MEDINFO 2001
V. Patel et al. (Eds)
Amsterdam: IOS Press
© 2001 IMIA. All rights reserved

GALEN Ten Years On: Tasks and Supporting Tools

Jeremy Rogers[a], Angus Roberts[a], Danny Solomon[a], Egbert van der Haring[b]
[a]Christopher Wroe, Pieter Zanstra[b], Alan Rector[a]

[a]Medical Informatics Group, University of Manchester, UK & [b]University of Nijmegen, Netherlands

Abstract

The GALEN technology has matured over more than a decade of use. We describe a set of software tools and associated methodologies that together are supporting ontological engineering in a production, rather than a research setting.

Keywords:

Integrated Advanced Information Management Systems, Software Design, Classification, Terminology

Introduction

Medical terminology is recognised as one of the foundational resources needed if our hopes for healthcare computing are to be realised [1]. Traditional terminologies, optimised for direct human use, inherently constrain our ability to recruit computers to deliver our hopes for extensive data analysis, sharing and re-use. New kinds of terminology, designed for computation, are required.

One approach to a solution has been by gradual evolution of those terminologies originally designed for human use [2][3][4][5]. The GALEN program, by contrast, proposed a paradigm shift [6]: replace static look-up terminologies with a Common Reference Model and an automatic classification inference engine, in a Terminology Server.

Sophisticated tools support the building and maintenance of the GALEN resources, and these have now matured to the point where they are used in production as well as research environments. The range of these tools illustrates both the power and the complexity of building sophisticated terminological resources.

Most of the tools and methods have been described individually in previous papers. This paper gives an overview of the entire process. An outline chronology of the challenges that motivated the overall development programme is also presented.

Figure 1 summarises diagrammatically how the tools and methodologies interrelate. The various knowledge bases are shown, linked by arrows representing the broad knowledge management processes that interrelate them. These arrows pass through symbols representing the software tools that implement or facilitate the knowledge management.

Four main activity streams are indicated in the diagram, and discussed in this paper:

- GRAIL Authoring: constructing the central core of a common reference model
- Authoring in Intermediate Representation (Intermediate Representation): linking external knowledge to the common reference model
- Quality assurance of all involved knowledge bases
- Delivering the result; localising and tailoring the common reference model to provide specific applications with what they need

A fifth activity discussed runs orthogonally to the other four and is not evident in the diagram:

- Collaborative working on all activities

GRAIL Authoring: The KnoME

History: The common reference model (Common Reference Model) [7][8] is a formal model within which concepts may be defined, described and automatically classified according to formal criteria for equivalence and subsumption.

Initial implementations of the Common Reference Model authoring environment were closely coupled with the inference engine. An early priority was to re-implement the authoring environment as a true client application – the Knowledge Management Environment (KnoME) – communicating with a terminology server engine through a common client API [9]. This architecture allowed different server and compiler client applications to be developed, substituted and compared. This in turn motivated extensions to and clarifications of the specification of all components. The resulting separation enables us to take advantage of more powerful component implementations as they arise.

The KnoME supports all the activities in the 'GRAIL Authoring' area of figure 1, as well as some QA activities. It is composed of several tools, centred around a GRAIL editor and a browser for the internal form:

