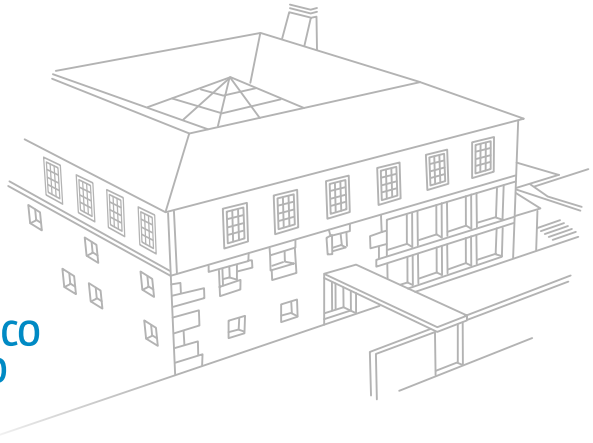

ESTGF | POLITÉCNICO DO PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Guias Clínicos na Web

DESIGNAÇÃO DO MESTRADO Mestrado em Engenharia Informática

AUTOR Luís Miguel Carvalho da Cunha

ORIENTADOR(ES) Professor Doutor Luís da Costa Lima

ANO 2015

Agradecimentos

A realização deste projeto marca uma das mais importantes etapas da minha vida. Gostaria de agradecer a todos aqueles que deram a sua contribuição para a realização e conclusão deste projeto com sucesso. A todos eles deixo aqui o meu agradecimento sincero.

Quero fazer um agradecimento especial ao Professor Doutor Luís Lima, pela forma como orientou o meu trabalho. As notas dominantes da sua orientação foram a utilidade das suas recomendações e a cordialidade com que sempre me recebeu. Estou grato por ambas e também pela liberdade de ação que me permitiu, que foi decisiva para que este trabalho contribuísse para o meu desenvolvimento pessoal.

Em segundo lugar, agradeço à minha família e amigos que de alguma forma contribuíram para o desenvolvimento e escrita desta dissertação.

Resumo

A prevenção e tratamento da doença e a gestão dos cuidados de saúde em geral têm assumido uma complexidade crescente. A falta de tempo por parte dos prestadores de serviços clínicos para acompanharem todos os desenvolvimentos na medicina, face a um aumento do número de diferentes opções de tratamento e gestão da doença, exige a existência de ferramentas de apoio que permitam a prática de cuidados de saúde suportados na medicina baseada em evidências. Uma das ferramentas mais importantes são os guias clínicos, principalmente quando estão disponíveis para o prestador de serviços em todos os momentos e locais em que deles necessita.

Neste trabalho vamos apresentar o que são guias clínicos, as suas vantagens, origens, objetivos, riscos, benefícios e formas de representação de conhecimento clínico. Iremos também mostrar como podem apoiar a tomada de decisão por parte dos prestadores de serviços. Por último, vamos apresentar uma proposta de representação de guias clínicos e da sua implementação através de um portal Web, de fácil utilização por parte dos prestadores de cuidados de saúde, permitindo a execução de guias clínicos com interligação de registos de saúde eletrónicos. Resultando um protótipo executável, como prova de conceito, que concretiza as propostas apresentadas.

Palavras chave: Guias clínicos interpretáveis por computador, Sistema de apoio a decisão clinica, Guias clínicos, Representação do conhecimento.

Abstract

The prevention and treatment of disease and the management of health care in general have assumed an increasing complexity. The providers of clinical services do not have the time to monitor all developments in medicine, facing an overwhelming increase in the number of different treatment and disease management options, requiring the existence of tools to allow the practice of health care supported on evidence-based medicine. One of the most important tools are clinical guidelines, especially when they are available to the service provider at all times and places in which they are need.

In this work we present clinical guidelines, its advantages, origin, objectives, risks, benefits, and forms of representation of clinical knowledge. We also show how you can support service providers decision-making. Finally, we will present a proposal for representation of clinical guidelines and their implementation through a Web portal, easy to use by health care providers, allowing the execution of Clinical Guidelines with interconnection with electronic health records. This work resulted in an executable prototype, as proof of concept, formalizing the proposals submitted.

Keywords: Computer-interpretable guidleines, Clinical decision suporte system, Clinical guidelines, Knowledge representation.

Índice

Índice de Figuras

Índice de Tabelas

Convenções, termos e abreviações

A correta interpretação deste documento exige o conhecimento de algumas convenções e termos específicos, que são descritos a seguir.

- ACL - Lista de controlo de acesso
- ANAES - *Agence Nationale d'Accréditation et d'Evaluation en Santé*
- API - *Application Programming Interface*
- ASTM - Sociedade Americana de Testes e Materiais
- BPM - *Business Process Management*
- CDSS - *Clinical Decision Support System*
- CIG - *Computer-Interpretable Guideline*
- CPG - Prática do Guia Clínico
- CRUD - *Create, Read, Update e Delete*
- DTD - Definição de tipo de documento
- e-CG - Guia Clínico eletrónico
- ECR - *Randomized Controlled Trials*
- EMR - Electronic Medical Record
- EPR - Registo electrónico do paciente
- EUA - Estados Unidos da América
- GC - Guia Clínico
- GCW - Guia Clínico Web
- GLIF - *Guideline Interchange Format*
- HCE - Histórico Clínico Eletrónico
- HL7 - *Health Level 7*[1]
- HTML - *HyperText Markup Language*
- IM - Informática Médica
- IOM - *Institue of Medicine*
- IA - Inteligência Artificial
- MHB - *Many-Headed Bridge*
- MLM - *Medical Logic Module*
- NAPAQH - Parceria Nacional para a Qualidade na Saúde
- NICE - *National Institute for Health and Care Excellence*
- NOC - Normas de Orientação Clínica

[1] Para saber mais aceder a https://pt.wikipedia.org/wiki/Health_Level_7

- OAW - *OpenArchitectureWare*
- OCL - *Object constraint language*
- ORM - *Object-relational mapping*
- PHP - *Hypertext Preprocessor*
- QoI - Qualidade de informação
- RDF - *Resource Description Framework*
- RIM - Modelo de Referencia de Informação
- RSE - Registo de saúde eletrónico
- SAD - Sistema de apoio à decisão
- TD - Teoria da decisão
- TNM - *Task-Network Model*
- UML - *Unified Modeling Language*
- UMLS - *Unified Medical Language System*
- XML – *eXtensible Markup Language*
- XSL - *eXtensible style sheets*

1. Introdução

1.1 Contextualização

A prevenção e tratamento da doença e a gestão dos cuidados de saúde em geral têm assumido uma complexidade crescente. Por um lado o número de diferentes opções para a gestão de doentes tem aumentado consideravelmente nas duas últimas décadas [1], tornando-se crucial conjugar a explosão de conhecimento e técnicas na área da saúde com a necessidade de uma utilização racional [1] dos recursos existentes.

Durante os anos 1970 a 1980, os investigadores de serviços de saúde descobriram várias evidências de variações substanciais na prática clínica, sugerindo que a medicina era muitas vezes mais arte do que ciência [2].

Há uma tendência crescente, dentro da classe médica, para que as decisões clínicas sejam suportadas na medicina baseada em evidências (provas científicas sólidas) [1]. Esta tendência impulsionou o desenvolvimento de Guias Clínicos (GC) [1], ou seja, documentos de apoio à prática clínica contendo os passos a seguir e questões a ter em conta na gestão da doença de um doente, para evitar práticas e resultados abaixo dos padrões [2]. Inicialmente os GCs eram apenas baseados nas melhores práticas resultantes das recomendações de especialistas médicos. Atualmente, os guias considerados como tendo um grau mais elevado de certeza são os baseados em *Randomized Controlled Trials* (ECR) [3].

Os GCs têm suscitado o interesse da comunidade científica de Inteligência Artificial (IA) no desenvolvimento de ferramentas, sistemas e linguagens específicas para o suporte à sua conceção e implementação prática [2], ou seja na conceção de *Computer-interpretable Guidelines* (CIG).

1.2 Metodologia de Investigação

Este trabalho foi desenvolvido durante o último ano do Mestrado de Engenharia Informática. Para conduzir toda a investigação foi utilizada a metodologia Investigação-Ação (*action research*). Existem várias definições para esta metodologia, uma das definições mais

simples e que retrata perfeitamente o nosso problema associado aos guias clínicos é "estudo de uma situação social realizado pelos envolvidos nessa situação, a fim de melhorar a sua prática e a qualidade da sua compreensão" [4], ou seja, pretendemos ajudar os prestadores de cuidados e os pacientes/doentes a tomarem decisões mais acertadas e de uma forma simplificada (através de um browser Web). O modelo que melhor retrata o processo de pesquisa usado é o modelo de O'Leary's [4] que representa um processo cíclico que ganha forma com o conhecimento obtido gradualmente.

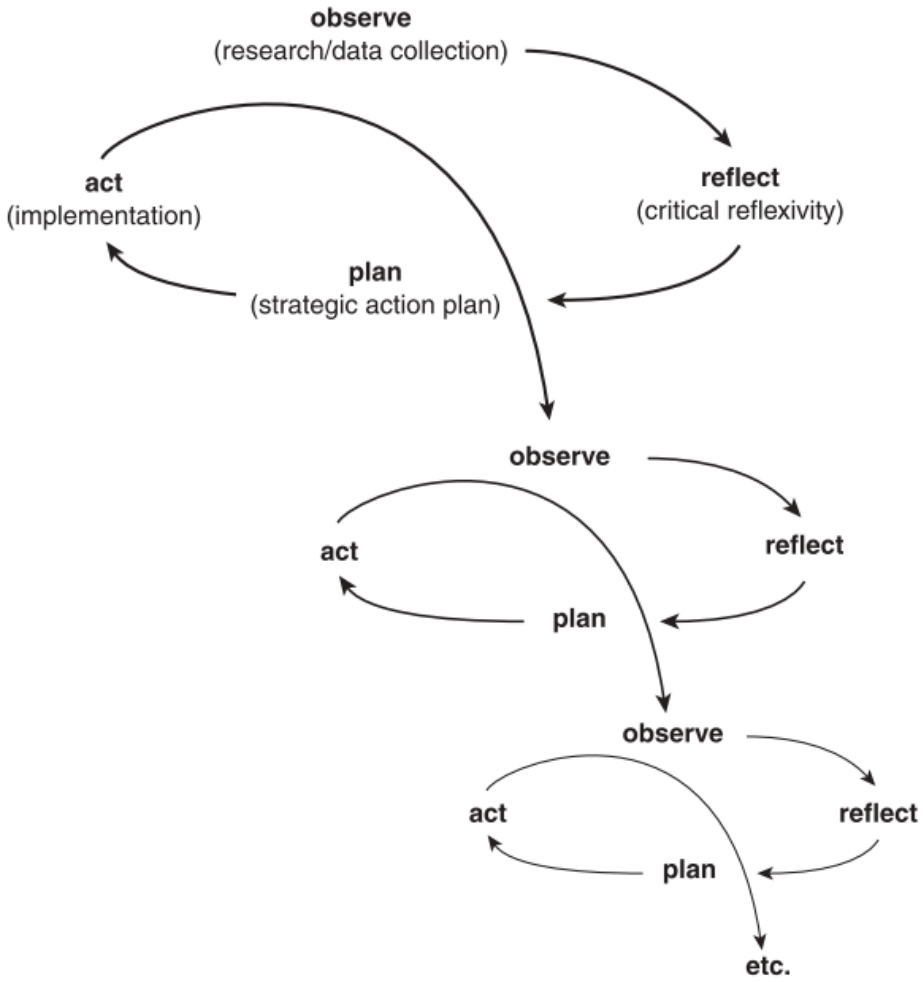

Figura 1 - Ciclo de pesquisa de O'Leary's [4]

Inicialmente foi feito um estudo prévio e uma análise através de algumas pesquisas generalizadas sobre GCs. Estas pesquisas permitiram obter uma compreensão geral sobre o tema e ainda dar origem à escolha e proposta deste tema.

Para dar início ao desenvolvimento deste projeto elaborou-se um plano de desenvolvimento. Plano que começou por uma etapa de investigação mais aprofundada com vários estudos prévios, e uma revisão de documentos resultando na elaboração e apresentação de um

shortpaper, que permitiu obter conhecimento sobre o estado da arte, a definição de guias clínicos, algumas linguagens de representação de GCs e alguns problemas e necessidades de e-CG. Nesta etapa e nas seguintes as pesquisas tiveram como base:

- b-on (Biblioteca do conhecimento online) - http://www.b-on.pt/
- SD (*Science Direct*) - http://www.sciencedirect.com/
- DOAJ (*Directory of open access journals*) - http://doaj.org/
- RCAAP (Repositório científico de acesso aberto de Portugal) - http://projeto.rcaap.pt/

Após a conclusão desta primeira etapa foi feito um estudo mais aprofundado sobre o estado da arte, de forma a obter conhecimento sobre a origem e os objetivos dos GCs, os seus benefícios, os seus riscos e problemas, os problemas de informatização de GCs, a qualidade de um GC, tentativas de soluções para os problemas e os modelos de representação de GCs. Foi ainda feita pesquisa sobre aplicações que permitam/ajudam a implementação de GCs, e ainda projetos que tentaram fazer uma implementação de GCs na Web.

Por fim, a última parte deste projeto consiste no desenvolvimento de uma aplicação Web que permita a representação interna de GCs, a execução de GCs e a interação com um histórico clínico de um paciente.

Em toda a pesquisa efetuada houve uma especial atenção na tentativa de utilização de documentos de revistas científicas indexadas para que as fontes tenham o máximo de credibilidade e confiabilidade.

1.3 Objetivos

Este trabalho tem como principal objetivo a especificação e construção de um sistema de representação e execução de Guias Clínicos Web (GCWs). Define-se GCW como sendo um CIG que pode ser usado no apoio à prática clínica através de um browser Internet.

Os resultados que se pretende obter com este trabalho são:

- Proposta de desenvolvimento do modelo de representação de CGWs;
- Desenvolvimento de um protótipo de criação de GCWs
- Desenvolvimento de mecanismo de execução de GCWs;
- Proposta de interligação com registos eletrónicos de saúde.

Este trabalho tem como propósito ajudar o prestador de cuidados de saúde na consulta de GC e na decisão a tomar. A possibilidade de aceder a um GC através de um browser Web simplifica o acesso à informação médica reduzindo o tempo de consulta e, nomeadamente,

dá a opção ao prestador de cuidados de utilizar um dispositivo móvel, que poderá ajudar à tomada de decisão no local onde é necessária.

1.4 Estrutura do Relatório

Para evitar ambiguidades de interpretação, ao longo deste relatório podem ser encontrados vários termos em inglês. Assume-se a opção de não traduzir estes termos/conceitos para diminuir eventuais ambiguidades.

Este relatório está organizado pelos seguintes capítulos, sendo que a sua divisão e ordenação encontra-se da seguinte forma:

1. Introdução

Neste capítulo, faz-se um enquadramento do trabalho, dando uma descrição geral do problema.

2. Guias Clínicos

Neste capítulo, pretende-se dar a conhecer o que são GCs, apresentando a sua importância, a sua definição e as suas principais características, bem como as suas origens, objetivos, limitações, riscos e benefícios.

3. Guias interpretáveis por computador

Neste capítulo, iremos falar da representação de GCs em formato digital e sistemas de apoio à decisão baseados em GCs apresentando obstáculos para a sua representação.

4. Modelos de Representação de GCs

Neste capítulo, iremos apresentar algumas linguagens de representação de guias clínicos em formato eletrónico como Arden Syntax, Asbru, GLIF3, PRO*forma* e ainda uma comparação.

5. Projetos de investigação Análogos

Neste capitulo pretende-se dar a conhecer alguns dos projetos que servem de apoio ao

desenvolvimento de GCs.

6. Registos de saúde

Este projeto pretende que o sistema de execução de GCs consiga apresentar uma decisão com base também no histórico do registo clínico de um paciente. Para isso é necessário uma simulação/interligação com o Registo médico eletrónico (EMR). Pretende-se assim dar a conhecer o sistema de EMR que iremos usar/simular.

7. Modelo de representação proposto

Neste capítulo, vai ser apresentado o modelo para representação de GCs em formato eletrónico e o portal Web desenvolvido.

8. Conclusões

Esta é última secção que conclui o trabalho. Neste capítulo apresenta-se as conclusões retiradas deste projeto.

9. Trabalho Futuro

Neste capítulo apresenta-se as sugestões para trabalho futuro.

2. Guias Clínicos

2.1 Introdução

Um Guia Clínico (GC) é definido como sendo "*declarações sistematicamente desenvolvidas para auxiliar decisões dos prestadores de cuidados e dos pacientes sobre os cuidados de saúde apropriados para circunstâncias clínicas específicas*"[5]. O desenvolvimento de GC, em geral, tem acelerado significativamente desde meados dos anos oitenta. Associações médicas e agências governamentais nos Estados Unidos, Canadá, Austrália, Grã-Bretanha e Itália, entre outros, têm coordenado esforços para disseminar o conhecimento sobre o desenvolvimento de guias, a sua divulgação e implementação [6] [7].

Os GCs eram inicialmente baseados apenas no consenso de especialistas da área da saúde, mas rapidamente foi reconhecida a necessidade de, na sua conceção, realizar uma análise sistemática da bibliografia existente e incluir a síntese das melhores evidências científicas disponíveis – medicina baseada em evidências. Para além desta mudança de paradigma na criação de guias, o processo de desenvolvimento passou a englobar não só profissionais de saúde, mas também pacientes, adquirindo um carácter multidisciplinar. Atualmente é consensual a necessidade da integração da evidência proporcionada por pesquisas clinicamente relevantes, da experiência do clínico e das preferências do paciente [8]. Por sua vez, constituem-se em posicionamentos ou recomendações, sistematicamente desenvolvidas para orientar médicos e pacientes acerca dos cuidados de saúde apropriados, em circunstâncias clínicas específicas [8].

Os GCs têm sido promovidos num cenário de dados que mostram que há uma lacuna significativa na aplicação da base de evidências para a prática clínica [9] [10]. Estudos nos EUA e na Holanda têm mostrado que cerca de 30-40% dos pacientes não recebem cuidados de acordo com a evidência científica atual e cerca de 20-25% dos cuidados prestados não são necessários ou são até potencialmente prejudiciais [11] [12]. Estudos mais recentes mostram que essas tendências persistem, embora variem consideravelmente segundo a condição clínica e também pela localização geográfica [10].

Os GC suportam os processos de tomada de decisão nos cuidados com o paciente, sendo por isso central a representação do conhecimento. Desde cedo atraíram a atenção da comunidade de Inteligência Artificial (IA), levando ao desenvolvimento de modelos específicos, ferramentas e linguagens para apoiar a sua conceção e implementação, no que pode ser chamado de Guias Interpretáveis por Computador, CIG (*Computer-interpretable Guidelines*) na versão original em inglês [13] [14] [1].

Os GCs são um auxílio à tomada de decisões num sentido ótimo do que é melhor para o doente. O tratamento mais adequado, condições do tratamento, medicação, redução de custos e recursos na prestação de cuidados, oferecendo os meios mais rápidos de proporcionar o uso consciente, explícito e judicioso da melhor evidência atual [15]. GCs são baseados nos resultados de investigação científica, seguida de discussão e expressão da opinião de especialistas, de forma a tornar a prática médica mais eficaz, eficiente e transparente. Os GCs podem ser baseados em consenso, que resulta da concordância de opiniões entre especialistas, ou baseados na evidência em que se trata de um guia desenvolvido após a extração e revisão de informação científica a partir da bibliografia existente e da evidência clínica, fazendo a distinção entre o que é prova científica e o que é opinião. Estes, são mais valiosos para os prestadores de cuidados de saúde e para os pacientes, porque as suas recomendações são baseadas em provas científicas e suportam decisões capazes de produzir melhores resultados.

2.2 Origens

O desenvolvimento de guias, em geral, tem acelerado significativamente desde meados dos anos 1980. As associações médicas e agências governamentais nos Estados Unidos, Canadá, Austrália, Grã-Bretanha e Itália, entre outros, juntaram-se para disseminar o conhecimento sobre o desenvolvimento de guias, a sua difusão, disseminação e implementação [6] [7]. O movimento para desenvolver e divulgar a prática de GCs está enraizado, tanto na necessidade de limitar como em restringir a prática de cuidados de saúde no sistema dos Estados Unidos e está claramente ligado à medicina baseada em evidências [16].

Em reconhecimento do importante papel que a qualidade das iniciativas de cuidados, orientações práticas, especialmente clínicas, pode trazer no sistema de saúde no Canadá, a Associação Médica Canadiana estabeleceu um Comitê de Qualidade de Cuidados em 1990 e posteriormente, facilitou o desenvolvimento da Parceria Nacional para a Qualidade na Saúde (NAPAQH) e duas conferências de consenso nacional sobre guias de prática clínica [16].

2.3 Objetivos

Os objetivos dos guias clínicos são:

1) Melhorar a qualidade de atendimento ao paciente e resultados dos cuidados de saúde [17];
2) Identificar e preencher lacunas no conhecimento e priorizar as atividades de pesquisa [17] [18];
3) Fornecer orientações para os consumidores e informar e capacitar os doentes [17];
4) Informar a política pública [17];
5) Fornecer suporte ao controlo de qualidade, incluindo a auditorias das práticas clínicas ou hospitalares [17];
6) Descrever os cuidados adequados com base nas melhores evidências científicas disponíveis e ampliar o consenso [19];
7) Reduzir a variação inadequada na prática [19];
8) Promover o uso eficiente de recursos [19] [17];
9) Fornecer o foco para uma educação continua [19];
10) Realçar lacunas na literatura existente e sugerir apropriadas pesquisas no futuro [17] [19];

2.4 Riscos e benefícios

O benefício mais fundamental da implementação e utilização de GCs é a melhoria dos resultados de saúde [15]. Assim o objetivo principal dos GCs é, ajudar na tomada de decisão clínica por parte dos médicos que não podem integrar todos os dados publicados sobre as novas tecnologias e conhecimentos na sua prática quotidiana [20].

Em circunstâncias como pré-hospitalar e medicina de emergência, o uso de GCs pode beneficiar os resultados de saúde, com base na sua capacidade para simplificar a tomada de decisões clínicas.

Os GCs oferecem uma solução, tornando mais provável que os pacientes sejam tratados da mesma maneira, independentemente de onde ou por quem eles são tratados, melhorando a qualidade das decisões clínicas. Os GCs podem derrubar as crenças de médicos acostumados a práticas obsoletas, melhoram a consistência do atendimento, e fornecem recomendações oficiais que tranquilizam os profissionais sobre a adequação das suas políticas de tratamento. Orientações baseadas numa avaliação crítica da evidência científica (GCs baseados em evidência) esclarecem quais são as intervenções de benefício

comprovado e documentam a qualidade dos dados de suporte [15], dissuadindo as intervenções que são ineficazes.

Os GCs apoiam as atividades de melhoria de qualidade. Sendo um ponto de referência comum para auditorias prospetivas e retrospetivas dos clínicos ou hospitais, estes fornecem critérios (testes, tratamento e objetivo do tratamento) de avaliação para a classificação das melhores práticas de cuidados [15].

A descoberta de que os "GCs podem ser eficazes na melhoria da eficiência (muitas vezes por meio da padronização de cuidados) e na otimização do valor para o dinheiro" [15], ou seja, alguns GCs reduzem gastos com o internamento, medicamentos, cirurgias e outros procedimentos. A divulgação e a adesão aos guias também pode melhorar a imagem pública, através de mensagens de compromisso com a excelência e a qualidade. Estas mensagens podem promover a boa vontade, o apoio político, e (em alguns sistemas de cuidados de saúde) a receita. Muitos acreditam que o motivo económico por trás dos GCs é a principal razão para a sua popularidade [15].

2.5 Limitações

Apesar da importância atribuída ao GC, a efetiva transferência do seu conteúdo para a prática clínica é uma meta ainda não alcançada [21], algumas dessas barreiras são:

- Textos narrativos que não são usados pelo médico, além de, por razões de tempo e de compreensão, podem não encontrar tudo o que precisam [21];
- A ambiguidade e falta de clareza das recomendações é também um problema generalizado, especialmente nos guias locais [21];
- Dificuldade em fornecer recomendações individualizadas por má interoperabilidade com seu histórico clínico eletrónico (HCE), a falta de detalhes práticos adicionais a serem feitos pelos prestadores de cuidados durante as suas ações de aplicação e por fim, a baixa adesão dos médicos a GCs em papel como GCs computadorizados (CIG) [21].

2.6 Qualidade

O Instituto de Medicina (IOM)[2] dos EUA definiu qualidade como sendo "o grau em que os serviços de saúde para indivíduos e populações aumenta a probabilidade dos resultados de

[2] Para saber mais aceder a http://iom.nationalacademies.org/

saúde desejados e que são consistentes com o conhecimento profissional atual". Boa qualidade significa fornecer aos pacientes serviços adequados de forma tecnicamente competente, com boa comunicação, tomada de decisão compartilhada, e sensibilidade cultural. Em termos práticos, a má qualidade pode significar “muito cuidado” (por exemplo, proporcionando exames desnecessários, medicamentos e procedimentos, com riscos associados e efeitos colaterais), muito *pouco de cuidado* (por exemplo, não fornecer um teste de diagnóstico indicado ou um procedimento cirúrgico de forma a salvar uma vida), ou o *cuidado errado* (por exemplo, a prescrição de medicamentos que não devem ser administrados em conjunto, o uso de técnicas cirúrgicas inadequadas) [11].

