

KAOS- β : Analysing EIS architecture using KAOS

Malihe Tabatabaie, Fiona A.C. Polack, and Richard F. Paige¹

Department of Computer Science, University of York, YO10 5DD, York, UK
{malihetb, fiona, paige}@cs.york.ac.uk

Abstract. There is increasing demand for approaches to develop more effective enterprise information systems (EIS). A possible solution is to focus on eliciting and elaborating *goals* prior to capturing EIS requirements. Focusing on EIS goals could help the developer team and other stakeholders (particularly decision makers) achieve a shared and structured understanding of the goals of the EIS and the overall enterprise. We present an investigation of the use of the KAOS approach to goal modelling in the domain of EIS. As a result, we propose a *tailoring* of KAOS for this domain; the tailoring has been developed through empirical studies. Our tailoring, called KAOS- β , is described, and an empirical evaluation is presented, investigating its applicability for defining and structuring the goals of *groups* of EIS.

1 Introduction

This paper describes an approach to the analysis of Enterprise Information Systems (EIS) based on the well-known KAOS goal-based approach. The approach uses structuring and modelling of the goals of EIS to address some of the common challenges of EIS [7]:

- lack of clear links between EIS and the organisation’s key priorities; lack of agreed measures of success;
- lack of effective engagement with stakeholders;
- lack of understanding of and contact with the supply industry at senior levels in the organisation;
- lack of ability to visualise complex software systems.

Elsewhere, we review current EIS structure and development [21, 20]. An enterprise is a collection of small or medium-sized businesses (SMEs) and partners that operates as a single organisation. Most EIS are designed and developed for the whole enterprise, rather than for a single business unit or department of the enterprise. They can deal with the problems of a large organisation (which includes different SMEs or partners), and they can deal with the problems of an organisation that includes different departments [21]. This leads to a definition of EIS as a software system that integrates the business processes of organisation(s) to improve their functioning [20, P. 24].

The challenges of EIS vary across enterprises, with many challenges related to scale and complexity. Some factors that create complexity in EIS are [21]:

- size and growth of information technology (IT), information systems (IS) and the organisation itself;

- the interactions between different IT and IS;
- the involvement of many different parties in the constructions and use of IT and IS;
- and,
- the ever-increasing rate of organisational and social change.

It is these factors that give rise to the failures of communication and understanding identified as challenges [7], above.

We identify the systems architecture as the key part of an EIS, and propose the goal-oriented approach to facilitate communication and understanding in the development of EIS. Through a structured process of analysis and modelling, EIS developers are led towards an understanding of the priorities of the enterprise organisation and its organisational goals. Analysis requires identification of a sufficiently-complete sets of goals, by engagement with the different group of stakeholders. Engagement with senior staff helps to create understanding between developers and organisational seniors. Structuring the goals of EIS leads towards a vision of the EIS-to-be and systems-as-is, and can address the challenge of EIS structural visualisation.

The need to identify enterprise-level goals led us to review well-known goal-oriented approaches. The KAOS approach is identified as the best match to EIS needs, and section 3 shows how the approach is used to analyse EIS goals. Like most goal-oriented approaches, KAOS provides heuristics rules for analysis and presentation. Whilst [13] describes how heuristic rules build on each other, and presents a case study in which KAOS is applied in steps, there is no general systematic guide to application of KAOS or validation of its results. Based on the example of a stroke-care EIS, a structured, flexible process, KAOS- β (KAOS-beta), is devised to guide goal development. KAOS- β exploits KAOS heuristics and addresses its limitations for enterprise-level analysis. The evaluation (section 4) addresses some of the challenges of evaluating such a process, and presents some results of evaluating KAOS- β . Finally the section 5 summarises the challenges, results and future directions of this work.

2 Goal-Oriented Approaches

There are at least 15 distinct goal-oriented approaches (GOAs) [11], from many areas of computer science¹. Existing GOAs address system-level goals, so need adaptation for EIS-level goal analysis. Approaches generally lack goal evaluation guidance. However, four approaches were identified as possible candidates for EIS goal analysis: GSN [12], KAOS [13], GBRAM [2], and i^* [3]. i^* is a well-known approach that has been applied to health care [1]; it claims to encourage the involvement of stakeholders in requirement analysis, and to help the developers to achieve deep understanding of the domain; however, it proved difficult to adapt i^* to the level of EIS goal analysis. GSN is a goal-structuring notation used for presenting the structure of arguments, for example in safety certification, and has been applied in research contexts to requirements analysis [9]. GSN structures an argument using goals, for presentation to stakeholders; in this way it can be a bridge between developers and stakeholders, but has