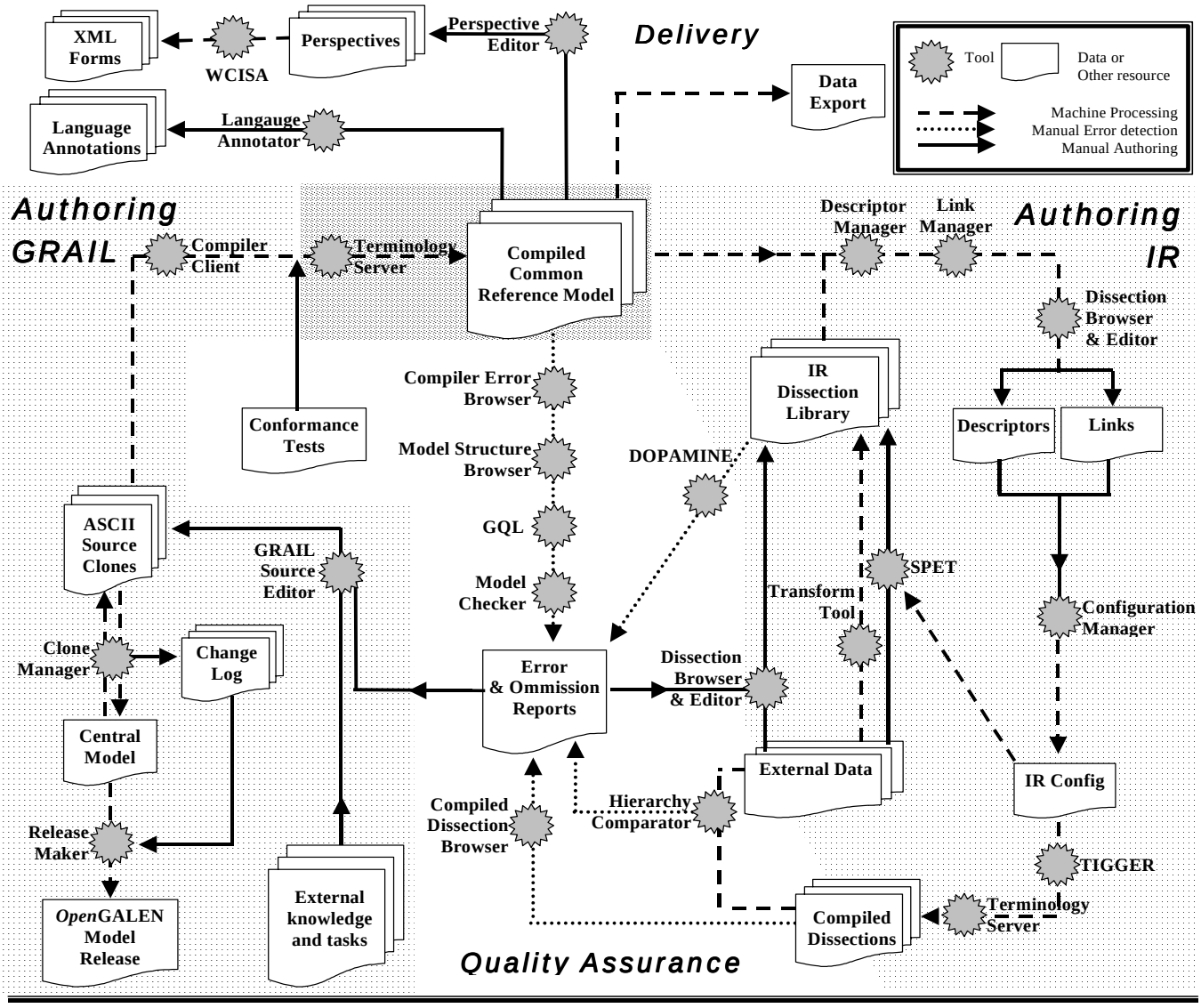

Figure 1: Diagrammatic representation of GALEN Tools and Tasks. Shaded areas denote major activities.

GRAIL Source Editor: The Common Reference Model is authored in a description logic dialect called GRAIL [10]. Primary authoring occurs in text files that are then compiled to an internal form. In common with many description logic tools, 'roll-back' or 'undoing' of the internal form is technically difficult, so revisions must be recompiled from the source files. Furthermore, GRAIL syntax demands that concepts be defined before they are used, and so the order of the source files is critical. Source file management is, therefore, a critical task.

Initially the Common Reference Model sources were divided between some 30 files. However, as it grew, the sources became unmanageable by normal manual means: authors could not locate code that needed changing, and much effort was spent re-ordering source statements.

