FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Monitorização Integrada de Aplicações Empresariais

Rui Miguel Ferreira Azevedo

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Rui Filipe Lima Maranhão de Abreu (Doutor)

28 de Junho de 2010

Monitorização Integrada de Aplicações Empresariais

Rui Miguel Ferreira Azevedo

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo júri:

Presidente: José Manuel Magalhães Cruz (Professor Auxiliar)
Vogal Externo: Ricardo Jorge Fernandes Chaves (Professor Auxiliar)
Orientador: Rui Filipe Lima Maranhão de Abreu (Professor Auxiliar Convidado)

---

20 de Julho de 2010

Resumo

Nos últimos anos assistimos a um considerável aumento de complexidade das aplicações informáticas. Cada vez mais criamos grandes sistemas como resposta para o aumento das crescentes necessidades com que a indústria se depara. Esses sistemas, embora tragam vantagens competitivas, também criam uma maior dependência para o funcionamento do negócio. Tendo isso em atenção seria de esperar que fossem feitos esforços à mesma escala para melhorar os mecanismos de monitorização e análise de problemas das aplicações. No entanto, na realidade em muitos casos não é isso que se verifica.
A I2S, SA desenvolve soluções para o mercado de seguros e tem presente a necessidade de melhorar os meios para diagnosticar e corrigir situações anormais, melhorando dessa forma o serviço prestado aos seus clientes. As suas soluções podem atingir um elevado grau de complexidade, com diversos sistemas distribuídos a operarem em conjunto. Qualquer problema que aconteça com as suas aplicações provoca grandes incómodos, afectando o negócio dos seus clientes, e devido à sua complexidade por vezes torna-se difícil de permita encurtar o tempo gasto a diagnosticar o que de mal ocorreu tem potencial para ser uma ferramenta de bastante utilidade.
Este projecto "Monitorização integrada de aplicações empresariais"pretende atacar essa frente por explorar que é a área de monitorização e análise de erros de aplicações informáticas, dotando a empresa de uma ferramenta que permita reunir os dados dos diversos constituintes dos seus sistemas e analisar essa informação de forma fácil e rápida, permitindo fornecer a solução para um problema de forma automatizada.
Este documento apresenta todo o estudo efectuado nesse âmbito assim como o desenho da solução e os passos da implementação que foram entretanto realizados. É apresentado o estudo de técnicas relevantes, como o *Autonomic Computing*, e a razão da escolha do Eclipse TPTP para a solução, descrevendo as suas funcionalidades e enquadrando a solução com essa tecnologia.
Actualmente com o fruto deste trabalho a I2S possui uma ferramenta que cumpre com os objectivos de análise e determinação de problemas.

Abstract

Recent years have witnessed a considerable increase in complexity of computer software. Increasingly large systems were created in response to the growing needs that the industry faces. Those complex systems, although bringing competitive advantages, can also create greater dependence to the operation of the business. With that in mind it would be expected that efforts were made by the same scale to improve the monitoring and problem determination mechanisms of the software. However, in reality this is not the case.
I2S, SA develops solutions for the insurance market and understands the need to improve facilities for diagnosing and correcting abnormalities, thereby improving the service provided to its customers. Their solutions can achieve a high degree of complexity, with several distributed systems operating together. Any problem that happens with their applications affects their customers business, and because of the systems complexity sometimes its difficult to realize what is operating abnormally. A tool that allows to shorten the time spent to diagnose what is wrong has the potential to be a very useful tool.
The project "Integrated Monitoring of Business Applications"aims to attack this front, the monitoring and problem determination area of computer applications, providing the company with a tool that allows to gather data from the several components of their systems and to analyze this information quickly and easily, providing a solution for the problem automatically.
This document presents the entire study made, as well as the solution design and implementation steps that have since been made. In this document it is presented the study of relevant techniques made such as *Autonomic Computing*, and why the Eclipse TPTP was chose for the solution, describing its features and fitting the proposed solution with this Technology.
Currently, with the results of this work, I2S has a tool that meets the the objectives of log analysis and problem determination.

Agradecimentos

Não posso deixar de prestar homenagem a todas as pessoas que directa ou indirectamente me apoiaram neste projecto pois a sua ajuda foi preponderante no trabalho realizado.

Agradeço à Faculdade de Engenharia da Universidade do Porto e todas as pessoas que a constituem pelo apoio e pelo conhecimento que me transmitiram.

À I2S e seus responsáveis por acreditarem no meu valor e pelo apoio e oportunidade de poder contribuir num projecto tão interessante deixo aqui o meu muito obrigado.

Agradeço também às pessoas que directamente supervisionaram este trabalho, o Eng. Paulo Bastos e o Professor Rui Maranhão da FEUP, claramente pessoas com elevados conhecimentos e que com as suas sugestões e aconselhamento contribuíram para que este projecto fosse bem sucedido.

Aos meus colegas Adão Pinto, Bruno Lopes, Daniel Barciela, Hélder Branco, Marcelo Barreira e Sérgio Brandão, com os quais partilhei mais de perto estes últimos meses e que sempre mostraram uma total disponibilidade para me ajudar no que fosse preciso. Trabalhar com vocês é um prazer.

Aos meus colegas de curso por todos os momentos de trabalho e diversão que partilhamos ao longo destes últimos anos.

Agradeço ainda publicamente à minha família, por sempre me terem apoiado e possibilitado atingir os meus objectivos.

Rui Azevedo

Conteúdo

CONTEÚDO

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

I2S	Informática, Sistemas e Serviços
EDA	Event driven applications
TCP/IP	Transmission Control Protocol/Internet Protocol
IBM	International Business Machines
CBE	Common Base Event
WSDM	Web Services DIstributed Management
XML	Extensible Markup Language
TPTP	Test and Performance Tools Platform
AC	Autonomic Computing
TI	Tecnologias de Informação
ACT	Autonomic Computing Toolkit
GLA	Generic Log Analyzer
LTA	Log and Trace Analyzer
BD	Base de Dados
AME	Autonomic Management Engine
ISC	Integrated Solutions Console
RU	Recovery Unit
MTTF	Median-Time-To-Fail
MTTR	Median-Time-To-Recover
ROC	Recovery Oriented Computing
ISO	Organisation Internationale de Normalisation
CMMI	Capability Maturity Model Integration
LAP	Log Audit Profile
IDE	Integrated Development Environment
NASA	National Aeronautics and Space Administration
RCP	Rich Client Platform
OSGI	Open Services Gateway initiative
SWT	Standard Widget Toolkit
U2TP	UML 2 Test Profile
JVMPI	Java Virtual Machine Profiler Interface
JVMTI	Java Virtual Machine Tool Interface
EMF	Eclipse Modelling Framework
WAS	Websphere Application Server

Capítulo 1

Introdução

Este primeiro capítulo pretende introduzir o leitor ao problema abordado nesta Dissertação. A correcta definição do problema permitirá enquadrar o leitor com o rumo do projecto e os seus objectivos.

1.1 Contexto/Enquadramento

1.1.1 A Área de Seguros

À semelhança do que acontece com um grande número de indústrias e serviços na sociedade actual, o negócio de seguros também depende de aplicações informáticas para facilitar e agilizar processos de negócio. Todos os intervenientes na vida de uma apólice, desde a companhia de seguros até ao cliente final, passando ainda pelos mediadores, necessitam de uma ligação que facilmente os mantenham em contacto. Os serviços usados para ligar todos os intervenientes referidos podem variar, no entanto é comum a todas as companhias processos de simulação de prémios, consulta de apólices, efectuar pagamentos de apólices, partilha de informação de negócio entre a companhia e mediadores através de uma mesma infraestrutura informática, consulta de estado de processos, etc. Todos estes serviços e outros mais são assegurados por complexos sistemas, cuja fiabilidade e estabilidade são muitíssimo importantes. Tendo isso em conta, é importante prevenir situações potencialmente negativas para o sistema, utilizando técnicas de monitorização das aplicações, para reconhecer esse tipo de situações e antecipar as acções a tomar para recuperar o normal estado de funcionamento tão cedo quanto possível.

1.1.2 I2S

A I2S, SA [eSSa] é uma empresa de concepção, desenvolvimento e implementação de soluções informáticas de elevado nível de integração e flexibilidade para o mercado global de seguros. Fundada em 1984 na cidade do Porto, local que acolhe a sua sede, conta actualmente com uma delegação em Lisboa e uma associada no Rio de Janeiro, a I2S Brasil. As soluções desenvolvidas na I2S, SA são usadas em vários países da Europa e da África, e ainda no Brasil. Estas soluções englobam suporte para os diferentes intervenientes da actividade seguradora, sendo apresentados na tabela 1.1 os produtos desenvolvidos na empresa, junto com uma breve descrição dos seus objectivos[eSSa].

A I2S, SA conta com 140 colaboradores com formação superior, especialistas em tecnologias de informação e na área de seguros. É detentora de uma certificação ISO 9001:2000, sendo igualmente uma empresa IBM Premier Business Partner e Microsoft Gold Certified Partner. De momento decorre também na empresa o processo de certificação CMMI. Um dos factores de sucesso da I2S prende-se com a constante melhoria dos processos de desenvolvimento, que se reflectem no aumento de qualidade do produto final disponibilizado ao cliente, fazem da I2S, SA a empresa líder da área de desenvolvimento de soluções para a actividade seguradora no mercado nacional, e ainda reconhecimento à escala internacional da qualidade do software desenvolvido.

A I2S, SA conta com uma vasta carteira de clientes, nacionais e internacionais, apresentados nas figuras 1.1, 1.2 e 1.3 divididos por área de negócio:

Para além dos clientes apresentados, a I2S, SA conta ainda com mais de 200 mediadores e corretores de seguros em Portugal, Cabo Verde e Angola, na sua carteira de clientes.

1.2 Motivação

O projecto "Monitorização Integrada de Aplicações Empresariais"descrito nesta tese nasce do contínuo esforço da I2S, SA para melhorar os seus produtos e serviços. Em sistemas de larga escala, compostos por várias máquinas distribuídas, e cujas disponibilidade e fiabilidade têm de ser, no pior dos casos, altas, é necessário um grau de qualidade no software muito alto. No entanto, por melhores processos de desenvolvimento e testes que se façam, é possível que o produto final possua alguns defeitos que podem, a determinada altura, afectar negativamente o normal funcionamento das aplicações. Nestes casos torna-se necessário diagnosticar e resolver o problema. É para este contexto que este projecto pretende desenhar uma solução, que após implementada será denominada de LAP.

Produto	Descrição
gis Não Vida	Desenvolvido a partir de 1989 em plataforma IBM iSeries, inclui um conjunto de módulos que, de uma forma integrada, suportam as actividades associadas às diferentes áreas funcionais de uma companhia de seguros Não Vida, contemplando todos os tipos de produto.
gis Vida	Desenvolvido a partir de 1990 em plataforma IBM iSeries, inclui um conjunto de módulos aplicacionais para suporte à gestão integrada de uma companhia de seguros do Ramo Vida, considerando todas as áreas funcionais e todos os tipos de produto.
gis Fundos de Pensões	Desenvolvido a partir de 1993 em ambiente Windows, destina-se à gestão da actividade de Sociedades Gestoras de Fundos de Pensões - avaliação, estudos actuariais e gestão de pensionistas - podendo adaptar-se a qualquer tipo de fundo.
egis	Arquitectura de *e-insurance*, multicanal e integradora, que permite implementar soluções de *front-office* para os diferentes canais de comercialização - delegações, mediadores, bancasseguros, *home insurance* - e facilmente integradas com outros sistemas, internos e externos à companhia.
gis Mediadores	Aplicação em ambiente Windows para informatização global da actividade de mediadores de seguros (incluindo correctores), e que interliga com a informática central das companhias.
gis SIG	É um conjunto de módulos que, de uma forma integrada, satisfazem as necessidades de tratamento administrativo- financeiro de uma companhia de seguros, sendo de realçar o nível de integração entre si e com toda a área técnica, com base no conceito de *Data Warehouse*. Inclui os módulos: • Contabilidade • Gestão Financeira (Caixa e Bancos) • Gestão de Pagamentos • Gestão de Imobilizado • *Data Warehouse/EIS*
gisdoc	Sistema destinado ao tratamento, arquivo electrónico, circulação e controlo de documentos digitalizados (*Workflow*), integrados ou não com as aplicações GIS.

Tabela 1.1: Soluções desenvolvidas pela I2S

O diagnóstico de problemas em sistemas distribuídos, compostos por várias tecnologias diferentes em conjunto funcionamento pode revelar-se de uma extrema dificuldade. Uma aplicação que permita, de forma transparente, aglomerar informação de várias fontes

e permitir a sua visualização será sempre uma ferramenta de elevado interesse para o diagnóstico de problemas. Da mesma forma uma ferramenta que permita remotamente recolher informação, diagnosticar e resolver, a uma determinada escala, os problemas terá um elevado valor para a empresa. Igualmente se usando uma aplicação for possível encurtar temporalmente o processo de resolução de problemas essa aplicação permitirá fornecer um melhor serviço aos clientes da I2S, SA e garantir uma maior disponibilidade das suas aplicações, o que se traduzirá em menores perdas monetárias para os clientes da I2S, SA devido a indisponibilidade dos sistemas informáticos. Este projecto pretende dotar a I2S de uma aplicação que cumpra as referidas metas para melhor servir os seus clientes.

É esperado que no final deste projecto a aplicação LAP esteja correcta e totalmente especificada e pronta a ser implementada. Esta aplicação será usada, numa primeira fase, apenas pela I2S, SA para monitorização, diagnóstico e resolução de problemas. No entanto o objectivo a médio-prazo será o de adicionar funcionalidades que permitam vender este produto aos clientes da I2S, SA, simplificando-o ao ponto de permitir que as equipas técnicas dos clientes da I2S, SA possam facilmente resolver autonomamente problemas que ocorram.

1.3 Estrutura da Dissertação

Esta dissertação está organizada em cinco capítulos. O primeiro capítulo apresenta apresenta o problema de uma forma superficial, a empresa de acolhimento e a motivação inerente a este projecto. O segundo capítulo descreve o estado de arte de várias técnicas usadas em monitorização, análise e resolução de problemas. No terceiro capítulo é descrito o problema em maior detalhe, definindo-o o mais claramente possível, delimitando a sua fronteira e requisitos. Logo após a definição do problema é apresentada a proposta de solução, justificando convenientemente as escolhas efectuadas e fazendo uma breve apresentação das tecnologias a serem usadas. No quarto capítulo é descrito todo o trabalho de implementação efectuado, junto com uma análise crítica das maiores dificuldades experimentadas no decorrer do projecto. No quinto capítulo são apresentadas as conclusões finais do projecto, junto com uma análise crítica do autor fazendo a retrospectiva do trabalho desenvolvido e algumas considerações pessoais, e é descrito o trabalho futuro a realizar nas próximas iterações deste projecto.

Figura 1.1: Clientes I2S - Ramo Vida[eSSd]

Figura 1.2: Clientes I2S - Ramo Não Vida[eSSc]

Figura 1.3: Clientes I2S - Sociedades Gestoras de Fundos de Pensões[eSSb]

Capítulo 2

Revisão Bibliográfica

Neste capítulo é exposto todo o estudo que foi elaborado no processo de levantamento do estado da arte, analisando trabalhos que já foram realizados e que partilham objectivos semelhantes aos desta dissertação. Muito embora os trabalhos analisados tenham objectivos semelhantes nenhum deles consegue ser uma solução válida que cumpra todos os requisitos deste projecto.

2.1 Event-Driven Applications

2.1.1 Apresentação do conceito

Event driven applications (EDA) é o termo atribuído a aplicações que devem o seu comportamento a eventos externos à aplicação. A ocorrência de um evento, o qual pode ser definido como uma mudança significativa de estado, define o próximo passo de execução da aplicação. Esses eventos podem ter variadas origens, desde vulgares dispositivos de entrada como teclado ou rato, mensagens enviadas a partir de outras aplicações ou threads ou até mesmo a partir de dados obtidos de sensores externos.

2.1.2 Estrutura de um evento

Cada ocorrência de um evento tem um cabeçalho e um corpo. O cabeçalho do evento contêm informação apenas acerca da ocorrência do evento, tais como um identificador, tipo de evento, nome, data, entre outros. O corpo do evento é usado para descrever o que realmente se passou. Um bom exemplo para melhor diferenciar as duas partes constituintes de um evento é a execução de uma inserção de dados, realizada numa tabela de uma base de dados. O cabeçalho poderá ter informação acerca da data, do tipo e identificação

do evento. O corpo conterá a informação de que foram inseridos dados numa determinada tabela.

2.1.3 Event Processing

A ocorrência de eventos pode ser decomposta em camadas diferentes, executadas sequencialmente[Mic06]:

- Event Generator — a primeira camada lógica é a geração de eventos. É o primeiro passo e ocorre quando é detectada uma mudança relevante que pode ter diversas origens como referido anteriormente, que posteriormente é transmitida à aplicação como um evento.

- Event Channel — é o canal que transporta a informação do evento entre o seu gerador e o motor de processamento de eventos. Pode assumir a forma de um canal de comunicação de TCP/IP ou um simples ficheiro. Este canal pode também em alguns casos garantir funções de fila de espera para processar os eventos.

- Event Processing Engine — Nesta camada lógica os eventos recebidos são avaliados, tendo em conta as regras de processamento previamente definidas, e as acções adequadas são escolhidas. O processamento de eventos é assegurado por motores de processamento de eventos, que na sua forma mais simples tratam cada evento individualmente, ou em alternativa podem guardar informação acerca do contexto de execução, passado e futuro, e com isso alterar as acções a tomar para os eventos.

- Downstream event-driven activity — Tomada de acções – Esta é a camada final, responsável por executar as acções escolhidas pelo motor de processamento de eventos. Estas acções podem tomar diversas formas, tais como simples notificações ao utilizador da aplicação ou acções mais complexas como gerar novos eventos ou lançar processos externos para tratar o evento.

Vulgarmente a programação usada neste tipo de aplicações divide o código em duas secções: selecção de eventos e tratamento de eventos.

A secção de selecção de eventos detecta a partir dos dados de entrada a ocorrência de um evento. Em programação a secção de selecção de eventos é normalmente codificado recorrendo a um ciclo infinito, que a cada execução procura detectar eventos. No entanto em sistemas embebidos é vulgar o uso de interrupções, em detrimento de ciclos. Convêm também esclarecer que o processo de detecção de eventos pode não ser tão linear quanto possa parecer à primeira vista pois, apesar de no caso mais simples uma simples entrada poder ser usada de forma unívoca para detectar um determinado evento, muitas vezes a detecção de um evento baseia-se em dados obtidos a partir de diversas entradas diferentes.

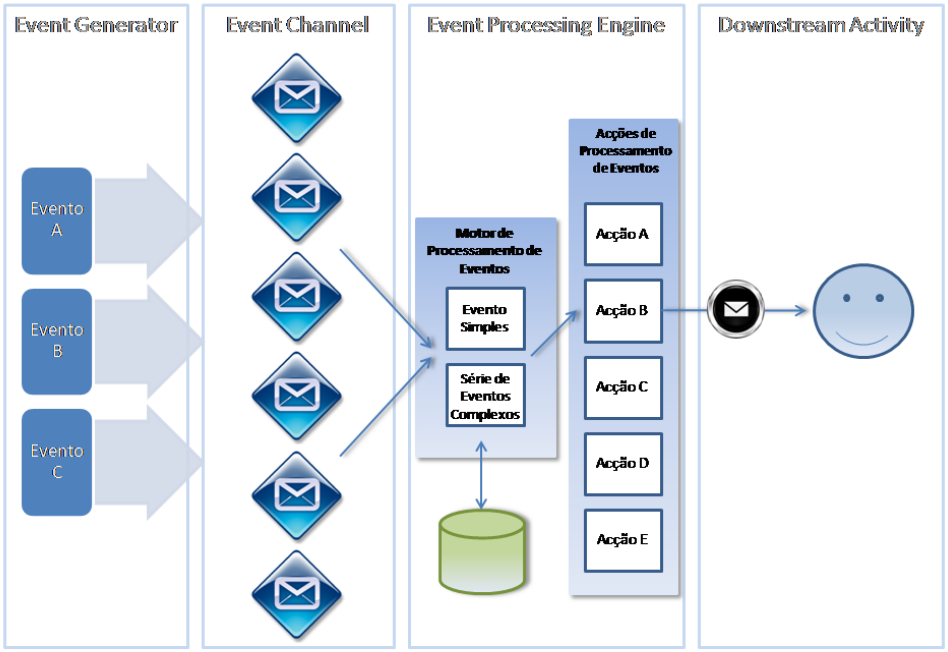

Figura 2.1: Gráfico das etapas de ocorrência de eventos

A detecção de eventos pode ainda basear-se em funções matemáticas que identificam e determinam a ocorrência de um evento a partir dos dados fornecidos à aplicação.

A secção de tratamento de eventos reage aquando da ocorrência de um evento significativo, isto é, um evento ao qual é pretendido responder, baseando-se em acções pré-definidas que foram associadas a cada um dos eventos. As acções a realizar tanto podem ser limitadas apenas à execução de uma simples parte de código ou algo mais complexo como efectuar o tratamento da informação recebida do evento e passar essa informação a outro componente, que por sua vez realizará mais acções.

2.1.4 Convenções

Não existe nenhuma convenção para a definição e notação de eventos, no entanto um esforço da IBM resultou na definição Common Base Events[IBM06] (CBE), parte do standard Web Services Distributed Management (WSDM). O objectivo do CBE é o de facilitar a comunicação de eventos entre componentes de software distintos, proporcionando um formato particularmente adequado para determinação de problemas e eventos, baseado em XML. O CBE foi inicialmente implementado e melhorado na aplicação Tivoli da IBM, sendo implementado posteriormente nas aplicações livres Apache Muse e Eclipse TPTP.

2.2 Autonomic Computing

2.2.1 Apresentação do conceito

Autonomic Computing [KC03] (AC) pode ser considerado como o próximo grande desafio que a indústria das tecnologias de informação (TI) conhecerá e sobre o qual recaí uma crescente importância. De facto, vivemos numa era em que o poder computacional já não é um factor limitativo para um grande número de áreas das TI, uma era em que temos variadas metodologias capazes de atacar diversos problemas e produzir resultados satisfatórios, uma era em que a partilha de informação é bastante facilitada e por isso mesmo o conhecimento é facilmente transmissível e adquirível possibilitando um avançar progressivo das tecnologias e dos que com elas trabalham, uma era em que cada vez mais as ferramentas permitem retirar esforço ao utilizador e melhorar significativamente os resultados reduzindo o trabalho necessário. No entanto continua a existir um aspecto descurado e que poderá dentro em breve travar a evolução das TI. Trata-se da complexidade dos sistemas. Os sistemas informáticos têm vindo a crescer em complexidade de forma assustadora, complicando largamente o desempenho de funções de instalação, manutenção e gerenciamento dos sistemas. O trabalho dos administradores de sistemas nunca foi tão complicado como na actualidade. Esta complexidade está a interferir com o funcionamento correcto dos sistemas, seja através da dificuldade em instalar e configurar um sistema, seja através da detecção de erros e recuperação do sistema para um estado funcional, seja através da dificuldade em tornar os sistemas mais robustos e independentes do controlador humano. Cabe-nos então responder ao próximo grande desafio das TI para assegurar o contínuo crescimento e qualidade dos sistemas de amanhã. A resposta a esse desafio é Autonomic Computing.