¹ A separate paper on the analysis of GOAs that was undertaken here is in preparation, and will be available on the lead author's website

little guidance on goal identification and validation [19]. KAOS and GBRAM address system-level requirements engineering, and have approaches for defining and refining system goals; KAOS is the better-documented approach. The Objectiver tool guides and supports the construction and documentation of KAOS models. Although KAOS gives little guidance on goal evaluation, it does validate system models using heuristics, for instance applying a “why” question, both bottom-up and top-down, to the goal model [13]. KAOS is thus the most appropriate basis for an EIS goal-analysis approach, but it needs some adaptation.

3 KAOS Adaptation for EIS

This section presents a brief summary of KAOS and introduces an example EIS whose goals are to be analysed (section 3.1). The EIS is then analysed using KAOS, leading to the derivation of KAOS- β .

KAOS is a goal-driven methodology, designed to elicit and validate requirements and to prove their consistency [6]. The developers of the Objectiver tool² state that KAOS extends the traditional “what question” approach to requirements with “why”, “who” and “when” questions; the approach comprises:

1. identifying and refining goals progressively until constraints that are assignable to individual agents are obtained;
2. identifying objects and actions progressively from goals;
3. deriving requirements on the objects and actions to meet the constraints; and
4. assigning the constraints, objects and actions to the agents composing the system.

KAOS develops a *goal model*, from which other models can be derived, covering the various views and aspects of the requirements engineering for a system: the *obstacle model*, the *agent model*, the *operation model*, the *object model* and the *behaviour model*. Lamsweerde [13] provides heuristic rules and patterns for the creation of the goal model. Preliminary goals are elicited by analysis of the current system, requirements and documentation, and through consideration of the KAOS taxonomy of goal categories. For each goal, sub-goals are identified by a process described as “identifying goals along refinement branches” [13]: a process of systematic challenge using structured “how” and “what” questions. During refinement, mappings between goals and agents are established and refined until there is one agent per goal. Obstacles, threats and conflicts related to the goals are identified. Lamsweerde’s guidelines are accompanied by advice on common pitfalls, reuse of refinements etc [13].

To develop a variant of KAOS for EIS goal analysis, KAOS was applied to a well-documented stroke-care EIS. The following section describes the example domain; in section 3.2 we outline the findings of the KAOS analysis. Section 3.3 summarises the KAOS- β process.

² www.objectiver.com

3.1 The Stroke-Care EIS

Stroke care is the management of medical procedures within a health service – here the UK’s NHS. The health service enterprise is made up of many business processes, and related organisations include hospitals and government bodies. NHS stroke care is an EIS within the wider health service enterprise: the organisational and medical systems depend on different business processes and partners and coherent EIS support. Stakeholders in the stroke-care enterprise include patients, their relatives, politicians, health specialists, IT specialists and health-service managers.

The strategic importance of stroke care (stroke is the third-greatest cause of death in the UK) has generated various EIS proposals and documents, representing the views of society and organisations who are interested in achieving the goals of stroke care. The UK Department of Health’s National Stroke Strategy (NSS) identifies the lasting and profound impact of strokes; it proposes a strategy for stroke care and stroke prevention [22], and is the main source in this study, because of its coverage of goals, non-functional requirements and soft goals. The NSS is backed up with: public documents on Danish electronic patient records [10] and US stroke care systems [17]; a regional health trust report covering IT for stroke care [14]; and some limited interview results.

3.2 Applying KAOS and Deriving the KAOS- β Process

The KAOS analysis follows the sequence of heuristic goal applications described for a case study by Lamsweerde [13]. The Objectiver tool was used to develop a KAOS goal model. This identified limitations of the KAOS, particularly in relation to modularity and traceability. Taking inspiration from other GOAs, KAOS was then adapted to give the KAOS- β process. Here, we highlight how KAOS has been adapted for EIS, and, in this way, outline how KAOS- β process differs from KAOS.