The source model editor was therefore devised as an integrated tool organising source file subdomains and maintaining compile order. Using this tool, the current 3Mb of source text is maintained as some 1250 separate files. They are presented in a uniaxial and arbitrarily deep tree such that, for example, all sources relating to anatomical partonomy are located in one branch and all those to pathology in another. Other features of the source editor include a precompiler check for syntax and statement ordering errors, and various string searching tools that operate across the whole source unit structure or identified branches of it. These are used extensively to locate statements that need editing.

Model Structure Browser: The original browser for the compiled form of the Common Reference Model presented everything that was known about a concept in a single interface with multiple panes. As the model grew in complexity the browser was reengineered: a central display shows only a concept in its type hierarchy. Other information, such as concept definition or constraints operating on the concept, may be requested as additional pop-up displays. Screen shots of these tools can be found at www.topthing.com/knome.

Authoring Intermediate Representation

History: authoring in GRAIL syntax is time consuming and requires extensive training. An easier syntax and less complex semantics were needed. The GALEN Intermediate

Representation (Intermediate Representation) and an associated toolset – SPET and TIGGER - for authoring and processing Intermediate Representation 'dissections' were born of this requirement [11][12][13].

A major value of the Common Reference Model is as an index and link to other resources such as static coding schemes or items in decision support systems and EPRs - surgical procedure codes in GALEN-IN-USE, drug interactions, indications, and other information in the PRODIGY GP protocol project. The processes of authoring application specific concepts and linking specialised knowledge to the Common Reference model are combined in a suite of associated tools described below.

Authoring in the Intermediate Representation: Intermediate Representations are 'soft'; different applications or users may redefine them. The SPET tool facilitates an entirely manual approach in which the meaning of rubrics from traditional schemes, such as ICD, is declared in Intermediate Representation. By contrast, schemes that are already at least partially compositional – such as the UK Clinical Terms V3 – may be automatically re-expressed directly in Intermediate Representation syntax.

The intermediate representation greatly simplifies the work of authoring. The trade off is that there are now three resources to manage: the Common Reference Model source files, the Intermediate Representation libraries, and the files defining the transformations between them.

Intermediate Representation-Dissection Libraries: Authors using the intermediate representation to create dissections typically work systematically through their source schemes, storing their work as many dissection files (e.g. one for each chapter of the original scheme). When collected at the centre as a 'library of dissections' the number of files becomes large. A dissection browser in the TIGGER supports a rough manual indexing of the cumulative dissection library.

Intermediate Representation- Configuration: Expressions composed in Intermediate Representation have explicit semantics, but are phrased using a vocabulary and structure which is specific to the application and must be transformed to the Common Reference Model. The TIGGER environment defines how to transform from an intermediate representation to the Common Reference Model. Fashioning such transformations requires analysis of the Intermediate Representation as used, a task supported by the descriptor and link manager subcomponents of TIGGER. These perform functions such as: retrieving all intermediate representation expressions using a specified descriptor or link; declaration of mappings from descriptors to other descriptors or to the Common Reference Model. .

The TIGGER toolset was devised for situations where both a single rigid Common Reference Model target ontology and a small number of broadly similar and related *Intermediate Representation* source ontologies for a given subdomain have already been established but require linking. However, the same environment has proved valuable in supporting an iterative 'refine and test' work cycle in the development of novel *Intermediate Representation* and Common Reference Model ontologies for new subdomains, such as a drug dictionary ontology for the UK PRODIGY project.

Processing Intermediate Representation: the output of the Intermediate Representation analysis is an explicitly declared Intermediate Representation to Common Reference Model mapping. This allows Intermediate Representation dissections using that particular Intermediate Representation ontology to be 'expanded' into the richer Common Reference Model ontology. TIGGER itself provides one implementation of the expansion algorithm [14] but the declared mapping information can also be exported as an Intermediate Representation configuration file for use in standalone implementations of the transformation algorithm.