2.2.2 Características e elementos de AC

Um sistema de AC pode ser caracterizado pela capacidade de se auto-gerir. Dentro dessa capacidade podem ser identificados quatro propriedades fulcrais apresentadas de seguida[kn:05]:

- Auto-configuração — A instalação e configuração de sistemas de larga escala são tarefas muito complexas e demoradas e por isso a introdução de erros não é invulgar nestas tarefas. Partindo desse pressuposto, as vantagens em ter um sistema capaz de se auto-configurar são mais que óbvias e desejáveis. Sistemas autonómicos deverão configurar-se de acordo com políticas de alto-nível que especificam objectivos pretendidos. Quando um novo componente for introduzido no sistema é expectável que este se auto-configure de novo e seja capaz de explorar o funcionamento do novo componente, publicitando o novo componente para que este seja conhecido e utilizado pelos outros.

- Auto-optimização — Várias aplicações contêm imensas opções de configuração que afectam bastante a performance das aplicações. Se considerarmos que essas opções terão impacto nas outras aplicações constituintes do sistema, a importância de uma configuração adequada assume um papel preponderante na qualidade do sistema. Os sistemas autónomos terão por essa razão uma função de constante monitorização e adequação da configuração, com vista a obter o melhor desempenho do sistema. item Auto-correcção — O diagnóstico e correcção de falhas em sistemas complexos pode revelar-se uma tarefa herculeana, envolvendo por vezes equipas inteiras de programadores durante várias semanas e em que por vezes a falha desaparece sem que se consiga detectar a sua causa. Espera-se de um sistema autónomo a monitorização, detecção e correcção de falhas (possivelmente através de recuperação para trás), permitindo que o sistema se mantenha em funcionamento mesmo quando alguma falha ocorre. Espera-se também que a detecção de falhas seja reportada com vista à sua correcção.

- Auto-protecção — A integridade do sistema nunca pode ser garantida na sua totalidade e por esse motivo protecções adicionais terão sempre no sistema um efeito de acrescento de qualidade. Por esse motivo, complementando a existência de firewalls e mecanismos de detecção de intrusões, um sistema autónomo deverá possuir os mecanismos necessários para antecipar problemas, baseado em relatórios prévios, e se defender contra um ataque.

2.2.3 Arquitectura e Componentes

O objectivo do AC é o de reduzir custos e complexidade de sistemas de larga escala, dotando-os de capacidades de auto-gestão, de acordo com os objectivos definidos pelos humanos. Para além da necessidade de diferentes tecnologias que cumpram funcionalidades deste tipo de sistemas, existe ainda um outro desafio, o de englobar todas as funções numa só arquitectura capaz de explorar cada funcionalidade ao máximo, resultando num serviço óptimo. Com esse fim foram identificados três objectivos que uma adequada arquitectura terá obrigatoriamente de cumprir:

- Deve descrever as interfaces externas e comportamentos esperados dos componentes individuais do sistema

- Deve descrever de que forma é possível formar componentes de forma a estes poderem cooperar para atingir o objectivo global do sistema

- Deve descrever como compor o sistema a partir dos componentes, para que este possa se auto-gerir, ultrapassando falhas que possam ocorrer

A arquitectura de um sistema de AC pode ser descrita através dos seus componentes e usando barramentos de comunicação como o meio de ligação entre eles. Uma possível arquitectura do sistema é apresentada na figura seguinte, que será descrita ao pormenor mais à frente.

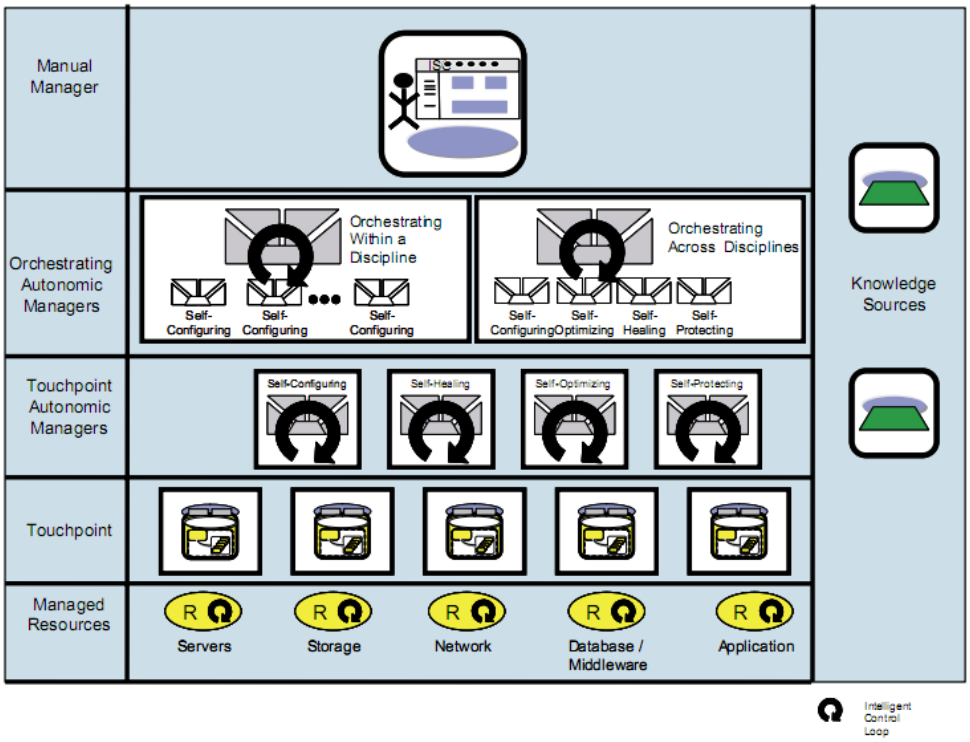

Figura 2.2: Arquitectura Sistema Autónomo[kn:05]

A camada inferior contem os recursos que compõem o sistema informático, os quais incluem recursos de software e hardware. A camada seguinte (Touchpoint) representa as interfaces de controlo dos recursos referidos anteriormente. As duas camadas seguintes representam automatismos de porções do sistema, utilizando Autonomic Managers, que são elementos que cumprem funções de análise, monitorização, planeamento e executam acções no sistema. Na terceira camada podemos ver quatro Autonomic Managers, cada um responsável por um aspecto de auto-gestão do sistema. Na camada quatro temos alguns Autonomic Managers que são responsáveis pelos Autonomic Managersda camada inferior. A camada superior ilustra um Manual Manager, que pode ser considerado o ponto de interacção do administrador do sistema com o sistema de AC. A partilha de informação entre elementos do sistema e o administrador é possível através dos knowledge sources.

2.2.4 Níveis de Maturidade

Várias soluções podem implementar os conceitos de *Autonomic Computing* e serem muito diferentes entre si, relativamente às suas funcionalidades. Na verdade é até possível que várias aplicações que foram desenvolvidas sem ter em atenção este conceito possam considerar-se como aplicações que o implementam em determinada medida. Tendo em atenção esse aspecto é interessante fazer uma análise que permita classificar as aplicações quanto ao nível de implementação do conceito de *Autonomic Computing*. Por isso a IBM especificou [kn:05] um modelo que permite aferir acerca das capacidade autónomas incorporadas numa determinada aplicação e situá-la em relação à adopção do conceito. Esse modelo analisa o nível de controlo que a aplicação para efectuar acções autonomamente e o nível de funcionalidades que a aplicação é capaz de efectuar. Com esses dois indicadores é depois possível adoptar o nível de implementação do conceito de *Autonomic Computing*.

Na figura seguinte [kn:05] esse modelo é apresentado.

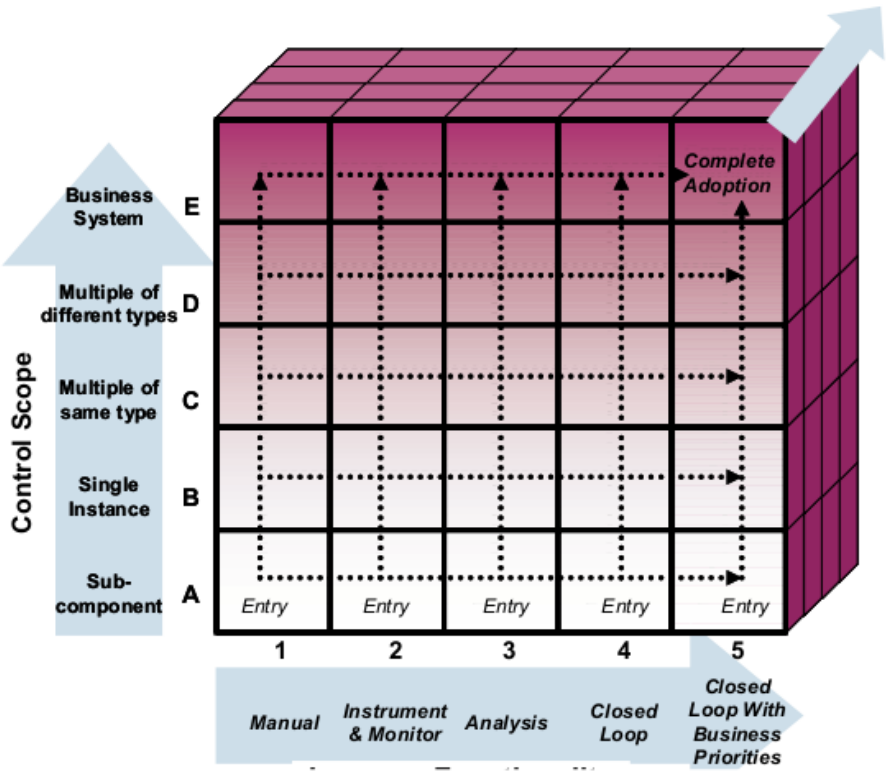

Figura 2.3: Modelo de adopção de capacidades de AC

2.2.5 IBM Autonomic Computing Toolkit

Tendo o esforço inicial da criação do conceito de Autonomic Computing sido feito pela IBM não é de estranhar a estreita ligação desta última na implementação do conceito, especialmente com o Autonomic Computing Toolkit (ACT). O ACT é um bom ponto de partida para a implementação de AC em sistemas, ao permitir poupar bastante tempo e trabalho através do uso de componentes de AC previamente desenvolvidos. Do ACT fazem parte vários componentes criados com o objectivo de ajudar no desenvolvimento de capacidades autonómicas para qualquer tipo de produtos[JLHNY04].

Um dos componentes do ACT é o Generic Log Adapter (GLA), cuja função é a de traduzir informação de um vulgar log de um programa para informação estruturada capaz de ser consumida por um Autonomic Manager.

Do ACT faz parte também o Log and Trace Analyzer (LTA), um automatismo existente para consumir a informação enviada pelo GLA, analisando-a e apresentando uma visão correlacionada dos eventos ocorridos. Dessa forma LTA cobre as acções de monitorização e análise, do ciclo de controlo de um Autonomic Manager. A sua utilização pode ainda ser expandida se adicionarmos uma base de dados de sintomas. Nessa situação, o LTA será capaz de, para além de recolher e analisar informação que lhe é passada, pesquisar na base de dados de sintomas resoluções para o incidente detectado e providenciar um conjunto de soluções, de acordo com a informação contida na BD.

Para uma implementação completa de um ciclo de controlo, isto é, monitorização, análise, planeamento e execução, existe ainda o Autonomic Management Engine (AME). O AME cumpre com todas as funções de um Autonomic Manager. O funcionamento do AME recorre ao uso dos chamados resource models. Estes por sua vez cumprem a função de monitorização, obtendo informação em períodos de tempo especificados constantes dos aspectos do sistema a monitorizar. No entanto importa dizer que o AME é de uma implementação simples do tipo black-box de um Autonomic Manager, e que por isso poderá revelar-se insuficente para casos de uso mais complexos.

Outro dos componentes do ACT é o Integrated Solutions Console (ISC). Este componente permite o controlo de vários elementos constituintes do ambiente, desde hardware e software IBM até aplicações de terceiros, por parte de um humano. Corresponde ao Human Manager na arquitectura de um sistema de AC. As suas funções podem incluir realizar operações de configuração, optimização, monitorização.

2.2.6 Limitações

Embora seja de extrema utilidade para o desenvolvimento de capacidades de computação autónoma, o ACT sofre de alguma imaturidade no seu desenvolvimento, e este aspecto é particularmente limitativo pois com isso a concepção de tais capacidades será sempre uma tarefa demorada e complexa. A inexistência ou a apenas parcial existência de

elementos da arquitectura típica de AC no ACT, como knowledge sources ou implementações completas de manageability interfaces, adicionam uma quantidade considerável de trabalho ao projecto.

Olhando a um nível mais abstracto, é possível verificar que a característica de auto-protecção é inexistente no ACT e a auto-correcção tem uma implementação bastante limitada. No caso da primeira, tudo terá de ser feito de raiz pois nenhum trabalho está feito e este envolve, para além da sua implementação, também um desenho de solução que seja compatível com o ACT de forma a permitir englobar tudo num único sistema. No caso do segundo, a auto-correcção está limitada pois as interfaces dos recursos apenas estão a incluir a componente de sensor, e para uma auto-correcção capaz é necessária também a componente de actuador que não se encontra implementada pois será sempre dependente das aplicações e/ou serviços a monitorizar.

O suporte de especificação de políticas do sistema é algo limitado pois apenas cumpre com essa funcionalidade usando ficheiros baseados em XML. Este aspecto poderá levantar questões de performance, especialmente para especificações de políticas mais longas. Quer isto dizer que para todos os sistemas desenvolvidos usando ACT será necessária uma análise cuidada para garantir que os parâmetros de qualidade do sistema serão cumpridos.

2.2.7 Melhoramentos

A enorme complexidade inerente à implementação de capacidades de computação autónoma implicam que, embora parte do trabalho esteja já feita de forma genérica, ainda assim muito há a desenvolver de forma a permitir a total criação de sistemas autónomos usando a ACT.

O ACT tem uma enorme carência de desenvolvimento das suas funcionalidades de auto-protecção. Por isso mesmo, mecanismos de verificação de controlo de acesso às funcionalidades do sistema seriam uma adenda de enorme interesse ao ACT. Esses mecanismos poderão ser implementados recorrendo-se do fluxo de informação existente entre aplicações, baseado em CBE, um pouco à semelhança do que acontece com os eventos, e das knowledge sources. Os acessos às funcionalidades do sistema serão identificados por eventos gerados, transmitidos aos autonomic managers correspondentes e verificados num registo, que terá de ser também ele implementado visto não existir. O objectivo será ter um registo que guarda informação de quem garantiu acesso às funcionalidades identificando-se correctamente. Um pedido feito ao sistema informático naturalmente gerará um evento que contem a identificação de quem fez o pedido, e essa identificação será posteriormente verificada no registo para garantir a sua validade. A ausência do autor do pedido no registo traduzir-se-a num pedido inválido e caberá ao sistema desencadear o actuador da interface do managed resource com vista à negação desse pedido.

Os knowledge sources, para além da falta de uma implementação de um registo, também carecem de estruturas de armazenamento de informação partilhada entre autonomic managers. Essas estruturas serão de enorme utilidade pois permitem partilha de informação entre autonomic managers e com isso uma melhor coordenação e desempenho entre estes. Por isto, o desenvolvimento de uma estrutura de dados capaz de reter a informação interessante durante a validade da mesma requererá planeamento e muito pensamento para o desenho de uma estrutura capaz corresponder às necessidades.

Outro dos aspectos cujo desenvolvimento melhorará significativamente a qualidade do ACT será a criação de um componente que funcionará em conjunto dos actuadores de um managed resource, traduzindo as instruções recebidas dos autonomic managers, possivelmente no formato CBE, em chamadas a funções. Este componente funcionará um pouco à semelhança do GLA, mas no sentido oposto, traduzindo as saídas dos autonomic managers em algo concreto para os componentes do sistema informático.

2.3 FLORA

2.3.1 Conceito

A arquitectura de um sistema, tipicamente construída de acordo com os objectivos de reusabilidade, adaptabilidade e performance, normalmente cria sistemas constituídos por dependências entre componentes, e que por isso podem falhar como um todo aquando de uma situação de erro, mesmo que esse erro ocorra apenas num componente do sistema. Com o intuito de ultrapassar essas situações de erros foi criada uma Framework que decompõe um sistema completo e isola as suas partes em recovery units (RU), tornando-as capazes de recuperar o seu funcionamento sem afectar o todo o funcionamento do sistema. O princípio de funcionamento desta Framework é o de recuperação local, isto é, permitir a recuperação de todo o sistema sem existir propagação de falhas, desde o ponto onde a primeira falha ocorre até à totalidade do sistema. A contenção de falhas pode ser vista como um passo lógico, no entanto a sua implementação pode ser complicada pois requer não só o isolamento e recuperação das RUs mas também a garantia de que nenhuma comunicação entre processos é perdida e também que o RU do sistema que falhou seja colocado num estado correcto após a sua reinicialização.

2.3.2 Vantagens

A recuperação local pode melhorar a disponibilidade de um sistema ao permitir que um componente que falhe recupere de seguida sem afectar os outros componentes. Experiências comprovam[STA09] que o uso da Framework FLORA num projecto pode implicar a adição de menos de 1% de linhas de código. Com uma adição tão pequena de linhas

de código é de facto positivo obter resultados que, no melhor dos casos, conseguiram aumentar a disponibilidade do sistema em 13%.

A Framework tem uma forma de funcionamento muito simples, o que por norma é desejável em mecanismos de tolerância a falhas, pois com maior complexidade cria-se um ambiente mais propicio à introdução de erros.

A Framework FLORA consegue bons resultados em situações em que o median time to fail (MTTF) é bastante diferente entre componentes. Isto porque, aquando da separação do sistema em RUs, tem em atenção este aspecto e impede que componentes com altos e baixos MTTF façam parte da mesma RU. Essa separação tem ainda em conta o median time to recovery (MTTR) dos componentes, pois componentes com valores altos de MTTR não devem ser misturados com componentes com baixos valores de MTTF, sob pena de as recuperações serem frequentes e demoradas, afectando a performance do sistema.

2.3.3 Limitações

Apesar de ser capaz de produzir bons resultados, esta Framework tem alguns pontos menos fortes que importa referir. Provavelmente o maior deles será a decomposição de um sistema numa arquitectura tal que permita um óptimo desempenho. Esta decomposição esbarrará na maioria dos casos nas boas práticas de Engenharia de Software, quanto à adaptabilidade e reusabilidade do sistema, o que provoca em sistemas existentes enormes custos de redesenho e implementação, e em sistemas novos limitará a sua capacidade de receber novas funcionalidades ou ser modificado no futuro. Para além disso, para obter uma recuperação local óptima é necessário escolher bem a divisão em módulos do sistema, o que nem sempre é fácil, principalmente quando os sistemas têm uma dimensão bastante grande e também quando existe muita interacção entre todos os módulos. Outro dos problemas que se coloca no uso desta Framework é a necessidade de uma análise cuidada das dependências funcionais e de dados entre componentes do sistema. Esta análise pode ser muito complicada de efectuar com projectos em que haja uma elevada dependência entre componentes e não pode ser descurada de forma alguma pois influencia a divisão do sistema em RUs e os resultados conseguidos. No caso da comunicação entre processos existe ainda um efeito colateral do uso da Framework, que é o overhead adicionado ao sistema. Quanto maior for a comunicação entre componentes e quanto maior for o número de RUs, maior será o overhead adicionado, o que provocará um decréscimo de performance do sistema.

Por fim um ponto que importa ter sempre em conta na criação de métodos tolerantes a falhas é o de que se esses métodos requerem a introdução de mais código podem também introduzir novas falhas. Portanto a concepção de uma Framework para obter recuperação local deverá ser um projecto bem supervisionado e revisto, obedecendo a boas práticas de

engenharia de software para evitar tanto quanto possível a introdução de novas falhas no sistema, as quais poderiam provocar a avaria do sistema sem possibilidade de recuperação, e consequentemente anular o objectivo inicial.

2.4 Recovery Oriented Computing

2.4.1 Conceito

Recovery-oriented computing (ROC) é um método para criar serviços fiáveis baseados na Internet. O seu nascimento foi potenciado pelas Universidades de Stanford e de Berkeley, com o objectivo de contornar a existência de falhas inevitáveis de software e reduzir os seus efeitos negativos através da recuperação das falhas, ao contrário de outras abordagens de tolerância a falhas que focam-se em tentar prevenir a ocorrência de falhas[PBB+02].

A abordagem ROC assenta o seu plano de acção em três premissas, vastamente suportadas pela experiência na produção de serviços de Internet:

- A taxa de falhas, tanto de software como de hardware, são crescentes e não podem ser desprezáveis
- Os sistemas não podem ser completamente modelados na execução de análises de fiabilidade e por isso os seus modos de falhas não podem ser previstos antecipadamente
- Erros humanos introduzidos por operadores do sistema ou durante acções de manutenção são uma grande fonte de falhas

Atendendo às premissas previamente referidas, é possível guiar o desenho de sistemas ROC focando algumas áreas concretas capazes de garantir os objectivos pretendidos e que serão apresentadas seguidamente:

- Isolamento e Redundância — A ocorrência de uma falha em determinada parte do sistema pode facilmente ser ultrapassada, sem perda de funcionalidades, pelo isolamento de determinada parte do sistema e pela sua substituição por outra parte redundante. A acção de isolar é também crucial para detectar erros e restringir comportamentos erróneos. Esta área de pesquisa compreende investigação tanto a nível de hardware como de software para isolar partes de sistemas.
- Suporte para desfazer acções entre todo o sistema — Inevitavelmente em todos os sistemas existe um grande ponto comum de falha que é o utilizador. Seja através do uso do sistema, da sua configuração ou da manutenção necessária, é possível introduzir várias falhas. Por isso mesmo, ROC pretende introduzir um mecanismo

de desfazer acções em todas as partes constituintes do sistema, sejam hardware ou software, de forma a mais facilmente reparar o que de errado foi feito e recuperar o sistema para um estado correcto. Esta área de pesquisa pretende descobrir os limites do que é exequível no âmbito de aplicações ROC, os custos entre funcionalidade/benefício do mecanismo de desfazer acções e a definição de um modelo adequado e prático.