Preliminary goal identification [13] discovered a number of top-level goals of the stroke-care EIS, capturing the strategic-level aspirations of the enterprise. As in KAOS [13], the documentation of each goal includes a name (a unique identifier) and a definition, describing the goal in natural language. The definition also identifies phenomena related to the goal that could be monitored and controlled in the system. In EIS goal analysis, traceability of the goals and requirements to their source is crucial, both in analysis and in the presentation of results to the enterprise. The outcome of goal analysis is a set of integrated plans (an architecture) for EIS; it is the starting point for system-level analysis and design activities, and these must be able to trace back to the sources used in devising the EIS architecture. To support traceability, the KAOS- β process adds a mandatory *source* feature to the documentation of every goal. This identifies the document pages/paragraphs that are the origin of each goal.

In KAOS, optional features of each goal are *Type*, *Category*, *Source*, *Priority*, *Stability*, *FitCriterion*, *FormalSpec*, and *Issue* [13]. Because the level at which KAOS- β is used (EIS architecture development rather than system-level requirements engineering), features such as *FitCriterion*, which relate to measurability of goals, are of limited value. However, contextual information is often needed: for example, it is useful to identify which specific stakeholders have an interest in a goal, and goals may include terms

or assumptions that need elaborating or disambiguating. Earlier application of GSN to the stroke-care example [21] had made significant use of the GSN *context* concept [12], and has thus been incorporated in to the KAOS- β goal documentation. KAOS- β also adds a *notes* feature, to record goal information that does not fit in any other field, perhaps regarding future consequence or the need to further information. With these modifications, KAOS- β uses the same notations for forms and models as KAOS [13], we use a tabular format for recording goal features.

The documentation of goals leads to further goals, as well as the modification or removal of some goals. Whilst iteration may be implicit in the KAOS guidelines, in KAOS- β , we make the possibility of iteration explicit, to help analysts in applying the process.

In KAOS, consideration of goal categories contributes to goal identification [13]. The KAOS categories (behavioural vs Soft goals, and their sub-categories) refer to system-level requirements, and are less helpful at the strategic enterprise level. Instead, we use a modular structuring concept, and we apply it during the refinement of top-level goals to derive sub-goals. In the stroke-care example, goal refinement led to goal-bloat – the graph of goals and subgoals becomes unmanageable. This is exacerbated by KAOS' reliance on a single goal model, with no modularity. KAOS- β modularity is inspired by GSN modularity. In the stroke-care goal analysis, modules are identified via participant or stakeholder groups: health care specialists (doctors, nurses, ambulance staff etc.); community members (patients, family of patients etc.); stakeholders who deal with systems development (IT specialists, domain specialists, decision makers etc.). This leads to modules of relevance to the enterprise as well as to the EIS design activities, such as an IT module and a social module.

The goal model records the refinement links between goals. In KAOS, a link is documented with details of the refinement that has been made: *Name*, *SysRef*, *Status* and *Tactic* [13]. In KAOS, the name feature is used to remove any ambiguity. System reference (*SysRef*) refers to system-to-be or system-as-is. Status records whether the goal is still under refinement. Tactic records the refinement tactic used to derive the subgoal [13]. In KAOS- β , the tactic feature is used (additionally) to provide source information – that is, which documents or other sources were used in arriving at the refinement. This supports traceability, and also helps in evaluating the goal model.

In KAOS, the heuristics of goal refinement result in detection of agents, consideration of the wishes of agents, and the allocation of goals to agents. In the KAOS- β process, these heuristic rules are consolidated into two steps, to identify agents and to link them to goals. The identification of agents is closely related to the identification of personnel and stakeholder groups that form the basis of KAOS- β modules. Whilst KAOS- β uses the KAOS heuristic of asking “who” questions to identify agents, it does not use any system modelling (KAOS proposes sequence diagrams etc to relate goals and agents) because this is inappropriate to the EIS level. KAOS- β uses KAOS agent categories: in the stroke-care case, non-human agents are related to the operational context of the EIS, whilst the human agents are different groups of users and stakeholders: Patients; Health specialists (doctors, nurses, etc.); System developer; System maintainer; Decision makers (Hospital Managers, Government, Domain experts, etc.); Health Staff (system operators, data entry personnel etc.).

The KAOS heuristics advise refining goals until there is one agent per goal. In enterprise goal analysis, it is important to record responsibilities, but it is unnecessary (and inappropriate) to break down goals in such detail, since these are not (yet) the basis for system-level transactions. In the enterprise design level, there may be many agents to each goal.