Viewing Expanded Dissections: the final result of dissection authoring and processing is a set of concepts in the Common Reference Model linked to identifiers for external objects. The external object identifiers can therefore be re-displayed in a hierarchical view that mirrors the type hierarchy of Common Reference Model concepts they link to. The Compiled Dissection Browser offers this functionality; it provides a useful early quality check as dissecting work progresses.

Checking and Quality Assurance

History: tools such as the Compiled Dissection Browser and Model Structure Browser support only *ad hoc* discovery of errors of fact in the knowledge bases. Additional tools are required to search for, characterise and manage errors in more systematic ways.

GRAIL Conformance Suite: The development of the knowledge bases occurred contemporaneously with development of the GRAIL server implementation itself. This risked introducing spurious errors arising from problems in the server rather than in the knowledge base. A conformance test, intended to systematically confirm that any new compiler/server combination behaves as anticipated, was instituted.

Grail Query Language (GQL): Authoring the conformance suite required an error trapping syntactic extension to GRAIL. The GRAIL Query Language (GQL) includes expressions that take whole GRAIL statements as an argument, to test whether the result of evaluating the GRAIL raises expected compiler errors. For example:

```
(Leg newSub Leg) shouldRaiseError.
```

Other GQL statements allow the automatic classification mechanisms to be tested. Small but complex models are built, and specific subsumption inferences that should or should not be made are tested for. For example:

```
(Fracture which hasLocation Femur)
shouldSubsume
(Fracture which hasLocation
   (Shaft which isStructuralComponentOf Femur).
```

GQL was subsequently extended to support query scripting: other statements take sets of Common Reference Model concepts as arguments and return defined subsets such as all concepts that are descendants of a given concept, and whose knowledge name also contains a certain string infix, or that are also simultaneously descendants of another concept. For example, the following GQL query tests for the presence of any muscles that are also classified as parts of the skeleton:

```
ComponentOfSkeleton descendents
  selectEach isKindOf Muscle
  size shouldBeEqualTo 0.
```

GQL thus offers a way to partially formalise a metamodel of the Common Reference Model: queries can be constructed to detect all instances where modelling has been authored in a particular but undesirable semantic style.

Cross Validation: Some external resources – especially classifications - already incorporate a type hierarchy. Linking the Common Reference Model to such resources offers the possibility to cross-validate. An alternative type hierarchy can be independently computed by reference to the semantic properties of the Common Reference Model objects linked to. Comparison of the two structures can identify errors in either the Common Reference Model or the original source [15]. However, systematic comparison by manual means alone is inefficient and error-prone. A tool, the Hierarchy Comparator, automates the comparison and assists with discovering the origin of and deciding what to do with any differences detected.

Visualisation: In situations where automatic cross validation is not possible, manual checking remains the only option. When manually authored dissections become individually large and detailed and collectively numerous, the problem of checking the dissection library itself for factual errors or omissions becomes acute. Visualisation aids such as DOPAMINE [16] – in which knowledge base extracts are presented as spreadsheets - increase the ability of human checkers to spot such problems.

Central Error Handling: GALEN implements several levels of indirection between sources and targets, to isolate each from changes in the other. The overall process of authoring and checking the knowledge bases is deliberately iterative. As a consequence the detection, documentation and resolution of errors becomes the main driver for that process. Error logging, consequently, is critical. Three separate tools currently manage error logging, but their integration is a major goal for future development.

Delivering the model

History: The GALEN Common Reference Model aims to be re-usable. Consequently it contains information that is more detailed than required by any single application. In addition to suppressing unwanted information, applications need user-friendly presentation of that information that is required. To achieve this three further tools are required..

Perspectives: A perspective is a set of filters and systematic simplifications of the Common Reference Model ontology (similar to running the Intermediate Representation to GRAIL expansion algorithm in reverse) which, when applied to the Common Reference Model as a whole, results in a reduced view of the Common Reference Model being visible to the outside. A set of several perspectives applied simultaneously may produce a sufficiently simplified and focussed view that can approximate closely to a true de-merged application specific vocabulary. For any given specific use or application, localising the Common Reference Model is therefore achieved by using an appropriate set of perspectives. Work on the perspectives mechanisms themselves, and on the best methodology to index sets of perspectives relevant to a specific application, is ongoing.