2.7 Evidência

A aplicabilidade da evidência na prática clínica é um requerimento para dar o melhor atendimento ao paciente. *Evidence-based medicine* assume muitas formas e é frequentemente vista com algum ceticismo por parte dos clínicos. No entanto, fundamental para a prática baseada em evidências é o conceito de que, embora a evidência possa recomendar estratégias de diagnóstico, tratamentos particulares ou planos de gestão, cada plano deve ser individualizado para refletir as características específicas das circunstâncias clínicas individuais. Para a compreensão, prática e de desenvolvimento de guias baseados em evidências as revisões sistemáticas são fundamentais. Estas, utilizam estratégias de pesquisa bem definidas e reprodutíveis para identificar evidências de que informam problemas clínicos; os dados são então avaliados pelo seu rigor metodológico e, quando tem qualidade e quantidade suficiente, pode ser confinado matematicamente usando técnicas de meta-análise, que fornecem estimativas mais precisas dos efeitos de como é incorporado um grande número de pacientes do que os estudos de origem [3].

Os guias baseados em evidências podem melhorar a qualidade dos cuidados dos pacientes através do suporte de intervenções de benefício comprovado, enquanto desencoraja intervenções desanimadoras que são ineficazes ou potencialmente nocivas. No entanto, os guias também podem ser enganosos e causar danos, especialmente se as recomendações são baseadas num conjunto de dados incompletos e/ou com falhas/imperfeitos, se eles são tendenciosos, ou se o processo de desenvolvimento é realizado de forma incorreta. Guias baseados em evidências são rigorosamente desenvolvidos e visam minimizar esses potenciais efeitos nocivos [3].

Guias baseados em evidências usam os princípios da medicina baseada em evidências: a realização de uma pesquisa abrangente sobre a literatura, avalia criticamente a qualidade

da evidência e gera recomendações enquanto considera as preferências e valores dos pacientes [3].

O processo de aplicar os princípios da evidência baseada em medicina ao processo de desenvolvimento do guia é constituído por várias etapas:

1. Definição da questão clínica que o guia irá abordar;
2. Definição dos critérios de elegibilidade para os estudos que serão incluídos nas recomendações;
3. Realização e avaliação de uma busca sistemática da literatura e da evidência.

A evidência é considerada de alta qualidade se o estudo minimiza potenciais distorções. Os desenhos de estudo podem ser considerados numa hierarquia, com revisões rigorosas realizadas sistematicamente e revisões da meta-análises de ensaios clínicos randomizados, são considerados evidências da "mais alta qualidade".

Um estudo observacional demonstrando um efeito de tratamento muito forte pode proporcionar uma melhor evidência do que um ensaio clínico randomizado (ECR) de fraca potência ou mal realizado em que não foi conseguido encontrar um efeito significativo do tratamento [3].

3. Guias interpretáveis por computador

3.1 Introdução

Os GCs tradicionais são baseados em texto e, normalmente, incluem critérios que descrevem a sua aplicabilidade a grupos específicos de pacientes, os processos recomendados de tratamento e uso adequado de materiais e procedimentos, oferecendo informações adicionais, tais como meios complementares de diagnóstico sugerido [22]. A maior parte são inacessíveis para os médicos que mais precisam deles. Além do problema da acessibilidade costumam ter várias páginas de texto e a dificuldade da sua leitura é acrescida.

Podemos encontrar diversos GCs no formato de texto em sites especializados, como por exemplo o Guia-Saúde (http://portal.guiasalud.es), o *National Guidelines Clearinghouse* (NGC, http://guideline.gov), *National Institute for Health and Care Excellence* (NICE, http://www.nice.org.uk/guidance/published?type=cg), *Guidelines International NetWork* (GIN, http://www.g-i-n.net/library/international-guidelines-library), em Portugal na Direção-Geral de Saúde (https://www.dgs.pt/directrizes-da-dgs.aspx) e na *British Columbia* (http://www2.gov.bc.ca/gov/content/health/practitioner-professional-resources/bc-guidelines).

Mesmo quando existem em formato eletrónico, e mesmo quando esse formato é acessível *online*, os médicos raramente têm tempo e meios para decidir qual dos vários GCs vai utilizar e qual aplicar ao paciente. Além disso, desenvolvimentos na área da saúde organizacional e profissional, muitas vezes implicam uma redução na acessibilidade ao GC, criando uma sobrecarga de informações significativas sobre os profissionais de saúde. Esses profissionais precisam processar mais dados do que nunca, continuamente e em curtos períodos de tempo. Considerações semelhantes aplicam-se à tarefa de avaliar a qualidade do GC. Para atender a todas estas necessidades dos profissionais de saúde, bem como administradores, e garantir a qualidade contínua do atendimento, são necessárias ferramentas de processamento de informações mais sofisticadas. Limitações que ocorrem devido ao estado da arte, análise de dados não estruturados e baseados em texto. Assim,

há uma necessidade urgente de facilitar a divulgação de guias e aplicações que usam representações legíveis por máquina e métodos computacionais automatizados [20].

O processo de informatização de GC é difícil. O ciclo de vida de um GC eletrónico inclui várias etapas [14]:

- desenvolvimento do guia baseado em papel;
- especificação do conteúdo de conhecimento;
- implementação do guia computadorizado;
- testes e validação;
- implantação e avaliação da eficácia do guia informatizado.

Um GC bem desenvolvido melhora a qualidade nos cuidados de saúde por reduzir a variabilidade e melhora a correção dos diagnósticos e terapias, enquanto desencoraja procedimentos cirúrgicos ineficazes e perigosos [23]

O uso de sistemas de apoio à decisão computadorizados pode agregar valor aos documentos de texto, através de uma oferta ativa, oferecendo na tomada de decisão provas justificadas para pacientes específicos [24], permitindo reduzir o tempo de consulta por parte dos prestadores de cuidados.

Sob a perspetiva da IA há uma alternativa [22] para formalizar GCs em texto num formato legível por computador que pode usado para apoiar os clínicos no seu trabalho diário. O resultado deste processo de transformação pode ser expresso em diferentes linguagens formais, como Asbru, EON, GLIF, PRO*forma*, SDA e GLARE. A adoção de um formalismo padrão melhora a interoperabilidade entre as diferentes ferramentas e sistemas, no entanto, nenhum deles foi amplamente adotado pelos investigadores. Embora cada linguagem seja projetada com diferentes particularidades (por exemplo, acionamento de eventos no caso de PRO*forma*, a inclusão de papéis em SAGE, ou o uso de funções de sincronização em GLIF3), a maioria delas partilham um conjunto de funções básicas [22].

A sustentabilidade e adaptabilidade às mudanças da tecnologia é uma característica fundamental para a geração de protótipos sucessivos e de versões finais do GC. A evolução contínua tanto da tecnologia como dos GCs não pode ser um obstáculo para uma geração de versões ágeis de CIGs. Para isso é necessário uma plataforma tecnológica para apoiar as diversas atividades durante todo o ciclo de vida da informatização do CIG para capturar as mudanças em que eles ocorrem e gerar automaticamente um novo CIG mantendo a compatibilidade com o CIG desenvolvido nas versões anteriores sobre a tecnologia desenvolvida ou adaptada por outros.

A disponibilidade de um sistema Web facilita o acesso, intercomunicação com especialistas, administração e ainda permite gravar todas as informações produzidas durante a aplicação da CIG a um paciente. Um exemplo é o *e-GuidesMed* acessível em http://www.e-guidesmed.ehu.es [21].

Várias das principais tarefas envolvidas nos GCs, que beneficiariam de suporte automatizado, incluem a especificação (autoria) e manutenção do GC, a recuperação de guias apropriados para cada paciente, a execução de aplicações de guias e avaliação retrospetiva da qualidade de aplicações de guias. O suporte a cuidados de saúde baseados em GCs implica um balanceamento entre o papel do prestador dos cuidados e um sistema automatizado, cada um com os seus pontos fortes. Por exemplo, os médicos têm melhor acesso a certos tipos de informações específicas do paciente clínico (como seu odor, a aparência da pele e estado mental), da medicina geral e do conhecimento do senso comum. Os sistemas automatizados têm melhor acesso e mais preciso às especificações dos guias e detetam mais facilmente padrões temporais pré-especificados nos dados do paciente [20].

3.2 Obstáculos

A conversão dos GCs em CIGs não é tarefa fácil. Existem vários obstáculos que necessitam de ser resolvidos:

- a criação de qualidade GC utilizando metodologias definidas;
- desenvolvimento e conversão sistemática do abundante conhecimento existente num CIG;
- manutenção e gestão do conhecimento e da infraestrutura [23];
- interpretação do conteúdo de um GC:
    - significado exato dos termos nem sempre está definido [25];
    - as recomendações nem sempre são claras [25];
    - uso de palavras vagas [25],
- como adquirir, verificar, localizar, executar (decisões a serem tomadas, restrições entre tarefas, restrições temporais) e avaliar guias formais e sistemas de suporte na prática diária [25];
- a integração desses conhecimentos num sistema de apoio à decisão (SAD) [23];
- como interligar o CDSS baseado em GCs com um sistema de informação do paciente [25] (EMR);
- como fornecer apoio à decisão a um prestador de cuidados na prática diária [25];
- como representar e partilhar GCs usando uma representação formal e ambígua [25].

3.3 Sistemas de apoio à decisão clínica

Os sistemas de apoio à decisão clínica (CDSS) pretendem proporcionar o "*conhecimento certo às pessoas certas, na forma certa, no momento certo*" [24]. Para isso é necessário fazer as "melhores" decisões possíveis. Em termos gerais, a Teoria da Decisão (TD) é um meio para analisar qual o conjunto de alternativas que deve ser escolhido quando há incerteza sobre os resultados, a fim de fazer uma opção. TD centra a sua atenção na identificação da "melhor" escolha. A noção de "melhor" tem significados diferentes, sendo o mais comum a que maximiza a utilidade esperada para o decisor [13].

Durante as duas últimas décadas, um grande número de estudos abordaram os efeitos de CDSSs na prática clínica [26]. A implementação de CDSS com base em GCs promete a melhor aceitação e aplicação de GCs na prática diária, porque estes sistemas são capazes de monitorizar as ações e observações dos prestadores de cuidados e fornecer acessória no ponto de cuidado (*care*) com base no GC. Afirma-se que os sistemas de apoio à decisão (baseados em guias) são de fato necessários para no futuro tomar decisões médicas em geral [25].

É amplamente aceite que a adoção de mecanismos de execução de guias na prática diária irá melhorar o atendimento ao paciente, através da uniformização dos procedimentos de cuidados. Um guia armazena conhecimento médico (declarativo) sobre procedimentos médicos. É importante o uso de um vocabulário comum e a adoção de uma das terminologias disponíveis para permitir a reutilização, aprender e compartilhar esse conhecimento modelado [24].

Um CDSS, tenta fornecer os "melhores" conselhos aos profissionais de saúde para prestarem cuidados com base nas recomendações dos guias. As revisões sistemáticas descobriram que CDSSs ativos, que fornecem conselhos não solicitados durante procedimentos de rotina de trabalho ao utilizador final, são significativamente mais eficazes na melhoria do desempenho profissional do que os sistemas passivos, que exigem que os usuários finais reconheçam que a consulta do CDSS seria útil [26].

A prestação do apoio à decisão no momento e no local de tomada de decisão e a integração do CDSS no fluxo de trabalho dos médicos, foram fatores que contribuíram para o sucesso de CDSSs [26]. Estes sistemas devem possibilitar iniciar, interromper e continuar a consulta ao GC, em qualquer momento, e por diferentes utilizadores. Isto exige que o estado do processo de avaliação das necessidades de pacientes individuais possa ser reconstruído pelo sistema. CDSS de consulta multidisciplinar também requer que os dados sejam armazenados numa de base de dados central que seja acessível a vários sistemas. No

entanto, a consulta CDSS simultânea para o mesmo paciente nunca deve levar a recomendações inconsistentes ou inconsistências da base de dados [26].

Para que seja possível reconstruir a lógica de uma recomendação, a reexecução do guia através do CDSS é necessário ter os dados do paciente disponíveis na base de dados. No entanto, isto pode levar a inconsistências se os dados clínicos subjacentes da recomendação mudaram [26]. Em muitos domínios clínicos, é um fenómeno natural que os dados clínicos mudam ao longo do tempo, como a maioria das observações clínicas e medições são inerentemente temporais. Para evitar essas inconsistências, é necessário armazenar algumas características das recomendações. Uma possibilidade é armazenar uma recomendação, juntamente com o seu tempo de geração na base de dados e número da versão do guia, que torna possível usar os dados clínicos originais, subjacente à recomendação, para reexecutar o guia e reconstruir o caminho das etapas do guia. Esta solução exige que sejam armazenados poucos dados adicionais, mas implica, e coloca uma carga adicional no mecanismo de execução do CDSS. Um segundo método possível para proporcionar uma recomendação de lógica é simplesmente armazenar a recomendação e todas as etapas do guia que levaram a ela. Esta solução não requer qualquer envolvimento do mecanismo de execução, e, portanto, não é afetada pelos dados clínicos dinâmicos. No entanto, esta solução também necessita que a informação seja armazenada na base de dados [26].

O problema de mudanças no conteúdo (como as evidências de estudos clínicos que se vão acumulando com o passar do tempo) do guia, os guias muitas vezes precisam de revisão, para garantir que ainda refletem a opinião clínica predominante. Quando isso acontece, também os CDSS baseada em GC também precisam de ser revistos. Lidar com este problema também requer que as informações relativas às recomendações sejam armazenadas [26].

Quando se armazena todas as recomendações, inclusive as suas etapas de orientação subjacentes numa base de dados, a lista de medidas de orientação subjacentes a uma recomendação específica pode ser sempre reconstruída, independentemente do guia formalizado que foi atualizado. Atualizando o conteúdo do guia no CDSS não afeta a informações das recomendações geradas previamente armazenados na base de dados. Mais uma vez, esta solução exige que mais informações sejam armazenadas, mas não requer qualquer envolvimento do mecanismo de execução CDSS para reconstruir a lógica da recomendação [26].

4. Modelos de Representação de GCs

4.1 Introdução

Uma das questões críticas para a implementação de um GC baseado em computador é o modelo de representação. Durante toda a pesquisa foram encontrados vários modelos de representação, sendo eles: Arden Syntax [25], GLIF/GLIF3 [25], Asbru [25], PRO*forma* [25], T-HELPER [27], EON/DHARMA [25], GUIDE/PatMan [28] [29], Prodigy [28], DILEMMA [30], PRESTIGE [31], ONCOCIN [20], GEM [20], HELEN [32], HGML [33], Stepper [34]. Além dos modelos de representação foram encontradas *frameworks* e ferramentas de ajuda para o desenvolvimento, implementação e execução de GC, como: o DeGel [24], SEBASTIAN [35], HeCaSe2 [22] [24], Arezzo [36], GLARE [24], GLEE [37] [24], SAGE [24], NewGuide [23] [24], SPEM [23] [24], GASTON [26], Asgaard [27], GEM [20].

Dos modelos de representação mencionados apenas vão ser abordados: Arden Syntax, Asbru, GLIF e PRO*forma*. A escolha destes quatro modelos tem como base os seguintes critérios: a diferença de representação, a quantidade e a qualidade de referências/informação disponível. Além destes critérios, a **Error! Reference source not found.** ajudou a definir a escolha, através dela verificamos que Arden Syntax teve como base o primeiro modelo de representação e que desde a sua origem até ao presente ainda tem atualizações de software e suporte, GLIF3 e PRO*forma* que surgiram de uma evolução continua, e Asbru que desde a sua criação tem-se mantido original e por ter contribuído para a criação de novos modelos.

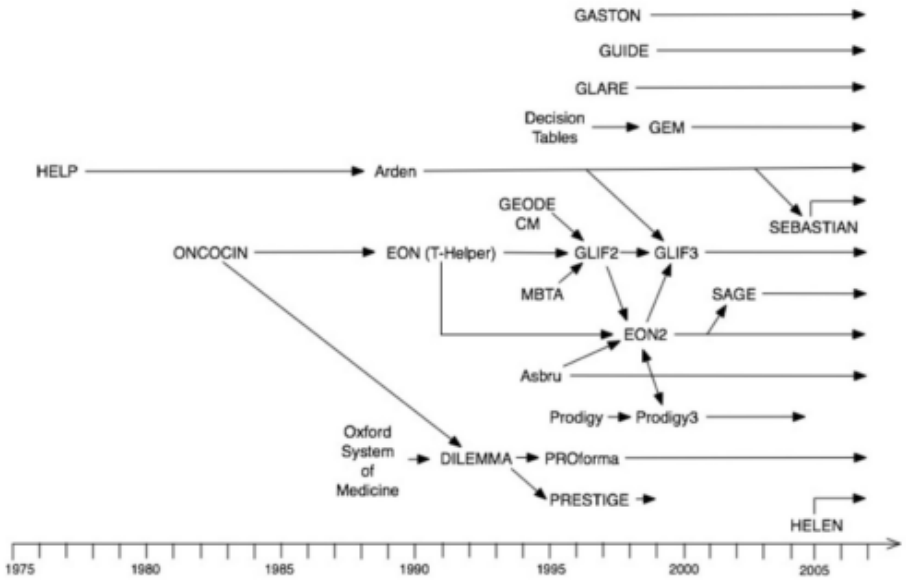

Figura 2 - História da criação de modelos de representação até 2007 [25]

4.2 Arden Syntax

4.2.1 Introdução

Foi criada em 1989 na conferência Arden Homestead em Harriman, Nova York [25], como resposta à incapacidade de partilha de conhecimento médico entre o pessoal médico e sistemas de informação entre diferentes instituições [25]. Define uma representação para guias modulares: *Medical Logic Modules* (MLM, deve conter lógica suficiente para fazer uma única decisão médica [38]), que representam decisões clínicas individuais [39], centrando-se na partilha de guias modulares independentes [25], ou seja foi projetada para apoiar a tomada de decisão clínica particular. O utilizador final da Arden Syntax é o clínico e os MLMs são destinados a ser escritos e usados por médicos com pouca ou nenhuma formação em programação [38].

A Arden Syntax foi aceite em 1992 como um padrão, pela Sociedade Americana de Testes e Materiais (ASTM) e transposta para o HL7 em 1997/1998. Este padrão encontra-se na versão 2.9 e está disponível na área de livraria do site HL7 (www.HL7.org) [38].

A Arden Syntax traz suporte especial para funções temporais, garantindo que cada elemento e cada evento tem uma data/hora associada que é clinicamente significativo. Muitas funções de tempo são fornecidas para ajudar os utilizadores de destino a especificar a data/hora em MLM. Com qualquer outra língua, essas definições seriam mais

dependentes da pessoa responsável pela implementação do MLM. A Arden Syntax permite que esta definição seja explícita [38].

4.2.2 Modelação

Na Arden Syntax, os guias são modelados como (uma coleção de) modelos médicos lógicos (MLM). Um MLM (ficheiro ASCII) é um híbrido entre uma regra de produção (uma regra "if-then") e um formalismo processual [38]. Cada MLM representa uma decisão única e contém *slots* que são agrupadas em três categorias: *Library*, *Maintenance* e *Knowledge*. As categorias de *Maintenance* e *Library* descrevem a pragmática do MLM (por exemplo: título, versão, explicação e palavras-chave) e a categoria de *Knowledge* descreve a lógica de um MLM [25].

Como os MLMs são para serem partilhados entre várias instituições, as categorias de *Maintenance* e *Library* contêm a documentação necessária para cada MLM [25], oferecendo informações explicativas e links para facilitar a pesquisa por meio de uma base de conhecimento, permitindo a sua manutenção [39]. As *slots* da categoria de *Maintenance* incluem os atributos nome do MLM (ficheiro), autor, versão, instituição, especialista, data da última modificação e status de validação (por exemplo, *testing*, *research*, *production* ou expired).

As *slots* na categoria *Library* são usadas para documentação e consistem no objetivo do MLM, uma explicação mais detalhada (que pode, por exemplo, ser mostrada para os utilizadores quando eles recebem mensagens de MLM geradas) e uma série de palavras-chave (por exemplo, usadas para categorizar MLMs) [25].

O conhecimento médico é armazenado na categoria *Knowledge* [25]. Esta categoria contém as *slots* que especificam o que o MLM faz [39], sendo composta por cinco *slots* obrigatórias (*type*, *data*, *evoke*, *logic* e *action*) e duas *slots* opcionais (*priority* e *urgency*) [25].

A ***slot data*** define os termos usados no MLM [39].

A ***slot type*** é utilizada para obter os valores dos conceitos mencionados no MLM do sistema de informação da clínica local, tais como o registo eletrónico do paciente (EPR). Por

exemplo, a linha '*hematocrit := read last {'hematocrit'};*' indica que o valor do conceito '*hematocrit*' corresponde ao último valor *hematocrit*, por exemplo num EPR [25].

A ***slot evoke***, especifica o contexto no qual um MLM deve ser executado. Os MLMs podem ser executados como um resultado de três tipos diferentes de eventos: operações de base de dados, eventos temporais e notificações externas [25].

A ***slot logic*** contém os critérios de decisão que podem levar a uma determinada ação. Essas expressões lógicas são implementadas como regras de produção e contêm conceitos que são definidos na *slot data* (por exemplo, '*Hematocrit*'). A Arden Syntax suporta vários tipos de operadores (lógicos, de lista, temporais e de agregação). Os operadores booleanos usam uma lógica três valores, em que o valor de "*null*" é considerado como desconhecido. Quando a condição é avaliada "*false*" ou "*null*", a execução do MLM termina. Quando uma expressão lógica é avaliada como "*true*", a ***slot action*** é executada [25] [39], realizando as ações que estão especificados nessa slot. As ações típicas incluem o envio de uma mensagem para um prestador de cuidados de saúde, adicionando uma interpretação para o registo do paciente, retornando um resultado para chamar um MLM, e evocando outros MLMs [25].

4.2.3 Portabilidade

Arden Syntax faz com que o conhecimento seja portátil, mas os MLMs desenvolvidos para um ambiente não são facilmente incorporados dentro de outro ambiente [38]. Os esquemas da base de dados, vocabulário (terminologia), e métodos de acesso variam muito de modo que qualquer codificação de conhecimento clínico (tal como um MLM), deve ser adaptada à instituição local a fim de utilizar o repositório local clínico [39]. Isto dificulta a partilha de conhecimento. As adaptações (referências isoladas) ao ambiente de dados local são feitas através de chaves ["{}"], num MLM, muitas vezes referido como o "*curly braces problem*" [38].

Como resultado, os compiladores da Arden Syntax diferem de arquitetura e do sistema de informação específico de cada hospital [39]. Para reduzir/ultrapassar este problema a versão 3.0 da Arden Syntax que está em desenvolvimento pela HL7, está prevista para ser baseada em XML [39]. Desta forma através do uso de XML não á necessidade da criação de um compilador que muda de instituição para instituição [39]. Assim o uso de XML é uma solução para o problema de compilação de Arden Syntax.

4.3 Asbru

4.3.1 Introdução

Asbru foi desenvolvido de forma colaborativa na Ben Gurion University e na Universidade de Tecnologia de Vienna [28].

Esta abordagem visa a representação de GCs como sendo *skeletal-plans*, que são esquemas constituídos por vários níveis de detalhe, incorporando restrições temporais *time-oriented* e ainda intenções explícitas *intention-based* que são condições e/ou resultados a garantir durante o tratamento.

Os *skeletal-plan* fornecem uma forma de reutilizar o conhecimento processual específico de um domínio, deixando espaço para a flexibilidade de tempo de execução para alcançar objetivos particulares. A representação de *skeletal-plan* é similar à noção de planos em PRO*forma*.

Esta linguagem permite flexibilidade durante o tratamento de forma a poderem ser executados planos alternativos conforme a evolução observada dos resultados [28] [40].

4.3.2 Modelação

Em Asbru o elemento *plan* é o mais importante. Além de atributos administrativos (como por exemplo, o título do *plan*), cada *plan* contém os seguintes atributos (chamados de papéis de conhecimento em Asbru), que descrevem a sua funcionalidade: *preferences, intentions, conditions, effects e plan body* [25].

As ***preferences*** restringem a aplicabilidade de um plano (exemplo: seleção de critérios: exatos ou ajustáveis) e expressa um tipo de comportamento de um plano (exemplo: tipo de estratégia: agressivo ou normal) de forma a atingir um determinado objetivo [30].

As ***intentions*** são padrões temporais que são usadas para modelar os objetivos do plano, por exemplo: o processo *intention* de administrar insulina regular, duas vezes por dia; uma *intention* resultado para manter a glicemia de jejum num determinado intervalo de pelo menos 5 dias por semana, que são atribuídos pesos individuais, significando a sua importância relativa. Tal conhecimento é necessário para determinar, por exemplo, se um prestador de cuidados ainda está a seguir a maior parte do guia, ou, pelo menos, o seu espírito. Esse provedor pode aplicar o guia de forma modificada, como é de fato o que acontece em 50% dos casos que foram inspecionados [40].

As ***conditions*** tal como as *intentions* são também padrões temporais e são usadas para mudar o estado de um plano durante o seu tempo de execução Esses estados podem ser *started*, *suspended*, *reactivated*, *aborted* e *completed*. Estes estados estão ainda dependentes de pre-condições, que são:

- *filter-preconditions* e *setup-preconditions* - condições que necessitam de ser válidas caso um plano seja considerado aplicável;
- *suspend-conditions* - que determinam quando um plano ativo deve ser (temporariamente) suspenso;
- *abort-conditions* - determinam quando um plano está ativo ou em suspensão tem que ser abortado;
- *completed-conditions* - determinam quando um plano é (com sucesso ou não) concluído [25] [30].