- Suporte integrado de diagnóstico — Parte do processo de recuperar de falhas engloba detectar a falha e essa detecção será sempre tanto melhor quanto mais rapidamente conseguir diagnosticar a ocorrência de uma falha. Dessa forma podem ser tomadas acções de correcção que simultaneamente recuperam o sistema e evitam a propagação de falhas, sendo que neste último caso pode fazer a diferença entre perceber realmente o que de facto correu mal ou obter uma visão limitada da origem da falha. Pretende-se então criar um sistema de diagnóstico incorporado no sistema global, que permite detectar autonomamente falhas e as suas origens, através da auto-monitorização constante de todos os componentes do sistema assim como dos componentes dos quais dependem. É tão pretendido que os componentes do sistema sejam capazes de interagir entre eles fornecendo informação quanto às dependências, recursos e pedidos recebidos do utilizador ao sistema de diagnóstico. Pesquisa nesta área envolve a criação de interfaces e frameworks de teste para os componentes, verificação de software e algoritmos de análise de origem de falhas, de forma a dotar o sistema de funcionalidades de contenção de falhas.

- Mecanismos de recuperação — Os mecanismos de recuperação são elementos vitais para manter o sistema a funcionar correctamente, pois a sua eficiência e fiabilidade será sempre colocada em prova aquando da ocorrência de uma falha, tratando-se dos mecanismos responsáveis pela recuperação da falha. Por isso mesmo, num sistema ROC é esperado que estes mecanismos sejam constantemente sujeitos a diversos testes que possam garantir um elevado grau de confiança nos mecanismos. Esta área engloba investigação de métricas de qualidade dos mecanismos, desenho de testes adequados e a integração destes mecanismos com os restantes componentes.

- Design para alta modularidade, mensurabilidade e reiniciamento — Muitas vezes uma das técnicas eficazes contra na prevenção de falhas é o reiniciamento de componentes do sistema. Muitas falhas são de difícil replicação, por vezes causadas por corrupções de memória ou problemas de concorrência resultantes de um elevado tempo de funcionamento, em que a partir de certa altura um componente atinge um ponto de falha. Por isso mesmo esta área de pesquisa procura encontrar resposta às necessidades de reiniciamento de componentes através de técnicas adequadas para esse efeito, tendo em conta as dependências funcionais.

- Medidas de disponibilidade — A produção de medidas eficazes na avaliação da disponibilidade de um sistema é essencial para analisar os resultados dos avanços produzidos na área de ROC. A introdução de falhas e perturbações no sistema leva a uma análise do impacto resultante, o qual se for medido adequadamente pode fornecer níveis de qualidade.

2.4.2 Análise crítica

Apesar de o ROC basear-se em pressupostos válidos e relevantes no âmbito de criação de mecanismos de software para tolerância a falhas, a verdade é que por vezes sobrepõe-se um pouco a outras abordagens para tolerância a falhas já existentes. A característica que distingue melhor o ROC será o suporte para desfazer operações em todo o sistema. Esta funcionalidade, por si só, poderá ser muito poderosa para recuperar de falhas. No entanto tem desvantagem ao ser uma operação que necessita de reparação, após andar para trás, para restabelecer o correcto estado do sistema. Esta operação poderá inserir novos erros, caso a reparação seja inadequada.

Capítulo 3

Proposta de Solução

Este capítulo começa por apresentar em detalhe o problema e posteriormente passa à descrição dos seus requisitos. Tendo em conta essa primeira análise é de seguida apresentada a proposta de solução, abordando os conceitos teóricos desta e a sua arquitectura, justificando a sua adequação a este problema.

3.1 Problema

3.1.1 Âmbito

Devido ao aumento das necessidades de fiabilidade e capacidade de resposta do *software*, motivadas pelo aumento do uso das aplicações a nível empresarial, assistiu-se ao emprego de novas arquitecturas no acto de projectar sistemas capazes de responder adequadamente às exigências referidas. Uma dessas arquitecturas bastante vulgar no mundo actual é a arquitectura distribuída. Num sistema concebido com este tipo de arquitectura os componentes do sistema deixam de estar todos localizados na mesma máquina, sendo comum que vários componentes se possam encontrar em várias máquinas diferentes. O nível de granularidade destes componentes não é fixo, sendo normalmente limitado pelas capacidades das tecnologias. No entanto este tipo de arquitectura acarreta alguns problemas, sendo um deles a dificuldade em monitorizar e diagnosticar problemas. Seja devido ao número de componentes isolados que compõem o sistema, às suas ligações físicas, a um erro do utilizador ou outra razão qualquer, a verdade é que no final se detecta um funcionamento incorrecto do sistema mas torna-se difícil perceber qual a sua causa. Este tipo de problema poderá ter consequências bastante negativas, tanto para o produtor de *software* pela deterioração da imagem do seu produto como para o cliente final que vê as aplicações de suporte ao negócio falharem e com isso acumular prejuízos. Nesse sentido torna-se importante desenvolver mecanismos que permitam facilitar a detecção de

ocorrências anómalas e prejudiciais ao normal funcionamento dos sistemas e, se possível, antecipar essas situações através de contínua monitorização das aplicações constituintes do sistema.

3.1.2 Objectivos

Numa arquitectura de sistemas distribuídos, poderão existir várias aplicações ou componentes heterogéneos, que podem usar tecnologias diferentes e que comunicam de forma diferente o seu estado, e um sistema de monitorização e detecção de erros só será útil se conseguir abranger a totalidade dos constituintes do sistema e fornecer, dessa forma, uma visão global.

Este trabalho pode ser visto como uma união de três pontos de vista diferentes: o da monitorização de aplicações, que em aplicações de média ou grande dimensão assume uma elevada complexidade, o da análise de problemas que requer não só informação adequada das aplicações mas também um mecanismo eficiente e capaz de detectar tais situações adequadamente, e o da empresa que necessita de uma solução que se adeqúe aos seus produtos com o menor esforço possível.

Olhando para o problema de monitorização de uma perspectiva de alto nível, e segundo Jack Shirazi[Shi03], a escolha de uma ferramenta de monitorização deverá ser realizada atendendo aos seguintes critérios:

- Componentes de monitorização e *logging* - Todos os aspectos importantes do sistema a monitorizar deverão ser registados. Os problemas de performance advêm essencialmente de três localizações genéricas: processamento em componentes, *overhead* comunicações entre componentes e interfaces entre componentes.

- Reduzido *overhead* - O processo de monitorização deverá impor uma pequena carga no sistema, normalmente menos de 5% dos recursos disponíveis são necessários para uma monitorização eficiente. Idealmente valores de 1% serão desejáveis. Isto porque é pretendido que o processo de monitorização seja capaz de monitorizar eficientemente o sistema sem afectar directamente a performance deste. Permite igualmente que o sistema possa estar sempre activo, mesmo em ambientes de produção, sem que afectar a performance.

- Pedidos mapeados para métodos - Os pedidos recebidos pelo sistema deverão mapear métodos que serão monitorizados, assim como os componentes e as comunicações. Se este mapeamento for fácil de ser efectuado é possível detectar mais eficazmente quais os pedidos que estão a causar problemas.

- Armazenamento de informação e sua granularidade - Toda a informação recebida pelo sistema de monitorização deverá ser guardada de forma persistente, de forma a

separar o processo de análise da execução do servidor. Assim, tornar-se-a possível mais tarde reanalisar a informação. A granularidade da informação deverá ser ponderada pois demasiada informação poderá apenas complicar o processo de análise e torná-lo menos eficiente, enquanto que reduzida informação poderá revelar-se insuficiente para determinar a causa de um problema.

- Escalabilidade - A aplicação de monitorização deverá conseguir escalar-se de forma a acompanhar diferentes configurações possíveis do produto a monitorizar.

A análise de problemas toma particular importância pois é um passo resultante do processo de monitorização e é necessário para dar importância à monitorização. Sem análise de problemas não faria sentido monitorizar. A informação utilizada na análise de problemas deverá ser clara o suficiente para que seja possível diagnosticar correctamente o(s) problema(s) existente(s) e sugerir soluções. Para isso, este processo deverá fornecer não só ferramentas de análise mas também mecanismos que permitam facilmente filtrar enormes quantidades de informação segundo vários critérios, detectar eventos específicos, agrupar informação relacionada a pedido do utilizador de forma a obter uma vista universal do que se passa em determinada parte do sistema.

As aplicações da I2S usam diferentes tecnologias e por isso um dos desafios é encontrar uma forma que permita aglomerar informação de diversas fontes diferentes, com formatos diferentes, num único formato universal. Existe outro requisito que é o de não ser possível alterar as aplicações já existentes, logo todo o trabalho terá de ser feito com as aplicações que já existem no seu estado actual. Isto implica que a aplicação deverá ser projectada para funcionar com as aplicações actuais da I2S, sendo flexível o suficiente para permitir a sua adaptação a diversas configurações de sistemas em uso.

Após descrever o problema e contextualizá-lo, torna-se possível apresentar os requisitos específicos principais deste projecto.

- Visualização uniforme da informação recebida do LAP Services ou carregada das aplicações da I2S
- Filtrar a informação mostrada segundo vários critérios
- Mostrar informações acerca das estatísticas das transacções
- Interface configurável e extensível
- Obter as soluções disponíveis para cada problema encontrado
- Permitir fazer drill-down à informação mostrada acerca de uma transacção para obter maior grau de detalhe

Este trabalho pretende idealizar e desenhar uma solução que cumpra os requisitos referidos anteriormente e que se integre com as soluções de negócio da I2S, SA. De referir

ainda que este trabalho será articulado com outros dois trabalhos no âmbito do mesmo projecto. Esses outros dois trabalhos darão origem a duas outras teses de mestrado.

3.1.3 Arquitectura geral do projecto

Como referido, devido a restrições de tempo, o projecto LAP foi dividido em três componentes. Cada componente corresponde a uma tese de mestrado diferente, no entanto uma visão global do projecto torna-se necessária para a sua melhor compreensão. Assim, de seguida é apresentada a arquitectura geral do projecto, referindo os seus componentes e uma breve descrição de cada um deles.

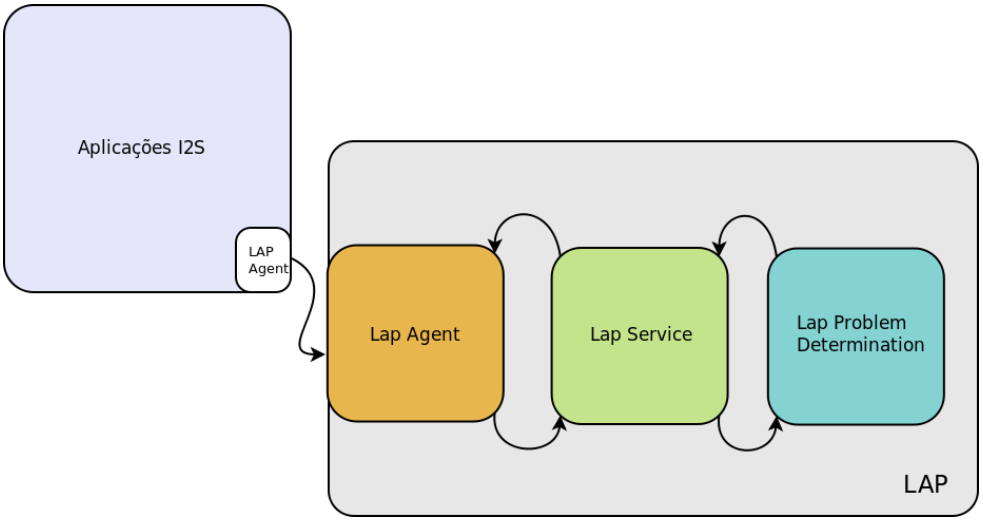

Figura 3.1: Arquitectura geral LAP

Na figura é possível observar a divisão em três componentes, do projecto LAP, como referido anteriormente. O *LAP Agent* é responsável por redireccionar a saída de *logging* das aplicações I2S para o LAP. O ponto de entrada dessa informação no LAP, através do *LAP Agent*, deverá fornecer mecanismos de persistência da informação e também métodos robustos de enviar dados para o *LAP Services*, sem perda de informação. O *LAP Services* providencia um conjunto de serviços para tratamento da informação. Estes serviços deverão incluir agrupamento e correlacionamento de informação por critérios, conversão para um formato de dados universal e passagem dessa informação processada ao *LAP Problem Determination*. O *LAP Problem Determination*, o alvo de trabalho desta tese, deverá ser capaz de importar a informação do *LAP Services* e providenciar funcionalidades de filtragem e análise de problemas, assim como monitorização dos registos e apresentação de soluções para situações de erro.

De acordo com o modelo de maturidade das capacidades de *Autonomic Computing* sugerido pela IBM[kn:05], é possível caracterizar o LAP como uma aplicação de *Autonomic Computing* de grau 3, a nível de funcionalidades, pois incluirá capacidades de

análise de diversas origens de dados e oferecer sugestões de soluções para problemas, e a nível de controlo será de grau 4 pois actuará sobre conjuntos de componentes heterogéneos.

3.1.4 Fronteira do projecto

Em qualquer projecto é útil delimitar bem a sua fronteira, especificando o que é esperado do projecto, o que é possível e também o que não é possível realizar. Este projecto terá de obedecer às orientações globais, apresentadas de seguida:

- Linguagem de programação - Embora não exista uma restrição imperativa neste caso, a escolha da tecnologia a usar no desenvolvimento do projecto deverá ser a da linguagem Java, pois já existem alguns componentes anteriores desenvolvidos nesta linguagem e é uma tecnologia que é suportada em todos os sistemas de produção criados pela I2S. O uso de uma tecnologia alternativa, não sendo proibida, implicará uma boa justificação.
- Não alteração de código de aplicações existentes - Este ponto, como o anterior, não sendo obrigatório, é muito recomendável, e será necessária uma boa justificação para quebrar esta regra.
- Independência de componentes - Os componentes do sistema de a desenvolver deverão ser independentes entre si, permitindo que o seu funcionamento não seja dependente de qualquer outro componente.
- Modificações de código de tecnologias de suporte - Por questões de suporte e mais fácil actualização de versões, o uso de bibliotecas ou código *open source* deverá restringir-se a APIs ou pontos de extensão existentes e dever-se-a evitar a todo o custo a modificação de código.
- Integração com aplicações I2S - O resultado final deverá ser facilmente integrado com as aplicações desenvolvidas e comercializadas pela I2S.

3.2 Solução

3.2.1 Proposta de solução

Várias metodologias e abordagens ao problema de monitorização e análise de problemas foram consideradas, estando o seu estudo descrito no capítulo "Revisão Bibliográfica"deste relatório. No entanto tendo em conta os requisitos do projecto LAP apenas uma tecnologia consegue cumprir integralmente com os objectivos. Essa tecnologia é o *Autonomic Computing Toolkit*, que embora não esteja disponível presentemente, deu origem

ao projecto Eclipse TPTP. Após uma análise de ferramentas de monitorização e análise de problemas não foi possível encontrar outra alternativa que não passasse pelo projecto de monitorização do Eclipse TPTP. Ao contrário de outras soluções, o Eclipse TPTP fornece vários componentes capazes de realizar as funcionalidades pretendidas e possui vários pontos de extensão que permitem personalizar o seu funcionamento à realidade da I2S. Desde logo três componentes destacam-se no que a este projecto dizem respeito:

- GLA - Permite converter informação para um formato comum, CBE, e cujo suporte a diferentes formatos de informação é facilmente extensível
- LTA - Proporciona operações de filtragem e visualização de informação importada através do GLA
- Motor de análise - Um motor de análise consegue detectar situações de erros ou problemas nas aplicações, necessitando apenas de uma base de dados de sintomas

O seu código é aberto o que acrescenta maior valor pois permitirá divergir livremente desta tecnologia no futuro, caso os responsáveis do projecto assim o entendam e não acarreta custos. Não foi encontrada outra ferramenta pronta a ser trabalhada que cumprisse com os objectivos mínimos para poder sequer chegar a ser considerada. Tendo em conta esse aspecto, existiu a possibilidade de criar de raíz uma ferramenta que cumprisse os requisitos do projecto. Esta opção tinha a desvantagem de ser uma abordagem mais ousada e poderia não ser possível, dentro do tempo previsto de realização desta dissertação ser possível apresentar resultados. Por essas razões foi decidido avançar com o uso do Eclipse TPTP como base e personalizar o projecto para se adaptar às necessidades da I2S. É de todo desejável que haja um mecanismo que permita aglomerar informação e tratar um problema de uma forma simplificada. Por essa razão foi decidido estender a base do Eclipse TPTP com funcionalidades de um *Issue Tracker* [1]. Tendo em conta que a plataforma usada será do Eclipse, a escolha de um sistema de gestão de tarefas deu preferência a sistemas que se integrem naturalmente com esta plataforma. Por essa razão, e por apresentar funcionalidades adequadas aos requisitos deste projecto, como é o caso de utilização de repositórios locais e/ou externos e mecanismos de importação/exportação de *issues*, optou-se pelo sistema de gestão de tarefas e de ciclo de vida de aplicações do Eclipse, o Mylyn. Este projecto goza de grande popularidade e, sendo desenvolvido pela Fundação Eclipse, dá uma confiança extra no que toca à qualidade de integração com a plataforma Eclipse.
A versatibilidade da plataforma Eclipse também permitirá a fácil personalização de um navegador de contéudos, mais adequado à aplicação. Esse objectivo será atingido através

[1] Um *issue tracker* é um programa criado com o objectivo de gerir a correcção de erros ou a realização de tarefas, no processo de desenvolvimento de *software*

da *Common Navigator Framework* da plataforma Eclipse. A vista de navegação de conteúdos existente no LTA do Eclipse TPTP é algo limitado, apenas apresentando informação acerca dos objectos genéricos com que o Eclipse TPTP trabalha. Foi considerado que esta vista não era muito "amigável"para o utilizador e no sentido de maximizar a usabilidade da aplicação, foi decidido criar uma vista personalizável, adaptada às necessidades da I2S. Nesse sentido está neste momento a ser desenhada uma vista que permita replicar a estrutura dos sistemas da I2S, obrigatoriamente adaptável a diferentes cenários, e que permite uma navegação lógica por todos os componentes das soluções I2S. Um exemplo daquilo que existe actualmente e aquilo que é pretendido pode ser visualizado nas figuras 3.2 e 3.3.

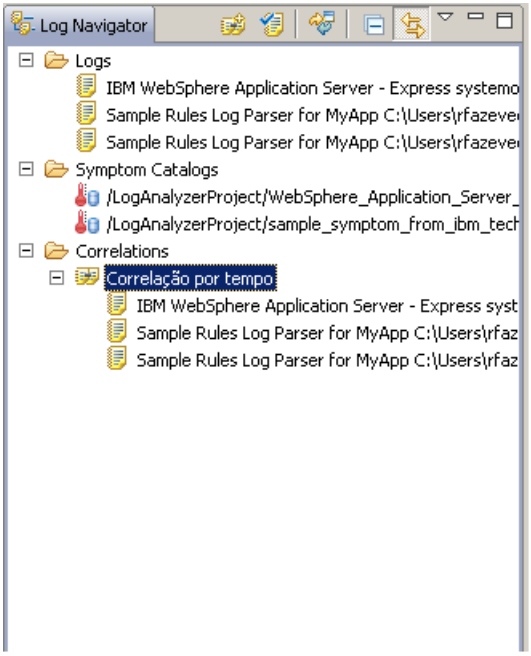

Figura 3.2: Vista *Navigator* disponível no LTA do Eclipse TPTP

3.3 Análise Tecnológica

3.3.1 Plataforma Eclipse

3.3.1.1 Apresentação

O Eclipse [Fouh] nasceu inicialmente como uma plataforma para implementação de ambientes de desenvolvimento, pelas mãos da IBM Canadá, tendo sido mais tarde, mais concretamente em 2001, lançado com uma licença *Open Source*. Essa abertura à comunidade tornou o Eclipse cada vez mais popular e permitiu aumentar imensamente as suas

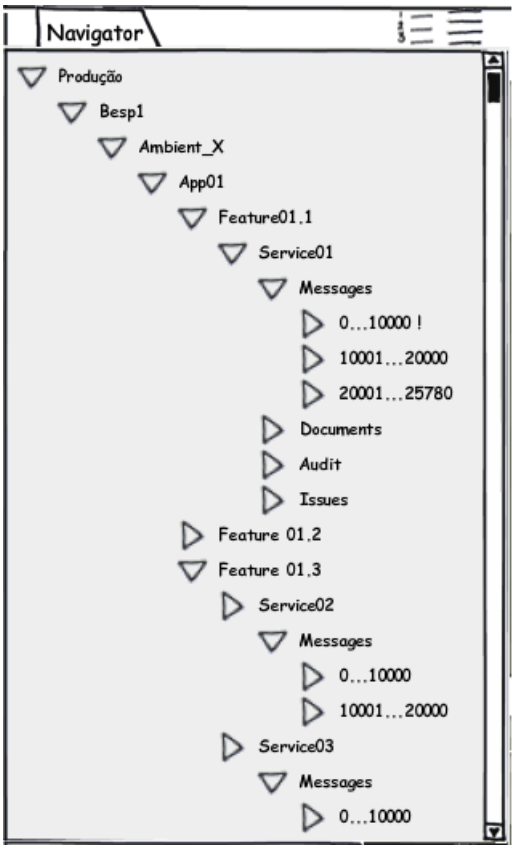

Figura 3.3: Exemplo de vista *Navigator* que replica a hierarquia existente nas aplicações da I2S

capacidades, através de contribuições da comunidade. Actualmente o seu desenvolvimento é coordenado pela fundação *The Eclipse Foundation*. Actualmente vários produtos comerciais são baseados nesta plataforma, sendo que um dos mais conhecidos será o ambiente de desenvolvimento da própria IBM, denominado *Websphere Rapid Application Development*.

3.3.1.2 Funcionalidades

O maior trunfo da plataforma Eclipse é a sua arquitectura modular. Em cima de um sistema mínimo de *runtime* são adicionados vários *plugins* que permitirão à plataforma desempenhar as funções desejadas, sejam elas quais forem. Na realidade, e apesar de a plataforma Eclipse ser maioritariamente reconhecida pelo seu IDE de Java, a plataforma Eclipse é utilizada em diversas ferramentas [Foum], inclusivamente para auxiliar a NASA nas suas missões espaciais [Foul].

Os *plugins* que adicionam funcionalidades à plataforma Eclipse usam pontos de extensão bem definidos para se ligarem à plataforma, e podem eles próprios adicionar pontos de extensões adicionais e dessa forma adicionar novas capacidades de expansibilidade.

A figura 3.4 [Foud] representa de uma forma simplificada a arquitectura da plataforma Eclipse e as ligações com novas funcionalidades.

