of Health Information Systems (HIS-EVAL). IMIA Yearbook of Medical Informatics 2006. Methods Inf. Med. 45(suppl. 1), 121–123 (2006)
5. Talmon, J.: Evaluation and implementation: A call for action. IMIA Yearbook of Medical Informatics 2006. Methods Inf. Med. 45(suppl. 1), 16–19 (2006)
6. Haux, R., Winter, A., Ammenwerth, E., Brigl, B.: Strategic Information Management in Hospitals. Springer, New York (2004)

7. Cimino, J.J., Zhu, X.: The Practical Impact of Ontologies on Biomedical Informatics. IMIA Yearbook of Medical Informatics 2006. Methods Inf. Med. 45(suppl. 1), 124–135 (2006)
8. Winter, A.F., Ammenwerth, E., Bott, O.J., Brigl, B., Buchauer, A., Gräber, S., Grant, A., Häber, A., Hasselbring, W., Haux, R., Heinrich, A., Janssen, H., Kock, I., Penger, O.-S., Prokosch, H.-U., Terstappen, A., Winter, A.: Strategic Information Management Plans: The Basis for systematic Information Management in Hospitals. International Journal of Medical Informatics 64(2-3), 99–109 (2001)
9. Winter, A., Brigl, B., Wendt, T.: Modeling Hospital Information Systems (Part 1): The Revised Three-Layer Graph-Based Meta Model 3LGM2. Methods Inf. Med. 42(5), 544–551 (2003)
10. Karp, R.M.: Reducibility Among Combinatorial Problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum, New York (1972)
11. Hübner-Bloder, G., Ammenwerth, E., Brigl, B., Winter, A.: Specification of a reference model for the domain layer of a hospital information system. Stud. Health Technol. Inform. 116, 497–502 (2005)
12. Cormen, T.H., Leiserson, C., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, Cambridge (2001)
13. Winter, A., Brigl, B., Funkat, G., Häber, A., Heller, O., Wendt, T.: 3LGM^2-Modeling to Support Management of Health Information Systems. International Journal of Medical Informatics 76(2-3), 145–150 (2007)
14. van Bemmel, J.H. (ed.): Handbook of Medical Informatics. Springer, Heidelberg (1997)
15. Wendt, T., Häber, A., Brigl, B., Winter, A.: Modeling Hospital Information Systems (Part 2): Using the 3LGM2 Tool for Modeling Patient Record Management. Methods Inf. Med. 43(3), 256–267 (2004)