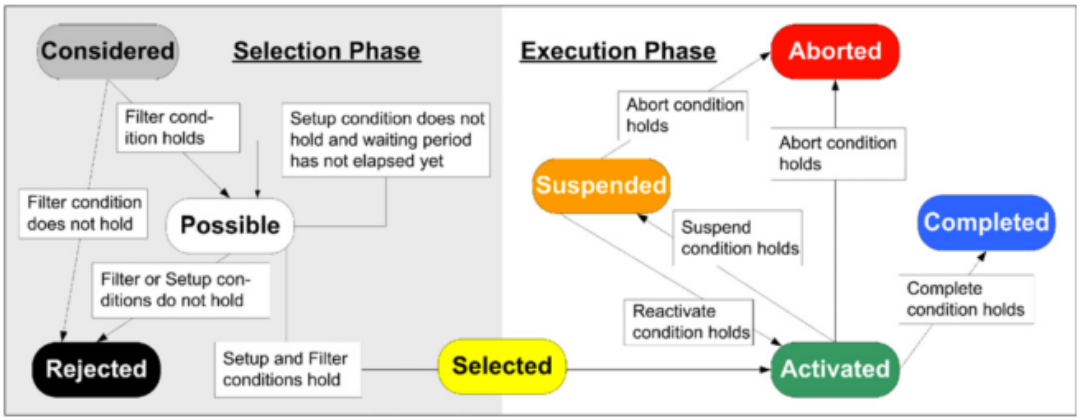

Figura 3 - O modelo de transição de estado de um plano de Asbru [41]

Os ***effects*** descrevem a relação entre os argumentos dos planos e parâmetros mensuráveis através de funções matemáticas, e ainda fornecem a probabilidade de ocorrência.

O ***plan body*** é um conjunto de *subplans* (*composite plans* [24]) e/ou *actions* 'atomic plans [24], são planos que não contêm sub-planos) que têm de ser executados sempre que o plano é considerado adequado (com base em pressupostos, intenções ou efeitos). Um plano é decomposto em sub-planos até não ser possível a sua decomposição. Todos os sub-planos têm os mesmo componentes que o plano (*preferences, intentions, conditions, effects e plan body*) [42]. Os *subplans* podem ser executados em sequência (sequentially), em paralelo (todos ou apenas alguns, *parallel*), em qualquer ordem (*unordered*) ou periodicamente (any-order) [25].

4.3.3 Linguagem

O Asbru TNM é definido como uma definição de tipo de documento (DTD)[3] que define a estrutura dos diferentes elementos do *Task-Network Model* (TNM). A linguagem de controlo de fluxo e a linguagem de expressão são definidos formalmente por meio de XML, com base no DTD [25].

Um aspeto importante da linguagem de expressão é o conceito de anotações de tempo, que são usados na complexa especificação de padrões e restrições temporais em que uma ação deve ocorrer, ou que uma condição deve ser cumprida.

Outras línguas, relacionadas com Asbru como MHB (Many-Headed Bridge) estão em desenvolvimento com o objetivo de fazer a ligação entre orientações informais (por exemplo, textual) e formais [25].

4.4 Guideline Interchange Format (GLIF)

4.4.1 Introdução

A GLIF foi criada pelo laboratório InterMed, num projeto conjunto de grupos de Informática Biomédica nas Universidades de Harvard, Colômbia e Stanford, para servir como um formato de representação comum para GCs. O objetivo do GLIF é facilitar a partilha de GCS entre diferentes instituições [25]. A versão atual do GLIF é GLIF3 (versão 3), e é uma atualização da versão anterior (GLIF2) que tinha várias limitações como: a especificação incompleta em relação à interpretação e execução de guias por computador, abordagem (*adhoc*) para a definição de dados do paciente e ações clínicas, falta de uma especificação para as expressões lógicas e um conjunto limitado de modelos de decisão [43] [30]. GLIF3 responde às limitações de GLIF2, e às novas exigências para a modelagem de orientações, tais como a representação do estado clínico do paciente) [43] [30].

GLIF3 permite a codificação de um GC em três níveis: (1) num fluxograma conceptual (ajuda os autores e utilizadores a visualizar e entender um GC), (2)

especificação computável que pode ser verificada para a consistência, e (3) integridade lógica (entre o nível do fluxograma e o nível de execução, define formalmente lógica, critérios, definições de itens de dados do paciente, ações clínicas e do fluxo das orientações) [43] [30];

[3]A **definição de tipo de documento**, ou simplesmente **DTD**, contém as regras que definem quais as *tags* que podem ser usadas num documento XML e quais os valores válidos.

GLIF3 é uma abordagem baseada em tarefas (*task-based*) e segue o modelo TNM. De forma a facilitar o mapeamento dos dados dos pacientes na EMR que são necessários para a execução de GCs e a uniformizar os conceitos médicos, GLIF3 possui um modelo de dados estruturado, que é baseado no padrão Modelo de Referência de Informação (RIM) de Saúde Nível 7 (HL7) ou no modelo *Unified Medical Language System* (UMLS) [44].

4.4.2 Modelação

O modelo GLIF é orientado a objetos (*class instances*) extensíveis com uma sintaxe estruturada baseada na *Resource Description Framework* (RDF) [44], e consiste num conjunto de classes que descrevem as características típicas de um guia (por exemplo: *decisions* e *actions*), atributos das classes e os tipos de dados para os valores dos atributos. Em GLIF3, todas as classes, atributos e as relações são descritas através de *Unified Modeling Language (UML)*, o objeto *guideline* encapsula um *sub-guideline*. Este objeto contém uma série de atributos que são de natureza administrativa (por exemplo: nome e autor), mas também atributos que descrevem as capacidades de um guia (por exemplo: a intenção do guia e critérios de elegibilidade).

Um guia GLIF consiste numa coleção de *steps* que são ligados num grafo direcionado (fluxograma) [30]. GLIF consiste em cinco etapas (*decision step*, *action step, patiente state step, branch step, synchronization step*) [30], em que estas permitem especificar (programaticamente) e coordenar a execução da aplicação, e gravar as ações computacionais ou clínicas (ou seja, permite ter um histórico da execução).

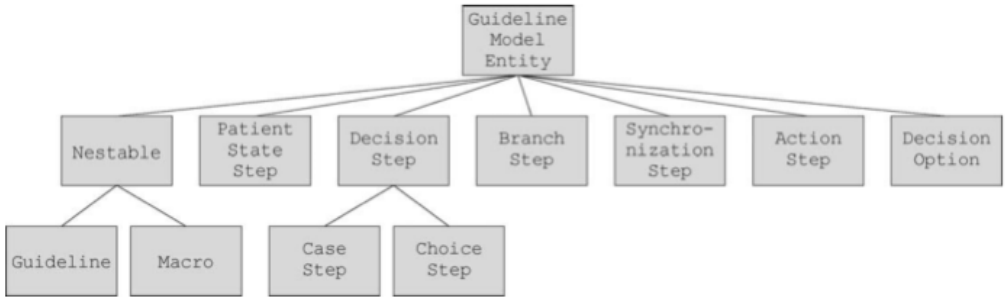

Figura 4 - Classes principais de GLIF

As ***decision steps*** são usadas para codificar os pontos de decisão [37]. Um ponto de decisão contém um conjunto de decisões lógicas. Estas decisões são especificadas na classe *decision*_step. Um atributo booleano chamado *automatic_decision* estabelece uma distinção entre as decisões que são executadas automaticamente (*case step*) e aquelas que têm de ser aprovadas por um agente externo (*choice step*), como um médico, um prestador

de cuidados de saúde, ou outro *software*. [44] [30]. Um exemplo de um *case step* é uma decisão que é seguida por uma ação orientada para a programação como para a recuperação de dados ou invocação de um *subguideline*, a decisão pode ser executada sem sobrecarregar o utilizador para aprovação. Um exemplo de uma *choice step* é quando as ações após a etapa de decisão pode envolver riscos significativos, outras considerações clínicas que possam mitigar a decisão, ou os critérios de decisão são ambíguos, é essencial para obter a escolha da decisão aprovada por um agente externo [44].

Uma tarefa de decisão contém um certo número de expressões lógicas [30], e uma opção de decisão que é uma combinação de uma referência (como um apontador) para a próxima etapa (destino/direção do fluxograma) e um critério de seleção. Durante a execução, os critérios de decisão de todas as opções são avaliados [44]. Os critérios de decisão são modelados usando uma *based-OCL*[4] *language (Object constraint language)* chamado GELLO [24].

Os ***patient state steps*** são etapas usadas para especificar os estados de gestão ou fisiopatológicos de um paciente nos contextos específicos da aplicação [37]. Um *patient state step* serve como um rótulo que descreve o estado atual em que o doente se encontra depois de ter realizado as etapas anteriores, também pode ser usado como um ponto de entrada no guia, dependendo do estado corrente do paciente (por exemplo, o paciente volta para a família com a pressão arterial elevada). O estado de cada paciente contém atributos que descrevem o estado do paciente (por exemplo, a pressão arterial é superior 140/90 durante a última semana). Sempre que este estado ocorre na prática, o guia que contém a correspondente *Patient state step* é executado [30].

A ***branch step*** com a ***Synchronization step****,* em conjunto são usadas para modelar os vários caminhos em simultâneo ao longo do GC. Através da programação e coordenação de tarefas simultâneas (em paralelo) ou tarefas com execução arbitrária [37]. Várias etapas de guias que seguem a *Branch step* convergem para a *Synchronization step*. Quando uma *branch* que começou num passo de ramificação anterior atinge o correspondente passo de *synchronization*, um atributo *continuation* especifica se todos, alguns, ou um dos anteriores passos foram completados antes que o fluxo de controlo se mova para a etapa seguinte [25]. O atributo *continuation* é uma expressão lógica [30], por exemplo: *Step_A* ou *Step_B*, indica que o fluxo deve continuar uma vez que um dos *steps* esteja completo [44].

[4] Para saber mais sobre OCL pode aceder em https://pt.wikipedia.org/wiki/OCL

As ***actions steps*** são as ações do modelo de tarefas que são (ou devem ser) realizadas. Existem 3 tipos de ações definidas:

- *medically oriented* - são recomendações para um tratamento específico;
- *programming-oriented* - é a recuperação de dados a partir do registo eletrónico do paciente ou o envio de uma mensagem para um prestador de cuidados;
- *control-oriented* - invocam estruturas aninhadas como (sub)GCs ou macros para dar suporte à especificação recursiva [30].

As ***macro steps*** fornecem um meio para especificar declarativamente um padrão processual que aparece nos guias, como uma única construção que é realizada por um conjunto de *steps* GLIF3 [28]. Internamente, a macro consiste em duas etapas: uma *decision step* e uma *action step* [30]. Permitindo a reutilização de guias que são usados com frequência.[30]

4.4.3 Ferramentas

Para o desenvolvimento do modelo GLIF em termos de classes, atributos, criação das relações entre os elementos fundamentais, os conceitos da RIM e ainda da camada de conhecimento médico são utilizadas duas ferramentas: Protégé[5] e GEODE[6] [30].

GLEE é um mecanismo de execução de guias codificados no formato GLIF3 e capaz de ser integrado com sistemas de informação clínica em instituições locais [43]. GLEE tem 3 níveis de abstração: *data* (contém a EMR com um repositório de GCs e monitorização de eventos clínicos), *business logic* (motor de execução formado pelo servidor e clientes. O servidor interage com o nível *data* e os clientes com os utilizadores (através de interfaces)), e *o user interface* (onde estão localizadas as aplicações clinicas que trocam dados com níveis superiores) [24]. Além de GLEE existe o GESDOR (*Guideline Execution by Semantic Decomposition of Representation*) que é uma melhoria de GLEE e que permite a representação de GCs independente da linguagem de representação [48,49].

[5] É um editor e uma framework, gratuito e de código aberto para a construção de sistemas inteligentes [60].

[6] Sistema que combina GCs com a entrada de dados estruturados dos pacientes e a recuperação de dados através de um base de dados clinica [30]

4.5 PRO*forma*

4.5.1 Introdução

No Reino Unido, o *Advanced Computation Laboratory of Cancer Research* iniciou em 1998 o desenvolvimento do formalismo PRO*forma*. O nome PRO*forma* é o resultado de uma concatenação dos termos *proxy* ("*authorized to act for another*") e *formalize* ("*give definite from to*") [30].

O objetivo deste formalismo é o desenvolvimento de sistemas especialistas mais confiáveis que possam auxiliar os cuidados de saúde a um paciente de forma ativa, através de suporte à decisão e gestão do fluxo de trabalho [45].

PRO*forma* foi criado de forma a prestar suporte a um processo completo desde a aquisição do conhecimento até a execução do mesmo. Possui uma linguagem semi-formal (sintaxe declarativa [46]) para descrever procedimentos de apoio à decisão médica, uma ferramenta para construir modelos (baseado em argumentação lógica [46]) e uma ferramenta de execução. A construção de GCs é realizada em duas fases: a primeira num nível mais alto através de um editor gráfico (similar ao GLIF) e na segunda o desenho gráfico é instanciado com o conhecimento necessário para a ratificação do mesmo [30].

Similar ao GLIF, PRO*forma* também representa o GC como um fluxograma em que os nós são instâncias de um conjunto de classes pré-definidas [25].

4.5.2 Modelo

Em PRO*forma*, cada guia é modelado como sendo um plano que consiste numa sequência de tarefas [30]. A tarefa (*Keystone*) define quatro tipos: *plans*, *decisions*, *actions* e *enquiries* [45] [30].

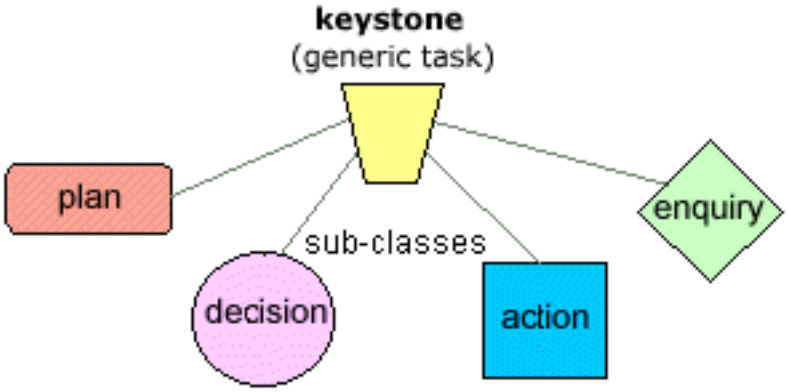

Figura 5 - Modelo de tarefas PRO*forma*

Todas as tarefas são derivadas/estendidas a partir da *keystone/root task*. A ***keystone*** contém um conjunto de atributos que são comuns a todas as quatro tarefas derivadas, inclui atributos administrativos como o nome, legenda ou descrição, e também atributos que descrevem as capacidades de uma tarefa como:

- *goals*: (por exemplo, *'achieve(normal_respiration)*') [30];
- *state*: indica o estado de execução em que se encontra uma tarefa;
- *precondition*: é a condição que deve ser satisfeita para que seja iniciado a execução da tarefa [46];
- *wait_condition*: é uma condição que quando é satisfeita, permite que a tarefa seja iniciada sem esperar que as outras restrições estejam válidas [46]. Uma condição só pode ativar uma ou várias tarefas uma única vez;
- *postcondition*: é uma condição que tem que ser assumida como verdade quando a tarefa é concluída [46];
- *antecedent_tasks*: é um conjunto de identificadores que indicam quais as tarefas que necessitam estar completas para que esta tarefa seja iniciada [46];
- *trigger*: é uma mensagem que permite que as tarefas sejam iniciadas/reiniciadas sem que as suas restrições estejam satisfeitas [46];
- *nbr_cycles_expression*: uma expressão que, quando é avaliada indica o tempo que a tarefa deve ser ciclicamente promulgada [46];
- *cycle_until*: indica que a tarefa deve ser ciclicamente promulgada até a sua expressão ser verdade, ('*cycle(integer, interval)*' [30]);
- *parent_plan*: indica se a tarefa faz parte de um plano e qual o plano a que pertence [46];
- *confirmatory*: se a tarefa deve ser confirmada pelo utilizador antes de ser executada [46];

Os ***plans*** são os elementos básicos da construção de um GC e podem conter coleções de tarefas de qualquer tipo, incluindo outros *plans*.

Plans definem (1) uma sequência ordenada de tarefas, (2) restrições lógicas e temporais na sua promulgação e (3) as circunstâncias em que um *plan* deve ser anulado ou terminado (por exemplo, exceções). Além dos atributos comuns que são definidos na *keystone*, a tarefa *plan* contém atributos adicionais, tais como componentes, restrições de agendamento e temporais e condições de aborto (*abort_condition*) ou rescisão (*termination_condition*). O atributo *components* é um recipiente que contém um conjunto de instâncias de tarefas, semelhante ao atributo *steps* em GLIF.

Por exemplo, um guia que consiste em quatro instâncias de tarefas (por exemplo, *history*, *diagnosis*, *therapy*, *follow-up*), é modelado através de uma instância *plan* de qual o atributo *components* contém referências a essas quatro instâncias de tarefa [30]. A ordem entre essas instâncias de tarefa é definida por dois tipos de restrições: de agendamento e temporais. A ordem das tarefas de restrições de agendamento num plano através de condições qualitativas (por exemplo, a tarefa *history* é executada *before* da tarefa *diagnosis*). A ordem das tarefas de restrições temporais usando condições temporais (por exemplo, a tarefa de *follow-up* é executada 'depois de um período de 10 semanas'). Ao utilizar estes dois tipos de restrições, as tarefas num plano não são modeladas como um fluxograma tradicional em que os elementos do guia são ordenados apenas por restrições de agendamento [30].

Outra maneira de direcionar o fluxo do GC em PRO*forma* é através de condições de aborto ou rescisão [30]. Cada tarefa passa através de uma série de estados: *dormant*, *in progress*, *aborted*, *terminated* e *performed*. Cada tarefa começa com o estado inicial *dormant*. Executando uma determinada tarefa, vai alterar o seu estado de *dormant* para *in progress* [30]. Sempre que uma tarefa é concluída normalmente, o seu estado passa para *performed*. É possível forçar a rescisão ou o aborto de um plano através das condições de rescisão e aborto [30].

A tarefa ***decision*** é representada como sendo um conjunto de possíveis resultados candidatos dos mais diversos tipos de esquemas (expressões lógicas) que apoiam ou se opõem a cada candidato. Todos os candidatos são associados com um conjunto de esquemas. Esquemas consistem em regras, variáveis qualitativas, ponderações quantitativas, e fatores de segurança e candidatos de suporte (+) ou oposição (-), estabelecendo uma ordem de preferência entre eles [30].

.

As ***actions*** são tipicamente procedimentos clínicos, em que a sua execução é feita por um agente externo (como a administração de uma injeção) [45] [30], e são sempre atômicas e não podem ser decompostas [30]. Normalmente uma *action* consiste na emissão de uma mensagem a um utilizador ou uma chamada a um programa externo por meio de uma interface de programação de aplicações pré-definido (API – *Application Programming Interface*) [30]. Por exemplo:

- "dar ibuprofeno, 10mg" - mostra uma mensagem ao utilizador,
- "*call*(*print*(*leaflet1*)" - executa um procedimento externo para imprimir um folheto.

As ***enquiries*** são tipicamente tarefas/pedidos encarregues de adquirir informações (clínicas ou administrativas), ou dados, necessários para a orientação poder prosseguir [45] [30]. Esta informação pode ser obtida a partir de um utilizador clínico ou pode ser diretamente extraída de um software externo de um dispositivo agente ou hardware (por exemplo, EPR ou monitor paciente). Portanto, como foi o caso com a definição de uma *action*, a classe *enquiry* contém atributos que definem a forma de recuperação de dados.

4.5.3 Linguagem

Guias em PRO*forma* são armazenadas (como instâncias de classes de tarefas) usando uma linguagem de representação de conhecimento com controlo de fluxo *time-oriented* chamada Red (R^2L) [25] [30].

Um guia, escrito em R^2L, é uma especificação declarativa de tarefas e das suas (inter)relações organizadas na hierarquia dos seus planos e componentes [30]. Esta linguagem também contém uma linguagem de expressão formal para expressar objetivos (por exemplo, '*achieve(normal_respiration*)'), condições (por exemplo, '*peak_flow* < 30', *'risk_level = severe'*) e sistemas de argumento ('*diagnosis = oesophagitis and liver_disease = absent then cimetidine*: +') [25]. Antes da execução, orientações na língua R^2L são traduzidos para outra língua, chamado L_{R2L} ('Lógica de R^2L'), uma linguagem baseada na lógica de predicados [30].

A execução de guias PRO*forma* é suportada por dois mecanismos de execução, o Arezzo e o Tallis suites (implementação em java [45]). E o mecanismo de execução genérico GESDOR [25], sendo o Arezzo a primeira implementação que permitiu criar, visualizar e aprovar guias do PRO*forma* [25]

4.6 Análise

4.6.1 Primitivas

Primitivas são os elementos base num modelo de representação de GCs. Existem dois tipos de primitivas, as de representação e as de estrutura [47]. As primitivas de representação são constituídas pelas *actions*, *decisions*, *patient state* e *execution state*. As primitivas de estrutura são constituídas por *nesting* e *scheduling constraints* [47].

Uma *action* é uma tarefa clínica ou de intervenção/administrativa que é recomendada a ser executada, mantida/sustentada, ou retirada, durante o processo de aplicação do GC, por exemplo: medicação, prescrição, investigação clínica [47] [29] [28]. Estas são tipicamente usadas para representar tarefas clínicas específicas [29], e podem ser classificadas em três categorias: *clinical interventions*, *data collections*, e *wait actions*. *Clinical intervention*s são as ações que lidam diretamente com a gestão e tratamento de pacientes, como receitas e operações. *Data collections*, por outro lado, são as ações que não tratam pacientes diretamente, em vez disso, só são usadas para obter informação sobre o paciente, como observações e exames. As *wait conditions* são circunstâncias específicas, em que mesmo, que não se faça nada aos pacientes, o seu estado fisiopatológico subjacente ainda pode mudar ao longo do tempo [29].

A *decision* é uma seleção geralmente de uma opção de um conjunto de alternativas baseadas em critérios pré-definidos num GC [47] [29].

A *patient state* é uma descrição de um tratamento individual baseado em *actions* executadas e *decisions* que foram feitas dentro do contexto de um GC [29] [47].

O *execution state* é uma descrição do estado em que se encontra uma tarefa, durante o processo de aplicação do GC [29], tal como *actions* e *decisions* definidas anteriormente, durante o processo de execução do GC [47].

Nesting é a capacidade de permite ter múltiplos níveis de abstração no modelo de representação [47].

Scheduling constraints na representação de primitivas são usadas para representar a ordem temporal destas primitivas em execução. Estas restrições são tipicamente definidas como uma sequência ou concorrência de representação de primitivas, mas podem também ser definidas como *alternatives* e *loops*. Primitivas numa sequência devem ser executadas uma por uma, de acordo com uma programação/horário específico, que é normalmente definida como uma ordem parcial. Primitivas em concorrência devem ser executadas em paralelo [29]. As *alternatives* são um conjunto de primitivas em que a sua execução depende de

critérios específicos, são geralmente definidas em conjunto com uma *decision* primitiva, em que nesse caso a alternativa que deve de ser seguida depende do resultado da *decision*. *Loops* são execuções repetidas de primitivas [29].

As *decisions* e *actions* são as primitivas chave que devem ser fornecidas em qualquer modelo de representação [29] [47].

4.6.2 Comparação

Esta comparação tem como base as primitivas de representação comuns e estruturas dos processos dos diferentes modelos.

Todas as linguagens acima descritas usam TNM, em que a modelação dos GC é feita através de uma sequencia de classes (por exemplo, fluxograma), com exceção da Arden Syntax em que os GC são modelados como (uma coleção) regras modulares independentes. Como resultado, Arden Syntax é mais adequada para representar orientações simples, tais como sistemas de alertas, mas menos adequado para orientações de várias etapas complexas [25].

O plan pode ser definido como "uma organização ordenada de partes de uma conceção global ou objetiva" [28]. Podemos concluir que Asbru e PRO*forma* utilizam uma única classe genérica do objecto *plan* [28]. GLIF tem dois tipos de planos: *Guidelines* e *Macros*. Todas as metodologias fornecem mecanismos para simplificar os planos de *top-level* e para apoiar a reutilização de sub-guias [28]. Em Arden Syntax embora um MLM possa invocar outros MLMs, Arden Syntax por si só, não modela nenhuma estrutura destas invocações [47]. Assim todos os modelos exceto Arden syntax permitem ***nesting***.

Actions em Arden Syntax são representadas por *action slots* nos seus MLMs, e são usadas para codificar a tarefa clínica que vai ser executada. Em Asbru os planos são decompostos de uma forma recursiva até se tornarem planos atômicos (considerados *actions*) que podem ser diretamente executados para fins clínicos específicos. Em PRO*forma* representam intervenções clínicas e são codificadas pelas *actions* e *enquiries*. GLIF utiliza as *actions* para especificar o "trabalho" a fazer pelo sistema de apoio decisão, pelo prestador ou ainda por um agente externo. Assim todos os mecanismos permitem representar, ***actions***.

Em Arden Syntax, as *decisions* são codificadas na *logic slot* do MLM. Asbru não representa as *decisions* explicitamente como um componente independente, em vez disso, codifica as *decisions* em *conditions e preferences* de um *plan* que definem os critérios de transição de um estado para outro *plan*. Em GLIF as *decisions* são representadas pelas *decision steps* que podem ser do tipo *case*_step e *choice_step*. Em todas as metodologias, exceto

PRO*forma*, a tomada de uma decisão é explicitamente relacionada/ligada a uma escolha. PROforma representa as *decisions* na tarefa *decision*.