Delivery to Applications in XML: The 'What can I say about…?' tool (WCISA) applies perspective sets and generates XML specifications of sample Stuctured Data Entry forms and form fragments. This allows full prototyping of the mechanism to deliver the Common Reference Model via a forms based approach to be conducted.

Language Annotation Database: Generation of natural language phrases for underlying Common Reference Model expressions, in multiple languages, has been a feature of GALEN from the beginning [17]. Originally conceived as functionality primarily required for final rendering to end users, it has since become an important component of the knowledge base checking process: many GRAIL representations of knowledge are large and difficult to read. Generating language directly from GRAIL expressions in the Common Reference Model facilitates the process of comprehending what another author has said, whether or not both authors speak the same natural language. However, implementing large-scale language generation across the whole Common Reference Model requires that each Common Reference Model concept carries one or more lexical annotations. The Language Annotator tool maintains an annotations database in step with successive Common Reference Model releases.

Collaborative development

History: The development of the Common Reference Model has always been performed by more than one author at any one time. As the Common Reference Model became larger and more interwoven, rigorous collaborative working procedures have become necessary.

Clone Manager: The source unit editor has been extended to provide a multi-author and distributed modelling environment with full check-in/check-out and cloning of source units. Authors working together do so via individual remote cloned copies of a single centralised canonical GRAIL source file store. Elements of the central model source store may be checked-out to individual clones, edited remotely and revised versions checked back in. Other authors are updated automatically with revisions whenever their clones connect with the central model. .

Release Maker: Periodically further development of the Common Reference Model is frozen and a snapshot external release is created. These are the sources posted on

the OpenGALEN website. The release procedure is now encapsulated in a Release Maker tool. This includes a number of final routine quality checks, as well as a tool to help inspect the full audit trail and change logs in order to prepare a more concise summary of Common Reference Model changes since the last version.

Discussion

The description above describes and chronicles the development of one set of linked resources and supporting tools based on GALEN technology. These resources are developed and used in five broad activity streams, but these overlap and their interactions drive continuous quality assurance. Figure 1 illustrates the existence of many potential iterative quality assurance cycles and subcycles within the overall workflow.

The methodology set out owes much of its complexity to the fact that the knowledge to be represented is evolving. These resources must change consistently to produce a coherent integrated whole. The iterative development methodology relies on a combination of careful detection and analysis of unexpected behaviour together with methods to encode and build on the lessons learnt.

Historically, data linked to the Common Reference Model originated external to this environment – *e.g.* classification centres. The environment described is supporting *de novo* co-operative development of large terminologies and ontologies. Within the PRODIGY project [18] a new resource – a drug ontology and populated drug index – is being developed entirely within this environment but will be exported as a free standing resource [19].

Future work: the collaborative GRAIL and Intermediate Representation authoring tools are currently being consolidated. A significant future challenge will be addressing version control issues: at any one time multiple versions of both the knowledge bases and of the various software tools and engines exist. Reengineering the underlying formalism and inference engine, for example to FACT [20], is also under consideration.

Acknowledgements

The work detailed here has been funded by the EC Framework III and IV programs and by the NHS Executive.

References

[1] Cimino JJ. Desiderata for controlled medical vocabularies in the twenty-first century. *Methods Inf Med*. 1998 Nov;37(4-5):394-403

[2] Spackman KA, Campbell KE. Compositional concept representation using SNOMED: Towards further convergence of clinical terminologies. *Proc AMIA Symp. 1998*;:740-744

[3] Campbell KE, Tuttle MS, Spackman KA. A "lexically-suggested logical closure" metric for medical terminology maturity. *Proc AMIA Symp. 1998;*:785-789

[4] O'Neil M, Payne C, Read J. Read Codes Version 3: A user led terminology. *Methods Inf Med* 1995;34:187-192.