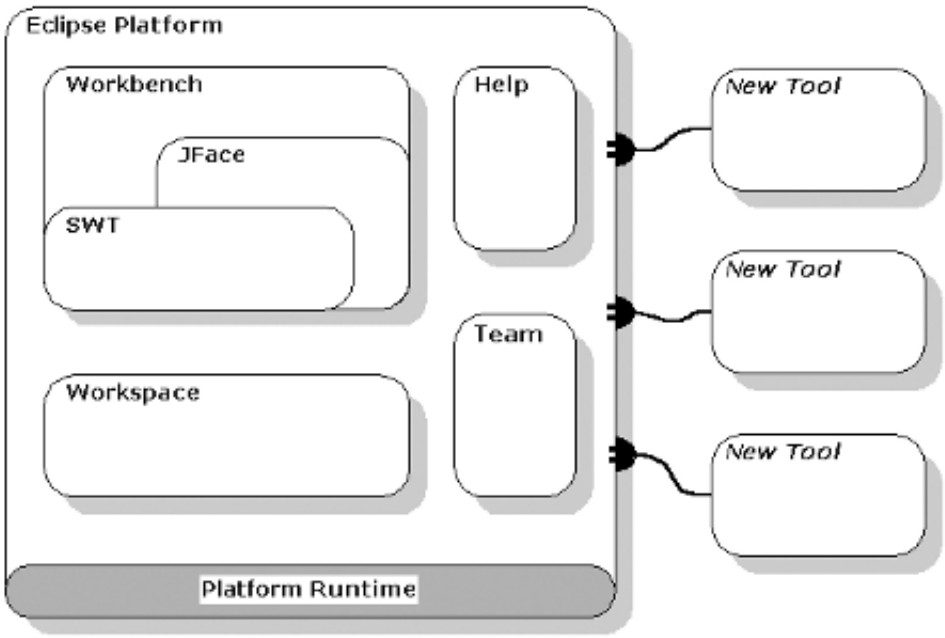

Figura 3.4: Arquitectura da plataforma Eclipse[Fouc]

A arquitectura, como já referido, torna fácil a criação novas aplicações de vários âmbitos. Para esses casos é disponibilizada a *Rich Client Platform*, composta por apenas cinco componentes, aos quais é possível adicionar as funcionalidades pretendidas. Esses cinco componentes são:

- Equinox OSGI - Framework usada para implementar a arquitectura de *plugins*
- Core - Núcleo da plataforma Eclipse, responsável por iniciar a aplicação e correr os *plugins*
- SWT - Uma framework portátil para criação de interfaces gráficas
- JFace - Adiciona o modelo Model-View-Controller à tecnologia SWT, assim como tratamento e editores de texto
- Eclipse Workbench - Possibilita a adição de vistas, perspectivas, editores e *wizards*

3.3.1.3 Análise Crítica

Uma grande vantagem da plataforma Eclipse é a sua versatilidade, devida a uma arquitectura cuidada, baseada em componentes e com um grau de simplicidade elevado, o que permite a sua utilização em diversos campos com resultados muito positivos.

3.3.2 Eclipse TPTP

3.3.2.1 Apresentação

O Eclipse TPTP é um projecto criado em Agosto de 2004[Foue] pela Fundação Eclipse como uma evolução do anterior projecto Hyades. Baseia-se em trabalho desenvolvido pela IBM e no seu IBM Autonomic Computing Toolkit. O seu objectivo é criar uma plataforma genérica, extensível e baseada em standards, com o intuito de ser usada no desenvolvimento de *software*, nomeadamente na criação ferramentas de teste e performance especializadas, diferenciadas e interoperáveis.[Foug] Este projecto *Open-Source* conta com o apoio de várias organizações no seu desenvolvimento, como a Intel, a IBM, a Scapa Technologies e a OC Systems.

3.3.2.2 Funcionalidades

O Eclipse TPTP permite fornecer funcionalidades para todo período de vida de uma aplicação, desde a análise do código desenvolvido, nas fases iniciais, até à monitorização do funcionamento da aplicação final.

Este projecto divide-se em quatro áreas de desenvolvimento, cada uma com objectivos distintos, que serão apresentados de seguida:

- TPTP Platform - Este sub-projecto pretende fornecer a infraestrutura base nas áreas de interface com o utilizador, modelos de dados EMF e recolha de dados e controlo de comunicações, para os outros sub-projectos. Providencia ainda pontos de extensão para a adição de novas funcionalidades ou extensão de funcionalidades já existentes.

- TPTP Testing Tools - Adiciona funcionalidades para a fase de testes no desenvolvimento de código. Disponibiliza editores de testes, desenvolvimento e execução de testes e ainda análises e relatórios da execução de testes.

- TPTP Tracing and Profiling Tools - Disponibiliza ferramentas para as fases de *tracing* e *profiling*, no desenvolvimento de uma aplicação. Suporta sistemas isolados ou distribuídos.

- TPTP Monitoring - O seu objectivo é fornecer os meios para reunir informação e analisá-la. Na prática é capaz de reunir um conjunto de informações, normalmente através de *logs*, de várias aplicações e fornecer ferramentas para analisá-los em conjunto para monitorizar e diagnosticar eventuais problemas. É capaz ainda de recolher informação da máquina onde as aplicações se encontram em funcionamento.

A figura 3.5 apresenta a arquitectura do projecto TPTP.

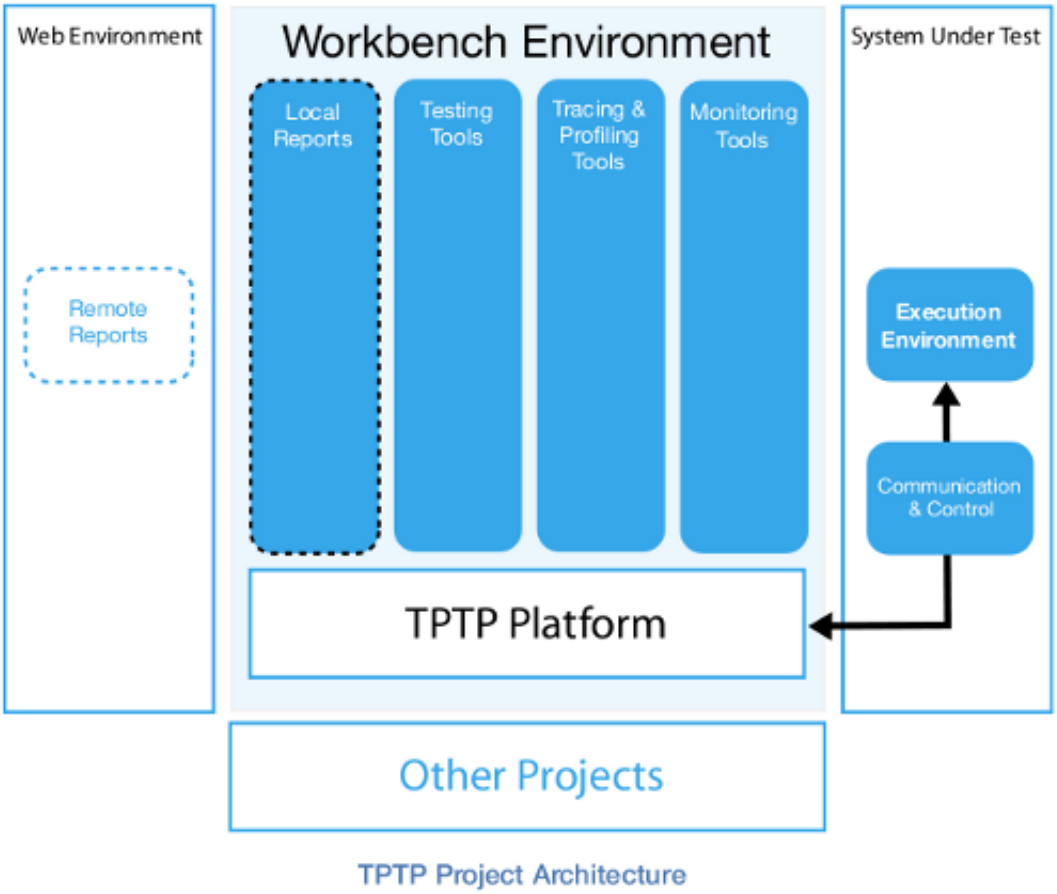

Figura 3.5: Arquitectura do projecto Eclipse TPTP[Fouf]

O objectivo da interoperabilidade deste projecto requereu que os modelos de dados utilizados fossem baseados em standards definidos. Assim, para o modelo de dados da área de testes foi usado o formato UML 2 Test Profile (U2TP) definido pelo *Object Management Group*. O modelo de dados usado internamente para representar os ficheiros de logs foi o Common Base Event (CBE), um standard proposto pelo consórcio OASIS. O modelo de dados escolhido para a área de *Trace* é muito próximo dos formatos JVMPI e JVMTI. Para modelo de dados de estatísticas foi usado um modelo que se mapeia facilmente para os formatos usado pelo JMX e pelo Microsoft Perfmon.

As funcionalidades revelantes para este projecto estão implementadas no sub-projecto de monitorização do TPTP[WC04]. Existem três componentes que se adequam àquilo que é pretendido com a solução: o GLA, o LTA e a análise de sintomas. O GLA é um componente facilmente extensível que permite importar informação a partir de virtualmente qualquer tipo de ficheiro de *log*. Este componente converte para um formato único adequado, o formato Common Base Event[IBM06], para permitir a análise conjunto da

informação de todas as fontes. O formato CBE é um formato genérico, definido pela IBM, que pode ser usado para expressar diversos tipos de situações, sendo adequado particularmente para expressar eventos de negócio e para situações de determinação de problemas. A utilização de formatos de dados diferentes por aplicações diferentes provoca dificuldade na altura de juntar toda a informação para obter uma visão única do sistema. Por outro lado, o uso de formatos específicos normalmente implica que quem utiliza a informação necessita de conhecer o formato e, normalmente, também alguma informação acerca do contexto. Com o formato CBE pretende-se que essas dependências caiam e o uso de um formato único e capaz de especificar qualquer tipo de evento seja capaz de melhorar os aspectos mencionados. Atendendo às vantagens mencionadas anteriormente, é de todo preferível o uso deste formato, tanto numa perspectiva de melhor escolha para a solução como também numa perspectiva de garantir a interoperabilidade no futuro. A utilização de formatos standard foi preferida pela empresa na execução deste projecto devido à necessidade de implementar uma solução capaz de interagir com ambientes heterogéneos, e que por isso, necessitam de um elemento comum que permita essa comunicação. O formato interno usado pelo Eclipse TPTP, como referido anteriormente, é o CBE, e esse foi mais um aspecto que pesou no sentido da adopção deste formato na solução. Uma figura com o resumo da estrutura do formato CBE é apresentado de seguida. A figura 3.6 apresenta a arquitectura do projecto TPTP.

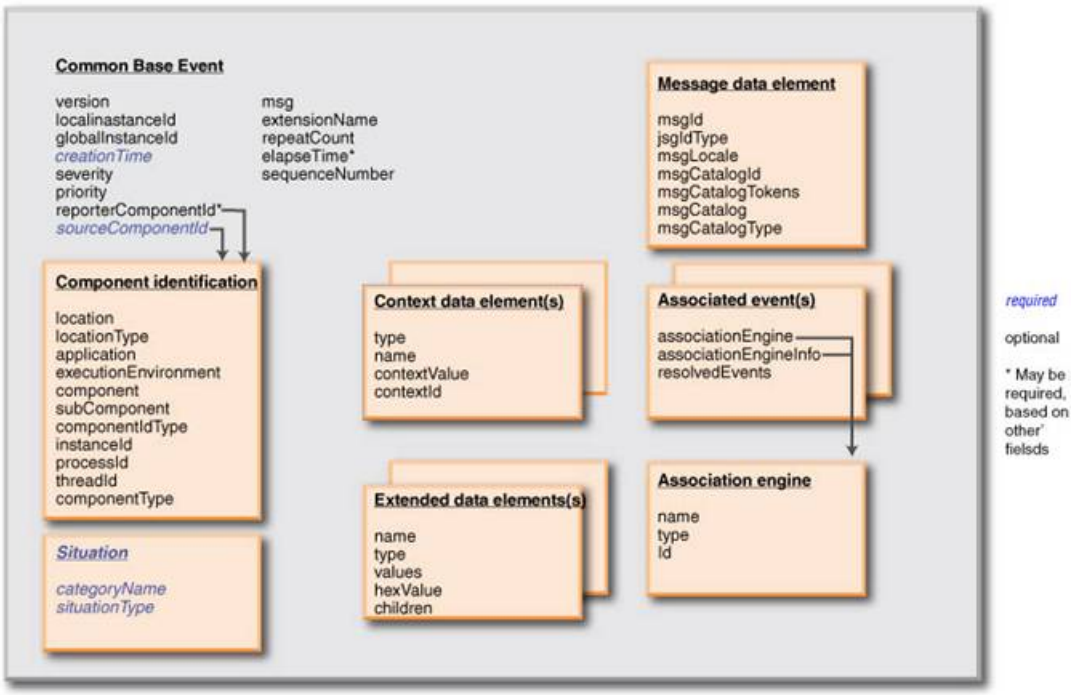

Figura 3.6: Esquema do formato CBE[IBM06]

Como se pode ver na figura, o formato contempla praticamente todo o tipo de informação acerca de eventos, sendo altamente adaptável, fruto da opcionalidade da maioria dos campos. A especificação do formato e as boas práticas de uso estão disponíveis em [IBM06].

O componente LTA do Eclipse TPTP permite visualizar a informação de uma forma uniformizada, uma vez que trabalha com registos no formato CBE, e tem funcionalidades de navegação, filtragem e destaque de informação, assim como automatização de acções através de *wizards* e outros elementos gráficos, que facilitam bastante ao seu uso.

O componente final no ciclo de tratamento de informação é o motor de análise. Este motor permite analisar através de expressões XPATH[2] e bases de dados de sintomas no formato XMI toda a informação importada. Nas bases de dados de sintomas estão descritos os sintomas de situações conhecidas de erros e possíveis recomendações para a solução dos mesmos. O que o motor de análise faz é pesquisar na informação que foi importada e convertida para o formato CBE os sintomas descritos na base de dados de sintomas. No final do processo de análise, são devolvidos os resultados com todos os sintomas conhecidos detectados e restante informação associada, como uma descrição textual da situação ou as possíveis acções de resolução.
Como referido anteriormente, este projecto articula-se com outros dois, que em conjunto irão criar a solução completa. Por isso mesmo a ligação entre as três peças é um aspecto bastante importante. No caso deste projecto, a interacção limita-se ao projecto "LAP Services". Embora nesta fase do projecto não seja objectivo criar uma ligação automática entre estes dois projectos, foi feito um pequeno levantamento das possibilidades de integração dessa funcionalidade. O Eclipse TPTP já dispõe de um mecanismo baseado em agentes capaz de executar acções remotas e recolher informação, que é enviada para o LTA. Apenas no futuro próximo está previsto implementar a funcionalidade de recolha automática de informação e ligação ao LAP Services, não fazendo parte dos requisitos deste projecto. No entanto fica o registo desta funcionalidade que foi alvo de uma pequena investigação.

3.3.2.3 Análise Crítica

Este projecto tem uma importância assinalável, tanto pela qualidade da solução desenvolvida mas também pelos serviços que oferece aos seus utilizadores. Por um lado este projecto disponibiliza soluções úteis que permitem acompanhar toda a vida de uma aplicação. Por outro lado permite ainda a quem desenvolve *software* não ficar agarrado a uma solução específica, seja comercial ou não, através do uso de modelos de dados *standards* que permitem manter a independência entre a informação e a aplicação utilizada.

[2] XML Path Language

3.3.3 Mylyn

3.3.3.1 Apresentação

O Mylyn é um *plugin* para a plataforma Eclipse que permite gerir tarefas[Foui]. Essas tarefas são unidades de trabalho, podendo assumir a forma de um *bug report* ou uma nota para o próprio.

3.3.3.2 Funcionalidades

O Mylyn permite a utilização de um repositório local ou de repositórios remotos. Estes últimos podem ser de vários tipos pois existem conectores para os tipos de repositórios mais utilizados. Existe ainda pontos de extensão para adicionar capacidades de comunicar com novos tipos de repositórios. Nos repositórios é guardada informação acerca das tarefas. O Mylyn permite a fácil edição destas tarefas, através de modificação local dos atributos, como por exemplo mudar o estado de "Aberta"para "Concluída", e posterior sincronização com o repositório. O trabalho nas tarefas é monitorizado pelo Mylyn, que permite associar os elementos modificados durante a resolução de determinada tarefa, assim como guardar informação relativamente ao tempo gasto[Fouk].

Existem várias API's disponibilizadas neste *plugin* que permitem uma fácil cooperação com outros *plugins* ou a personalização e adição de funcionalidades. Estas API's permitem expor facilmente os dados do Mylyn e a sua reutilização por outros elementos. Segue-se uma breve exposição das API's disponibilizadas pelo Mylyn[Fouj]:

- Commons API - Fornece acesso a funcionalidades comuns usadas por todo o *plugin*
- Discovery API - Trata os aspectos relacionados com o estabelecimento de ligação com os repositórios
- Tasks API - Providencia acesso à lista de tarefas existentes e aos métodos disponíveis para a sua edição
- Context API - Disponibiliza acesso ao comportamento definido a aplicar ao Eclipse, relativo a vários aspectos da apresentação de informação
- Team API - Funcionalidades que facilitam o uso de sistemas de controlo de versões
- Monitor API - Funcionalidades de monitorização do trabalho realizado
- Bugzilla API - Funcionalidades disponibilizadas relativas ao conector para o tipo de repositório Bugzilla

3.3.3.3 Análise Crítica

Este *plugin* incorpora-se bastante bem com a plataforma Eclipse e, por isso mesmo e pelas suas funcionalidades, é uma boa ferramenta que poderá contribuir para o aumento da eficiência do trabalho e/ou para facilitar a gestão de tarefas. As API's disponibilizadas conferem uma grande versatilidade a este *plugin* permitindo inclusive que seja possível utilizá-lo como solução para problemas para os quais não foi construído a pensar.

3.4 Análise e justificação da solução escolhida

A solução escolhida foi encontrada através de uma decisão que resultou da ponderação de vários pontos, tais como rapidez de execução da solução, qualidade do resultado final, flexibilidade, expansibilidade e custo. A solução escolhida foi aquela que foi considerada no global a que atingia os objectivos pretendidos com a melhor relação custo/qualidade.
Esta solução tem como grandes vantagens o facto de utilizar componentes de código aberto, o que permite diminuir tanto os custos de implementação assim como o tempo necessário para ter o produto em funcionamento. Ao utilizar componentes de código aberto está-se igualmente a utilizar código que é utilizado por uma maior base de utilizadores, o que aliado a um adequado suporte possibilita uma maior detecção e correcção de erros no software. Seguindo esse raciocínio e tendo em conta que o projecto Eclipse TPTP teve origem em 2004 é normal assumir que o código estará neste momento suficientemente maduro e também bastante robusto para ser considerado como uma solução de qualidade.
No entanto a solução escolhida inclui também alguns aspectos menos positivos, tais como a dificuldade de introduzir novas funcionalidades não previstas nas API's do produto e que podem quebrar a compatibilidade com versões futuras dos componentes desse mesmo produto. Nesse cenário a adição de novas funcionalidades não previstas necessitará do apoio dos desenvolvedores do Eclipse TPTP, através da criação das facilidades necessárias, o que devido aos recursos limitados deste tipo de projectos poderá muito bem ser considerado como algo de baixa prioridade, e dessa forma atrasar a introdução de novas funcionalidades. O lançamento e a actualização para novas versões do Eclipse TPTP pode também incluir alterações à arquitectura do projecto e por isso exigir a adaptação da solução desenvolvida à nova versão dos componentes, e dessa forma introduzir um maior custo a médio/longo prazo para adaptação das funcionalidades acrescentadas.
Como já referido neste relatório, não existem produtos comparáveis ao Eclipse TPTP a nível de funcionalidades. No entanto existia a alternativa de desenvolver uma solução inteiramente de raiz, ou em alternativa, uma solução híbrida, envolvendo menos componentes do Eclipse TPTP e compensando através da adição de funcionalidades desenvolvidas pelo autor. Essas alternativas não foram escolhidas pois embora resolvam alguns

dos aspectos negativos da solução escolhida introduziam eles próprios desvantagens incomportáveis. No primeiro caso, uma solução inteiramente desenvolvida certamente teria custos muito superiores para ter um nível de versatilidade e expansibilidade equiparável ao Eclipse TPTP. Essa solução demoraria igualmente bastante mais tempo para ser executada. A segunda alternativa foi descartada pois foi considerado que poderia introduzir custos adicionais, principalmente a nível de integração dos componentes da solução, sem que fosse claro que existisse aumento na qualidade da solução final.

3.5 Conclusão

Neste capítulo foi apresentada a proposta de solução e as tecnologias que dela farão parte. As decisões tomadas foram apresentadas e justificadas e o trabalho a realizar foi descrito de uma forma sumária. No próximo capítulo a implementação da solução aprofundará cada um dos componentes e a sua forma de integração na solução final.

Capítulo 4

Implementação e Resultados

Neste capítulo é descrita a um nível mais baixo a solução proposta no capítulo anterior assim como os detalhes de implementação da solução.

4.1 Desenvolvimento de adaptadores de suporte às aplicações da I2S para o GLA

O Eclipse TPTP dispõe de um componente, o GLA, capaz de converter informação de diversas fontes diferentes e com formatos diferentes para um formato único e adequado para ser trabalhado pela aplicação LTA. Para satisfazer as necessidades da I2S neste campo foram criados três adaptadores para os tipos de *logs* usados nas aplicações da I2S. Assim foram criados conversores para os seguintes logs:

- JMeter - Logs da aplicação JMeter, aplicação de análise de performance e testes de carga. O formato de logs suportado é do tipo:
  `(\d{4}/\d{2}/\d{2}\s\d{2}:\d{2}:\d{2})\sINFO\s{1,2}-(.*)`
  As informações estão codificadas no formato *AAAA/MM/DD HH:MM:ss SEVERIDADE - MSG*

- Log4J - Logs das aplicações da I2S (que utilizam o sistema de *logging* Log4J. Converte logs no formato:
  `(\d{4}-\d{2}-\d{2}\s\d{2}:\d{2}:\d{2},\d{3})`
  `\s\[(\w+)\]\s(DEBUG)\s(.*)`
  A informação neste tipo de *logs* aparece no formato *AAAA-MM-DD HH:MM:ss:SSS SEVERIDADE TIPO MSG*

- WAS - Logs do servidor de aplicações usado pela I2S. Logs deste tipo contêm entradas com o formato:

`[DD-MM-AAAA HH:MM:ss:SSS TMZ] código msg`

Este tipo de registo apresenta a data com indicação da zona horária onde foi gerado, um código de oito caracteres que identifica a situação ocorrida de a mensagem por extenso.

Com o desenvolvimento destes adaptadores passou a ser possível por parte da I2S analisar em conjunto informação de diversas fontes usadas nos seus sistemas.

Um exemplo do processo de criação destes adaptadores encontra-se documentado no Anexo C.

4.2 Divisão lógica de componentes do TPTP - Monitorização

Como já foi apresentado anteriormente neste relatório, a implementação da solução final utiliza componentes do Eclipse TPTP. Esses componentes encontram-se dispersos pela totalidade do projecto TPTP, e por isso mesmo foi útil realizar um estudo da plataforma e documentar os componentes que serão usados. Será útil apresentar uma divisão lógica desses elementos com vista à especificação de quais os *plugins* que farão parte da solução. De lembrar que a utilização destes elementos do sub-projecto de monitorização do TPTP requer a utilização dos componentes constituintes da plataforma do TPTP.
Dividindo os elementos a usar pela sua funcionalidade, teremos três grupos de elementos.