Arden Syntax não permite representar as primitivas *patient state* ou *execution state*. A forma de dar a volta é através do uso de estados intermédios com link para MLMs relacionados [47]. *GLIF* tem o ***patient state*** que define um contexto particular do paciente, é caracterizado por condições do paciente (por exemplo, hipertensão) e/ou os seus tratamentos (por exemplo, tomar uma dose baixa de diuréticos) e possivelmente por situações clínicas (por exemplo, ambulatórios). GLIF possui a ***branch steps*** e ***synchronization steps*** que são caminhos paralelos de um plano. PRO*forma* e Asbru conseguem ter estes caminhos paralelos como execução sequencial sem ter ***branch steps*** e ***synchronization steps*.** Os processos de cuidados podem envolver atividades sequenciais, interativas, e as possivelmente simultâneas que ocorrem ao longo do tempo.

PRO*forma* usa diferentes construções para especificar a execução de planos sequenciais contra planos paralelos. PRO*forma* usa *scheduling constraints* para governar a execução da tarefa, sendo que a execução sequencial e em paralelo são implicitamente suportadas [28]. Asbru representa *scheduling constraints* como sequências, concorrentes ou como ciclos (*loops*) e todas estas são representadas dentro do seu *plan-body*. GLIF é muito expressivo na representação de *shedule constraints* e representa estas como um *flowchart*.

Modelo de representação	***Actions***	***Decisions***	***Patient State***	***Execution State***
Arden Syntax	*action slot*	*decision slot*	*no*	*no*
Asbru	*plan*	*condition, preference*	*temporal patterns*	*plan state*
PRO*forma*	*action, enquiry*	*decision*	*n/a*	*task state*
GLIF	*action step*	*decision step*	*patientsate step*	*no*

Tabela 1 - Comparação primitivas de representação [29]

Modelo de representação	***Schedule constraints***	***Nesting***	***Modeling patient data***

Arden Syntax	*Module invocation*	*no*	*no*
Asbru	*plan*	*plan-body*	*n/a*
PRO*forma*	*Constraint satisfaction graph, enquiry*	*plan*	*n/a*
GLIF	*flowchart*	*subguideline*	*tree-layer domain ontology*

Tabela 2 - Comparação das primitivas de estrutura [29]

Assim podemos concluir que o GLIF se destaca das outras linguagens, apesar de não especificar a primitiva *execution state* GLIF preenche todas as outras primitivas de uma forma concreta, completa no que diz respeito a especificação, permite representar *schedule constraints* como um fluxograma e é a única linguagem que tem definida a primitiva para a ligação com a modelação de dados dos pacientes.

5. Projetos de investigação Análogos

5.1 Introdução

Existem várias ferramentas de ajuda ao desenvolvimento de GCs, algumas delas são: Arezzo [24], *DeGeL* [24], *GLARE* [23], *GLEE* [37], *HeCaSe2* [24], *NewGuide* [24], *SAGE* [24] e *SPEM* [24]. Neste capítulo iremos abordar o DeGeL por ser uma tecnologia distribuída baseada em Web e ainda o e-GuidesMed [23] e o projeto PRESGUID [48] porque é semelhante ao projeto que pretendemos desenvolver.

5.2 DeGeL

5.2.1 Introdução

DeGeL surge em 2001 na universidade de Ben Gurion em Israel [24] e tem como objetivo fornecer suporte automatizado para a especificação e implementação de GCs no tratamento de pacientes, com a particularidade nos que têm condições crónicas como os diabetes, a hipertensão e a depressão [49]. É uma arquitetura modular com tecnologia distribuída baseada na Web, concebida para apoiar (i) o desenvolvimento e implementação de aplicações de orientação clínicas e (ii) a criação de uma Biblioteca Digital de guias eletrónicos. Permite por isso, a aquisição de conhecimento, manutenção, recuperação e promulgação [49].

5.2.2 Representação

A sua especificação tem como base uma representação híbrida (*multiple-format*) de forma a conseguir representar eletronicamente todas as especificações dos editores e as diferentes necessidades de cada tarefa *guideline-based-care* [49].

No método DeGeL o guia primeiramente é convertido de texto livre (*free-text)* para uma semi-estrutura semântica (*semantically semi-structured text*) por um médico especialista, em

seguida para uma representação semi-formal por um médico especialista e um engenheiro de conhecimento usando um editor de marcação, e por fim, para uma total representação formal (*formal* linguagem de execução como Asbru ou GLIF, sendo Asbru a linguagem de defeito) por um engenheiro de conhecimento.

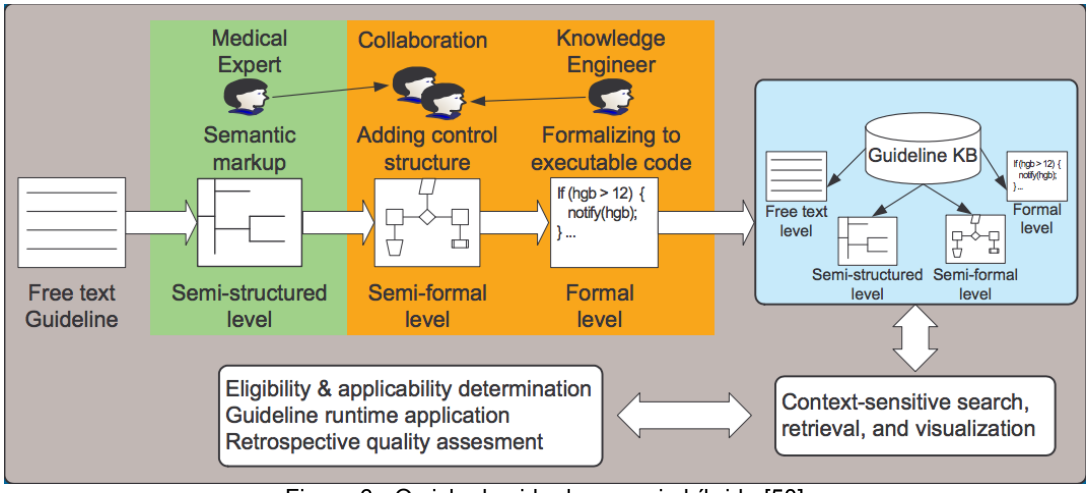

Figura 6 - O ciclo de vida de um guia híbrido [50]

Esta *framework* fornece várias ferramentas como:

- **URUZ**: Ferramenta que permite fazer a conversão do texto livre para *semantic markup* numa estrutura de texto e numa linguagem semi-formal, ou seja, permite fazer a conversão do GC em texto para um formato que permite uma representação compreensível em linguagem máquina.
- **VisiGuide**: Visualização e navegação de um conjunto de guias numa ontologia alvo.
- **Vaidurya**: Motor de pesquisa que fornece pesquisa baseada em contexto. Vaidurya utiliza a ferramenta VisiGuide para facilitar a visualização, navegação, e exploração dos resultados obtidos na pesquisa. Ajuda a determinar qual o GC que é mais relevante.
- **IndexiGuide**: Permite a classificação semântica dos guias para facilitar a sua recuperação mais tarde. Automatização do processo de classificação ao longo de vários eixos semânticos (exemplo: *diagnosis*, *therapy*), e definição do seu tipo (exemplo: *screening*).
- **Spock**: Permite a execução (desde começar e resumir) do GC [24].
- **QualiGuide**: apoia a avaliação da qualidade e a adesão médica ao GC [24] [49].
- **GESHER**: aquisição de guias [49].
- **Dipole:** auxilia os clínicos a determinar a elegibilidade do paciente e aplicabilidade do GC [24].

Fornece ainda suporte a várias tarefas durante o desenvolvimento e a implementação de um guia sendo que:

Durante a especificação, permite a verificação do processo de especificação (sintaxe) e a validação contra os seus objetivos (semântica). Durante a execução, permite a determinação de elegibilidade do paciente e da aplicabilidade do guia, a visualização de uma ou mais orientações potencialmente aplicáveis, a aplicação (execução) do guia, avaliação das ações dos provedores de qualidade, a modificação de programas de orientação ou de provedores, e a avaliação da eficácia do guia [49].

5.3 e-GuidesMed

5.3.1 Introdução

O portal Web e-GuidesMed permite a definição de CIGs e a adição de informações complementares exigidas pelos utilizadores, sendo também capaz de gerar versões executáveis independentes da plataforma [23]. Assim foi desenvolvido um CDSS baseado em guias clínicos eletrónicos como sendo uma ferramenta de consulta e de educação [51].

É possível aceder e executar CIGs através do portal e-GuidesMed (www.e-guidesmed.ehu.es). Os CIGs têm como base de desenvolvimento a tecnologia GuidMEx [51]. e-GuidesMed oferece a possibilidade de navegar graficamente no conhecimento e guardar/manter o registo da execução de um guia a um paciente em particular.

e-GuidesMed tinha a intenção de encontrar uma solução tecnológica para atingir uma tradução eficaz do conhecimento através de:

1. fornecer o melhor conhecimento, disponível no melhor formato num CDSS;
2. superar barreiras que fornecem um CDSS úteis para tradução e divulgação de conhecimento;
3. suportar a melhoria contínua do conhecimento e da metodologia de desenvolvimento [23].

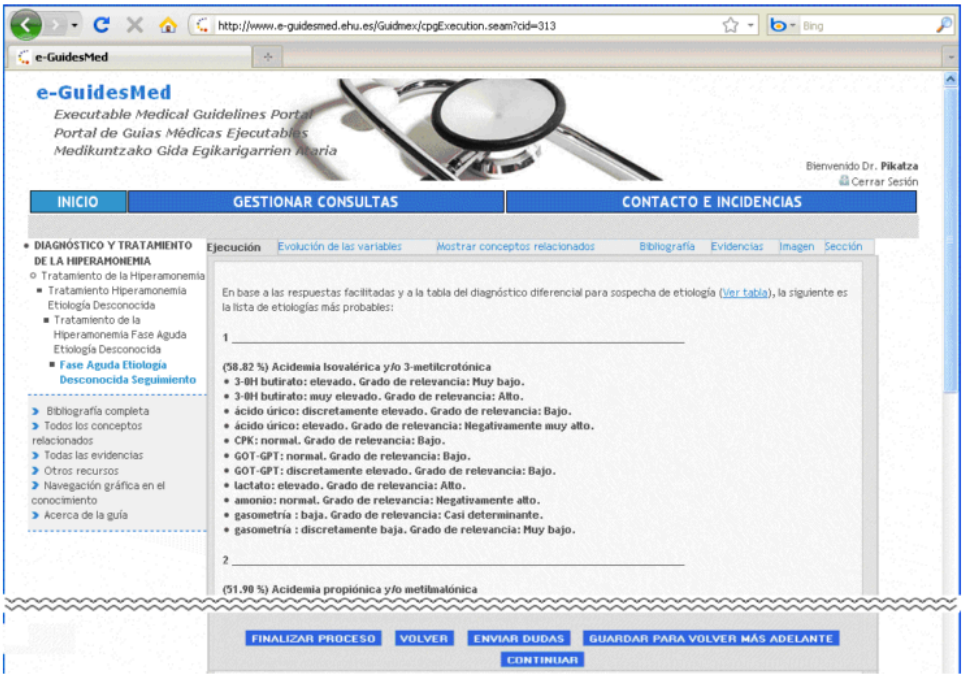

Figura 7 - Execução de um e-CG no portal e-GuidesMed

5.3.2 Representação

Possui uma estrutura de conceitos (*archetypes*) com base nos modelos de referência e conceitos definidos pela *openEHR*. As atividades prescritas pelo GC são representadas como instâncias desses conceitos. Devido ao grande número de instâncias conceito a serem definidas para cada e-CG, foi desenvolvido um editor para recolher as informações do GC mais qualquer conteúdo complementar. Este editor é gerado automaticamente a partir de um metamodelo, utilizando ferramentas fornecidas pelo EMF. As transformações *Model-to-Text* para gerar o executável e-CG são feitas através da *OpenArchitectureWare* (OAW).

Alguns nós do e-CG podem incluir sistemas de classificação para a implementação de tarefas de diagnóstico. Através do uso de *Fuzzy Cognitive Maps* e *Fuzzy Inference Systems*. A execução é realizada por Mairi: um *Web-Service* do motor de inferência EHSIS [23].

A execução e-CG é conseguida utilizando ferramentas de BPM.

Foi desenvolvido uma aplicação de pesquisa de terminologia médica UMLS, que inclui um navegador gráfico da relação entre conceitos médicos. Esta ferramenta é útil para identificar conceitos médicos durante o trabalho multidisciplinar com os especialistas. Foi também desenvolvido um editor de e-GC. Para a representação à base de standards do e-CG, criamos o modelo de referência A-POM (*Archetype-based Process Object Model*) com base no openEHR *Archetype Object Model* [23].

Os nós do e-CG podem ser classificados em: *Question*, *Decision*, *Recommendation, Action, Calculation* e *Final*. Qualquer sistema de classificação que poderia ser requerido por um e-CG é executado nos nós *Calculation* utilizando dados do paciente.

Diferentes e-CG podem ter seções comuns, a ferramenta de edição oferece a possibilidade de definir níveis de abstração maior do que *Nodes* usando *Sections* e *Sets*. A *Section* é formada apenas por *Nodes*, enquanto *Sets* podem conter *Sections* e *Sets*. Toda a orientação é representada como um conjunto. Para cada um destes elementos, foi definido um standard utilizando os resultados da iniciativa openEHR. Os modelos dos GCs criados usando as ferramentas de edição são transformadas em instâncias desses standards por meio de um padrão (*template*) para executar transformações *Model-to-Text*.

Essas transformações permitem gerar instâncias de standards que irá armazenar todas as informações do GC num formato independente da plataforma. Este projeto permite a reutilização do conhecimento nas diferentes plataformas, o que faz com que os esforços de desenvolvimento e de normalização da criação de um e-CG aconteçam apenas uma vez. Uma visão geral da criação e execução e-CG.

e-GuidesMed é constituído por 3 tipos de *roles*: *practitioner* (executa GCs), *expert* (resolve qualquer dúvida que surja durante a execução), *administrator* (administra os GCs e utilizadores) [51].

5.4 PRESGUID

5.4.1 Introdução

O projeto PRESGUID (acessível em http://cybertim.timone.univ-mrs.fr/recherche/projets-recherche/xga/projet-presguid/) visa atender as necessidades de informação dos médicos na sua prática combinando GCs com documentação regularmente atualizada de evidências científicas e documentação fiável sobre prescrições de drogas. Foi desenhado como um serviço online que permite a consulta de GCs computorizados ligados a uma base dados de drogas (Vidal, acessível em https://www.vidal.fr/) permitindo ao paciente especificar o apoio a decisão.

Utiliza uma abordagem pragmática para a implementação dos guias de texto ANAES num formato interpretado por computador, e reforça a recomendação de tratamento, através da referência de drogas da base de dados Vidal.

PRESGUID fornece um sistema Web baseado em GCs que recebe como dados de entrada dados clínicos e do paciente, retornando recomendações. Caso uma recomendação faça a prescrição de drogas o sistema consulta a base de dados de drogas e fornece informação sobre a medicação [48].

5.4.2 Modelação

PRESGUID é inspirado em GLIF, e um guia é uma sequência das *decision* e *action steps*. As *action steps* são usados para recolher dados do paciente, computorizar e assinar valores (por exemplo: calcular o índice de massa corporal através do peso e da altura), e mostrar mensagens ou recomendações. As *decision steps* permitem a avaliação de uma condição composta por um ou mais critérios. Os critérios são derivados dos dados das *action steps* anteriores. Utiliza os clássicos operadores comparativos, operadores lógicos e funções matemáticas para produzir as expressões a serem avaliadas.

5.4.2.1 Dados do paciente

Os dados do paciente durante a execução do guia são representados pela classe *DataItem* que é composta por vários pares de valores de atributos. Os seus valores podem ser do tipo numérico ou texto, sendo limitados por um intervalo (mínimo e máximo) ou por um conjunto de valores definidos numa lista de itens.

O uso de *standards* de classificação é a chave para a interoperabilidade entre GCS e EMRs.

5.4.2.2 Authoring tool

Foi desenvolvido um editor gráfico para criar e manter a lógica e o conteúdo do GC através de uma estrutura no formato XML. As classes deste modelo são instanciadas como *tags* de XML, contendo subelementos e atributos que detalham a estrutura lógica e conteúdo do GC. Neste editor o GC é apresentado com uma árvore de decisão que permite operações de *drag and drop*, fornecendo formulários específicos para estruturar cada elemento do guia (*steps, decision criteria, recomendations* ...). Também permite associar documentos didáticos (referências bibliográficas, definições de conceitos médicos, documentos multimédia, entre outros) com dados do paciente e outros elementos do GC que podem ser apresentados durante a consulta ao GC.

5.4.2.3 Interface do utilizador

A interface do utilizador é gerada automaticamente, a informação necessária para a interface do utilizador é representada em XML, e através *eXtensible Style sheets* (XSL) é

gerado o HTML de forma a organizar a informação e é criado os formulários e controlos que permitem a entrada de dados.

5.4.2.4 Drug prescription

A *drug prescription* é representada através da tag *XML DRUG_PRESCRIPTION* inserida dentro de um *tag* mensagem que recomenda uma terapia relacionada com a droga.

6. Registos de saúde eletrónicos

6.1 Introdução

Para este projeto pretende-se fazer integração com registos eletrónicos de saúde. Existem várias estruturas de informação de dados de registos eletrónicos. Nestes links (http://www.emrandhipaa.com/emr-and-ehr-vendors/ e http://www.emrandhipaa.com/administrator-2/2006/02/21/overwhelming-list-of-emr-companies/) é possível encontrar uma vasta lista de ferramentas de EMR. Esta diversidade e o acesso a dados de um paciente que foi tratado em várias estruturas (Hospitais/Clinicas) é um problema complexo. De todas essas estruturas foi decido usar o OpenEMR porque da lista apresentada [52] verificou-se que é gratuita e de código aberto e cumpre com todos os requisitos definidos no capítulo seguinte.

O aumento da quantidade e complexidade das informações e dos conhecimentos relativos à saúde, levou a que uma das componentes mais importantes de qualquer organização de saúde, seja o processamento de informação, e qualquer sistema de informação é tão bom quanto a qualidade dos dados e da informação que o suportam. Consequentemente, emerge a necessidade de mudança no paradigma da saúde e dos serviços prestados, evitando a falta de informação de um individuo. O Registo de Saúde Eletrónico (RSE) poderá funcionar como solução para estes problemas, na medida em que facilita a partilha de informação do histórico clínico do individuo entre os profissionais da saúde.

6.2 Definição

Na literatura consultada encontramos diferentes definições para o conceito de RSE.

Definição 1: É um sistema agregador de informação relativa aos antecedentes e ao estado de saúde atual, físico e mental, de cada cidadão, num formato suscetível de processamento informatizado, armazenado e transmitido de forma segura, e acessível, por múltiplos

utilizadores independentemente do momento ou local de acesso, desde que devidamente autorizados. A sua finalidade primária é o suporte a cuidados de saúde integrados, com continuidade, eficientes e com qualidade [53].

Definição 2: Registo médico completo ou documentação equivalente, em for- mato eletrónico, dos antecedentes e do estado de saúde atual, físico e mental, de uma pessoa, que permite obter prontamente estes dados para fins de tratamento médico e outros, estreitamente conexos [53].

Definição 3: Repositório de informação seguro, acessível em tempo real no ponto de prestação de cuidados, centrado no paciente, orientado para profissionais clínicos. O RSE apoia a tomada de decisão clínica através do acesso a registos de informação sobre a saúde de um paciente onde e quando se torna necessário, incorporando formas de suporte à decisão baseada na evidência [53].

Definição 4: Um repositório de informação a respeito da saúde de indivíduos, numa forma processável eletronicamente [54].

Sintetizando, e tendo por base todas estas definições, podemos dizer que o Registo de Saúde Eletrónico permite o armazenamento de todo o histórico clínico de um indivíduo em formato eletrónico, bem como a gestão de toda essa informação de forma segura e confidencial.

No entanto, é importante desenvolver modelos que permitam partilhar dados clínicos, aumentar a longevidade dos dados, melhorar a qualidade e tornar os dados independentes da tecnologia usada.

6.3 OpenEMR

6.3.1 Introdução

OpenEMR é uma ferramenta de gestão da prática médica que suporta registos eletrónicos de saúde. Possui certificação "*2014 ONC Ambulatory EHR Certification*". É gratuita e *open source,* sendo uma das ferramentas mais usadas atualmente. Possui vários *demos*,

manuais de suporte (instalação, atualização, configuração e utilizador e desenvolvimento), e suporte gratuito em vários fóruns [55].

Esta ferramenta foi a escolhida de entre outras para ser integrada no nosso projeto, devido à sua principal característica de ser gratuita e de código aberto juntamente com a existência de uma *RESTful api* [56] [57], que permitiu fazer a ligação entre o nosso sistema de representação e execução de guias clínicos e um sistema de registos eletrónicos de saúde.

6.3.2 Características

Esta ferramenta tem as seguintes características:

- Gratuito e código aberto;
- Possui certificado ONC, mais precisamente "*2014 ONC Ambulatory EHR Certification*";
- Permite gerir toda a informação sobre o paciente;
- Agendamentos relacionados com o paciente;
- Registos eletrónicos médicos;
- Fazer prescrições médicas;
- Regras de decisões clinicas;
- Um portal Web para o paciente;
- Permite gerar relatórios (*reports*);
- Possui vários idiomas;
- Segurança;
- Suporte;
- Comunidade[58].

Fornece ainda uma REST API [57], que permitiu fazer a ligação com este projeto.

7. Modelo de representação proposto

7.1 Enquadramento

Este projeto visa poder ser acessível em qualquer lugar e disponível a qualquer hora. O conceito de computação ubíqua é muito importante para disseminação de um CDSS. A integração do uso deste projeto no dia-a-dia de um prestador de cuidados com a capacidade de ser usado em larga escala, ser centralizado (centralização/coordenação de dados), a divulgação do conhecimento, e de poder ser usado durante um pedido de socorro, leva a que o uso seja natural. O modelo proposto tem como base de conceitos e ideias do GLIF. Tal como o projeto iniciado em 2004 e que não foi terminado, o projeto PRESGUID [59], também utiliza GLIF como base. As *actions steps e decision steps* tanto do projeto PREGUID como do GLIF são das tarefas mais importantes. Por isso foram reproduzidas no modelo proposto tendo como conceito base o projeto PRESGUID.

Assim, o modelo proposto deve conter um conjunto de tarefas genéricas em que seja capaz representar todas as facetas simples, bem como diagnósticos e tratamento complexos. Este conjunto deve ser compreensível em um nível funcional por autores do guia e em um nível executável pelo sistema de apoio à decisão informatizado [25].

7.2 Arquitetura do sistema e tecnologias

A arquitetura consiste num servidor Web com recurso a NGINX (servidor proxy de HTTP) e php. PHP foi selecionado para este projeto por ser uma linguagem de programação do lado do servidor (principal característica), multi-plataforma, velocidade, robustez e acesso a dados, facilidade de manutenção e instalação. Permite o acesso a diferentes bases de dados, sendo MySQL a que iremos utilizar para este projeto. Oferece ainda liberdade de escolha do sistema operativo e do servidor Web a usar.

Como pretendemos fazer um website e sendo o PHP uma linguagem especialmente desenhada para programação Web esta é a linguagem mais indicada a usar.

Para diminuir o tempo de desenvolvimento foi usada uma Framework que tem inúmeras vantagens como: um desenvolvimento mais rápido, dá estabilidade à aplicação, reduz a repetição de código (não reinventar a roda), tem soluções eficazes para problemas frequentes, organização, MVC, Base de dados, *Logs*, Geração automática de código, Segurança, Testes, Cache, Librarias, Licença MIT[7], Escalabilidade. A *framework* usada para o desenvolvimento foi *Symfony2*, porque tem excelente documentação, elevado suporte e taxa de atualizações, permite a utilização de um ORM (Doctrine), tem uma grande popularidade/comunidade, e em termos de funcionalidade permite o uso de um sistema de *templating* (*Twig*), permite a geração/validação de formulários, permite gerar uma lista de controlo de acesso (ACL) muito facilmente. Além deste conjunto de fatores traz facilidade na manutenção, atualização e modernização da aplicação.

Para a ligação com o OpenEMR, foi usada uma RESTful API que responde com o formato de XML, permitindo assim a ligação entre a aplicação e OpenEMR.

A figura seguinte apresenta um esquema da relação entre as tecnologias usadas.

[7] Para saber mais consulte https://en.wikipedia.org/wiki/MIT_License.