In KAOS, the heuristics and patterns related to *analysing obstacles, threats and conflicts* are part of goal refinement. This interesting step allows the designers to detect and address (through additional goals) interactions among goals and requirements [13]. In KAOS- β , the analysis of goal interaction and the resultant iteration is raised to the status of a step in the process. For example, the stroke-care strategy documents yield conflicting goals relating to the ease of access to patient data (eg. by patients and for emergency stroke treatments) and data security (eg. the need to limit who can access personal data). As in KAOS, conflicts are identified in the goal diagram.

3.3 The Structure of KAOS- β

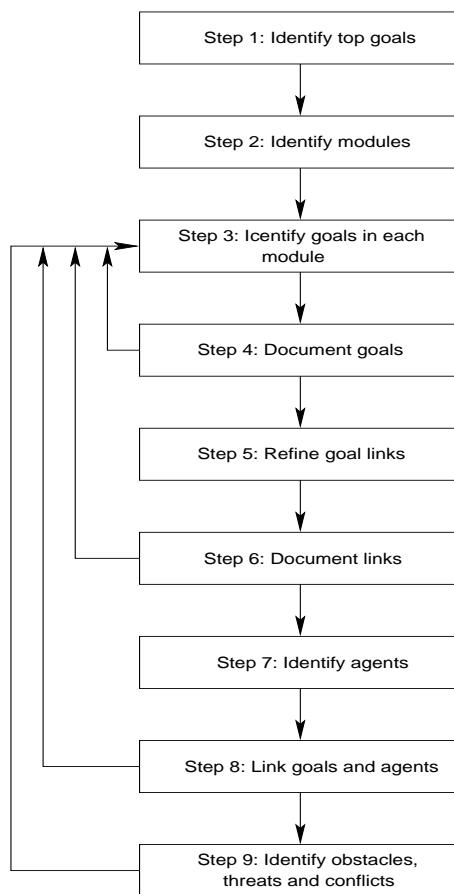


Fig. 1. Structure of the KAOS-Beta Process

The KAOS method is, in principle, applicable to EIS goal analysis. Most of the changes proposed are either the need for more detail to support traceability, or the need to provide clearer guidance on the path or paths that can be taken to develop the goal analysis. Omissions in KAOS- β are related to level: KAOS- β is applied at the level of enterprise strategy, to derive an EIS architecture, rather than at system level to engineer requirements.

Figure 1 summarises the steps and most likely iterations of the KAOS- β process. As in KAOS, *step 1* concerns identification of the top-level goals. For an EIS, this involves searching for strategic objectives, business goals, domain-specific objectives, and goals that could be refined by analysing requirements and problems.

In *step 2* the designers use modularity to structure the goal model. The heuristic used in the example to derive modules is to

consider participant groups, many of whom are later designated as agents. Modularity is a critical step: from this point, each KAOS- β step is applied to each module.

Within each module, *step 3* is akin to KAOS goal refinement. In *step 4* the designers document the goals, which leads to identification of missing, redundant, or mis-specified goals: the documentation process is thus part of the validation of the identified goals. Iteration of steps 3 and 4 is usually required.

In *steps 5 and 6* the designers refine goals and document the refinement links, creating traceability within the goal structure. Iteration occurs as new goals are identified.

Step 7 detects agents in the same way as KAOS. In *step 8* the designers associate the agents to goals. In *step 9* the designers search for obstacles, threats, and conflicts, with the appropriate iteration to step 3 to check for any changes in the goal set or the links. All these iterations are part of evaluating and improving the goal structure.

4 Evaluation of KAOS- β

The KAOS- β process model is a straightforward extension and modification of KAOS, adapted to the level of EIS goals. Evaluation is a non-trivial problem, particularly in the context of a small project that does not have the capacity to apply the process in many realistic situations. There is little guidance on process evaluation, and that which exists is often inapplicable. For example, Gruhn [8] proposes an approach evaluates application of a process in a specific context; the approach also focuses on temporal and dynamic aspects of process application: neither is appropriate here.

The KAOS- β development is considering many angles for evaluation: this paper focuses on three: general process characteristics; internal validity (or soundness) and external validity (based on [4]).

Internal validity addresses the detail of the process: the structure, applicability and heuristics, as well as how the outputs are validated. The KAOS- β process model is represented as a series of steps. Clearly identifying a process, its steps and links makes it possible to address its internal validity. It can be seen that the process and the steps themselves conform to most of the process characteristics listed above; KAOS- β also offers the potential for evolvability and adaptability.