4.2.1 GLA - Generic Log Analyzer

O GLA é constituído por vários componentes, alguns dos quais relevantes para a implementação e que serão descritos de seguida.

4.2.1.1 Vistas

A vista disponibilizada pelo GLA é apresentada na tabela 4.1.

Plugin	org.eclipse.tptp.lta.gla.ui.epi
Package	org.eclipse.hyades.logging.adapter.ui.internal.presentation
Classe	AdapterEditor
Função	Editor para definir um adaptador do GLA
Dependências	org.eclipse.ui;bundle-version="[3.4.0,4.0.0)", org.eclipse.core.runtime;bundle-version="[3.4.0,4.0.0)", org.eclipse.hyades.lta.logging.adapter.ui;bundle-version="[4.5.0,5.0.0)", org.eclipse.core.resources;bundle-version="[3.4.0,4.0.0)"

Tabela 4.1: Descrição das Vistas Eclipse TPTP - GLA

O *wizard* providenciado pelo GLA é descrito na tabela 4.2.

4.2.1.2 Wizards

Plugin	org.eclipse.hyades.logging.adapter.ui
Package	org.eclipse.hyades.logging.adapter.ui.internal.presentation
Classe	AdapterModelWizard
Função	Wizard para criar um novo adaptador do GLA
Dependências	-

Tabela 4.2: Descrição Wizard Eclipse TPTP - GLA

Os componentes de contexto constituintes do GLA são descritos na tabela 4.3.

4.2.1.3 Componentes de contexto

Plugin	org.eclipse.hyades.logging.adapter.ui
Package	org.eclipse.hyades.logging.adapter.ui.extension.internal.impl
Classes	Todas as definidas neste package
Função	Objectos criados que permitem definir o contexto de um adaptador GLA
Dependências	-

Tabela 4.3: Componentes de contexto GLA

4.2.2 LTA - Log and Trace Analyzer

O LTA disponibiliza vários componentes de interesse para este projecto. Os seus componentes mais relevantes serão apresentados de seguida.

4.2.2.1 Vistas

A localização das várias classes que constituem a vista de propriedades do LTA é apresentada de forma resumida na tabela 4.4.

A vista para os *logs* é descrita na tabela 4.5.

O navegador de contéudo pertencente ao LTA é apresentado na tabela 4.6.

4.2.2.2 Wizards

O *wizard* para importar logs é descrito na tabela 4.7.

O *wizard* para criar uma nova correlação é apresentado na tabela 4.8.

4.2.3 Acções

Várias acções possibilitam simplificar a realização de tarefas comuns e frequentes no uso da aplicação.

Plugin	org.eclipse.tptp.platform.lta.log.views
Package	org.eclipse.tptp.platform.log.views.internal.views
Classe	Várias classes que definem a vista de cada uma das *tabs* da área de propriedades
Função	Vista de propriedades de um evento
Dependências	org.eclipse.ui;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.ide;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.views;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.editors;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.console;bundle-version="[3.1.0,4.0.0)", org.eclipse.hyades.lta.analysis.engine;bundle-version="[4.2.0,5.0.0)", org.eclipse.hyades.lta.resources.database;bundle-version="[4.1.0,5.0.0)", org.eclipse.ui.views.properties.tabbed;bundle-version="[3.2.0,4.0.0)", org.eclipse.tptp.platform.common.ui;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.platform.lta.la.core;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.platform.models.lta;bundle-version="[4.5.0,5.0.0)", org.eclipse.tptp.platform.lta.common; visibility:=reexport;bundle-version="[4.5.0,5.0.0)"

Tabela 4.4: Descrição das Vistas de Eclipse TPTP - LTA (1)

Plugin	org.eclipse.tptp.platform.lta.log.views
Package	org.eclipse.tptp.platform.log.views.internal.views
Classe	LogViewer
Função	Vista do conteúdo dos ficheiros de *log*
Dependências	org.eclipse.ui;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.ide;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.views;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.editors;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.console;bundle-version="[3.1.0,4.0.0)", org.eclipse.hyades.lta.analysis.engine;bundle-version="[4.2.0,5.0.0)", org.eclipse.hyades.lta.resources.database;bundle-version="[4.1.0,5.0.0)", org.eclipse.ui.views.properties.tabbed;bundle-version="[3.2.0,4.0.0)" org.eclipse.tptp.platform.common.ui;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.platform.lta.la.core;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.platform.models.lta;bundle-version="[4.5.0,5.0.0)", org.eclipse.tptp.platform.lta.common; visibility:=reexport;bundle-version="[4.5.0,5.0.0)"

Tabela 4.5: Descrição das Vistas de Eclipse TPTP - LTA (2)

A acção para abrir a vista de propriedades é descrita de forma resumida na tabela 4.9.

A acção para abrir uma vista para o *log* encontra-se resumida na tabela 4.10.

Plugin	org.eclipse.tptp.platform.lta.log.views
Package	org.eclipse.tptp.platform.log.views.internal.navigator
Classe	LogNavigator
Função	Vista do navegador de conteúdo
Dependências	org.eclipse.ui;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.ide;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.views;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.editors;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.console;bundle-version="[3.1.0,4.0.0)", org.eclipse.hyades.lta.analysis.engine;bundle-version="[4.2.0,5.0.0)", org.eclipse.hyades.lta.resources.database;bundle-version="[4.1.0,5.0.0)", org.eclipse.ui.views.properties.tabbed;bundle-version="[3.2.0,4.0.0)", org.eclipse.tptp.platform.common.ui;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.platform.lta.la.core;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.platform.models.lta;bundle-version="[4.5.0,5.0.0)", org.eclipse.tptp.platform.lta.common; visibility:=reexport;bundle-version="[4.5.0,5.0.0)"

Tabela 4.6: Descrição das Vistas de Eclipse TPTP - LTA (3)

Plugin	org.eclipse.tptp.monitoring.lta.logui
Package	org.eclipse.tptp.monitoring.logui.internal.wizards
Classe	ImportLogWizard
Função	Wizard para importar *logs*
Dependências	org.eclipse.ui;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.forms;bundle-version="[3.2.0,4.0.0)", org.eclipse.core.runtime;bundle-version="[3.2.0,4.0.0)", org.eclipse.hyades.lta.logging.parsers;bundle-version="[4.2.0,5.0.0)", com.ibm.icu;bundle-version="[3.4.0,4.0.0)", org.eclipse.ui.ide;bundle-version="[3.2.0,4.0.0)", org.eclipse.tptp.platform.common;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.platform.common.ui;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.platform.lta.log.views;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.monitoring.lta.la.core;bundle-version="[4.5.0,5.0.0)", org.eclipse.tptp.platform.models.lta;bundle-version="[4.5.0,5.0.0)", org.eclipse.tptp.platform.common.ui.trace;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.platform.lta.common;bundle-version="[4.5.0,5.0.0)"

Tabela 4.7: Descrição dos Wizards de Eclipse TPTP - LTA (1)

4.2.4 Análise de sintomas

A análise de sintomas é o componente que permite realizar a análise dos ficheiros de *logs*. Este componente é apresentado de forma resumida de seguida.

Plugin	org.eclipse.tptp.platform.lta.log.views
Package	org.eclipse.tptp.platform.log.views.internal.navigator.wizards
Classe	NewCorrelationWizard
Função	Wizard para criar uma nova correlação *logs*
Dependências	org.eclipse.ui;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.ide;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.views;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.editors;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.console;bundle-version="[3.1.0,4.0.0)", org.eclipse.hyades.lta.analysis.engine;bundle-version="[4.2.0,5.0.0)", org.eclipse.hyades.lta.resources.database;bundle-version="[4.1.0,5.0.0)", org.eclipse.ui.views.properties.tabbed;bundle-version="[3.2.0,4.0.0)", org.eclipse.tptp.platform.common.ui;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.platform.lta.la.core;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.platform.models.lta;bundle-version="[4.5.0,5.0.0)", org.eclipse.tptp.platform.lta.common; visibility:=reexport;bundle-version="[4.5.0,5.0.0)"

Tabela 4.8: Descrição dos Wizards de Eclipse TPTP - LTA (2)

Plugin	org.eclipse.tptp.platform.lta.log.views
Package	org.eclipse.tptp.platform.log.views.internal.actions
Classe	OpenPropertyViewAction
Função	Abre a vista de propriedades *logs*
Dependências	org.eclipse.ui;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.ide;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.views;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.editors;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.console;bundle-version="[3.1.0,4.0.0)", org.eclipse.hyades.lta.analysis.engine;bundle-version="[4.2.0,5.0.0)", org.eclipse.hyades.lta.resources.database;bundle-version="[4.1.0,5.0.0)", org.eclipse.ui.views.properties.tabbed;bundle-version="[3.2.0,4.0.0)", org.eclipse.tptp.platform.common.ui;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.platform.lta.la.core;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.platform.models.lta;bundle-version="[4.5.0,5.0.0)", org.eclipse.tptp.platform.lta.common; visibility:=reexport;bundle-version="[4.5.0,5.0.0)"

Tabela 4.9: Descrição dos Acções de Eclipse TPTP - LTA (1)

4.2.4.1 Vistas

A vista do editor de bases de dados de sintomas, descrita na tabela 4.11, permite modificar as bases de dados de sintomas.

A tabela 4.12 resume as informações acerca da vista dos resultados das análises efectuadas aos ficheiros de *log*.

Plugin	org.eclipse.tptp.platform.lta.log.views
Package	org.eclipse.tptp.platform.log.views.internal.actions
Classe	OpenLogViewAction
Função	Abre a vista para o *log logs*
Dependências	org.eclipse.ui;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.ide;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.views;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.editors;bundle-version="[3.2.0,4.0.0)", org.eclipse.ui.console;bundle-version="[3.1.0,4.0.0)", org.eclipse.hyades.lta.analysis.engine;bundle-version="[4.2.0,5.0.0)", org.eclipse.hyades.lta.resources.database;bundle-version="[4.1.0,5.0.0)", org.eclipse.ui.views.properties.tabbed;bundle-version="[3.2.0,4.0.0)", org.eclipse.tptp.platform.common.ui;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.platform.lta.la.core;bundle-version="[4.3.0,5.0.0)", org.eclipse.tptp.platform.models.lta;bundle-version="[4.5.0,5.0.0)", org.eclipse.tptp.platform.lta.common; visibility:=reexport;bundle-version="[4.5.0,5.0.0)"

Tabela 4.10: Descrição dos Acções de Eclipse TPTP - LTA (2)

Plugin	org.eclipse.tptp.monitoring.lta.symptom.editor
Package	org.eclipse.tptp.symptom.internal.presentation
Classe	SymptomEditor
Função	Vista do editor de bases de dados de sintomas
Dependências	org.eclipse.ui;bundle-version="[3.4.0,4.0.0)", org.eclipse.core.runtime;bundle-version="[3.4.0,4.0.0)", org.eclipse.hyades.lta.sdb; visibility:=reexport;bundle-version="[4.5.0,5.0.0)", org.eclipse.ui.forms;bundle-version="[3.3.0,4.0.0)", org.eclipse.emf.edit;bundle-version="[2.2.0,3.0.0)", org.eclipse.emf.edit.ui;bundle-version="[2.2.0,3.0.0)", org.eclipse.emf.common.ui;bundle-version="[2.2.0,3.0.0)", org.eclipse.emf.ecore.edit;bundle-version="[2.2.0,3.0.0)", org.eclipse.ui.ide;bundle-version="[3.4.0,4.0.0)", org.eclipse.ui.workbench.texteditor;bundle-version="[3.4.0,4.0.0)", org.eclipse.tptp.platform.common.ui;bundle-version="[4.4.0,5.0.0)", org.eclipse.ui.editors;bundle-version="[3.4.0,4.0.0)", org.eclipse.tptp.platform.models.lta;bundle-version="[4.5.0,5.0.0)"

Tabela 4.11: Descrição das Vistas de Eclipse TPTP - Editor de Sintomas

Plugin	org.eclipse.tptp.platform.analysis.core.ui
Package	org.eclipse.tptp.platform.analysis.core.ui.views
Classe	ResultsFrameView
Função	Vista do resultado da análise efectuada
Dependências	-

Tabela 4.12: Descrição das Vistas de Eclipse TPTP - Resultados de Análise

4.3 Estudo sobre expansibilidade do Eclipse TPTP

Sendo este um projecto a médio prazo, composto por várias iteracções, foi considerado relevante nesta fase o levantamento das capacidades de expansibilidade, nomeadamente no que toca à adição de novas funcionalidades e personalização das já existentes, nos casos relevantes para a solução da I2S. Estando implementado o suporte para os ficheiros de *logs* usados pela I2S, houve interesse em perceber como funcionaria a adição de novos tipos de correlação e novos motores de análise. Nesse sentido é apresentado de seguida o resultado desse estudo efectuado, entrar em detalhes de algoritmos a utilizar, uma vez que esses ainda não estão definidos.

4.3.1 Adição de um novo modo de correlação

A adição de um novo tipo de correlação pode ser efectuada criando um *plugin* para a plataforma Eclipse, usando o ponto de extensão do TPTP:
`org.eclipse.hyades.trace.ui.logInteractionView`
e criando uma classe que implemente a interface **ILogRecordCorrelationEngine**, implementando os seus métodos e especificando o algoritmo de correlação no método

```
public void correlate(
   CorrelationContainerProxy correlationContainerProxy,
   EList logFiles)
```

No caso da adição de novos motores de análise, é possível incorporá-los criando um *plugin* que se liga ao ponto de extensão
`org.eclipse.hyades.analysis.engine.LogAnalyzer`
e definindo uma classe que implementa a interface **org.eclipse.hyades.analysis.engine.ILogAnalyzer**, e definindo o algoritmo de análise no método

```
public String analyze(Object aObject, IAnalysisMonitor monitor)
```

4.4 Análise de performance de BD Derby

O Eclipse TPTP tem suporte para a utilização de uma base de dados, nos casos em que a informação a importar seja razoavelmente grande. É recomendável que, caso o tamanho combinado dos ficheiros importados excedam 4% do tamanho do heap da máquina virtual Java, se utilize esta funcionalidade.[IBMb] Isto deve-se ao facto de que a informação importada pela aplicação, utilizando o modo normal de importação, fica armazenada em ficheiros XMI no workspace da aplicação. Estes ficheiros baseados em XML facilmente atingem grandes tamanhos e penalizam a pesquisa e análise de informação. Por isso em alguns casos é justificável e até recomendável a utilização de uma base de dados para

guardar a informação. O TPTP apenas suporta nativamente bases de dados baseadas em Apache Derby (incluindo IBM Clouscape e Java DB). Este tipo de base de dados pode ser usada em dois modos diferentes de funcionamento: no modo embebido e no modo cliente/servidor. No modo embebido, esta base de dados tem algumas limitações tais como suporte apenas para acesso local e apenas de uma única aplicação [Foub], executando na máquina virtual JAVA, mas tem a vantagem de consumir menos recursos e ser mais facilmente configurado. No tradicional modo cliente/servidor, é criado um servidor desta BD com capacidades multi-utilizador e utilização remota, o que não obriga a que o servidor se encontre na mesma máquina que a aplicação que lhe acede [Foua]. Devido às diferentes características dos dois modos de funcionamento, foi considerado relevante realizar testes para perceber quais as diferenças de performance na utilização dos dois modos e qual o mais adequado. Para esse efeito foram criadas duas bases de dados novas com os dois modos de funcionamento, de acordo com os requisitos da aplicação [IBMa]. Foram feitas importações de dados usando o Eclipse TPTP, usando vários ficheiros de 10 e 20 Mbytes. Os resultados dos valores médios desses testes são apresentados de seguida.

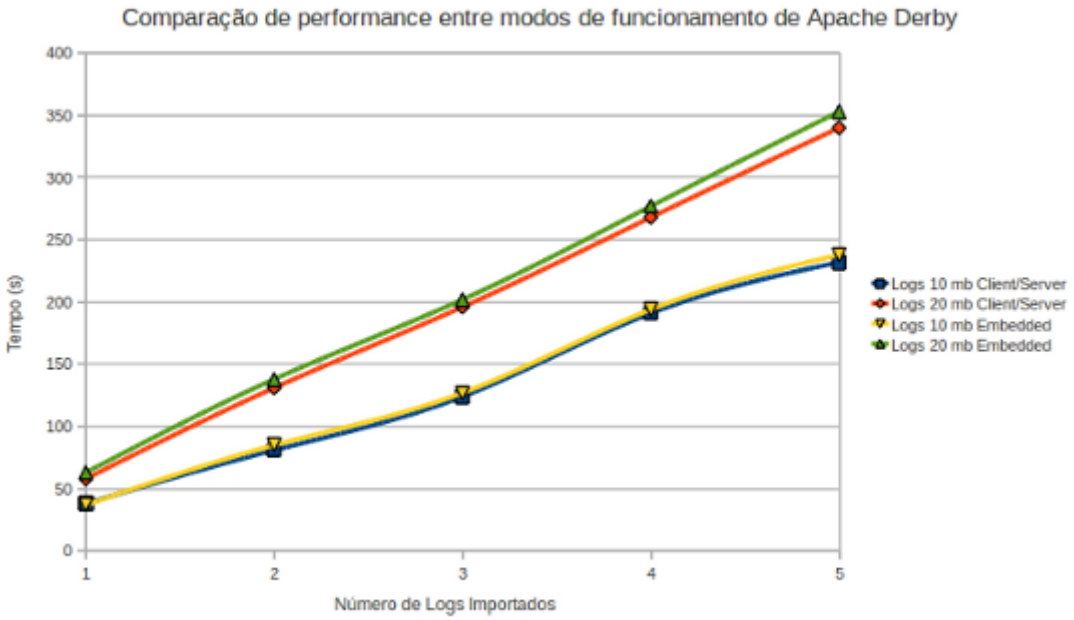

Figura 4.1: Comparação de performance entre modos de funcionamento de Apache Derby

4.4.1 Conclusão

Como seria de esperar, quando o motor da base de dados corre independentemente, a sua performance aumenta ligeiramente. Muito embora no global os ganhos não sejam muito consideráveis, há que considerar que a juntar a esse pequeno ganho de performance

existe ainda a vantagem de o motor da base de dados deixar de consumir memória da aplicação. Por essas razões o modo de funcionamento da base de dados mais eficiente para este projecto é o modo tradicional de cliente/servidor.

4.5 Dificuldades encontradas

A maior dificuldade encontrada na implementação deste projecto foi sem dúvida o cancelamento do suporte ao sub-projecto de Monitorização do Eclipse TPTP. O *timing* dessa decisão foi o pior possível relativamente à concretização deste projecto pois ocorreu numa altura em que o desenho e o planeamento da solução já tinha terminado e a implementação estava a arrancar. Esse anúncio obrigou a repensar todo o projecto, a pesquisar por outras alternativas, mesmo aquelas que são de um nível inferior mas que poderiam adaptar-se com um certo grau de esforço, e a focar o projecto em aspectos mais abstractos. Por outro lado com o anúncio colocou-se também a hipótese de utilizar a mesma base de uma forma mais liberal, alterando o código de acordo com as nossas necessidades pois deixou de ser necessário manter a estrutura do projecto intacta porque o suporte à ferramenta não existe mais. Posto isto, actualmente o projecto está a ser repensado e a implementação deslocou o seu foco para os aspectos que não estão dependentes da tecnologia, nomeadamente a definição da interface gráfica, de casos de uso e das *user stories*, trabalho esse que pode ser consultado no anexo B. Outra grande dificuldade encontrada na fase de implementação foi a documentação das ferramentas usadas, que nalguns casos não era a melhor e que com pequenos detalhes que mudam entre versões por vezes perdia-se imenso tempo. Também o facto de alguns projectos terem a documentação espalhada por diversos elementos sem ligação dificultou a tarefa de pesquisa e implementação.

4.6 Discussão

A solução proposta, após ter sido implementada, permite atingir os objectivos inicialmente definidos. Esta solução permite importar informação de várias fontes de dados através dos adaptadores desenvolvidos para o GLA. Estes adaptadores foram desenvolvidos tendo em conta o mapeamento entre o formato CBE e o formato original das fontes de dados, o qual foi proposto e aprovado. A solução permite ainda, através do LTA e do motor de análise do Eclipse TPTP, analisar a informação importada, e em conjunto com uma base de dados de sintomas adequada, detectar situações de erros e providenciar mais informação e/ou possíveis soluções. Esta solução permite monitorizar tanto as aplicações como as infraestructuras que as suportam. Permite ainda visualizar, filtrar, resumir e trabalhar as informações recolhidas pelo sistema de registo de eventos de negócio e de sistema. Está também projectada para num futuro próximo, aquando da integração do Mylyn, permitir a ligação ao sistema de suporte da I2S.

A solução pode ser classificada, utilizando a matriz de níveis de maturidade de aplicações de *Autonomic Computing* apresentada na figura 2.3 como uma aplicação que se encontra no nível médio. Esta análise foi realizada tendo em conta que a solução inclui funcionalidades de recolha de informação de diversas fontes e inclui ainda tecnologias de correlação, embora neste caso estas sejam bastante simples, o que corresponde ao nível 3. A nível de controlo implementado a aplicação classifica-se no nível C ao permitir obter informação de múltiplas instâncias de recursos homogéneos, tais como um conjunto de servidores de aplicações.
Esta solução será num futuro próximo incluída nas máquinas de desenvolvimento da I2S afim de começar a criação de uma base de dados de sintomas interna às aplicações desenvolvidas pela I2S.

Capítulo 5

Conclusões e Trabalho Futuro

Neste capítulo é feita uma análise crítica ao trabalho realizado, fazendo uma retrospectiva do projecto, observando os objectivos atingidos e quais os próximos passos deste projecto no futuro.

5.1 Retrospectiva

Este é sem dúvida um projecto ambicioso, mas também igualmente necessário. A necessidade de delegar esforço de análise e monitorização é agora tão necessário como foi a uns anos atrás delegar aspectos de negócio por parte de grande número de indústrias. É sem dúvida uma tarefa custosa, que necessitará de muito esforço e empenho, mas que produzirá no futuro grandes ganhos a quem estiver disposto a fazer esse investimento agora. A I2S está empenhada em fazer esse investimento em nome da qualidade dos seus serviços, para com os seus clientes. Realizar esta tese em âmbito empresarial e numa área tão recente como é o *Autonomic Computing* foi um desafio interessante e que permitiu aplicar várias capacidades adquiridas ao longo do curso.

Este projecto envolveu muita investigação e com certeza envolverá também bastante nas próximas iterações, o que é perfeitamente normal e aceitável quando se está a trabalhar na linha da frente de um novo conceito. Importa referir que essa característica aliada ao cancelamento do suporte da ferramenta escolhida para a solução e ainda o facto desta fase do projecto terminar daqui a cerca de dois meses fez com que esta tese neste momento não reflicta totalmente o trabalho deste projecto. No entanto há a assinalar o enorme enriquecimento que o projecto LAP teve no autor desta tese, adquirindo experiência e capacidades pessoais e profissionais no mundo empresarial, que de outra forma seriam difíceis de obter.