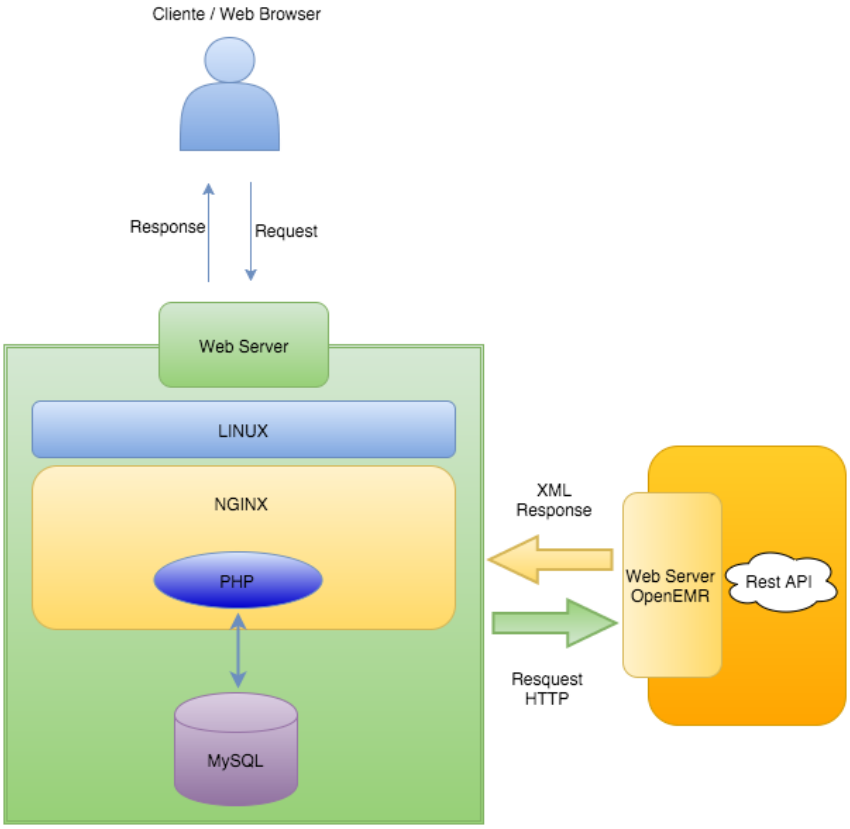

Figura 8 - Arquitetura do sistema

7.3 Representação

Tal como GLIF cada GC é modelado como sendo um sequência de tarefas (*task-based*) e segue o modelo TNM. O GC é constituído por um conjunto de entidades: *guideline, steps, actions, decisios, questions* e *transactions*.

A tarefa *guideline* contêm os atributos administrativos (como o titulo) e os atributos que descrevem a intenção do guia, apresentando semelhanças com o objeto *guideline* de GLIF.

A tarefa *step* representa o agrupamento de lógica comum, ou seja permite o agrupamento de *actions* e *decisions*. Esta entidade também apresenta características com a *macro step* de GLIF.

Actions são as ações do modelo de tarefas que podem ser realizadas. Estas podem ser definidas em 2 tipos, as *medically_oriented* em que representam recomendações médicas, e as *programming_oriented* em que dentro deste tipo podem conter atributos que correspondem a dados do paciente (por exemplo a idade e o género) em que o tipo de

resposta pode ser o mesmo que o atributo selecionado, do tipo *number* (quando a resposta é um numero) e do tipo *average* (obtenção da media do resultado conjunto das respostas).

Cada *action* pode ter várias *questions* que são usadas para construir as perguntas e ainda pode ter uma *transaction*. Quando o tipo da *action* exige apenas um pergunta e não existe nenhuma *question* configurada o sistema utiliza o titulo da action para construir a pergunta. Actions em GLIF têm mais um tipo de ação que é a *control-oriented*, este modelo não respresenta este tipo, mas os outros 2 tipos estão representados.

Uma *transaction* é a tarefa responsável pela transição de uma tarefa para outra, podendo transitar entre *actions* e *decisions*. No caso da transição de uma *decision* para outra tarefa a *transaction* é responsável pela transição de quando a condição da *decison* for verdadeira e quando for falsa. No caso das *actions* como não contêm condições quando é necessário uma transição esta é considerada como sendo sempre verdadeira.

A tarefa *decision* é constituída por *questions*, uma *condition* que define quando o resultado é verdadeiro, uma *transaction* e pode ter ligação a um *atributo*. No caso das *decisions* a tarefa *transaction* é de preenchimento obrigatório contendo sempre a representação do fluxo de para onde vai quando a condição é falsa ou verdadeira. A *condition* é constituída por um operador de comparação e um valor que define a veracidade da condição. A tarefa *decision* pode ainda ser configurada para depender de um valor de um atributo. Por exemplo uma *decision* em que a sua *question* tenha um valor em que dependa do género do paciente. Este modelo ao contrario do GLIF não tem o tipo de *decision* "case_*step*" apenas o conceito de que as decision têm que ser aprovadas por um agente externo. Assim esta tarefa também tem apresenta semelhanças com a tarefa *decision* de GLIF

Uma *question* é a representação da pergunta que é constituída por um titulo, uma descrição que é opcional e pode ter uma opção de um atributo.

As tarefas *action, decision e question* são as tarefas atómicas deste modelo de representação.

Assim a base de todo este modelo de representação ser o GLIF este modelo também apresenta algumas semelhanças com PRO*forma* nomeadamente os atributos *precondition, antecedent_tasks* que em conjunto se assemelham a entidade *transaction*, as *actions* de PRO*forma* assemelham-se a entidade *action* quando esta é do tipo "recomendation" e as enquiries assemelham-se a entidade *action* quando esta é diferente do tipo "recomendation".

Para uma melhor compreensão deste modelo de representação a figura que se segue mostra o esquema relacional da base de dados.

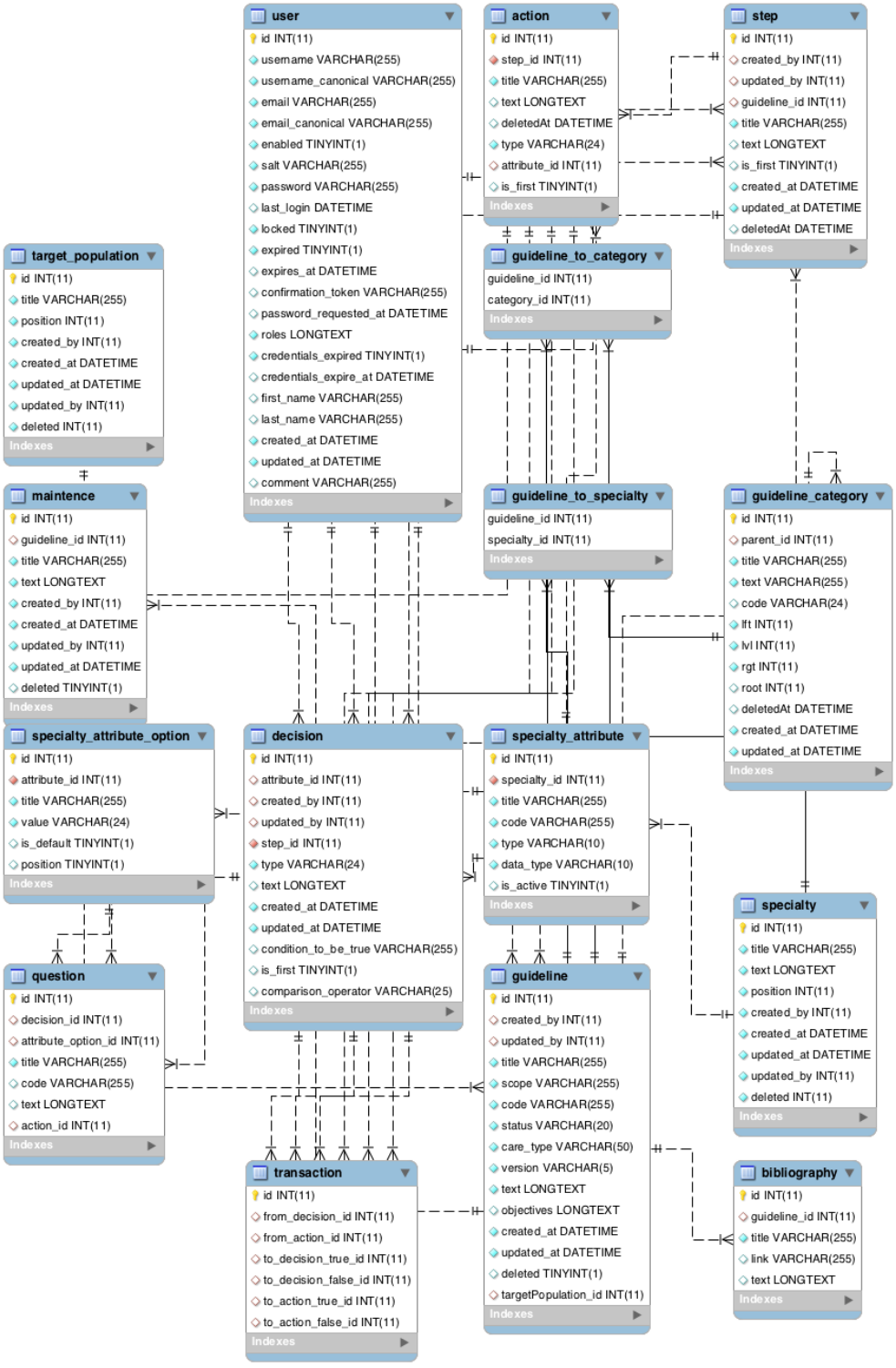

Figura 9 - Modelo da base de dados

As tabelas criadas foram usadas para representar a seguinte informação:

- guideline: Guarda os GCs;
- category: Guarda as categorias que um GC pode ter;
- specialty: Guarda as especialidades que um GC pode ter;

- specialty_atribute: Guarda os atributos de uma especialidade;
- target_population: Guarda a população destino do GC;
- bibliography: Guarda a bibliografia de um GC;
- user: Guarda os utilizadores da aplicação;
- step: Guarda a informação sobre o agrupamento de ações e decisões do GC;
- decision: Guardas as decisões do GC;
- action: Guarda as ações do GC;
- transaction: Guarda as transições do GC;
- question: Guarda as questões que uma ação ou uma decisão pode ter.

7.4 Caso de estudo

Para este estudo foi decidido usar GCs do BCGuidelines porque da lista de onde podemos encontrar guias clínicos descrita no capitulo 3.1, grande parte dos GCS aqui fornecidos têm o algoritmo clínico representado, que é bastante importante porque a compreensão do GC sem este algoritmo torna-se muito difícil por vezes quase impossível de se fazer por quem não pertence à área médica. Para além de podermos aceder e fazer o download gratuitamente no BCGuidelines podemos encontrar uma lista de GCs com muita e pouca complexidade.

Assim como a nossa área é informática e não medicina e a compreensão dos termos médicos poderia não ser a mais adequada, a escolha do GC teve como base a escolha de doenças conhecidas. Por isso o GC escolhido é sobre problemas de bebidas alcoólicas em que apresenta bastante complexidade devido a apresentar a ligação com dados do paciente como a idade e o género, condições de *worflow*, cálculos como medias, a apresentação de recomendações e ainda a interligação de GCs clínicos uma vez que este Guia é subdividido em 3 GCs.

7.5 Aplicação Web

A aplicação Web é apenas um protótipo que pretende mostrar a viabilidade do modelo proposto. Desta forma pretende-se dar a conhecer algumas das suas funcionalidades, sendo a sua estruturação a seguinte:

Roles: Como esta aplicação pretende ser usada por vários prestadores de cuidados, há uma necessidade de bloquear o acesso a áreas sensíveis (informações sobre outros utilizadores, configurações globais à aplicação). Assim a aplicação tem 3 ***roles***, que definem

as permissões*: ROLE_SUPER_ADMIN* (é o super utilizador, este utilizador tem permissões para todas as ações na aplicação e não pode ser removido), *ROLE_ADMIN* (tem mais permissões que a *ROLE_USER* e menos que a *ROLE_SUPER_ADMIN*, pretende-se que este utilizador seja por exemplo um médico especializado) e *ROLE_USER* (qualquer prestador de cuidados).

Login: Para aceder a qualquer ponto da aplicação o utilizador precisa de estar registado no sistema. Na página de login além da autenticação na aplicação permite a recuperação da password.

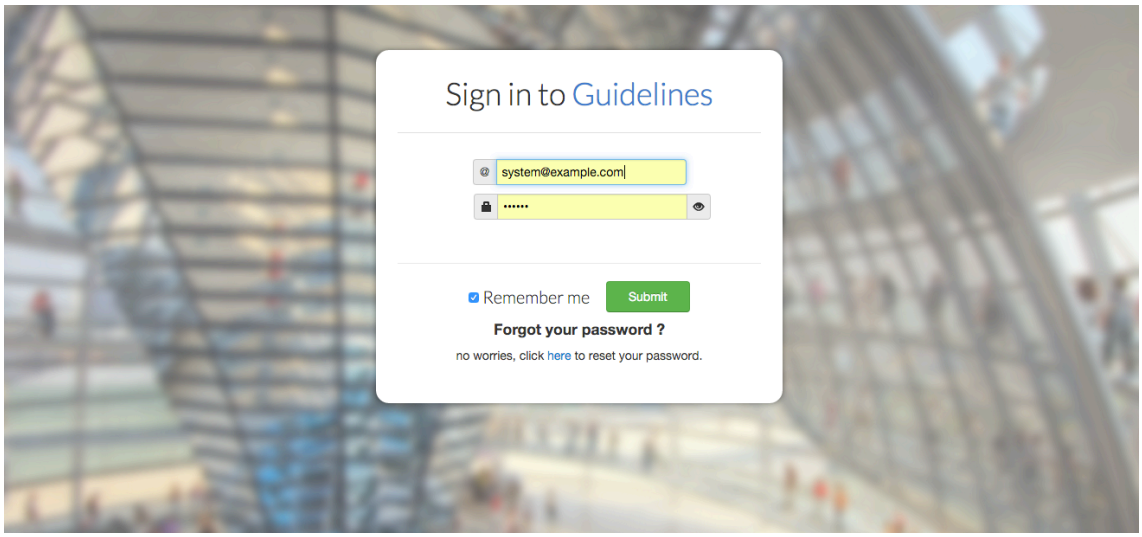

Figura 10 - Página de Login

Painel de controlo: Todos os utilizadores têm acesso ao painel de controlo onde podem ver estatísticas sobre GCs e utilizadores. Como podemos verificar na Figura **11**, esta página apresenta dois gráficos um sobre os utilizadores registados na aplicação pela hora do dia e outro gráfico sobre quantidade de GCs criados pelas diferentes horas do dia. Ambos os gráficos permitem a visualização do total e da média por hora, do dia corrente, dos últimos 7 dias e dos últimos 30 dias.

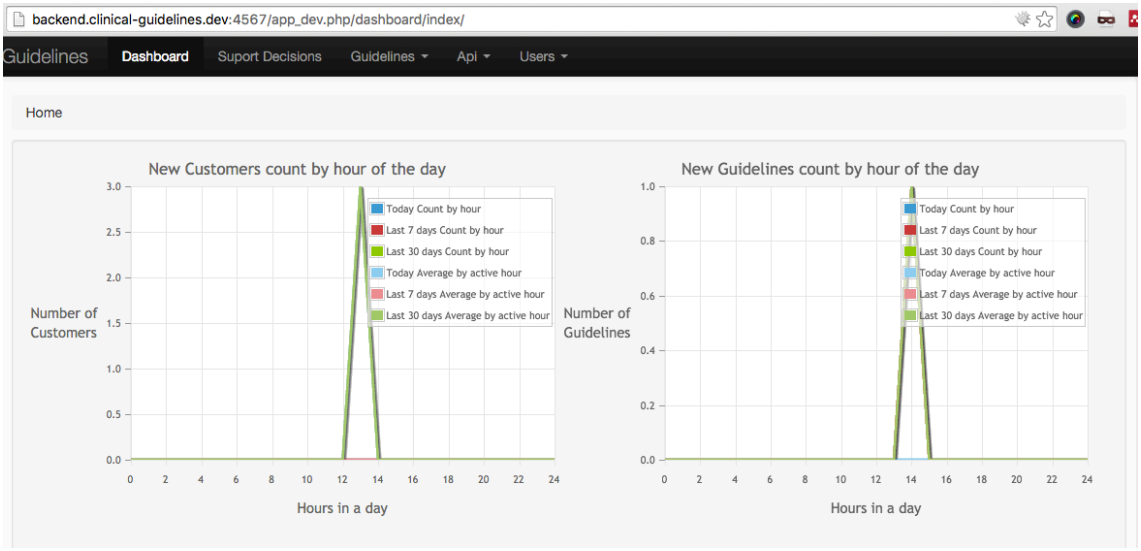

Figura 11 - Painel de controlo

Utilizadores: São os utilizadores do sistema (prestadores de cuidados). O sistema permite a criação de utilizadores, mas esse tem que ser criado por um utilizador com permissões suficientes (tem que ter a *ROLE_SUPER_ADMIN* ou a *ROLE_ADMIN)*. Assim como a criação, na eliminação de utilizadores aplica-se a mesma regra. Em relação à edição de um utilizador caso o utilizador corrente tenha alguma das *roles* de *administrador* pode editar qualquer utilizador, se só tiver a *ROLE_USER* só se pode editar a si próprio. A figura seguinte mostra a página de listagem dos utilizadores.

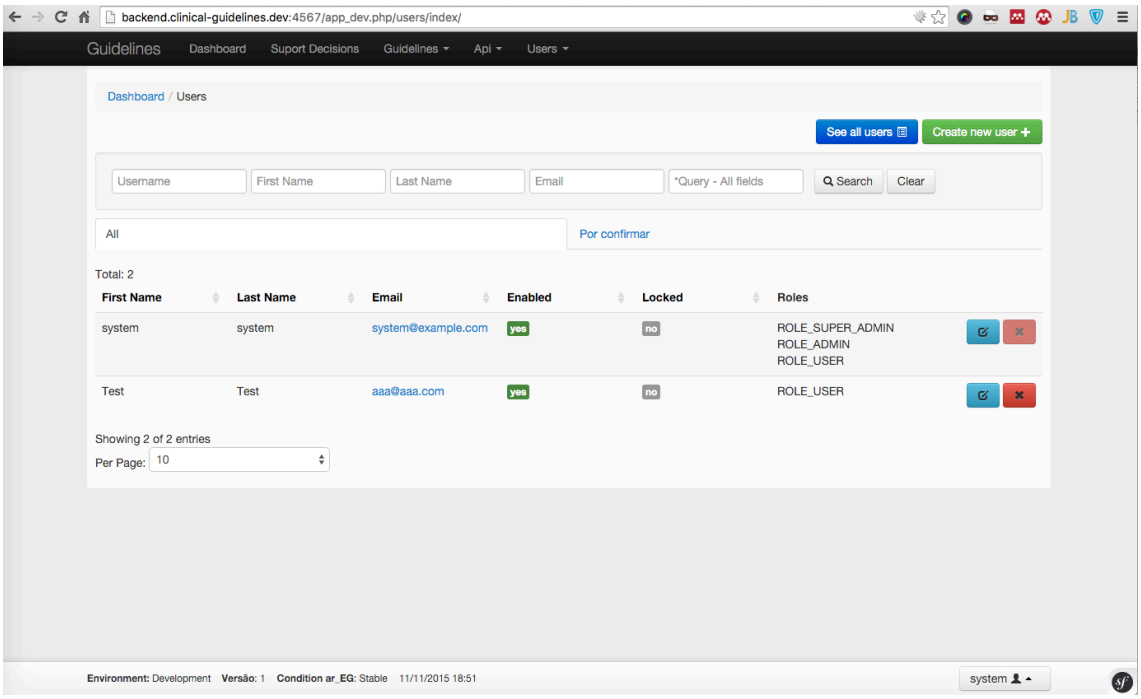

Figura 12 - Listagem dos utilizadores

Na figura seguinte podemos ver a edição dos dados de um utilizador. É permitida a edição do seu nome de utilizador, e-mail, primeiro e ultimo nome, activar e desactivar a sua conta, alterar a sua password e as suas roles. Caso o utilizador não tenha nenhuma das roles de admin não é permitido a alteração das suas roles.

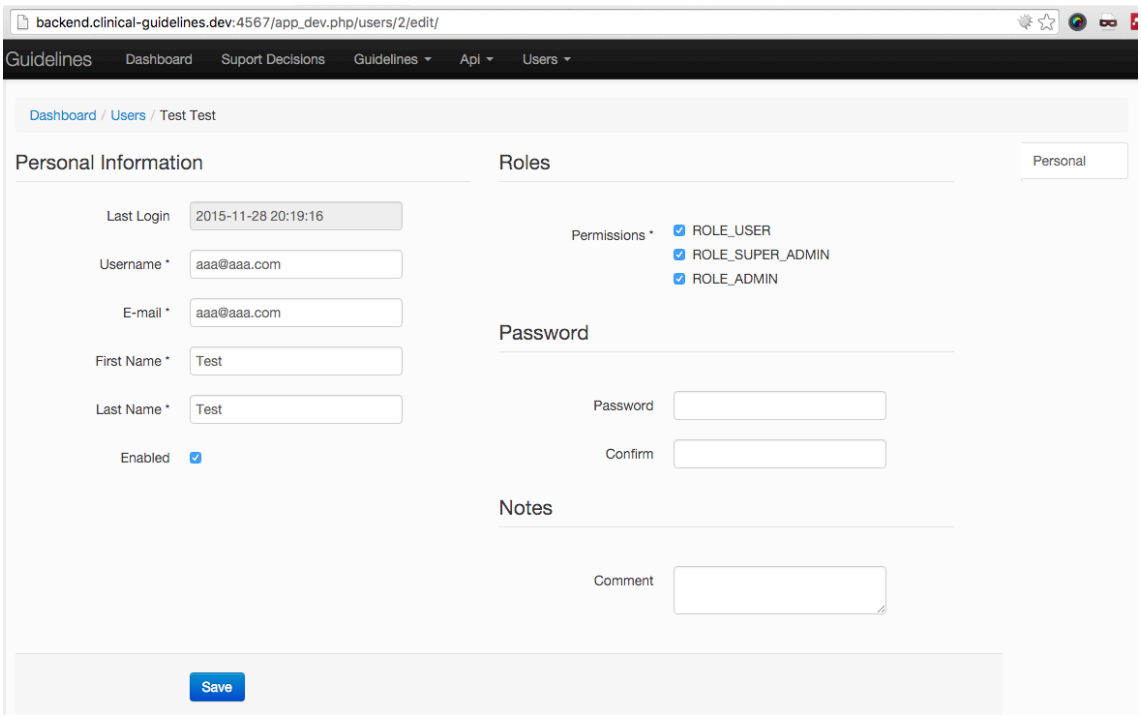

Figura 13 - Edição de um utilizador

API: A aplicação tem um protótipo de uma API. Este protótipo é uma RESTful API em json e têm o intuito de permitir o desenvolvimento de uma aplicação mobile. Como tal tem um sistema que permite a geração de documentação sobre a API. Na figura seguinte podemos ver um exemplo da documentação que é gerada.

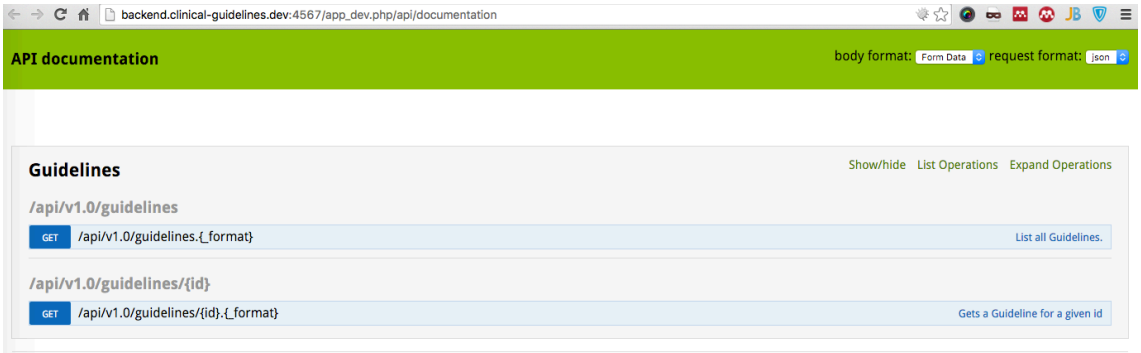

Figura 14 - Documentação gerada automaticamente da API

População destino: A população destino permite identificar a quem se destina o GC. Apenas os utilizadores com *roles* de administrador podem fazer o CRUD[8] desta informação.

[8] São as quatro operações básicas utilizadas em base de dados relacionais.

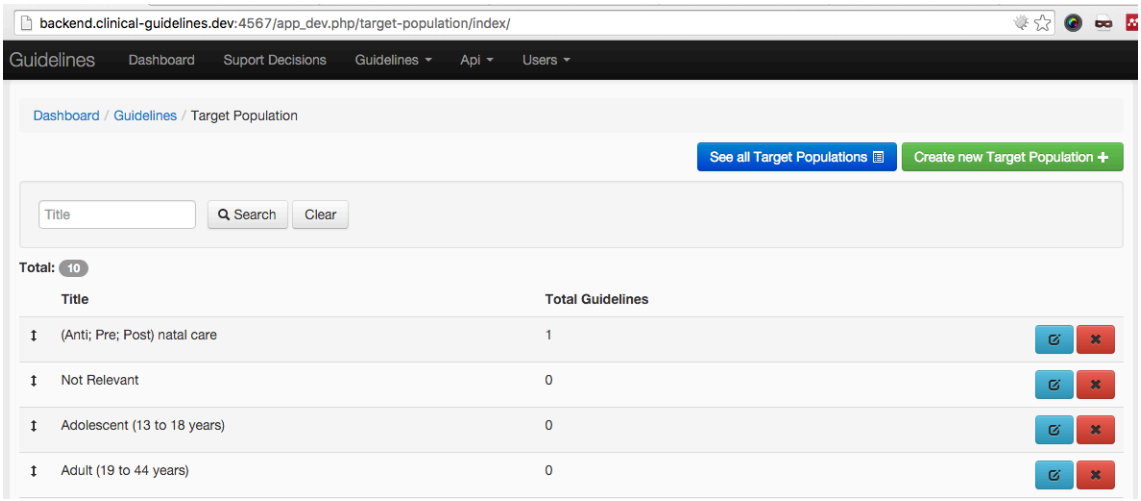

Figura 15 - Listagem população destino

Atributos: São características do paciente que são comuns numa especialidade dos GCs. Um exemplo de um atributo é o género. Os atributos podem ter opções como por exemplo: o atributo género pode ter uma opção “Homem” e outra opção “Mulher”. Apenas os utilizadores com roles de administrador podem administrar os atributos.