In terms of detail, KAOS is a widely-used approach to requirements engineering, and we can assume that it is internally valid. We can thus infer at least some internal validity for the KAOS- β process model through its derivation from KAOS, as summarised in section 3.3. The internal differences between KAOS- β and KAOS are derived by application of the process to an EIS. Here, internal validity can be shown by appeal to best practice: we add modularity to manage the scope of EIS goal analysis, and we add traceability so that goal models can be maintained, and can be used to initiate specific systems projects in the enterprise.

The KAOS- β process includes explicit iteration, to support the exploratory nature of goal analysis. The validity of the output of the process (the goal model) is improved through iterative development and oversight of the process and results by the enterprise stakeholders.

In terms of goal evaluation, KAOS- β Step 9, identifying obstacles and threats, requires the analyst to consider interaction and negative aspects of goals in evaluating

the goals. KAOS goes further, for example looking at the converse of goals; KAOS- β evolution should consider adding further goal evaluation heuristics as optional steps, or adaptation options for use in other EIS situations.

KAOS heuristics advise continuing goal identification “to the system boundary” as a stopping condition, and also advise that refinement should continue until each goal maps to one agent. For KAOS- β , we propose a validation check that defines the boundaries of each module and of the EIS (i.e. the scope of the goal analysis), and determines that the identified goals reach the boundaries: this is a coverage condition. The second validation check is that all goals should be mapped to agents – though we allow goals to map to many agents, as above: when the enterprise wishes to proceed to system development, the goal model would form the starting point for requirements engineering.

External validity looks at the general requirements for a valid process. Here, we use Ramsin’s criteria [16] for evaluating software engineering processes. However, first we consider the general requirements for a (software engineering) process. We draw on evaluations of process presentation techniques for UML [5] and Petri-nets [4, 15], and a general definition of process as a set of activities, associated results and a product [18], and add to the requirements of a process the need for defined inputs, defined outputs, and linkage between steps. The type and domain of input and output of the process shall be defined for the users of the process. The KAOS- β process outlined above conforms to these general process characteristics: it is a process to create an enterprise-level goal structure and EIS architecture; it comprises a set of well-defined activities, represented as steps, with associated results that link the activities. As a result a goal structure which is part of the software product design is created.

Good engineering practice also proposes that a process should be customisable, both in its ability to evolve, and in relation to adaptation to different situations. This cannot be assessed directly at this stage.

The criteria-based evaluation applied all Ramsin’s criteria [16], though space does not permit a complete presentation here. Three criteria require comment: (a) consideration of the clarity, rationality and consistency of the process definition which cannot be assessed until KAOS- β is fully reported; (b) coverage of generic lifecycle development activities is addressed in so far as is possible, but KAOS- β precedes most conventional lifecycle development activities; (c) support for umbrella activities (risk management, project management and quality assurance) have not yet been addressed. The other criteria are summarised in table 1.

This qualitative analysis illustrates that KAOS- β could be applied for some cases of EIS. We could get confidence about it by testing it in different cases. We believe that there is no one good process that could be applied to all types of EIS, hence instead we should focus on the best practices.

Table 1. Evaluation of KAOS- β against Ramsin's criteria [16]

Seamlessness and smoothness of transition between phases, stages and activities	√	KAOS- β retains and improves the flow between activities (steps), facilitating iteration or omission of optional steps
Basis in the requirements	√	KAOS- β address the needs of the example EIS goal analysis and is aligned with general definitions of process and of EIS
Testability and Tangibility of artefacts, and traceability to requirements	√	Artefacts are tangible: goals, agents, refinement (goal-model); testability is heuristic; KAOS- β traceability is explicit
Encouragement of active user involvement	√	Intended and facilitated in KAOS- β , as in KAOS
Practicability and practicality	(√)	KAOS- β is applicable EIS; it is a practical modification of KAOS. Full practicality needs umbrella activities (further work)
Manageability of complexity	√	KAOS- β 's clear activities and links minimize complexity
Extensibility / Configurability / Flexibility / Scalability	√	Addressed by iterative process, modularity; process potentially customisable
Application scope (Information Systems)	√	Scope is EIS goals/architecture

5 Conclusion and Future Work

KAOS- β is a modification of KAOS for EIS goal analysis. It has been developed through experimental application of KAOS to the stroke-care enterprise, a non-trivial example of EIS that includes different business processes, stakeholders and viewpoints. Some of the work being undertaken to evaluate KAOS- β and to validate the output of the process is outlined.