5.2 Satisfação dos Objectivos

A satisfação dos objectivos deste projecto foi bastante boa na medida em que o que os objectivos que foram inicialmente estabelecidos, e que eram mais teóricos, foram ultrapassados tendo o trabalho realizado não se limitado apenas ao desenho de uma solução mas também à implementação dessa mesma solução. A utilização dos componentes do Eclipse TPTP permitiu economizar tempo que foi usado para contemplar funcionalidades que não estavam previstas até uma fase adiantada do projecto, como foi o caso da incorporação de um *issue tracker*. Não tivesse existido o anúncio do fim do projecto Eclipse TPTP, às quais tanto o autor desta tese como a empresa I2S foram alheias, e a implementação teria avançado mais e a aplicação poderia estar já em avançada fase de implementação.

5.3 Trabalho Futuro

No futuro imediato será necessário antes de mais decidir qual a abordagem a tomar no que toca ao uso do Eclipse TPTP. Todo o restante futuro trabalho será influenciado por essa decisão. Caso se decida avançar com o uso do Eclipse TPTP como estava inicialmente previsto os próximos passos deverão passar por concluir a implementação. Posteriormente poderão ser incluídas funcionalidades interessantes e que não foram ainda consideradas formalmente, tais como adicionar um motor de análise com mais funcionalidades e optimizado para o que a I2S pretende. Outra funcionalidade muito interessante e útil para levar esta aplicação para um nível superior de maturidade seria a capacidade de realização automática de acções de correcção e/ou recuperação. Outras funcionalidades menos prioritárias também poderão ser consideradas, como criar novos automatismos de importação de *logs*, melhoramento de performance e possivelmente substituir a funcionalidade de *Large Log Support* do LTA por uma mais eficiente e rápida computacionalmente. Paralelamente com este trabalho também deverá ser iniciado o processo de criação de bases de dados de sintomas para as aplicações da I2S, de forma a melhorar o suporte e a capacidade de detecção de situações anormais respeitantes aos sistemas da I2S.

Referências

[eSSa] I2S Informática Sistemas e Serviços SA. Apresentação da i2s. Disponível em http://www.i2s.pt/i2ssite/Empresa/download_doc.asp, acedido a última vez em 05 de Junho de 2010.

[eSSb] I2S Informática Sistemas e Serviços SA. Companhias de seguros - fundos de pensões. Disponível em http://www.i2s.pt/i2ssite/Clientes/Fundos.asp, acedido a última vez em 05 de Junho de 2010.

[eSSc] I2S Informática Sistemas e Serviços SA. Companhias de seguros - não vida. Disponível em http://www.i2s.pt/i2ssite/Clientes/NaoVida.asp, acedido a última vez em 05 de Junho de 2010.

[eSSd] I2S Informática Sistemas e Serviços SA. Companhias de seguros - vida. Disponível em http://www.i2s.pt/i2ssite/Clientes/Vida.asp, acedido a última vez em 05 de Junho de 2010.

[Foua] The Apache Foundation. Client/server environment. Disponível em http://db.apache.org/derby/docs/dev/getstart/cgsquck13780.html, acedido a última vez em 23 de Junho de 2010.

[Foub] The Apache Foundation. Embedded environment. Disponível em http://db.apache.org/derby/docs/dev/getstart/cgsquck35643.html, acedido a última vez em 23 de Junho de 2010.

[Fouc] The Eclipse Foundation. Eclipse platform - technical overview. Disponível em www.eclipse.org/whitepapers/eclipse-overview.pdf, acedido a última vez em 06 de Junho de 2010.

[Foud] The Eclipse Foundation. Eclipse platform technical oerview. Disponível em http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.html, acedido a última vez em 06 de Junho de 2010.

[Foue] The Eclipse Foundation. Eclipse test and performance tools platform faq. Disponível em http://www.eclipse.org/tptp/home/documents/webcontent/faq.php, acedido a última vez em 12 de Junho de 2010.

[Fouf] The Eclipse Foundation. Eclipse test and performance tools platform project datasheet. Disponível em http://www.eclipse.org/tptp/home/project_info/general/tptp_datasheet.pdf, acedido a última vez em 12 de Junho de 2010.

[Foug] The Eclipse Foundation. Eclipse test and performance tools platform project structure. Disponível em http://www.eclipse.org/tptp/home/project_info/general/EclipseTPTPProjectStructure.pdf, acedido a última vez em 12 de Junho de 2010.

[Fouh] The Eclipse Foundation. Faq where did eclipse come from? Disponível em http://wiki.eclipse.org/FAQ_Where_did_Eclipse_come_from%3F, acedido a última vez em 06 de Junho de 2010.

[Foui] The Eclipse Foundation. Mylyn faq. Disponível em http://wiki.eclipse.org/index.php/Mylyn/FAQ, acedido a última vez em 13 de Junho de 2010.

[Fouj] The Eclipse Foundation. Mylyn integrator reference. Disponível em http://wiki.eclipse.org/index.php/Mylyn/Integrator_Reference, acedido a última vez em 13 de Junho de 2010.

[Fouk] The Eclipse Foundation. Mylyn user guide. Disponível em http://wiki.eclipse.org/index.php/Mylyn/User_Guide, acedido a última vez em 13 de Junho de 2010.

[Foul] The Eclipse Foundation. Nasa uses eclipse for interplanetary operations. Disponível em http://www.eclipse.org/community/casestudies/NASAfinal.pdf, acedido a última vez em 06 de Junho de 2010.

[Foum] The Eclipse Foundation. Rich client platform (rcp) applications. Disponível em http://www.eclipse.org/community/rcp.php, acedido a última vez em 06 de Junho de 2010.

[IBMa] IBM. Creating the database and database tables. Disponível em http://publib.boulder.ibm.com/infocenter/iadthelp/v7r0/index.jsp?topic=/org.eclipse.tptp.platform.doc.user/tasks/tlglogdb.html, acedido a última vez em 23 de Junho de 2010.

[IBMb] IBM. Guidelines for using large log support. Disponível em http://publib.boulder.ibm.com/infocenter/iadthelp/v7r0/topic/org.eclipse.tptp.platform.doc.user/ref/rlglogguideline.html, acedido a última vez em 23 de Junho de 2010.

[IBM06] IBM. Best practices for the common base event and common event infrastructure, 2006. Disponível em http://download.boulder.ibm.com/ibmdl/pub/software/dw/library/autonomic/books/cbepractice/index.htm, acedido a última vez em 30 de Maio de 2010.

[JLHNY04] Bart Jacob, Richard Lanyon-Hogg, Devaprasad K. Nadgir e Amr F. Yassin. *A Pratical Guide to the IBM Autonomic Computing Toolkit*. IBM Redbooks, First edition, 2004.

[KC03] J. O. Kephart e D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, January 2003.

[kn:05] An architectural blueprint for autonomic computing. Technical report, IBM, Junho 2005.

[Mic06] Brenda M. Michelson. Event-driven architecture overview, Fevereiro 2006. Disponível em www.omg.org/soa/Uploaded%20Docs/EDA/bda2-2-06cc.pdf.

[PBB+02] David Patterson, Aaron Brown, Pete Broadwell, George Candea, Mike Chen, James Cutler, Patricia Enriquez, Armando Fox, Emre Kiciman, Matthew Merzbacher, David Oppenheimer, Naveen Sastry, William Tetzlaff, Jonathan Traupman e Noah Treuhaft. Recovery oriented computing (roc): Motivation, definition, techniques,. Technical report, Berkeley, CA, USA, 2002.

[Shi03] Jack Shirazi. *Java Performance Tuning*. O'Reilly, second edition edition, 2003.

[STA09] Hasan Sözer, Bedir Tekinerdogan e Mehmet Aksit. Flora: A framework for decomposing software architecture to introduce local recovery. *Software: Practice and Experience*, pages 869–889, Julho 2009.

[WC04] Daniel Worden e Nicholas Chase. Using the generic log adapter with the log and trace analyzer, Junho 2004.

REFERÊNCIAS

Anexo A

Análise de Funcionalidades de Filtragem, Ordenação e Destaque de entradas existentes no *Log and Trace Analyzer*

As funcionalidades mais importantes do LTA foram analisadas e são descritas de seguida.

A.1 Filtragem

A filtragem de logs é obtida através da aplicação de uma regra ou de um conjunto de regras definidas num filtro. A criação de filtros pode ser efectuada clicando no botão "Manage Filters...", na barra de ferramentas do LTA, ou em alternativa clicando na seta do lado direito do botão e escolhendo a opção "Manage Filters...".A filtragem de logs é obtida através da aplicação de uma regra ou de um conjunto de regras definidas num filtro. A criação de filtros pode ser efectuada clicando no botão "Manage Filters...", na barra de ferramentas do LTA, ou em alternativa clicando na seta do lado direito do botão e escolhendo a opção "Manage Filters...", como demonstrado na figura A.1.

Após clicar em "Manage Filters..." é aberta uma janela onde é possível adicionar, editar e remover filtros. Para criar um filtro clica-se em "New...", e na nova janela dá-se um nome ao filtro, selecciona-se o separador "Advanced" e clica-se no botão "Add..." do lado direito, como visível na figura A.2.

Após clicar no botão "Add..." pode ser construída uma regra com base nos atributos padrão do formato Common Base Event (excepto os campos ExtendedDataElement), operadores matemáticos de igualdade, desigualdade e de ordem. No exemplo escolheu-se criar uma regra que selecciona todas as entradas cujo campo mensagem comece por "com.i2s.", como visível na figura A.3.

Após clicar em OK na janela de adicionar regra voltamos para a janela do filtro, agora já com uma regra. Podem ser adicionadas novas regras e editar ou remover as já existentes. As regras podem se relacionar por via de uma condição lógica AND ou OR. Os resultados produzidos pelo filtro são dessa forma diferentes, como se pode constatar na figura A.4.

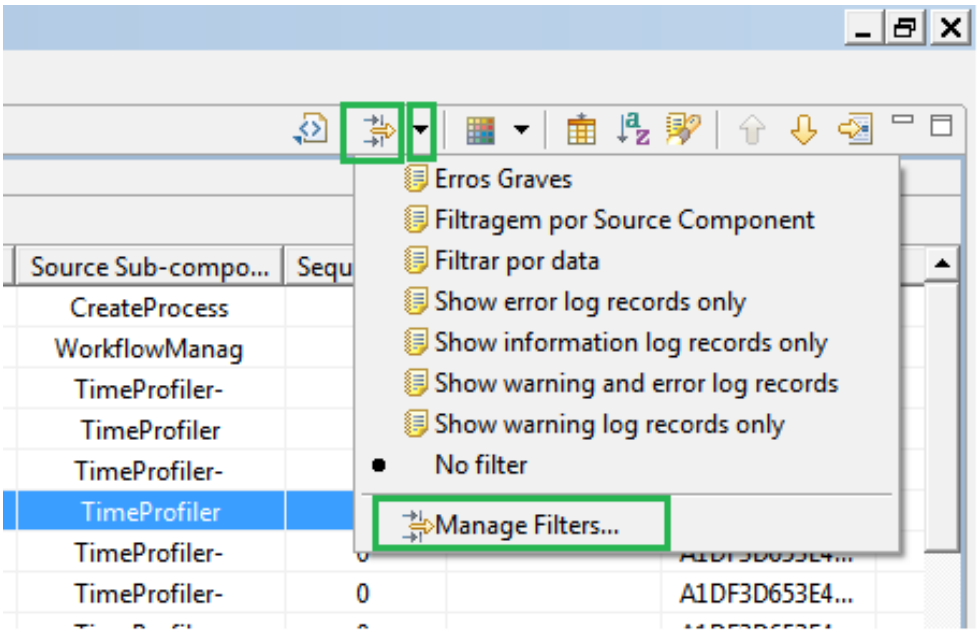

Figura A.1: Análise da funcionalidade de filtragem do LTA

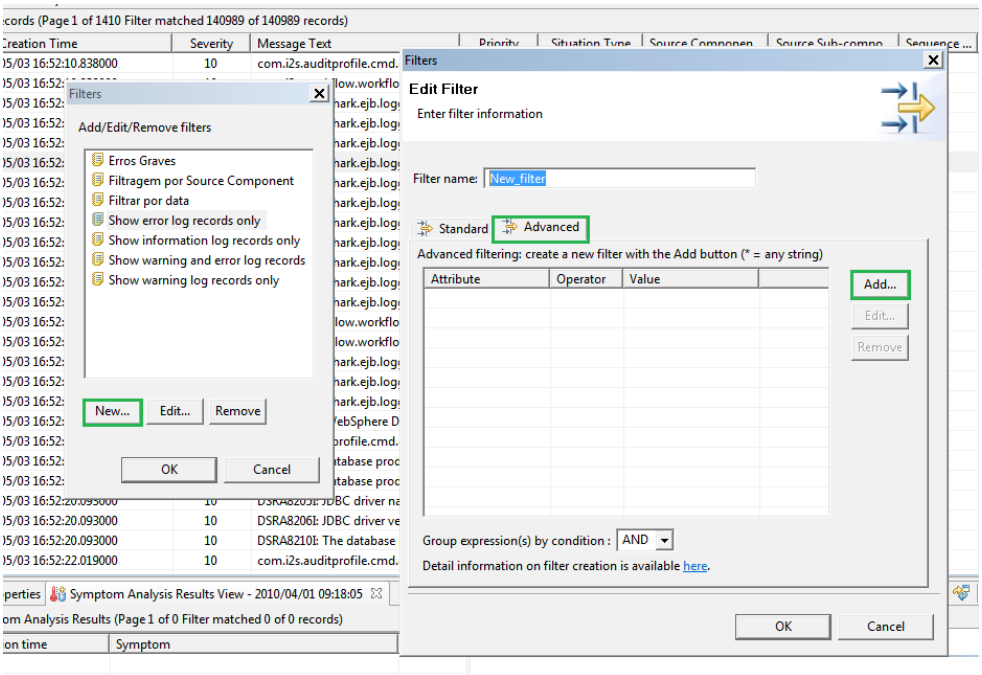

Figura A.2: Análise da funcionalidade de filtragem do LTA

Os resultados da aplicação deste filtro são os esperados, como pode ser visto na figura A.5. Logo por baixo do separador "Log View" pode ser identificado qual o filtro activo.

Para aplicar um filtro basta clicar na seta do lado direito do botão "Manage Filters..." e escolher a partir da lista, como demonstrado na figura A.6.

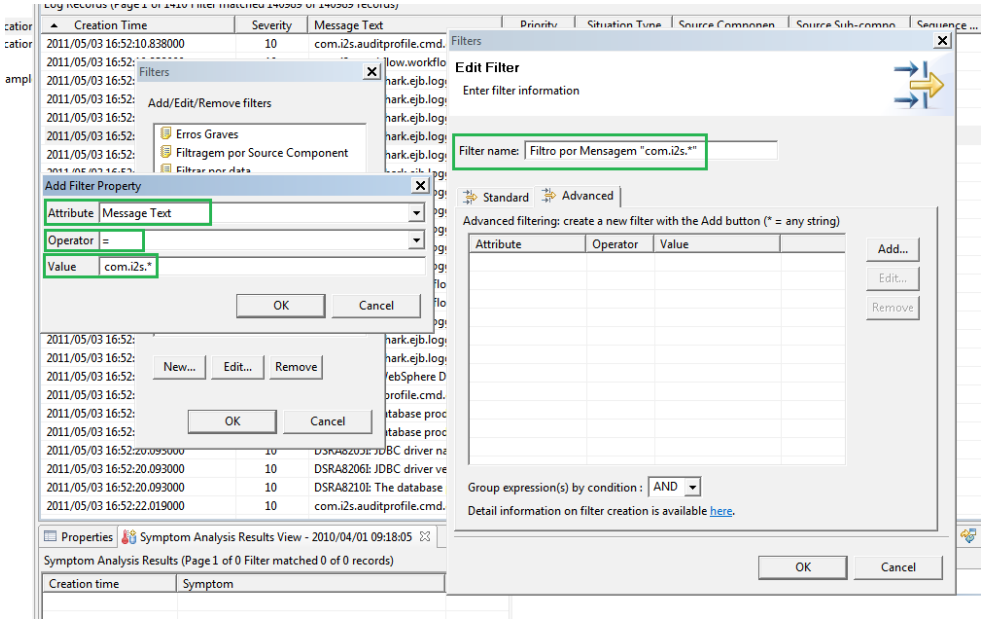

Figura A.3: Análise da funcionalidade de filtragem do LTA

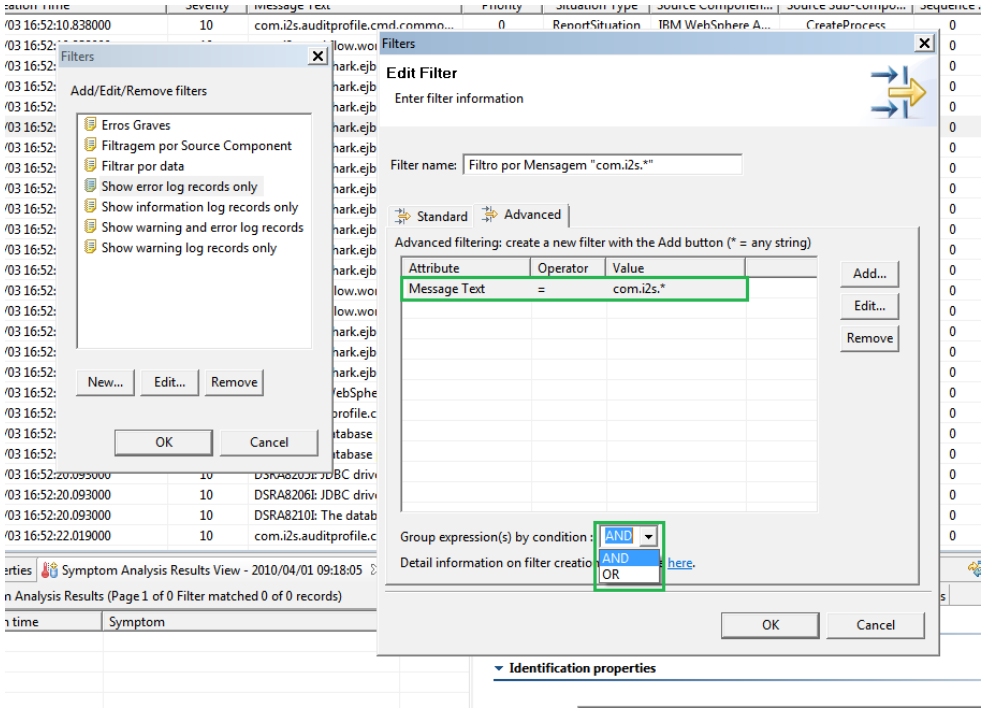

Figura A.4: Análise da funcionalidade de filtragem do LTA

Nota: Todas as funcionalidades de filtragem referidas anteriormente referem-se apenas a filtros de visualização. É também possível criar filtros de conteúdo para serem aplicados aquando da importação de logs. Com essa funcionalidade é possível filtrar o conteúdo diminuindo a quantidade de informação necessária importar. No entanto este

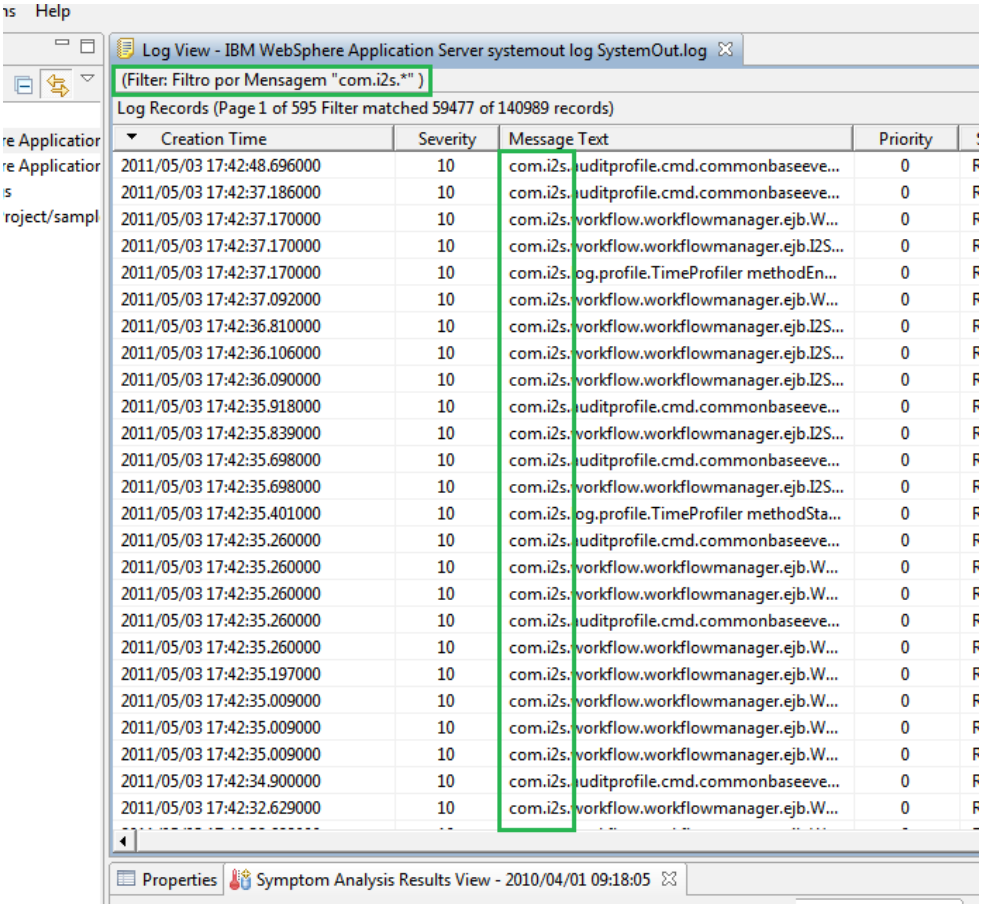

Figura A.5: Análise da funcionalidade de filtragem do LTA

tipo de filtragem implica que, caso se revele necessária informação que foi previamente filtrada, a repetição da importação de todo o log.

A.2 Ordenação de logs

Os logs podem ser ordenados segundo um ou vários campos do padrão Common Base Event. Para especificar quais os campos segundo os quais ordenar o log existem três métodos:
1º Método – Clicar no nome do atributo de coluna para ordenar segundo apenas um campo. Para ordenar segundo vários campos premir a tecla CTRL e clicar em todas as colunas segundo as quais se pretende ordenar o log. Neste último caso a ordem pela qual se clica nas colunas traduz-se na ordem dos atributos segundo os quais ordenar. As figuras A.7 e A.8 exemplificam este comportamento.

2º Método – Clicar no botão "Sort Columns..." e seleccionar de entre os atributos existentes (lado esquerdo) o(s) pretendido(s) passando-os para o lado direito, seleccionando também a ordem pretendida (ascendente ou descendente), como demonstrado na figura A.8.