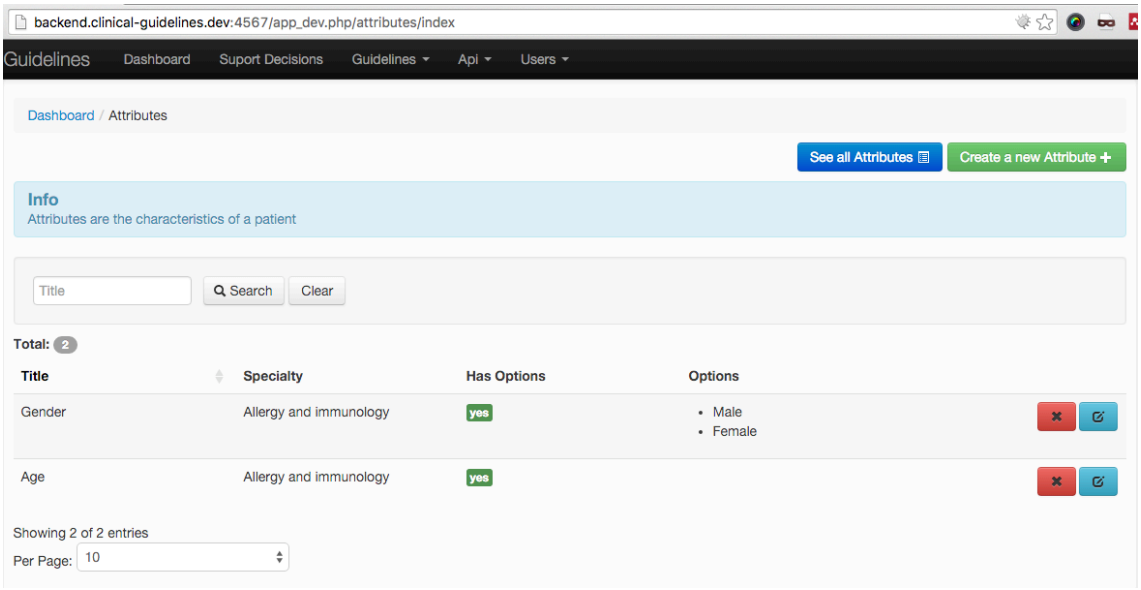

Figura 16 - Listagem de atributos

Categorias: São as categorias a que um GC pertence. Estas podem ter recursividade e são constituídas por um titulo e uma descrição. Apenas os utilizadores com *roles* de administrador podem administrar as categorias.

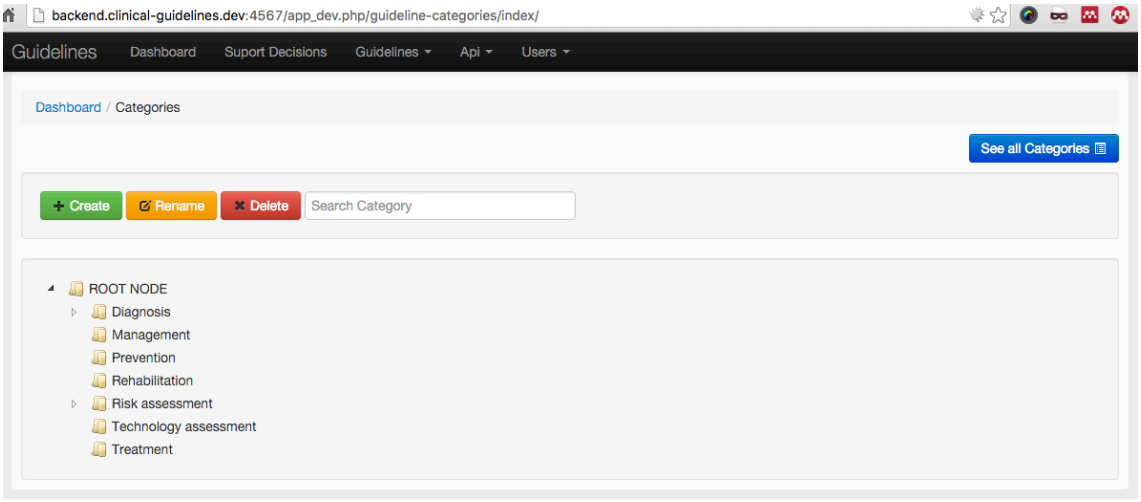

Figura 17 - Listagem das categorias dos GCs

Especialidades: São as especialidades médicas a que um GC pertence e são constituídas por um titulo e uma descrição. Apenas os utilizadores com roles de administrador podem administrar as especialidades.

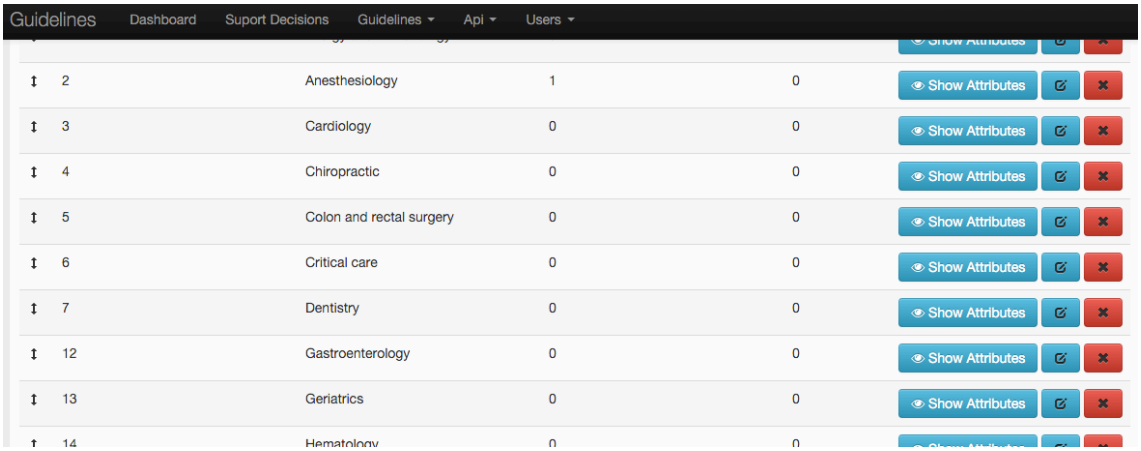

Figura 18 - Listagem das especialidades dos GCs

GCs: Todos os utilizadores têm permissões para administrar os GCs. O processo de criação de um GC exige algum conhecimento. Em cada GC existe a possibilidade de fazer a sua execução, gerar uma pré-visualização do fluxograma do GC e ainda a geração de um pdf do

GC. Cada GC é constituído por especialidades, categorias, código de diagnostico, estado do GC (aprovado, à espera de revisão, lixo, em construção), o tipo de cuidado, a população destino, o titulo, a descrição, objetivos, e âmbito, podendo ainda ter bibliografia associada.

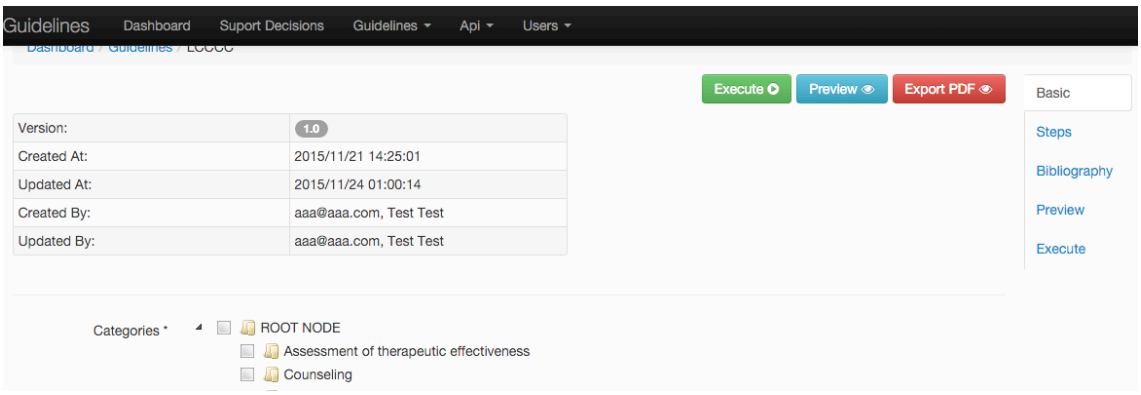

Figura 19 - Edição de um GC

A criação do GC não é uma tarefa fácil, a melhor forma de criar um GC é começar a do fim para o inicio. Esta abordagem contraria ao que é comum no dia a dia é devido ao fluxo do Workflow. Se seguíssemos o fluxo normal para definirmos por exemplo a transição da action_A para a action_B teríamos de criar as duas *actions* e depois editar a action_A e dizer que esta segue para a action_B. Se seguirmos a abordagem contraria e fazer a criação da action_B antes da action_A não precisamos de voltar à *action_A* para edita-la.

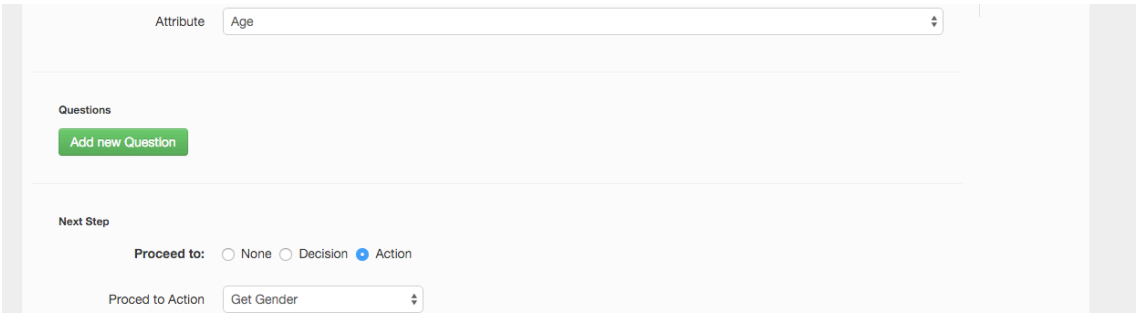

Figura 20 - Edição de uma *action* com atributo e transição

A mesma abordagem descrita anteriormente deve ser seguida na criação de *decisions*. A figura seguinte apresenta a edição de uma *decision* em que é possível configurar a transição para quando a condição seja verdadeira e falsa, a dependência de uma *question* de uma opção de um atributo.

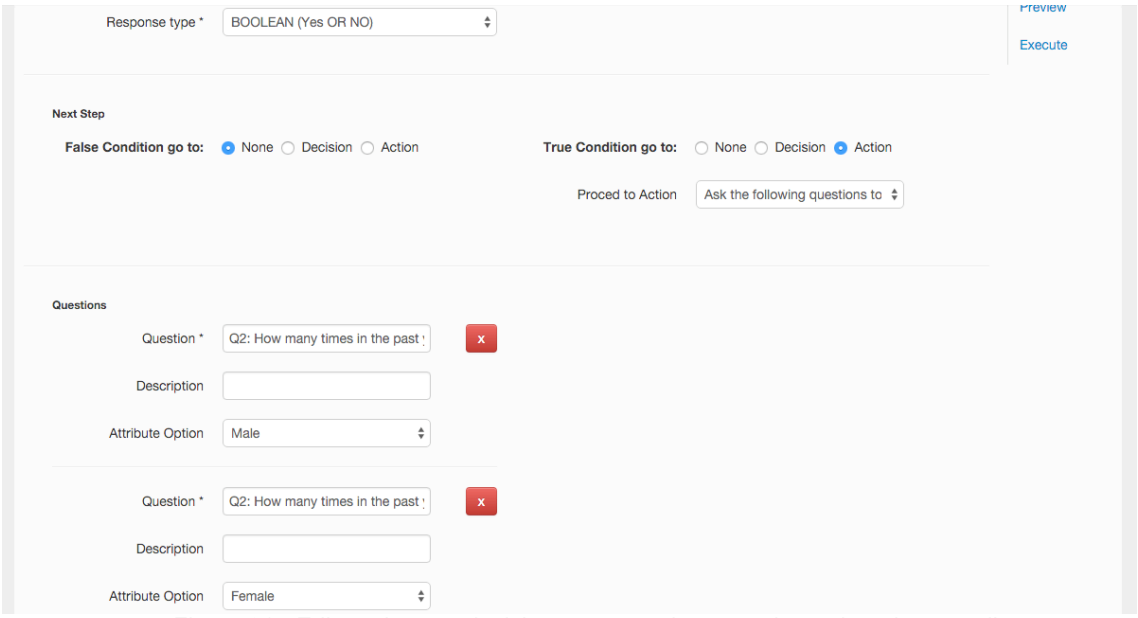

Figura 21 - Edição de uma *decision* com *questions* que dependem de um atributo

Como este modelo de representação funciona como um fluxograma a percepção das transições entre tarefas tornava-se difícil. Por isso foi criada uma funcionalidade que permite a pré-visualização das transições do GC, assim o utilizador que está a criar ou a editar o GC consegue obter ter uma perceção mais fácil do *Workflow* que o GC segue.

Esta pré-visualização mostra a execução do GC do inicio até ao fim agrupando as tarefas por *steps*.

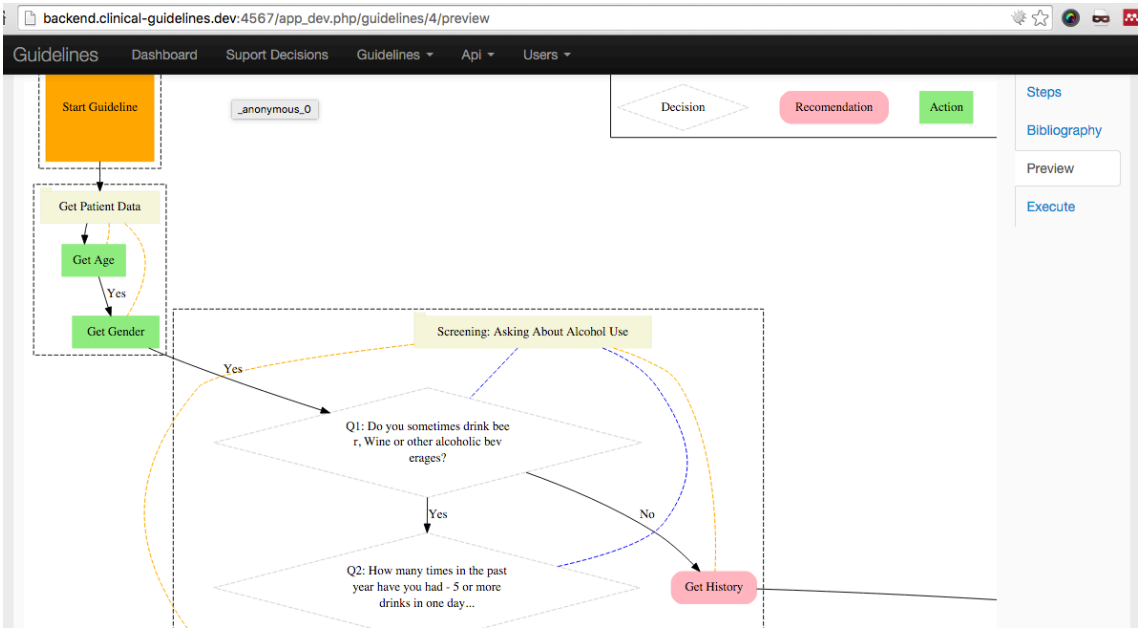

Figura 22 - Pré-visualização do *Worklfow* de um GC

Outra funcionalidade da aplicação é o protótipo de geração de um pdf do GC. Pela figura que se segue podemos verificar que o pdf não está formatado mas que apresenta todos os dados sobre o GC.

Figura 23 - Protótipo da geração do Pdf

Execução do GC: Todos os GCs podem ser executados. A interrupção e retoma posterior de execução da execução não é possível nesta versão. O primeiro passo da execução é a escolha do paciente. A lista dos pacientes é obtida através do OpenEMR. Na figura seguinte podemos ver um exemplo da lista obtida.

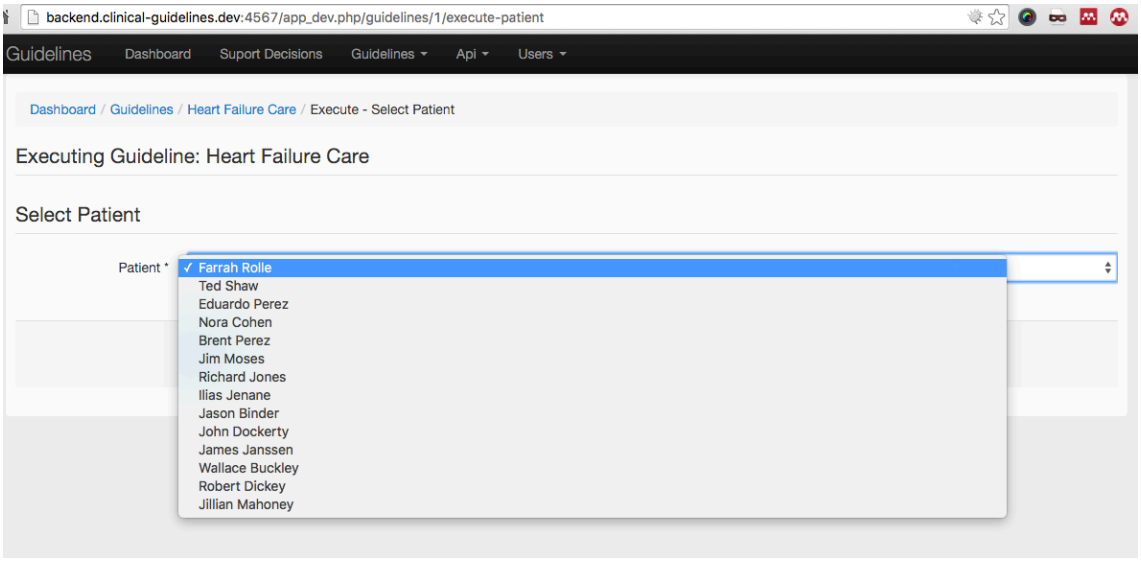

Figura 24 - Seleção do paciente

Após a seleção do paciente e durante a execução do GC caso seja necessário algum dado do paciente que possa ser disponibilizado o seu valor é preenchido automaticamente. Na figura seguinte podemos ver uma *action* que pretende obter o género o paciente, e que o valor do género do paciente foi automaticamente pré-selecionado com o valor correspondente.

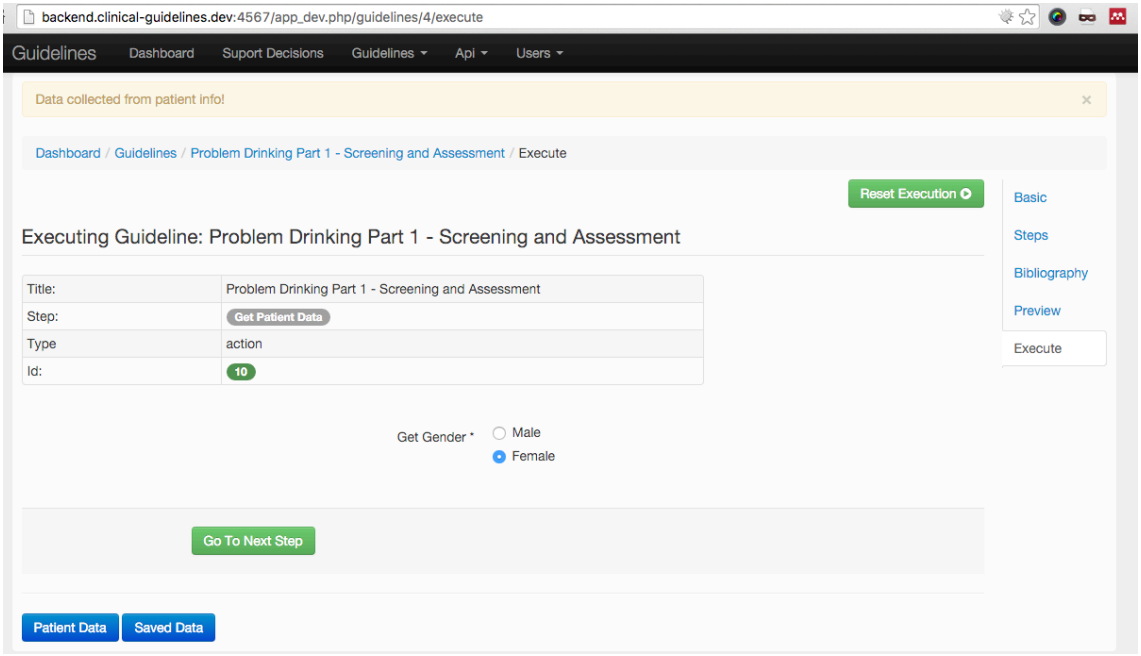

Figura 25 - Recolha de dados do paciente

Outros exemplos de *actions* é por exemplo a obtenção de uma média e uma recomendação. Na figura seguinte podemos ver uma *action* que tem como objetivo calcular a média semanal do número de bebidas alcoólicas que o paciente bebe. Na Figura 27 podemos um exemplo de uma recomendação (*medically_oriented*).

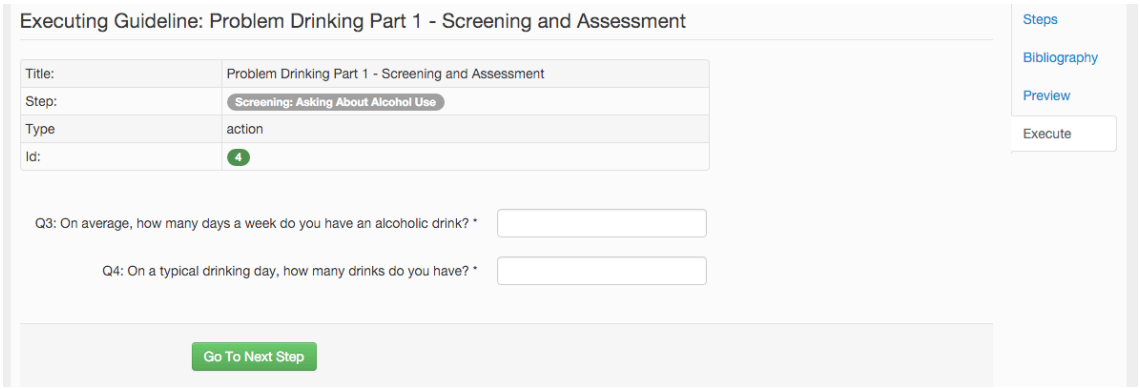

Figura 26 - *Action* para obter uma média

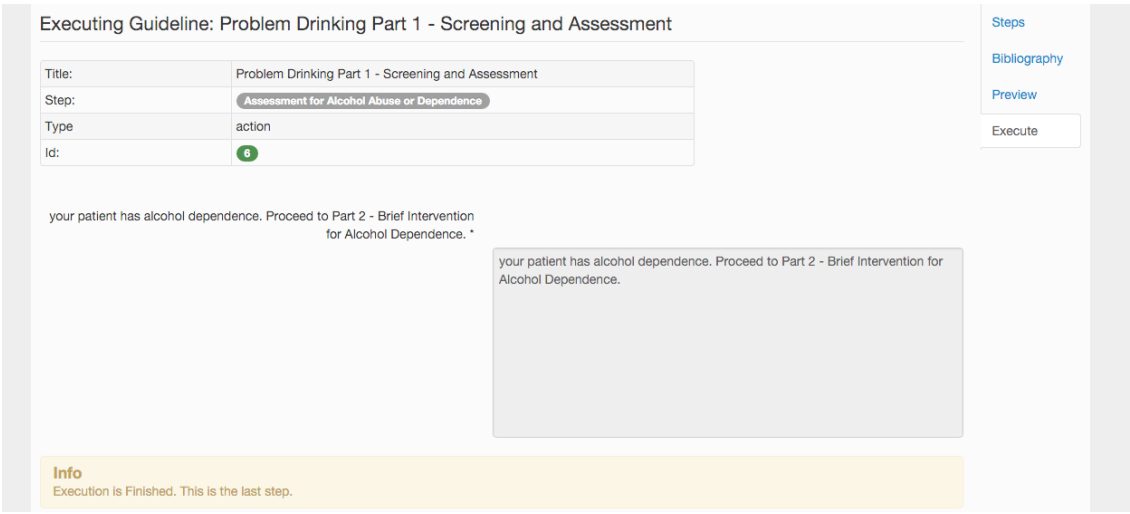

Figura 27 - Exemplo de uma recomendação

A figura seguinte mostra um exemplo de uma *decision* que tem uma condição. A condição tem um o operador de comparação maior ou igual e para que esta seja verdadeira tem que ter pelo menos uma resposta verdadeira.