Work is continuing on the development and documentation of KAOS- β . Support for traceability needs evaluating and testing. More work is needed in identifying and applying appropriate evaluation techniques. KAOS- β can be applied further within the identified modules of the stroke-care goal model, and should be applied to other EIS, to determine shortcomings and to improve the quality of the process model.

References

1. Yuan An, Prudence Dalrymple, Michelle Rogers, Patricia Gerrity, Jennifer Horkoff, and Eric Yu. Collaborative social modeling for designing a patient wellness tracking system in a nurse-managed healthcare center. In *International Conference on Design Science Research in Information Systems and Technology*. ACM, 2009.
2. Annie I. Antón. Goal-based requirements analysis. In *IEEE International Conference on Requirements Engineering*, pages 136–144, 1996.
3. Jaelson Castro, Manuel Kolp, and John Mylopoulos. Towards requirements-driven information systems engineering: the tropos project. *Information Systems*, 27(6):365–389, 2002.

4. Bill Curtis, Marc I. Kellner, and Jim Over. Process modeling. *Communications of the ACM*, 35(9):75–90, 1992.
5. Marlon Dumas and Arthur H.M. ter Hofstede. UML activity diagrams as a workflow specification language. In *International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, volume 2185 of *LNCS*, pages 76–90. Springer, 2001.
6. Emmanuelle Delor and Robert Darimont and Andr Rifaut. Software Quality Starts with the Modelling of Goal-Oriented Requirements. [Accessed 20 Sep 2009] Available at: www.objectiver.com, 2009.
7. RAE-BCS Working Group. The challenges of complex IT projects. Technical report, The Royal Academy of Engineering, 2004.
8. Volker Gruhn. Validation and verification of software process models. In *European symposium on Software development environments and CASE technology*, pages 271–286. Springer, 1991.
9. Ibrahim Habli, Weihang Wu, Katrina Attwood, and Tim Kelly. Extending Argumentation to Goal-Oriented Requirements Engineering . In *Advances in Conceptual Modeling - Foundations and Applications: ER Workshops*, volume 4802 of *LNCS*, pages 306–316. Springer, 2007.
10. Eben Harrell. In Denmark’s Electronic Health Records Program, a Lesson for the U.S. [online]. [Accessed 16 Sep 2009] Available at: www.time.com, April 2009.
11. Evangelia Kavakli and Pericles Loucopoulos. Goal driven requirements engineering: Analysis and critique of current methods. In John Krogstie, Terry A. Halpin, and Keng Siau, editors, *Information modeling methods and methodologies*, pages 102–124. IGI, 2005.
12. T. P. Kelly. *Arguing Safety – A systematic approach to managing Safety Cases*. PhD thesis, University of York, Department of Computer Science, 1998.
13. Axel Van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2009.
14. Leeds Team. Team notes: Research strategy on information systems for stroke care. Technical report, University of Leeds, August 2008.
15. Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
16. Raman Ramsin. *The Engineering of an Object-Oriented Software Development Methodology*. PhD thesis, University of York, Department of Computer Science, 2006.
17. Lee Schwamm, Arthur Pancioli, Joe Acker, Larry Goldstein, Richard Zorowitz, Timothy Shephard, Peter Moyer, Mark Gorman, Claiborne Johnston, Pamela Duncan, Phil Gorelick, Jeffery Frank, Steven Stranne, Renee Smith, William Federspiel, Katie Horton, Ellen Magnis, and Robert Adams. Recommendations for the establishment of stroke systems of care. *American Stroke Association*, 36:690–703, 2005.
18. Ian Sommerville. *Software Engineering*. Addison Wesley, 8 edition, 2006.
19. Malihe Tabatabaie. Applying gsn to stroke care. Technical report, University of York, 2009.
20. Malihe Tabatabaie, Richard Paige, and Christopher Kimble. Exploring the boundaries of enterprise information systems. In *Second York Doctoral Symposium on Computing*, volume YCS-2008-434, pages 27–34. Department of Computer Science, University of York, 2008.
21. Malihe Tabatabaie, Richard Paige, and Christopher Kimble. Exploring enterprise information systems. In Maria Manuela Cruz-Cunha, editor, *SOCIAL, MANAGERIAL, AND ORGANIZATIONAL DIMENSIONS OF ENTERPRISE INFORMATION SYSTEMS*, pages 415–432. IGI, 2010.
22. DH Stroke Policy Team. National Stroke Strategy. Department of Health: www.dh.gov.uk/en/Publicationsandstatistics/Publications/, 2007.