[5] Brown P, O'Neil M, Price C. Semantic definition of disorders in Version 3 of the Read Codes. *Methods Inf Med* 1998;37:415-419

[6] Rector AL, Zanstra PE, Solomon WD, Rogers JE, Baud R, Ceusters W, Claassen W, Kirby J, Rodrigues JM, Mori AR, van der Haring EJ, Wagner J. Reconciling users' needs and formal requirements: issues in developing a reusable ontology for medicine. *IEEE Trans Inf Technol Biomed*. 1998 Dec;2(4):229-42

[7] Rector A, Rogers JE ,Pole P (1996) The GALEN High Level Ontology. *Fourteenth International Congress of the European Federation for Medical Informatics*, MIE-96, Copenhagen, Denmark

[8] Rector AL, Rogers JE. Ontological Issues in using a Description Logic to Represent Medical Concepts: Part II - The GALEN High Level Schemata. *Methods Inf Med* 2000;

[9] Rector AL, Solomon WD, Nowlan WA, Rush TW, Zanstra PE, Claassen WM. Terminology Server for Medical Language and Medical Information Systems. *Methods Inf Med* 1995;34:147-157

[10] Rector A, Bechhofer S, Goble C, Horrocks I, Nowlan W, Solomon W. The GRAIL concept modelling language for medical terminology. *Artif Intell Med.* 1997 Feb;9(2):139-71.

[11] Rogers J.E., Solomon, W.D., Rector, A.L., Pole P.M.,P Zanstra, E van der Haring. Rubrics to Dissections to GRAIL to Classifications. *Stud Health Technol Inform*. 1997;43 Pt A:241-5

[12] Rogers J, Rector A. Terminological Systems: Bridging the Generation Gap. *Proc AMIA Symp. 1997*;: 610-614

[13] http://www.ehm.kun.nl/efcc/19981209/irconfig.rtf

[14] Solomon W.D., Roberts A., Rogers J.E., Wroe C.J. , Rector, A.L. Having our cake and eating it too: How the GALEN Intermediate Representation reconciles internal complexity with users' requirements for appropriateness and simplicity. *Proc AMIA Symp. 2000*;:819-23

[15] Rogers J.E., Price C, Rector, A.L, Solomon W.D., Smejko N. Validating Clinical Terminology Structures: Integration and Cross-Validation of Read Thesaurus and GALEN *Proc AMIA Symp. 1998*;:845-9.

[16] C. Wroe, W.D. Solomon, A.L. Rector and J.E. Rogers. DOPAMINE - A Tool for Visualizing Clinical Properties of Generic Drugs. Nada Lavrac, Silvia Miksch, Branko Kavsek (eds.): *The Fifth Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-2000), Workshop Notes of the 14th European Conference on Artificial Intelligence* (ECAI-2000), 2000: p61-65

[17] Wagner JC, Rogers JE, Baud RH, Scherrer J-R Natural language generation of surgical procedures *Int J Med Inf*; 1999 Feb-Mar;53(2-3):175-92

[18] Purves I. Prodigy, a computer assisted prescribing scheme. Interim data show that it is worth taking the scheme further.*BMJ*. 1996 Dec 14;313(7071):1549.

[19] Solomon WD, Wroe CJ, Rector AL, Rogers JE, Fistein JL, Johnson P. A reference terminology for drugs. *Proc AMIA Symp*. 1999;:152-6.

[20] Horrocks I. Using an expressive description logic: FaCT or Fiction. In: Cohn AG, Schubert LK, Shapiro SC, editors. Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference on Knowledge Representation (KR 98); 1998; San Francisco, CA: Morgan Kaufmann; 1998. p. 634-647.

Address for correspondence

Dr Jeremy Rogers (jeremy@cs.man.ac.uk)
www.opengalen.org