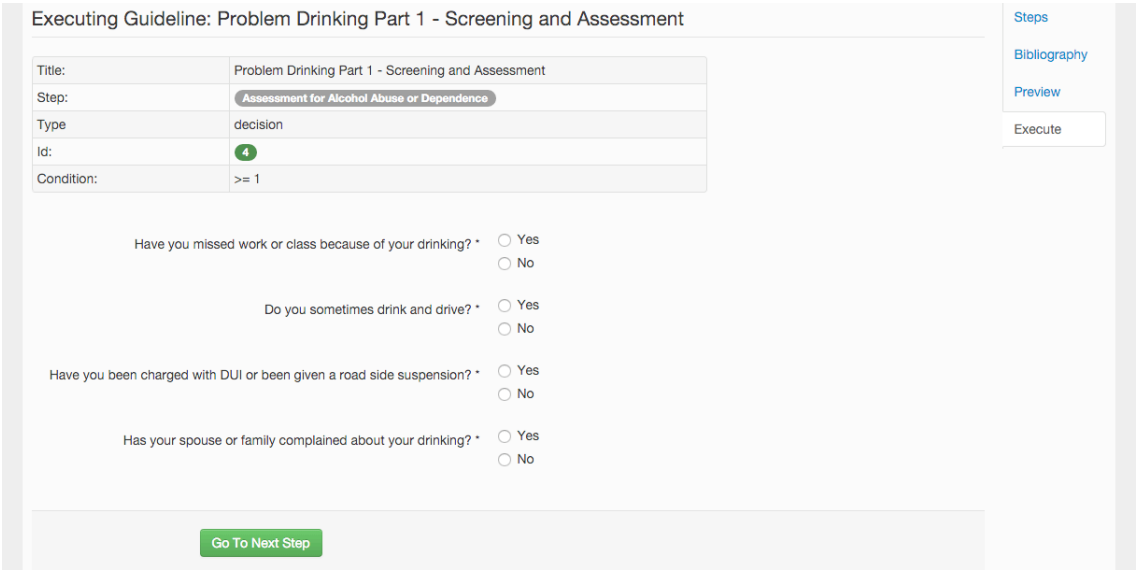

Figura 28 - Exemplo de uma *decision* durante a execução de um GC

8. Conclusões

Os GCs são de extrema importância como ferramenta informática no apoio à prestação de cuidados de saúde. Mas quando estão disponíveis apenas em papel aumenta a sua dificuldade de leitura e de interpretação. Estes como ferramenta devem de estar disponíveis a qualquer momento e em qualquer lugar e serem de fácil utilização para dar suporte no apoio à tomada de decisão por parte dos prestadores de cuidados. Mas para isso é necessário que a informação no GC esteja devidamente correta de forma a evitar maus procedimentos que podem mesmo levar à morte do paciente. Assim os GCs devem de ser baseados em evidências científicas de forma a garantir a qualidade da informação do GC.

Surge assim uma necessidade de informatizar os CGs em papel para um sistema informático que vai permitir uma disseminação do conhecimento, facilitando a leitura e o acesso aos prestadores de cuidados, aumento a qualidade da prática médica. Assim existe a necessidade da criação de CICs.

Para isso foram desenvolvidas varias abordagens de GCs umas com mais vantagens e desvantagens que outras mas nenhuma perfeita. Sendo que GLIF se destaca das outras pela capacidade de representação dos dados, fazendo uma abordagem em tarefas e funcionando como um fluxograma.

Dada a diversidade de linguagens, formas de representação e projetos que foram surgindo ao longo do tempo não parece ser viável a implementação de um único sistema que consiga abranger todas as vantagens no encontro de uma solução perfeita. No entanto sendo a internet um dos maiores meios de comunicação que permite a disseminação do conhecimento, torna-se essencial o desenvolvimento de sistemas de apoio à decisão que possam ser acedidos em qualquer lugar a qualquer hora. Assim projetos com o PRESGUID, e-GuidesMed e esta aplicação que têm como intuito a Web, são uma mais-valia para os prestadores de cuidados.

O principal foco deste projeto é ser direcionado para a Web juntamente com o desenvolvimento desta aplicação que é uma plataforma online e que pode ser acedida através de um simples navegador *Web,* é um bom protótipo que pode cumprir o objetivo de permitir ao prestador de cuidados obter apoio sobre qual a melhor decisão a tomar. Como é

um plataforma online e centralizada tem a vantagem de poder ser facilmente disseminada. Apesar do conceito Web estar evidenciado nos projetos PRESGUID e e-GuidesMed a tecnologia usada e as abordagens do modelo de representação são diferentes.

9. Trabalho Futuro

Apesar dos objetivos deste trabalho terem sido atingidos ainda existe muito trabalho futuro que pode ser feito, de forma a melhorar a aplicação, tornando a sua utilização mais fácil, mais rápida em termos de acesso e muito prática em casos de emergência médica.

No processo de criação pode-se melhorar o sistema de pré-visualização do sistema do fluxograma para permitir uma interação mais dinâmica, através da permissão da criação de tarefas como *actions* e *decisions*, e ainda permitindo a configuração de *transactions* através do arrastamento e possibilidade da criação das ligações entre tarefas. Tornando assim o processo de criação de GC mais fácil. Para agilizar o processo de consulta do GC o desenvolvimento de uma aplicação mobile (sistema operativo Android e sistema operativo iOS) é ideal para os prestadores de cuidados de emergência medica. Assim dos objetivos propostos para este trabalho o que diz respeito ao protótipo de criação de um GCW, apesar de ter sido atingido é um processo que pode ser melhorado.

Em relação ao objetivo do protótipo de execução este também pode ser melhorado. Uma funcionalidade que pode ser implementada, é o versionamento do GC e a gravação dos dados de execução do GC na base de dados, o que iria permitir fazer uma pausa e um resumo da execução do GC a partir do ponto onde se parou. O que iria permitir que as restrições temporais mais complexas de um GC fossem devidamente representas. Assim, objetivo do modelo de representação também poderia ser melhorado no diz respeito a restrições temporais. Outra possível melhoria é a atualização dos dados do paciente após a execução do GC. Por exemplo se for diagnosticado que o paciente tem pré-diabetes o sistema deveria de atualizar a informação do registo médico eletrónico.

10. Bibliografia

[1] A. Hommersom, P. Lucas, and M. Balser, “Meta-level Verification of the Quality of Medical Guidelines Using Interactive Theorem Proving,” 2004, pp. 654–666.

[2] C. Practice, “Guidelines.”

[3] W. Lim, D. M. Arnold, V. Bachanova, R. L. Haspel, R. P. Rosovsky, A. R. Shustov, and M. A. Crowther, “Evidence-based guidelines--an introduction.,” *Hematology Am. Soc. Hematol. Educ. Program*, pp. 26–30, 2008.

[4] “Action Research.” [Online]. Available: https://www.sagepub.com/sites/default/files/upm-binaries/36584_01_Koshy_et_al_Ch_01.pdf. [Accessed: 13-Oct-2015].

[5] M. J. Field and K. N. Lohr, “Clinical Practice Guidelines: Directions for a New Program,” *Institute of Medicine Committee to Advise the Public Health Service on Clinical Practice Guidelines*, vol. Washington. National Academies Press, p. 168, Jan-1990.

[6] J. S. Harris, L. S. Glass, C. Ossler, and P. Low, “Evidence-based design: the ACOEM Practice Guidelines Dissemination Project.,” *J. Occup. Environ. Med.*, vol. 42, no. 4, pp. 352–361, Apr. 2000.

[7] National Health and Medical Research Council (NHMRC), “Guidelines for the development and implementation of clinical practice guidelines,” National Health and Medical Research Council (Australia). Quality of Care and Health Outcomes Committee A.G.P.S., Canberra, 1995.

[8] A. T. Koutsavlis, Régie régionale de la santé et des services sociaux de Montréal-Centre, and Institut national de santé publique du Québec, *Disseminating practice guidelines to physicians*. Montréal: Régie régionale de la santé et des services sociaux de Montréal-Centre : Institut national de santé publique du Québec, 2001.

[9] T. Bodenheimer, “The American health care system--the movement for improved quality in health care,” *N. Engl. J. Med.*, vol. 340, no. 6, pp. 488–492, Feb. 1999.

[10] E. A. McGlynn, S. M. Asch, J. Adams, J. Keesey, J. Hicks, A. DeCristofaro, and E. A. Kerr, “The quality of health care delivered to adults in the United States.,” *N. Engl. J. Med.*, vol. 348, no. 26, pp. 2635–2645, Jun. 2003.

[11] M. Schuster, E. McGlynn, and R. Brook, “How good is the quality of health care in the United States?,” *Milbank Q.*, vol. 76, no. 4, pp. 517–563, 509, 2005.

[12] R. Grol, “Successes and failures in the implementation of evidence-based guidelines for clinical practice.,” *Med. Care*, vol. 39, no. 8 Suppl 2, pp. II46–I54, Aug. 2001.

[13] L. Lima, P. Novais, R. Costa, J. Cruz, and J. Neves, “Decision Making Based on Quality-of-Information a Clinical Guideline for Chronic Obstructive Pulmonary Disease Scenario,” *Distrib. Comput. …*, pp. 417–424, Jan. 2010.

[14] I. Colombet, A. R. Aguirre-Junco, S. Zunino, M. C. Jaulent, L. Leneveut, and G. Chatellier, “Electronic implementation of guidelines in the EsPeR system: A knowledge specification method,” *Int. J. Med. Inform.*, vol. 74, no. 7–8, pp. 597–604, 2005.

[15] “Benefits of Clinical Practice Guidelines.” [Online]. Available: http://www.emergencymedicalparamedic.com/benefits-of-clinical-practice-guidelines/.

[16] A. T. Koutsavlis, “Disseminating Practice Guidelines to Physicians,” Régie régionale de la santé et des services sociaux de Montréal-Centre : Institut national de santé

publique du Québec, Montréal, 2001.

[17] “Knowledge Base : Clinical practice guidelines.” [Online]. Available: http://ktclearinghouse.ca/knowledgebase/knowledgetoaction/action/interventions/strategies/clinicalpracticeguidelines.

[18] K. A. Monsen, C. Neely, G. Oftedahl, M. J. Kerr, P. Pietruszewski, and O. Farri, “Feasibility of encoding the Institute for Clinical Systems Improvement Depression Guideline using the Omaha System,” *J. Biomed. Inform.*, vol. 45, no. 4, pp. 719–725, Aug. 2012.

[19] “Clinical Practice Guidelines.” [Online]. Available: http://www.openclinical.org/guidelines.html.

[20] Y. Shahar, “Automated Support to Clinical Guidelines and Care Plans: The Intention-Oriented View,” *World Wide Web Internet Web Inf. Syst.*, pp. 1–6, 2002.

[21] “A implementação de guias clínicas: como pode ajudarnos a sua informatização?” [Online]. Available: http://apps.elsevier.es/watermark/ctl_servlet?_f=10&pident_articulo=90156994&pident_usuario=0&pcontactid=&pident_revista=37&ty=151&accion=L&origen=zonadelectura&web=zl.elsevier.es&lan=es&fichero=37v77n05a90156994pdf001.pdf.

[22] D. Isern, D. Sánchez, and A. Moreno, “Ontology-driven execution of clinical guidelines,” *Comput. Methods Programs Biomed.*, vol. 107, no. 2, pp. 122–139, Aug. 2012.

[23] K. S. Thaung, *Towards an Effective Knowledge Translation of Clinical Guidelines and Complementary Information*. 2012.

[24] D. Isern and A. Moreno, “Computer-based execution of clinical guidelines: a review,” *Int. J. Med. Inform.*, vol. 77, no. 12, pp. 787–808, 2008.

[25] P. De Clercq, K. Kaiser, and A. Hasman, “Computer-interpretable guideline formalisms,” *Stud. Heal. Technol. …*, vol. 139, pp. 22–43, 2008.

[26] R. Goud, A. Hasman, and N. Peek, “Development of a guideline-based decision support system with explanation facilities for outpatient therapy,” *Comput. Methods Programs Biomed.*, vol. 91, no. 2, pp. 145–153, 2008.

[27] Y. Shahar, O. Young, E. Shalom, M. Galperin, A. Mayaffit, R. Moskovitch, and A. Hessing, “A framework for a distributed, hybrid, multiple-ontology clinical-guideline library, and automated guideline-support tools.,” *J. Biomed. Inform.*, vol. 37, no. 5, pp. 325–44, Oct. 2004.

[28] M. Peleg, S. Tu, J. Bury, and P. Ciccarese, “Comparing computer-interpretable guideline models: a case-study approach,” *J. …*, vol. 10, no. 1, pp. 52–68, 2003.

[29] “Representation Primitives, Process Models and Patient Data in Computer-Interpretable Clinical Practice Guidelines: A Literature Review of Guideline Representation Models.” [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.18.4004&rep=rep1&type=pdf. [Accessed: 28-Oct-2015].

[30] “Approaches for creating computer-interpretable guidelines that facilitate decision support.” [Online]. Available: http://www.mums.ac.ir/shares/hisws/hisws/ha5.pdf.

[31] “OpenClinical: Prestige.” [Online]. Available: http://www.openclinical.org/gmm_prestige.html. [Accessed: 15-Oct-2015].

[32] “OpenClinical: HELEN.” [Online]. Available: http://www.openclinical.org/gmm_helen.html. [Accessed: 15-Oct-2015].

[33] “OpenClinical: HGML - Hypertext Guideline Markup Language.” [Online]. Available: http://www.openclinical.org/gmm_hgml.html. [Accessed: 15-Oct-2015].

[34] “OpenClinical: Stepper.” [Online]. Available: http://www.openclinical.org/gmm_stepper.html. [Accessed: 15-Oct-2015].

[35] “OpenClinical: SEBASTIAN.” [Online]. Available: http://www.openclinical.org/gmm_sebastian.html. [Accessed: 15-Oct-2015].

[36] “Arezzo ® Technical White Paper,” pp. 1–12, 2012.

[37] D. Wang, M. Peleg, and S. Tu, “Design and implementation of the GLIF3 guideline execution engine,” *J. Biomed. …*, vol. 37, no. 5, pp. 305–318, 2004.

[38] “OpenClinical: Arden Syntax.” [Online]. Available: http://www.openclinical.org/gmm_ardensyntax.html.

[39] S. Kim, P. J. Haug, R. A. Rocha, and I. Choi, “Modeling the Arden Syntax for medical decisions in XML,” *Int. J. Med. Inform.*, vol. 77, no. 10, pp. 650–656, 2008.

[40] “OpenClinical: Asbru.” [Online]. Available: http://www.openclinical.org/gmm_asbru.html.

[41] O. Young, Y. Shahar, Y. Liel, E. Lunenfeld, G. Bar, E. Shalom, S. B. Martins, L. T. Vaszar, T. Marom, and M. K. Goldstein, “Runtime application of Hybrid-Asbru clinical guidelines,” *J. Biomed. Inform.*, vol. 40, no. 5, pp. 507–526, Oct. 2007.

[42] “AsbruView.” [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.2694&rep=rep1&type=pdf.

[43] “OpenClinical: GLIF.” [Online]. Available: http://www.openclinical.org/gmm_glif.html.

[44] A. A. Boxwala, M. Peleg, S. Tu, O. Ogunyemi, Q. T. Zeng, D. Wang, V. L. Patel, R. A. Greenes, and E. H. Shortliffe, “GLIF3: a representation format for sharable computer-interpretable clinical practice guidelines,” *J. Biomed. Inform.*, vol. 37, no. 3, pp. 147–161, Jun. 2004.

[45] “OpenClinical: PROforma.” [Online]. Available: http://www.openclinical.org/gmm_proforma.html.

[46] M. A. Grando, D. Glasspool, and J. Fox, “A formal approach to the analysis of clinical computer-interpretable guideline modeling languages,” *Artif. Intell. Med.*, vol. 54, no. 1, pp. 1–13, Jan. 2012.

[47] “Representation of Clinical Practice Guidelines For Computer-Based Implementations,” 2001. [Online]. Available: http://mis.haifa.ac.il/~morpeleg/pubs/Dongwen_IMIA2001final.pdf. [Accessed: 28-Oct-2015].

[48] J. Dufour, D. Fieschi, and M. Fieschi, “BMC Medical Informatics and Decision Making through a web-based system – The PRESGUID project,” vol. 12, pp. 1–12, 2004.

[49] “OpenClinical: DeGeL.” [Online]. Available: http://www.openclinical.org/gmm_degel.html. [Accessed: 02-Sep-2015].

[50] “DeGeL_AIME03.”

[51] R. Barrena, J. M. Pikatza, A. Iruetaguena, and U. Segundo, “e-GuidesMed : Support for translation of clinical guidelines,” pp. 1–2, 2011.

[52] “Lista EMR and EHR.” [Online]. Available: http://www.emrandhipaa.com/emr-and-ehr-vendors/. [Accessed: 01-Sep-2015].

[53] D. De Estado, “RSE – Registo de Saúde Eletrónico,” 2009.

[54] S. Corregedor, “Prontuário Eletrônico em MT.”

[55] “OpenEMR Wiki Home Page - OpenEMR Project Wiki.” [Online]. Available: http://open-emr.org/wiki/index.php/OpenEMR_Wiki_Home_Page. [Accessed: 29-Aug-2015].

[56] “File:Clinical Decision Rules Manual.pdf - OpenEMR Project Wiki.” [Online]. Available: http://www.open-emr.org/wiki/index.php/File:Clinical_Decision_Rules_Manual.pdf. [Accessed: 20-Oct-2015].

[57] “opememr-api.” [Online]. Available: https://github.com/oemr501c3/openemr-api.

[58] “OpenEMR Features - OpenEMR Project Wiki.” [Online]. Available: http://open-emr.org/wiki/index.php/OpenEMR_Features#Electronic_Medical_Records. [Accessed: 29-Aug-2015].

[59] J.-C. Dufour, D. Fieschi, and M. Fieschi, “Coupling computer-interpretable guidelines with a drug-database through a web-based system--The PRESGUID project.,” *BMC Med. Inform. Decis. Mak.*, vol. 4, no. 1, p. 2, Mar. 2004.

[60] “Protégé.” [Online]. Available: http://protege.stanford.edu/. [Accessed: 07-Nov-2015].

11. Anexos

A apresentação do código desenvolvido da aplicação Web não foi considerado relevante para o entendimento do funcionamento do modelo de representação.

11.1 Entidades

As entidades representam as tarefas do modelo de representação. Aqui em anexo apenas vão ser representadas as entidades que permitem obter uma melhor percepção do motor de execução.

11.1.1 Entidade *Transaction*

```
<?php

namespace CGs\Bundle\GuidelinesBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

use Doctrine\Common\Collections\ArrayCollection;

use Gedmo\Mapping\Annotation as Gedmo;

/**

 * @ORM\Table(name="transaction")

 *

@ORM\Entity(repositoryClass="CGs\Bundle\GuidelinesBundle\Repository\TransactionRepo
sitory")

 * @ORM\HasLifecycleCallbacks()

 */

class Transaction {

    /**

     * @ORM\Column(type="integer")
```

```
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;
    /**
    *          @ORM\OneToOne(targetEntity="Decision",          inversedBy="transaction",
cascade={"persist","remove"})
     * @ORM\JoinColumns({
    *          @ORM\JoinColumn(name="from_decision_id",    referencedColumnName="id",
nullable=true, onDelete="CASCADE")
     * })
     */
    private $fromDecision;
    /**
    *          @ORM\ManyToOne(targetEntity="Action",          inversedBy="fromTransactions",
cascade={"persist","remove"})
    *          @ORM\JoinColumn(name="from_action_id",          referencedColumnName="id",
nullable=true)
     */
    private $fromAction;
    /**
    *       @ORM\ManyToOne(targetEntity="Decision",       inversedBy="toTransactionsTrue",
cascade={"persist","remove"})
    *       @ORM\JoinColumn(name="to_decision_true_id",       referencedColumnName="id",
nullable=true)
     */
    private $toDecisionTrue;
    /**
    *       @ORM\ManyToOne(targetEntity="Decision",       inversedBy="toTransactionsFalse",
cascade={"persist"})
```

```
     *      @ORM\JoinColumn(name="to_decision_false_id",      referencedColumnName="id",
nullable=true)
     */
    private $toDecisionFalse;
    /**
     *      @ORM\ManyToOne(targetEntity="Action",      inversedBy="toTransactionsTrue",
cascade={"persist"})
     *      @ORM\JoinColumn(name="to_action_true_id",      referencedColumnName="id",
nullable=true)
     */
    private $toActionTrue;
    /**
     *      @ORM\ManyToOne(targetEntity="Action",      inversedBy="toTransactionsFalse",
cascade={"persist"})
     *      @ORM\JoinColumn(name="to_action_false_id",      referencedColumnName="id",
nullable=true)
     */
    private $toActionFalse;
    public function getId() { return $this->id; }
    public function setId($id) { $this->id = $id; return $this; }
    public function getFromDecision() { return $this->fromDecision; }
    public function getFromAction() { return $this->fromAction; }
    public function getToDecisionTrue() { return $this->toDecisionTrue; }
    public function getToActionTrue() { return $this->toActionTrue; }
    public function getToDecisionFalse() { return $this->toDecisionFalse; }
    public function getToActionFalse() { return $this->toActionFalse; }
    public function setFromDecision($fromDecision) { $this->fromDecision = $fromDecision;
return $this; }
    public function setFromAction($fromAction) {  $this->fromAction = $fromAction;  return
$this; }
```

```
public function setToDecisionTrue($toDecisionTrue) { $this->toDecisionTrue = $toDecisionTrue; return $this; }

public function setToActionTrue($toActionTrue) { $this->toActionTrue = $toActionTrue; return $this; }

public function setToDecisionFalse($toDecisionFalse) { $this->toDecisionFalse = $toDecisionFalse;  return $this;  }

   public function setToActionFalse($toActionFalse) { $this->toActionFalse = $toActionFalse; return $this; }

}
```

11.1.2 Entidade *Decision*

```
<?php

namespace CGs\Bundle\GuidelinesBundle\Entity;

use Gedmo\Mapping\Annotation as Gedmo;

use Doctrine\ORM\Mapping as ORM;

use Doctrine\Common\Collections\ArrayCollection;

use CGs\Bundle\GuidelinesBundle\Entity\Restriction;

/**

 * @ORM\Table(name="decision")

 *

@ORM\Entity(repositoryClass="CGs\Bundle\GuidelinesBundle\Repository\DecisionRepository")

 */

class Decision {

   const TYPE_STRING = 'string';

   const TYPE_INT = 'number';

   const TYPE_CHOICE = 'choice';

   const TYPE_MULTIPLE_CHOICE = 'multiple_choice';

   const TYPE_BOOLEAN = 'boolean';

   const TYPE_BOOLEAN_GROUP = 'boolean_group';
```

```
/**
 * @ORM\Column(type="integer")
 * @ORM\Id
 * @ORM\GeneratedValue(strategy="AUTO")
 */
private $id;
/**
 * @ORM\Column(type="string", length=24)
 */
private $type;
/**
 * @ORM\Column(name="text", type="text", nullable=true)
 */
private $text;
/**
 * @var boolean $isFirst
 *
 * @ORM\Column(name="is_first", type="boolean", nullable=true)
 */
private $isFirst;
/**
 * @ORM\Column(name="comparison_operator", type="string", length=25, nullable=true)
 */
private $comparisonOperator;
/**
 * @ORM\Column(name="condition_to_be_true", type="string", length=255, nullable=true)
 */
private $conditionTobeTrue;
```

```
/**
 * @ORM\ManyToOne(targetEntity="Attribute", inversedBy="decisions")
 */
protected $attribute;
/**
 * @ORM\OneToMany(
 *      targetEntity="Question",
 *      mappedBy="decision",
 *      cascade={"persist"},
 *      orphanRemoval=true
 * )
 */
protected $questions;
/**
 * @var string $createdBy
 * @Gedmo\Blameable(on="create")
 * @ORM\ManyToOne(targetEntity="Sols\Bundle\UserBundle\Entity\User")
 * @ORM\JoinColumn(name="created_by", referencedColumnName="id")
 */
private $createdBy;
/**
 * @var User $updatedBy
 * @Gedmo\Blameable(on="update")
 * @ORM\ManyToOne(targetEntity="Sols\Bundle\UserBundle\Entity\User")
 * @ORM\JoinColumn(name="updated_by", referencedColumnName="id")
 */
private $updatedBy;
/**
```

```
     * @var \DateTime
     * @Gedmo\Timestampable(on="create")
     * @ORM\Column(name="created_at", type="datetime", nullable=false)
     */
    private $createdAt;
    /**
     * @var \DateTime
     * @Gedmo\Timestampable(on="update")
     * @ORM\Column(name="updated_at", type="datetime", nullable=false)
     */
    private $updatedAt;
    /**
     * @ORM\ManyToOne(targetEntity="Step", inversedBy="decisions", cascade={"persist"})
     * @ORM\JoinColumn(name="step_id", referencedColumnName="id", nullable=false)
     */
    protected $step;
    /**
     *          @ORM\OneToOne(targetEntity="Transaction",          mappedBy="fromDecision",
cascade={"persist"})
     */
    protected $transaction;
    /**
     *        @ORM\OneToMany(targetEntity="Transaction",        mappedBy="toDecisionTrue",
cascade={"persist"})
     */
    protected $toTransactionsTrue;
    /**
     *       @ORM\OneToMany(targetEntity="Transaction",       mappedBy="toDecisionFalse",
cascade={"persist"})
```

```
     */
    protected $toTransactionsFalse;
    public function __construct() {
        $this->questions = new ArrayCollection();
    }
    public function isFirst() { return $this->isFirst; }
    public function setIsFirst($isFirst) {
        $this->isFirst = $isFirst;
        return $this;
    }
    public function getTransaction() { return $this->transaction; }
    public function setTransaction($transaction) { $transaction->setFromDecision($this); $this->transaction = $transaction; return $this; }
    public function getConditionTobeTrue() { return $this->conditionTobeTrue; }
    public function setConditionTobeTrue($conditionTobeTrue) { $this->conditionTobeTrue = $conditionTobeTrue; return $this; }
    public function setId($id) {  $this->id = $id;  return $this; }
    public function getId() { return $this->id; }
    public function setText($text) { $this->text = $text; return $this; }
    public function getText() {  return $this->text; }
    public function setType($type) {
        $allowedTypes = $this->getTypesAllowed();
        if (!isset($allowedTypes[$type])) {
            throw new \InvalidArgumentException("Invalid type: " . $type);
        }
        $this->type = $type;
}
    public function getType() { return $this->type; }
    public function getStep() { return $this->step; }
```

```
public function setStep($step) { $this->step = $step; return $this; }
public function getAttribute() { return $this->attribute; }
public function setAttribute($attribute) { $this->attribute = $attribute;  return $this;  }
public function getTypesAllowed() {
    return array(
        self::TYPE_STRING => 'STRING',
        self::TYPE_INT => 'INTEGER',
        self::TYPE_CHOICE => 'ONE CHOICE',
        self::TYPE_MULTIPLE_CHOICE => 'MULTIPLE CHOICES',
        self::TYPE_BOOLEAN => 'BOOLEAN (Yes OR NO)',
        self::TYPE_BOOLEAN_GROUP => 'BOOLEAN GROUP (Multiple Yes Or No)'
    );
}
public function getQuestions() { return $this->questions; }
public function setQuestions($questions) { $this->questions = $questions;  return $this; }
public function addQuestion(Question $question) {
    if ($this->questions->contains($question)) {
        return $this;
    }
    $question->setDecision($this);
    $this->questions->add($question);
    return $this;
}
public function removeQuestion(Question $question) {
    if (!$this->questions->contains($question)) {
        return $this;
    }
    $question->setDecision(null);
```

```
        $this->questions->removeElement($question);
        return $this;
    }
    public function getToTransactionsTrue() { return $this->toTransactionsTrue; }
    public function getToTransactionsFalse() { return $this->toTransactionsFalse; }
public function setToTransactionsTrue($toTransactionsTrue) {
        $this->toTransactionsTrue = $toTransactionsTrue;
        return $this;
    }
    public function setToTransactionsFalse($toTransactionsFalse) {
        $this->toTransactionsFalse = $toTransactionsFalse;
        return $this;
    }
    public function __toString() {
        $str = '';
        foreach ($this->questions as $question) {
            $str .= $question->getTitle();
        }
        return $str;
    }
    public function getComparisonOperator() { return $this->comparisonOperator; }
    public function setComparisonOperator($comparisonOperator) {
        $this->comparisonOperator = $comparisonOperator;
        return $this;
    }
    public function getAllComparisonOperatores() {
        return array(
            '==' => '==  Equal',
```

```
            '===' => '=== Identical',
            '!=' => '!=  Not equal',
            '!==' => '!== Not identical',
            '>' => '>   Greater than',
            '>=' => '>=  Greater than or equal to',
            '<' => '<   Less than',
            '<=' => '<=  Less than or equal to',
            '<>' => '<>  Not equal',
        );
    }
}
```