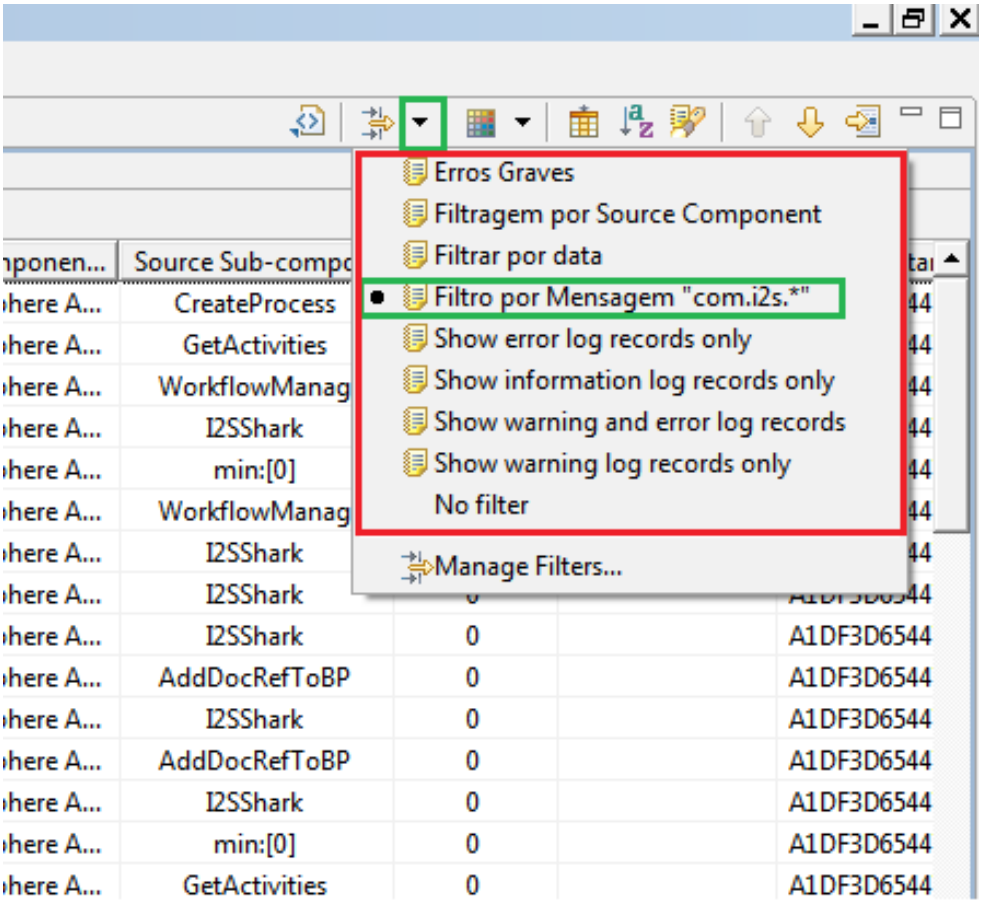

Figura A.6: Análise da funcionalidade de filtragem do LTA

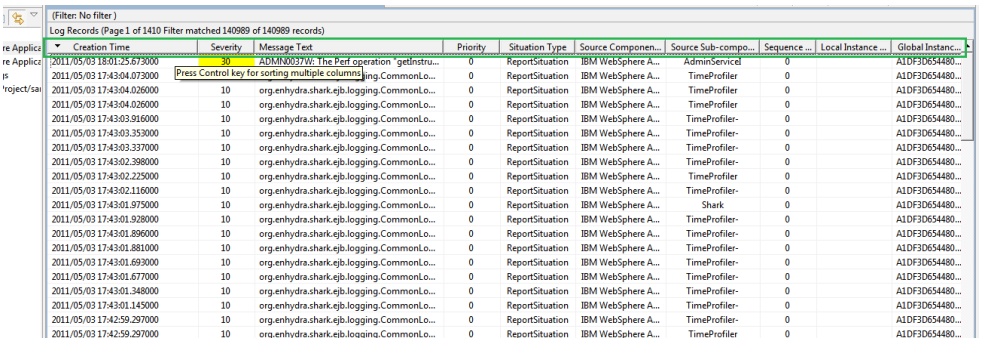

Figura A.7: Análise da funcionalidade de ordenação do LTA

3º Método – Através da barra de menu, seleccionando “Options”, seguidamente “Log View Preferences” e depois o separador “Sort Records”.

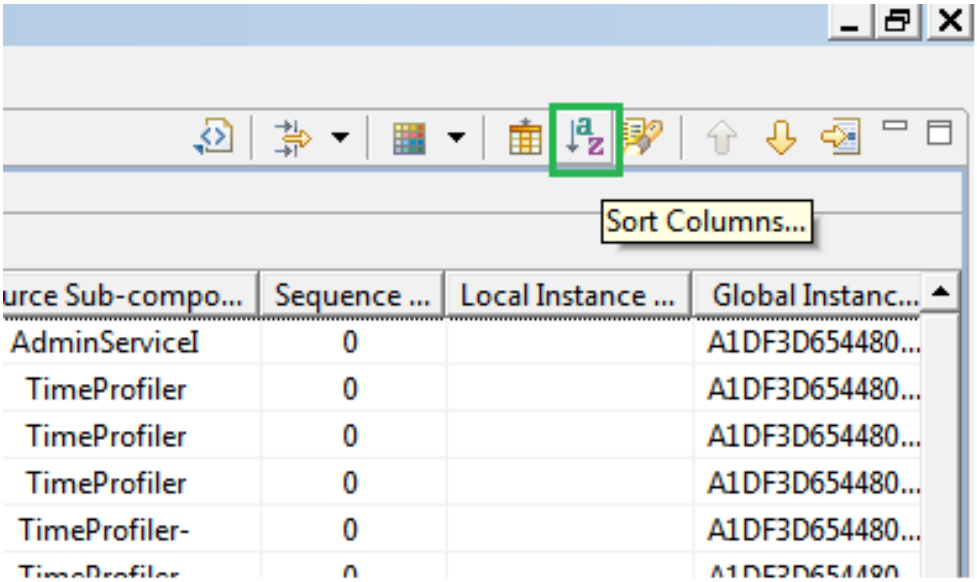

Figura A.8: Análise da funcionalidade de ordenação do LTA

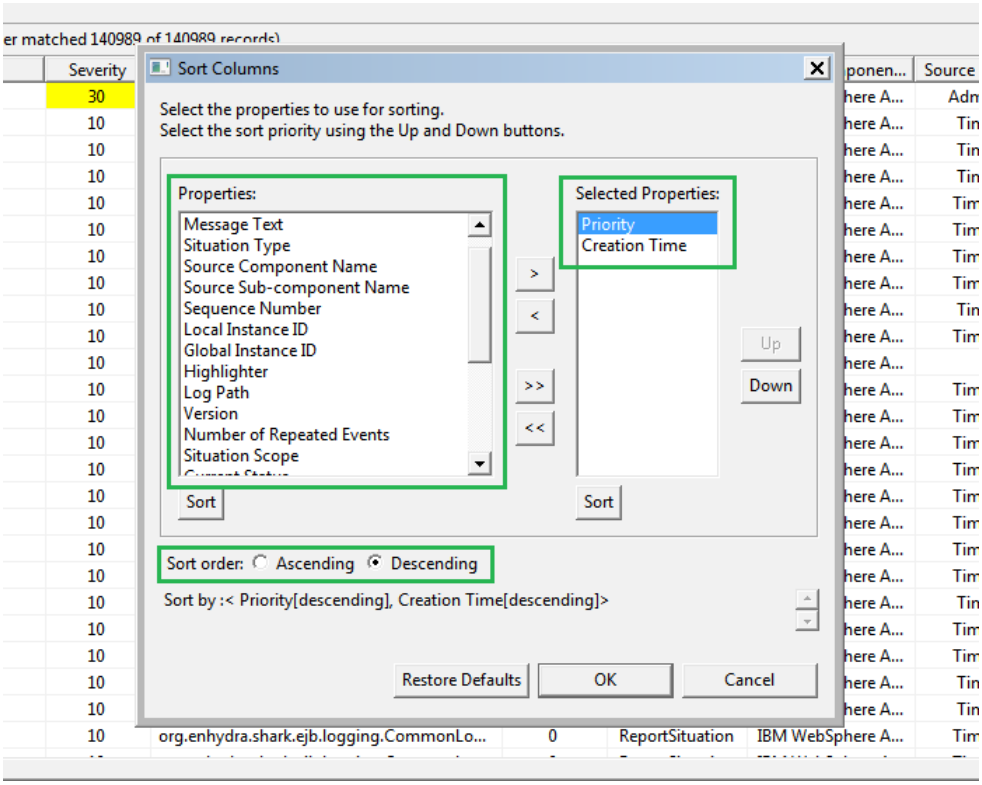

Figura A.9: Análise da funcionalidade de ordenação do LTA

A.3 Destaque de entradas

A funcionalidade de color coding está disponível em associação com os filtros, isto é, é possível usar color coding para "pintar" as entradas do log segundo várias condições que são definidas através das regras de um filtro. Podem ser usados vários filtros simultaneamente, sendo apenas necessário atribuir uma cor a um determinado filtro. Por exemplo, pode ser criado um filtro para todas as entradas cujo grau de severidade (campo "severity" do CBE) seja igual ou superior a 50. Após ser criado o filtro pode ser associada uma cor (neste caso vermelho), que fará com que as entradas do log que cumprem a regra definida no filtro fiquem realçadas. Para associar uma cor a um filtro basta clicar no botão "Highlight Events" na barra de ferramentas do LTA, como ilustrado na figura A.10.

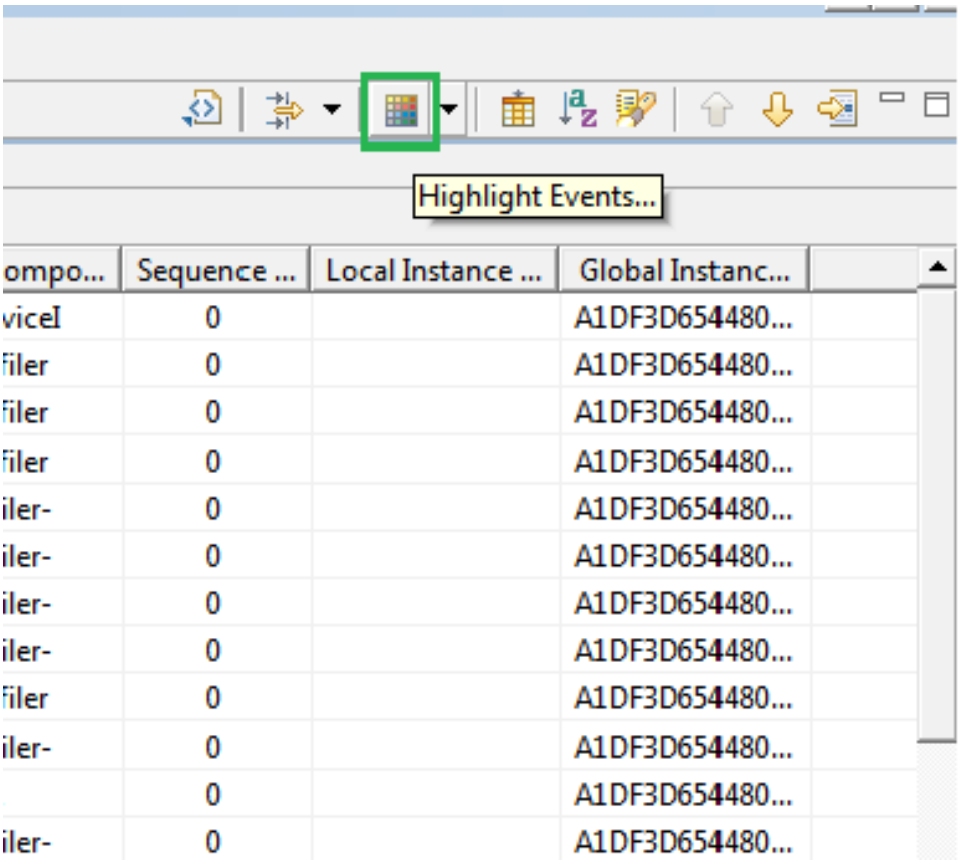

Figura A.10: Análise da funcionalidade de destaque de entradas do LTA

Seguidamente abre-se uma janela que lista todos os filtros existentes e na qual se podem activar os respectivos "color codings" através das checkbox existentes do lado esquerdo de cada filtro. Uma cor pode ser associada a um filtro fazendo duplo-clique no campo "Color" do filtro e seleccionando a cor, acção visível na figura A.11.

O resultado obtido será algo como ilustrado na figura A.12:

Nota: O uso de color coding, embora associado a um filtro, não implica que o filtro fique activo.

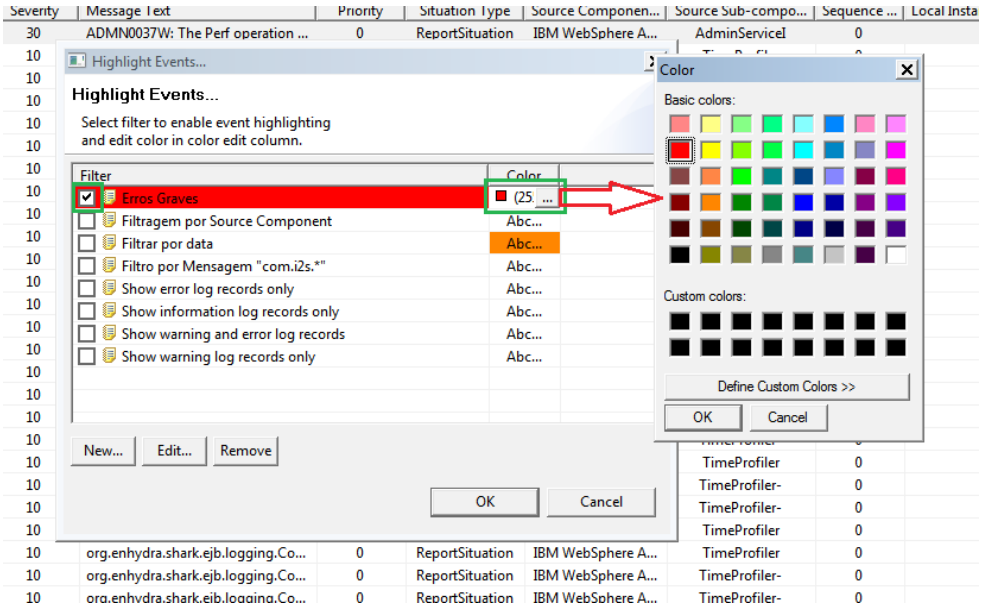

Figura A.11: Análise da funcionalidade de destaque de entradas do LTA

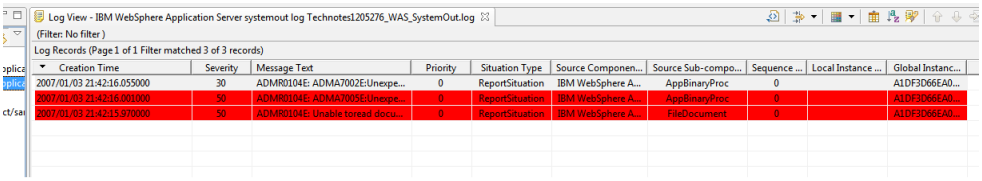

Figura A.12: Análise da funcionalidade de destaque de entradas do LTA

A.4 Selecção de atributos/colunas do *Log View*

Os atributos mostrados na vista "Log View" são configuráveis, podendo incluir qualquer campo do padrão Common Base Event. Para seleccionar quais os campos basta clicar no botão "Choose Columns..." da barra de ferramentas, visível na figura A.13.

Esta opção abre uma janela na qual temos os campos disponíveis do lado esquerdo e do lado direito os campos que estão a ser mostrados na vista "Log View", comportamento ilustrado na figura A.14.

Esta alteração também pode ser efectuada através da opção "Options -> Log View Preferences" da barra de menu, seleccionando depois o separador "Choose Columns...".

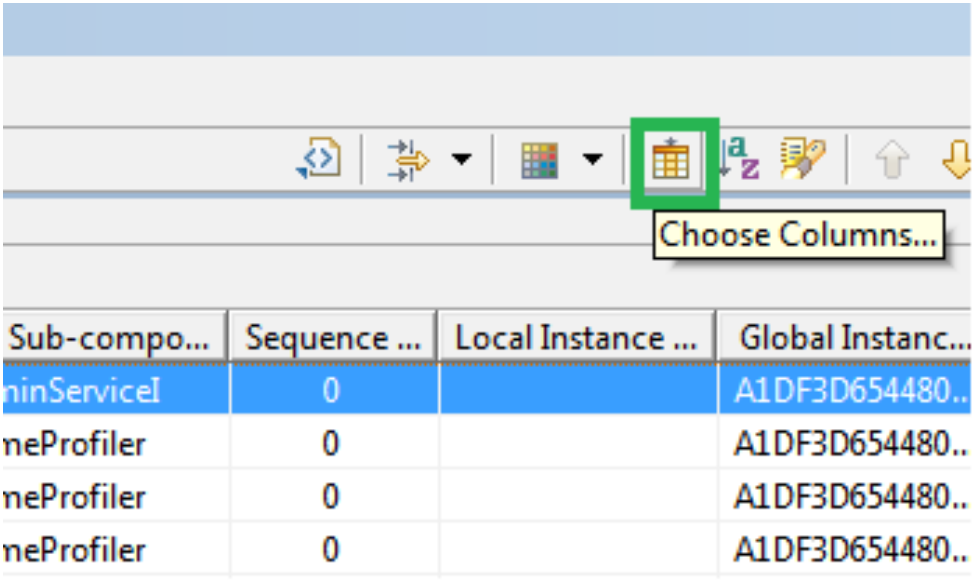

Figura A.13: Análise da funcionalidade de selecção de atributos do LTA

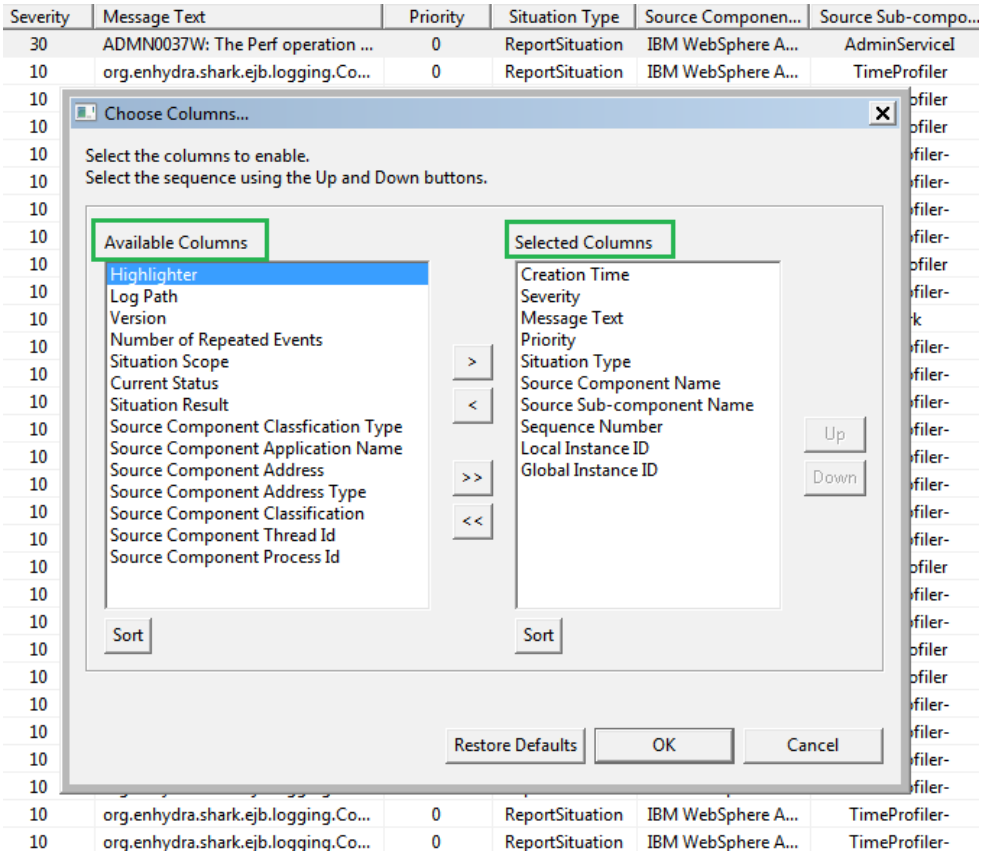

Figura A.14: Análise da funcionalidade de selecção de atributos do LTA

Anexo B

Levantamento de casos de uso, *user stories* e aspectos de interacção homem-máquina do LAP

Foram descritos os casos de uso, as *user stories* e documentados alguns aspectos de interacção homem-máquina, relativos ao projecto LAP.

B.1 Importar logs

A *user story* para importação de *logs* encontra-se resumida na tabela B.1.

User Story - US00	
Nome	Importar logs
Descrição	O utilizador escolhe a opção para importar um novo log. De seguida especifica qual o ficheiro a importar, qual o seu tipo e outras opções que sejam necessárias. Após confirmar as opções especificadas a aplicação importa o ficheiro e abre um visualizador para o conteúdo do ficheiro.
Casos de uso associados	CU00

Tabela B.1: Descrição da User Story - US00

O caso de uso para importar *logs* encontra-se resumido na tabela B.2.

B.2 Visualizar *Issue*

A *user story* para visualização de uma *issue* encontra-se resumida na tabela B.3.
O caso de uso para visualizar uma *issue* encontra-se resumido na tabela B.4.

B.2.1 Interacção homem-máquina - CU01

A vista das *Issues* localiza-se na parte inferior da vista *Navigator*, por defeito. Uma *Issue* pode ser aberta através de um duplo-clique em cima da *Issue*, através de um clique do botão direito do rato e depois seleccionando a opção "Visualizar"do menu de contexto

Caso de Uso - CU00	
Nome	Importar logs
Descrição	O utilizador importa um ficheiro de log
Prioridade	Alta
Pré-condição	A aplicação foi iniciada
Gatilhos	É confirmado qual o ficheiro a importar e os seus atributos
Cenário Principal	O utilizador selecciona a opção para importar logs e de seguida especifica as opções necessárias para realizar o processo de importação. No final do processo é aberta uma vista para o conteúdo do ficheiro
Cenários Alternativos	O utilizador pode cancelar a importação do log, durante a selecção das opções ou enquanto o processo de importação não terminar. Nesse caso nenhuma alteração é efectuada.
User story associada	US00

Tabela B.2: Descrição do Caso de Uso - CU00

User Story - US01	
Nome	Visualizar Issue
Descrição	O utilizador selecciona uma *Issue* da lista de Issues e usa a opção "Abrir". Os atributos da *Issue* são mostrados numa nova vista.
Casos de uso associados	CU01

Tabela B.3: Descrição da User Story - US01

Caso de Uso - CU01	
Nome	Visualizar *Issue*
Descrição	O utilizador visualiza o conteúdo de uma *Issue*
Prioridade	Alta
Pré-condição	A aplicação foi iniciada e a *Issue* já foi criada
Gatilhos	Foi seleccionada a operação "Abrir" da *Issue*
Cenário Principal	O utilizador usa o método "default" para abrir a *Issue*. É aberta uma vista com os atributos da *Issue* seleccionada
Cenários Alternativos	-
User story associada	US01

Tabela B.4: Descrição do Caso de Uso - CU01

ou premindo a tecla *ENTER*. A *Issue* abre por defeito junto com o separador que estava activo da última vez que esteve aberta, ou em alternativa com o separador *Overview* onde é apresentado um pequeno resumo dos elementos constituintes da *Issue* e a sua origem. A figura B.1 esquematiza a interacção homem-máquina referida.