11.1.3 Entidade *Action*

```
<?php
namespace CGs\Bundle\GuidelinesBundle\Entity;
use Doctrine\ORM\Mapping as ORM;
use Doctrine\Common\Collections\ArrayCollection;
use Gedmo\Mapping\Annotation as Gedmo;
/**
 * @ORM\Table(name="action")
 *
@ORM\Entity(repositoryClass="CGs\Bundle\GuidelinesBundle\Repository\ActionRepository"
)
 * @Gedmo\SoftDeleteable(fieldName="deletedAt", timeAware=false)
 * @ORM\HasLifecycleCallbacks()
 */
class Action {
    const TYPE_MEDICALLY_ORIENTED = 'recomendation';
    const TYPE_AVERAGE = 'average';
```

```
const TYPE_INT = 'number';
const TYPE_ATTRIBUTE = 'same_attribute';
/**
 * @ORM\Column(type="integer")
 * @ORM\Id
 * @ORM\GeneratedValue(strategy="AUTO")
 */
private $id;
/**
 * @ORM\Column(type="string", length=255)
 */
private $title;
/**
 * @ORM\Column(type="string", length=24)
 */
private $type;
/**
 * @ORM\Column(type="text", nullable=true)
 */
private $text;
/**
 * @ORM\Column(name="deletedAt", type="datetime", nullable=true)
 */
private $deletedAt;
/**
 * @ORM\OneToMany(
 *      targetEntity="Question",
 *      mappedBy="action",
```

```
     *      cascade={"persist"},
     *      orphanRemoval=true
     * )
     */
    protected $questions;
    /**
     * @ORM\ManyToOne(targetEntity="Attribute", inversedBy="actions")
     */
    protected $attribute;
    /**
     * @var boolean $isFirst
     *
     * @ORM\Column(name="is_first", type="boolean", nullable=true)
     */
    private $isFirst;
    /**
     * @ORM\ManyToOne(targetEntity="Step", inversedBy="actions", cascade={"persist"})
     * @ORM\JoinColumn(name="step_id", referencedColumnName="id", nullable=false)
     */
    protected $step;
    /**
     *          @ORM\OneToOne(targetEntity="Transaction",          mappedBy="fromAction",
cascade={"persist"})
     */
    protected $transaction;
    /**
     *          @ORM\OneToMany(targetEntity="Transaction",          mappedBy="toActionTrue",
cascade={"persist"})
     */
```

```
    protected $toTransactionsTrue;
    /**
     *          @ORM\OneToMany(targetEntity="Transaction",          mappedBy="toActionFalse",
cascade={"persist"})
     */
    protected $toTransactionsFalse;
    /**
     * Constructor
     */
    public function __construct() { $this->questions = new ArrayCollection; }
    public function isFirst() { return $this->isFirst; }
    public function setIsFirst($isFirst) { $this->isFirst = $isFirst;  return $this;  }
    /**
     * Get id
     * @return integer
     */
    public function getId() {  return $this->id; }
    public function setId($id) { $this->id = $id; return $this; }
/**
     * Set title
     * @param \String $title
     * @return Specialty
     */
    public function setTitle($title) { $this->title = $title; return $this; }
    /**
     * Get title
     * @return \String
     */
```

```
    public function getTitle() { return $this->title; }
    /**
     * Set text
     * @param string $text
     * @return Specialty
     */
    public function setText($text) { $this->text = $text;  return $this; }
    /**
     * Get text
     * @return string
     */
    public function getText() { return $this->text; }
/**
     * Add guidelines
     * @param \CGs\Bundle\GuidelinesBundle\Entity\Guideline $guidelines
     * @return Specialty
     */
    public function addGuideline(\CGs\Bundle\GuidelinesBundle\Entity\Guideline $guidelines)
{
        $this->guidelines[] = $guidelines;
        return $this;
    }
    /**
     * Remove guidelines
     * @param \CGs\Bundle\GuidelinesBundle\Entity\Guideline $guidelines
     */
    public function removeGuideline(\CGs\Bundle\GuidelinesBundle\Entity\Guideline
$guidelines) {
        $this->guidelines->removeElement($guidelines);
```

```
    }
    /**
     * Get guidelines
     * @return \Doctrine\Common\Collections\Collection
     */
    public function getGuidelines() { return $this->guidelines; }
    public function getStep() { return $this->step; }
    public function setStep($step) {
        $this->step = $step;
        return $this;
    }
public function getQuestions() { return $this->questions  }
public function setQuestions($questions) { $this->questions = $questions;  return $this;  }
    public function addQuestion(Question $question) {
        if ($this->questions->contains($question)) {
            return $this;
        }
        $question->setAction($this);
        $this->questions->add($question);
        return $this;
    }
    public function removeQuestion(Question $question) {
        if (!$this->questions->contains($question)) { return $this; }
        $question->setAction(null);
        $this->questions->removeElement($question);
        return $this;
    }
    public function getType() { return $this->type; }
```

```
    public function setType($type) { $this->type = $type; return $this; }
    public function getTransaction() { return $this->transaction; }
    public function getToTransactionsTrue() { return $this->toTransactionsTrue;  }
    public function getToTransactionsFalse() { return $this->toTransactionsFalse;  }
    public function setTransaction($transaction) { $transaction->setFromAction($this);  $this->transaction = $transaction;  return $this; }
    public function setToTransactionsTrue($toTransactionsTrue) {  $this->toTransactionsTrue = $toTransactionsTrue;  return $this; }
    public function setToTransactionsFalse($toTransactionsFalse) { $this->toTransactionsFalse = $toTransactionsFalse;  return $this; }
    public function getAttribute() { return $this->attribute; }
    public function setAttribute($attribute) { $this->attribute = $attribute; return $this; }
    public function __toString() { return $this->getTitle(); }
    public function getTypesAllowed() {
        return array(
            self::TYPE_AVERAGE => 'AVERAGE (Calculate avg from multiple Questions)',
            self::TYPE_MEDICALLY_ORIENTED => 'RECOMENDATION',
            self::TYPE_INT => 'NUMBER',
            self::TYPE_ATTRIBUTE => 'Same as attribute'
        );
    }
}
```

11.2 Motor de execução

```
<?php
namespace CGs\Bundle\GuidelinesBundle\Form;
use CGs\Bundle\GuidelinesBundle\Entity\Action;
use CGs\Bundle\GuidelinesBundle\Entity\Decision;
use Symfony\Component\Form\FormBuilderInterface;
```

```
use Symfony\Component\OptionsResolver\OptionsResolverInterface;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\HttpFoundation\Request;
class ExecutionAction extends AbstractType {
    const ACTION = 'action';
    const DECISION = 'decision';
    const IDENTIFIER_ACTION = 'action-';
    const IDENTIFIER_ACTION_RECOMENDATION = 'action-recomendation-';
    const IDENTIFIER_ACTION_QUESTION = 'action-question-';
    const IDENTIFIER_DECISION = 'decision-';
    const IDENTIFIER_DECISION_QUESTION = 'decision-question-';
    private $container = null;
    /**
     * @var \CGs\Bundle\GuidelinesBundle\Entity\Guideline $guideline
     */
    private $guideline = null;
    private $lastExecutedElement = null;
    private $currentElement = null;
    private $nextElement = null;
    public function __construct(array $options) {
        $this->container = $options['container'];
        $this->guideline = $options['guideline'];
    }
    /**
     * Build form.
     * @param FormBuilderInterface $builder
     * @param array $options
     */
```

```
public function buildForm(FormBuilderInterface $builder, array $options) {
    $builder->add('step', 'hidden');
    $element = $this->getCurrentElement();
    if ($element instanceof Decision) {
        $this->buildFormDecisions($builder, $element);
    } elseif ($element instanceof Action) {
        $this->buildFormActions($builder, $element);
    }
}
/**
 * Get patient attribute value.
 * @param type $attributeCode
 * @return type
 * @throws Exception
 */
protected function getPatientAttribute($attributeCode) {
    $patient = $this->container->get('session')->get('execution_patient');
    if (!empty($patient[$attributeCode])) {
        return $patient[$attributeCode];
    }
    throw new Exception("Patient not has attribute");
}
/**
 * Get the current element.
 * @return type
 */
public function getCurrentElement() {
    if ($this->currentElement) {
```

```
            return $this->currentElement;
        }
        $execution = $this->container->get('session')->get('execution');
        if (!empty($execution['next_executed_element'])) {
            $this->currentElement = $this->getElementByTypeAndId(
                $execution['next_executed_element']['type'],
$execution['next_executed_element']['id']
            );
        }
        if (empty($this->currentElement) && !empty($execution['last_executed_element'])) {
            $lastExecutedType = $execution['last_executed_element']['type'];
            if ($lastExecutedType == self::DECISION) {
                $this->lastExecutedElement = $this->getDecisionRepository()-
>find($execution['last_executed_element']['id']);
            } else if ($lastExecutedType == self::ACTION) {
                $this->lastExecutedElement = $this->getActionRepository()-
>find($execution['last_executed_element']['id']);
            }
            if (!empty($this->lastExecutedElement)) {
                $this->currentElement = $this->getNextTransaction($this->lastExecutedElement);
            }
        }
        if ($this->currentElement === null) {
            $first = $this->guideline->getFirstElement();
            $type = $this->getElementType($first);
            $this->currentElement = $this->getElementByTypeAndId($type, $first->getId());
        }
        return $this->currentElement;
    }
```

```
    /** Update current element with next element. */
    public function updateToNextElement() {  $this->currentElement = $this->nextElement; }
    /**
     * Save execution data
     * @param Request $request
     * @return type
     */
    public function save(Request $request) {
        $responses = [];
        /**
         * Verify condition to transaction
         */
        $params = $request->request->all();
        $keys = array();
        if (!empty($params['execution'])) {
            $keys = array_keys($params['execution']);
            $responses['response_params'] = $params = $params['execution'];
        }
        unset($responses['response_params']['_token']);
        unset($responses['response_params']['step']);
        $type = $this->getElementType($this->currentElement);
        if ($type == 'decision') {
            $condition = $this->currentElement->getConditionTobeTrue();
            $keys = $this->getKeysStartedBy($keys, self::IDENTIFIER_DECISION);
            if ($this->currentElement->getType() == 'boolean') {
                $isTrue = (bool) $params[self::IDENTIFIER_DECISION . $this->currentElement-
>getId() . '-value'];
                $this->processActionTransaction($isTrue);
```

```
        } else if ($this->currentElement->getType() == 'boolean_group') {
            $totalTrue = 0;
            foreach ($this->currentElement->getQuestions() as $question) {
                if (!empty($params[self::IDENTIFIER_DECISION_QUESTION . $question->getId() . '-value'])) {
                    $totalTrue++;
                    $responses['true_responses_questions'] = $question->getId();
                }
            }
            $isTrueComp = $this->compare($totalTrue, $this->currentElement->getComparisonOperator(), $condition);
            $this->processActionTransaction($isTrueComp);
        }
    } elseif ($type == 'action') {
        if ($this->currentElement->getType() == 'average') {
            $responses['average'] = $this->processAverage($params);
        }
        $this->nextElement = $this->getNextTransaction($this->currentElement);
    }
    /**
     * Save data to session
     */
    $execution = $this->container->get('session')->get('execution');
    $execution['elements'][] = array_merge(
            array(
        'type' => $this->getElementType($this->currentElement),
        'id' => $this->currentElement->getId()
            ), $responses
    );
```

```
        $execution['last_executed_element']['type'] = $this->getElementType($this->currentElement);
        $execution['last_executed_element']['id'] = $this->currentElement->getId();
        if ($this->nextElement) {
            $execution['next_executed_element']['type'] = $this->getElementType($this->nextElement);
            $execution['next_executed_element']['id'] = $this->nextElement->getId();
        } else {
            $execution['next_executed_element'] = [];
        }
        return $this->container->get('session')->set('execution', $execution);
    }
    /**
     * Check if current element has a next element.
     * @return boolean
     */
    public function hasMoreFlowCurrentElement() {
        return $this->getNextTransaction($this->currentElement);
    }
    /**
     * Sets the default options for this type.
     * @param OptionsResolverInterface $resolver The resolver for the options.
     */
    public function setDefaultOptions(OptionsResolverInterface $resolver) {
        $resolver->setDefaults(array());
    }
    /**
     * Returns the name of this type.
     * @return string The name of this type
```

```
     */
    public function getName() { return 'execution'; }
    /**
     * Get next element from passed element.
     * @param Decision|Action $element
     * @return null|Decision|Action
     */
    protected function getNextTransaction($element) {
        /**
         * @var \CGs\Bundle\GuidelinesBundle\Entity\Transaction $transaction
         */
        $transaction = $element->getTransaction();
        if (!is_object($transaction)) {
            return null;
        }
        //false condition
        if ($element instanceof Decision) {
            if ($transaction->getToActionFalse()) {
                return $this->getActionRepository()->find($transaction->getToActionFalse()->getId());
            } elseif ($transaction->getToDecisionFalse()) {
                return $this->getDecisionRepository()->find($transaction->getToDecisionFalse()->getId());
            }
        }
        //true condition
        if ($transaction->getToActionTrue()) {
            return $this->getActionRepository()->find($transaction->getToActionTrue()->getId());
        } elseif ($transaction->getToDecisionTrue()) {
```

```
            return $this->getDecisionRepository()->find($transaction->getToDecisionTrue()->getId());
        }
        return null;
    }
    /**
     * Build decisions
     * @param FormBuilderInterface $builder
     * @param Decision $decision
     */
    protected function buildFormDecisions(FormBuilderInterface $builder, Decision $decision)
{
        $patientValue = '';
        $type = $decision->getType();
        $attribute = $decision->getAttribute();
        if ($attribute) {
            $attributeCode = $attribute->getCode();
            $patientValue = $this->getPatientAttribute($attributeCode);
        }
        $fieldOptions = array();
        switch ($type):
            case Decision::TYPE_BOOLEAN:
                $fieldOptions['expanded'] = true;
                $fieldOptions['multiple'] = false;
                $fieldOptions['choices'] = array(
                    true => 'Yes',
                    false => 'No',
                );
                foreach ($decision->getQuestions() as $question) {
```

```
                if (empty($question->getAttributeOption())) {
                    $fieldOptions['label'] = $question->getTitle();
                    continue;
                }
                if ($question->getAttributeOption()->getValue() == $patientValue) {
                    $fieldOptions['label'] = $question->getTitle();
                }
            }
            $builder->add(self::IDENTIFIER_DECISION . $decision->getId() . '-value', 'choice', $fieldOptions);
            break;
        case Decision::TYPE_BOOLEAN_GROUP:
            $fieldOptions['expanded'] = true;
            $fieldOptions['multiple'] = false;
            $fieldOptions['choices'] = array(
                true => 'Yes',
                false => 'No'
            );
            foreach ($decision->getQuestions() as $question) {
                $fieldOptions['label'] = $question->getTitle();
                $builder->add(self::IDENTIFIER_DECISION_QUESTION . $question->getId() . '-value', 'choice', $fieldOptions);
            }
            break;
        case Decision::TYPE_CHOICE:
            $fieldOptions = array();
            $fieldOptions['expanded'] = true;
            $fieldOptions['multiple'] = false;
            foreach ($decision->getQuestions() as $question) {
```

```
                $fieldOptions['choices'][$question->getId()] = $question->getTitle();
            }
            $builder->add(self::IDENTIFIER_DECISION . $decision->getId() . '-value', 'choice',
$fieldOptions);
            break;
        case Decision::TYPE_MULTIPLE_CHOICE:
            $fieldOptions = array();
            $fieldOptions['expanded'] = true;
            $fieldOptions['multiple'] = false;
            foreach ($decision->getQuestions() as $question) {
                $fieldOptions['choices'][$question->getId()] = $question->getTitle();
            }
            $builder->add(self::IDENTIFIER_DECISION . $decision->getId() . '-value', 'choice',
$fieldOptions);
            break;
    endswitch;
}

/**
 * Build Actions
 * @param FormBuilderInterface $builder
 * @param Action $action
 */
protected function buildFormActions(FormBuilderInterface $builder, Action $action) {
    $patientValue = '';
    $type = $action->getType();
    $attribute = $action->getAttribute();
    if (is_object($attribute)) {
        $patientValue = $this->getPatientAttribute(strtolower($attribute->getCode()));
```

```
        }
        if (!empty($patientValue)) {
            $this->container
                ->get('session')
                ->getFlashBag()
                ->add('info-static-patient', 'Data collected from patient info!');
        }
        $fieldOptions = array();
        if ($attribute) {
            $fieldOptions['empty_data'] = $patientValue;
            $fieldOptions['data'] = $patientValue;
        }
        $questions = $action->getQuestions();
        if (!count($questions)) {
            $questions = array();
        }
        if ($type == 'same_attribute') {
            $type = $attribute->getType();
        }
        switch (true):
            case $type == 'radio' && $action->getType() == 'same_attribute' &&
empty($questions) && $attribute:
                $choices = [];
                foreach ($attribute->getAttributeOptions() as $option) {
                    $choices[$option->getValue()] = $option->getTitle();
                }
                $fieldOptions['label'] = $action->getTitle();
                $fieldOptions['choices'] = $choices;
```

```
            $fieldOptions['multiple'] = false;
            $fieldOptions['expanded'] = true;
            $builder->add(self::IDENTIFIER_ACTION . $action->getId() . '-value', 'choice', $fieldOptions);
            break;
        case $type == 'number' && !empty($questions):
            foreach ($questions as $question) {
                $fieldOptions['label'] = $question->getTitle();
                $builder->add(self::IDENTIFIER_ACTION_QUESTION . $question->getId() . '-value', 'number', $fieldOptions);
            }
            break;
        case $type == 'number' && empty($questions):
            $fieldOptions['label'] = $action->getTitle();
            $builder->add(self::IDENTIFIER_ACTION . $action->getId() . '-value', 'number', $fieldOptions);
            break;
        case $type == Action::TYPE_MEDICALLY_ORIENTED:
            $this->buildRecomendation($builder, $action);
            break;
        case $type == 'average':
            foreach ($questions as $question) {
                $fieldOptions = array();
                $fieldOptions['label'] = $question->getTitle();
                $builder->add(self::IDENTIFIER_ACTION_QUESTION . $question->getId() . '-value', 'number', $fieldOptions);
            }
            break;
    endswitch;
```

```
    }
    /**
     * Build recomendation
     * @param FormBuilderInterface $builder
     * @param Action $action
     */
    protected function buildRecomendation(FormBuilderInterface $builder, Action $action) {
        $builder->add(self::IDENTIFIER_ACTION_RECOMENDATION . $action->getId() . '-
value', 'textarea', array(
            'label' => $action->getTitle(),
            'empty_data' => $action->getText(),
            'data' => $action->getText(),
            'read_only' => true,
            'attr' => array(
                'rows' => '10',
                'read_only' => true,
                'class' => 'textarea-wysihtml5-prevew input-block-level'
            )
        ));
    }
    /**
     * Get Decision repository
     * @return \CGs\Bundle\GuidelinesBundle\Repository\DecisionRepository
     */
    protected function getDecisionRepository() {
        return $this->container->get('doctrine')->getManager()
                    ->getRepository('CGsGuidelinesBundle:Decision');
    }
```

```
    /**
     * Get Action repository
     * @return \CGs\Bundle\GuidelinesBundle\Repository\ActionRepository
     */
    protected function getActionRepository() {
        return $this->container->get('doctrine')->getManager()
->getRepository('CGsGuidelinesBundle:Action');
    }
    /**
     * Get type of element
     * @param type $ele
     * @return type
     */
    protected function getElementType($ele) {
        $classParts = explode('\\', get_class($ele));
        return strtolower($classParts[count($classParts) - 1]);
    }
    /**
     * Get keys from array started by passed string
     * @param array $array
     * @param string $start
     * @return type
     */
    protected function getKeysStartedBy(array $array, $start) {
        $keys = [];
        foreach ($array as $key => $row) {
            if ($this->startsWith($start, $key)) {
                $keys[] = $key;
```

```
        }
    }
    return $keys;
}
/**
 * Check if string start by another string
 * @param type $haystack
 * @param type $needle
 * @return type
 */
protected function startsWith($haystack, $needle) {
    // search backwards starting from haystack length characters from the end
    return $needle === "" || strrpos($haystack, $needle, -strlen($haystack)) !== false;
}
/**
 * Get element by type and id.
 * @param type $type
 * @param type $id
 * @return type
 * @throws Exception
 */
protected function getElementByTypeAndId($type, $id) {
    switch ($type) {
        case self::DECISION:
            return $this->getDecisionRepository()->find($id);
            break;
        case self::ACTION:
            return $this->getActionRepository()->find($id);
```

```
            break;
        default:
            throw new Exception('type not implemented!');
            break;
    }
}
/**
 * Compare condition
 * @param type $value1
 * @param type $op
 * @param type $value2
 * @return type
 */
protected function compare($value1, $op, $value2) {
    switch ($op) {
        case '==':
            return ($value1 == $value2);
            break;
        case '===':
            return ($value1 === $value2);
            break;
        case '!=':
            return ($value1 != $value2);
            break;
        case '!==':
            return ($value1 !== $value2);
            break;
        case '>=':
```

```
                return ($value1 >= $value2);
                break;
            case '<=':
                return ($value1 <= $value2);
                break;
            case '<':
                return ($value1 < $value2);
                break;
            case '>':
                return ($value1 > $value2);
                break;
            case '<>':
                return ($value1 <> $value2);
                break;
        }
    }
    /**
     * @param $params
     * @return string
     */
    public function processAverage($params)
        $average = '';
        foreach ($this->currentElement->getQuestions() as $q) {
            if (empty($average)) {
                $average = $params[self::IDENTIFIER_ACTION_QUESTION . $q->getId() . '-
value'];
            } else {
                $average = $average * $params[self::IDENTIFIER_ACTION_QUESTION . $q-
>getId() . '-value'];
```

```
            }
        }
        return (float) $average;
    }
    /**
     * @param $val
     */
    public function processActionTransaction($val) {
        $transaction = $this->currentElement->getTransaction();
        if ($val) {
            if ($transaction->getToActionTrue()) {
                $this->nextElement = $this->getActionRepository()->find($transaction->getToActionTrue()->getId());
            } elseif ($transaction->getToDecisionTrue()) {
                $this->nextElement = $this->getDecisionRepository()->find($transaction->getToDecisionTrue()->getId());
            }
        } else {
            if ($transaction->getToActionFalse()) {
                $this->nextElement = $this->getActionRepository()->find($transaction->getToActionFalse()->getId());
            } elseif ($transaction->getToDecisionFalse()) {
                $this->nextElement = $this->getDecisionRepository()->find($transaction->getToDecisionFalse()->getId());
            }
        }
    }
}
```