B.3 Visualização de erros

A *user story* para visualização de erros encontra-se resumida na tabela B.5.

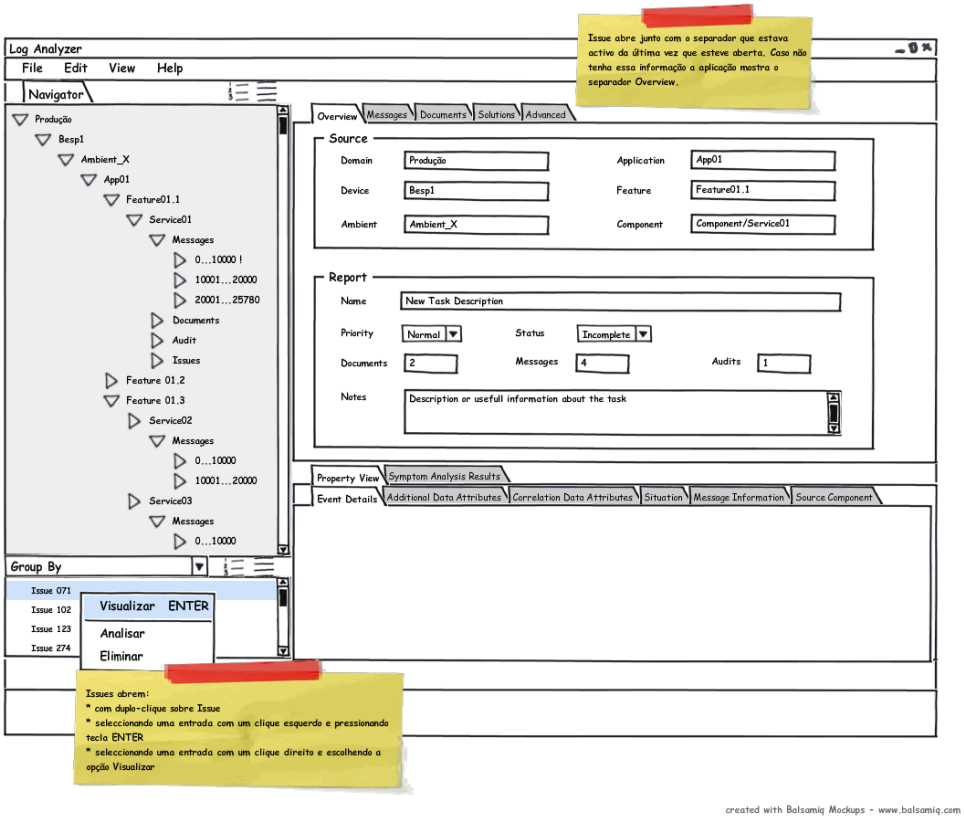

Figura B.1: Interacção homem-máquina - Visualizar Issue

User Story - US02	
Nome	Visualização de erros
Descrição	O utilizador selecciona a opção "Visualizar" de um ficheiro que contém eventos com erros. Ao visualizar, o ficheiro é aberto numa vista que mostra o seu conteúdo. O utilizador pode a partir daí navegar pelo conteúdo do ficheiro e visualizar os erros que estão registados.
Casos de uso associados	CU02

Tabela B.5: Descrição da User Story - US02

O caso de uso para visualizar erros encontra-se resumido na tabela B.6.

B.3.1 Interacção homem-máquina - CU02

Na vista *Navigator* os recursos monitorizados são apresentados de acordo com uma hierarquia configurável. Nos sub-campos de *Messages*, do *Navigator*, são disponibilizados os logs paginados de cada componente/serviço de determinada aplicação. Caso existam erros nos logs importados surge uma indicação (ponto de exclamação '!') à frente da página onde o erro se encontra (visível na figura). Fazendo duplo-clique sobre a página

Caso de Uso - CU02	
Nome	Visualização de erros
Descrição	O utilizador visualiza a informação que importou em busca de erros
Prioridade	Alta
Pré-condição	A informação disponível das aplicações monitorizadas já foi carregada para o programa
Gatilhos	Foi importado pelo menos um ficheiro contendo erros
Cenário Principal	O utilizador usa a operação “Abrir” nos ficheiros que tiverem erros. Posteriormente visualiza os erros através do visualizador de conteúdos do ficheiro
Cenários Alternativos	-
User story associada	US02

Tabela B.6: Descrição do Caso de Uso - CU02

onde o erro se encontra abre uma vista para o log (separador *Messages*), ou em alternativa, clicando com o botão direito do rato sobre a página dos logs e seleccionando a opção "Visualizar"a partir do menu de contexto ou premindo a tecla *ENTER*. Esta interacção encontra-se ilustrada na figura B.2

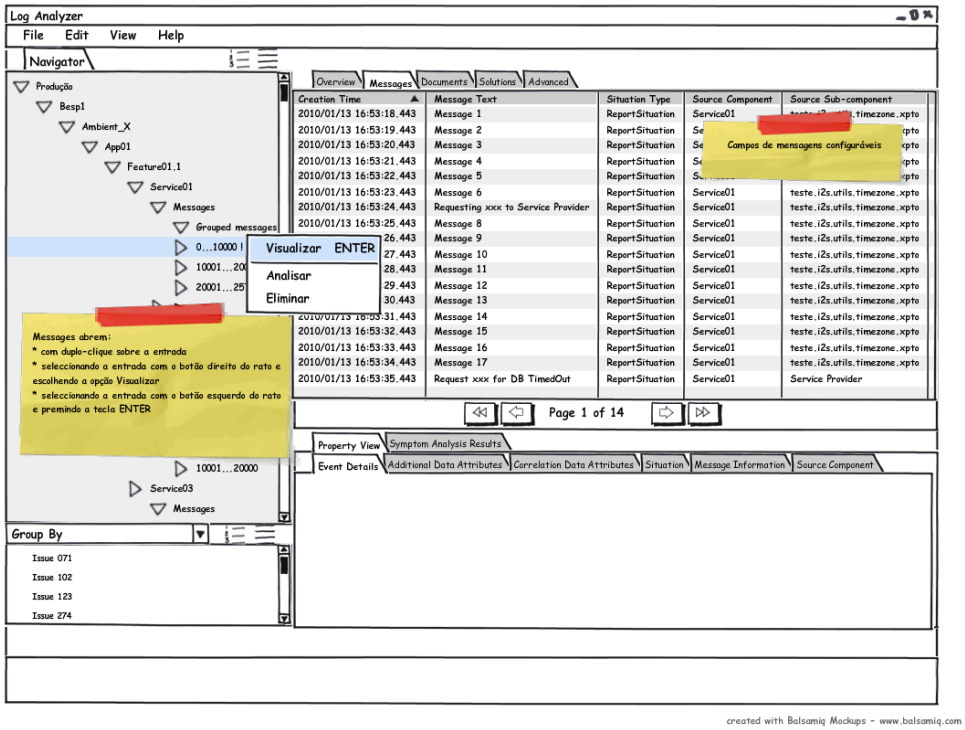

Figura B.2: Interacção homem-máquina - Visualização de erros

B.4 Criar correlação

A *user story* para criar uma nova correlação entre *logs* encontra-se resumida na tabela B.7.

User Story - US03	
Nome	Criar correlação
Descrição	O utilizador invoca a operação para criar uma nova correlação a partir de um ficheiro de mensagens, seleccionando de seguida o tipo de correlação a executar e sobre que ficheiros será executada. Após confirmar esses parâmetros a correlação é executada e no final do processo é aberta uma vista para o resultado da correlação
Casos de uso associados	CU03

Tabela B.7: Descrição da User Story - US03

O caso de uso para criar uma correlação entre ficheiros de *logs* encontra-se resumido na tabela B.8.

Caso de Uso - CU03	
Nome	Criar correlação
Descrição	O utilizador correlaciona um conjunto de mensagens
Prioridade	Normal
Pré-condição	Foram importados, pelo menos, dois ficheiros de log
Gatilhos	Foram confirmados os parâmetros da correlação e submetido o pedido de execução
Cenário Principal	O utilizador invoca a operação “Correlacionar”, seleccionando de as opções relevantes. No final da operação é aberta uma vista para o resultado da correlação
Cenários Alternativos	O utilizador cancela a execução da correlação durante o processo de escolha das opções. Nenhuma alteração é produzida.
User story associada	US03

Tabela B.8: Descrição do Caso de Uso - CU03

B.4.1 Interacção homem-máquina - CU03

Para criar uma correlação entre diferentes logs selecciona-se a opção "Correlate"do menu de contexto, clicando no log que se pretende correlacionar. Um esboço desta interacção é apresentado na figura B.3

Seguidamente, na janela que abre adiciona-se os logs com os quais se pretende correlacionar. Os logs disponíveis encontram-se do lado esquerdo e aparecem de acordo com o filtro seleccionado no cimo da área de selecção de logs. Os logs seleccionados para correlacionar encontram-se do lado direito. É possível adicionar ou remover entradas da lista dos logs a correlacionar utilizando os botões no centro da janela. Podem ser adicionadas selecções de logs ou todos os logs disponíveis. A figura B.4 representa esta interacção.

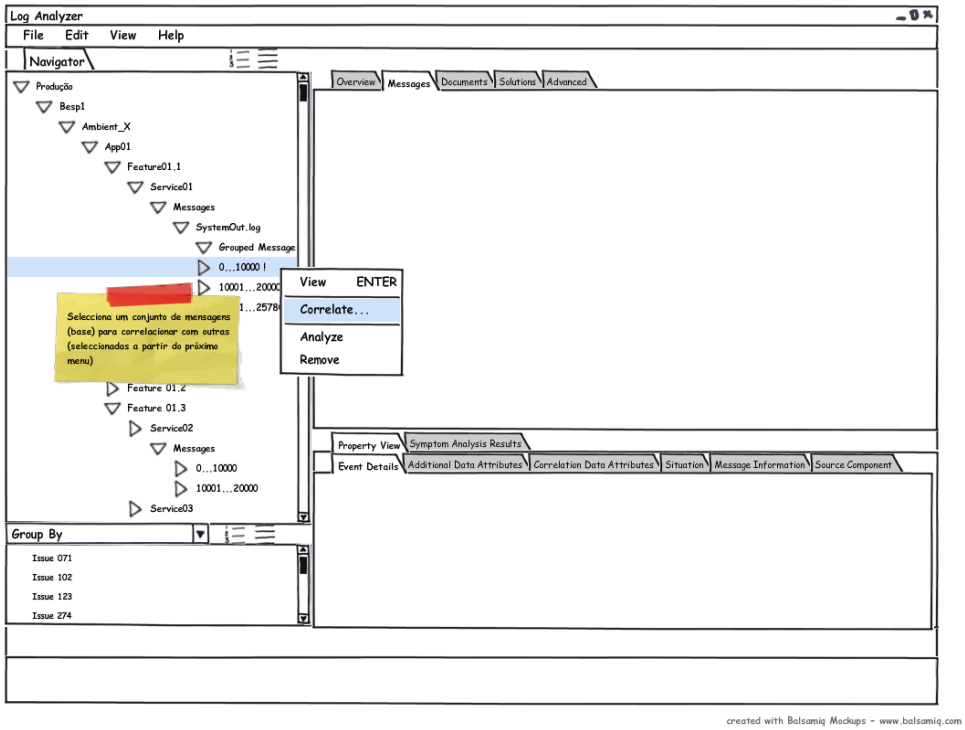

Figura B.3: Interacção homem-máquina - Criar correlação (1º passo)

Após seleccionar todos os pretendidos define-se o tipo de correlação que se pretende e a que Issue a correlação deverá ficar associada. No final clica-se em "Finish". Será aberto um novo separador com uma vista que permite visualizar o conteúdo dos logs correlacionados, como demonstrado na figura B.5.

B.5 Consultar uma auditoria ou documento

A *user story* para consultar uma audiria ou um documento encontra-se resumida na tabela B.9.

User Story - US04	
Nome	Consultar uma auditoria ou documento
Descrição	O utilizador selecciona a opção “Abrir” de uma auditoria ou documento. Após seleccionar a opção, e de acordo com o tipo de ficheiro, é aberto o editor adequado com o conteúdo do ficheiro.
Casos de uso associados	CU04

Tabela B.9: Descrição da User Story - US04

O caso de uso para consultar uma auditoria ou documento encontra-se resumido na tabela B.10.

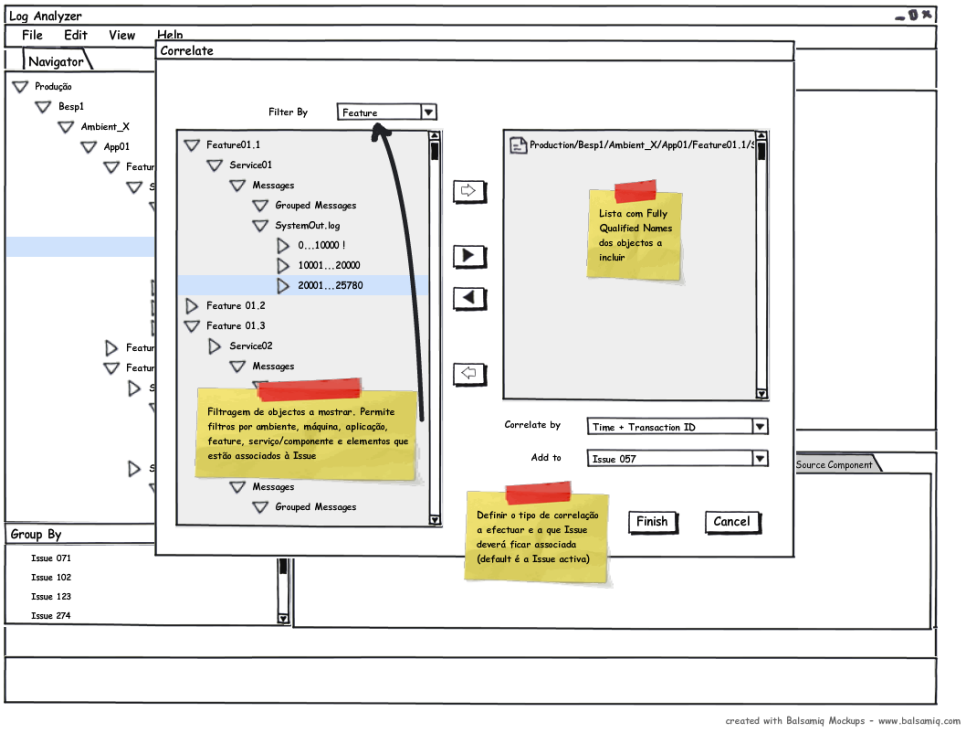

Figura B.4: Interacção homem-máquina - Criar correlação (2º passo)

Caso de Uso - CU03	
Nome	Consultar uma auditoria ou documento
Descrição	O utilizador visualiza o conteúdo de diferentes tipos de ficheiros
Prioridade	Normal
Pré-condição	Foi importado pelo menos um documento ou auditoria
Gatilhos	O ficheiro foi seleccionado e a operação "Abrir"
Cenário Principal	O utilizador invoca a operação "Abrir" sobre um documento. O documento é aberto e é mostrado o seu conteúdo
Cenários Alternativos	-
User story associada	US04

Tabela B.10: Descrição do Caso de Uso - CU04

B.5.1 Interacção homem-máquina - CU04

As auditorias ou documentos, quando disponíveis, encontram-se no nível inferior a *Audit* ou *Documents* respectivamente, no Navigator. Podem ser abertas através de um duplo-clique, através do menu de contexto (botão direito sobre a auditoria) seleccionando a opção Visualizar ou pressionando a tecla *ENTER*. A auditoria é aberta num separador *Documents* mostrando a informação que contem, como exemplificado na figura B.6.

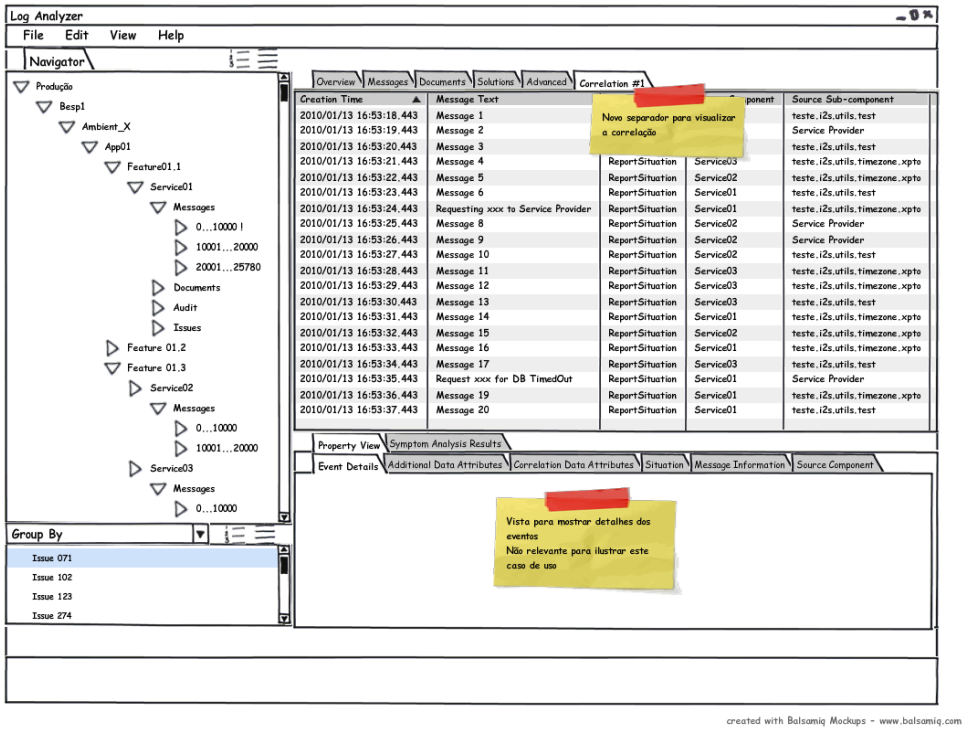

Figura B.5: Interacção homem-máquina - Criar correlação (3º passo)

B.6 Analisar logs recorrendo a uma base de dados de sintomas

A *user story* para analisar *logs* recorrendo a uma base de dados de sintomas encontra-se resumida na tabela B.11.

User Story - US05	
Nome	Analisar logs recorrendo a uma base de dados de sintomas
Descrição	O utilizador selecciona um log ou um conjunto de mensagens para analisar, em busca de problemas. Após seleccionar os ficheiros/mensagens a serem analisados, escolhe a opção “Analisar” e a selecção será analisada recorrendo às bases de dados de sintomas activas naquele momento, seleccionáveis a partir das opções do programa. Após a invocação da operação “Analisar” a análise é efectuada e os resultados da análise são mostrados numa nova vista.
Casos de uso associados	CU05

Tabela B.11: Descrição da User Story - US05

O caso de uso para analisar *logs* recorrendo a uma base de dados de sintomas encontra-se resumido na tabela B.12.

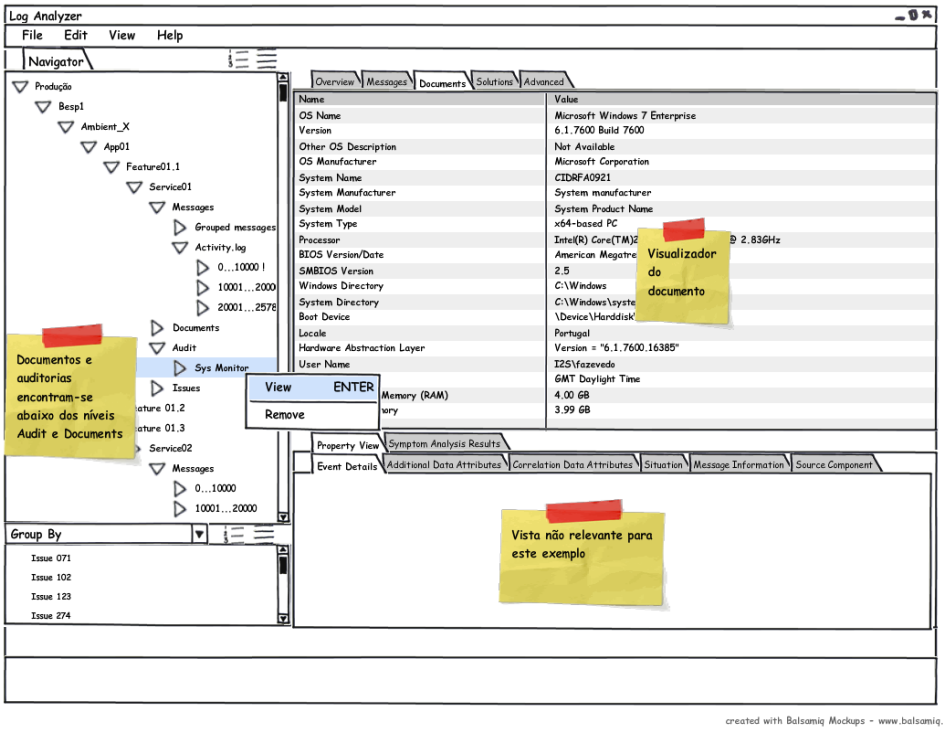

Figura B.6: Interacção homem-máquina - Consultar uma auditoria ou documento

Caso de Uso - CU05	
Nome	Analisar logs recorrendo a uma base de dados de sintomas
Descrição	O utilizador analisa uma selecção de mensagens com recurso a uma base de dados de sintomas
Prioridade	Alta
Pré-condição	Foi activa, pelo menos, uma base de dados de sintomas e importado pelo menos um ficheiro de log
Gatilhos	A operação "Analisar" foi invocada
Cenário Principal	O utilizador selecciona um log ou um conjunto de mensagens e analisa os eventos em busca de problemas. No final da análise são mostrados os resultados da análise
Cenários Alternativos	-
User story associada	US05

Tabela B.12: Descrição do Caso de Uso - CU05

B.6.1 Interacção homem-máquina - CU05

Uma análise de logs com recurso a uma base de dados de sintomas pode ser efectuada seleccionando o log, a partir do *Navigator*, ou um conjunto de entradas de logs e depois seleccionando a opção de analisar. No caso de ser seleccionado um log, esta função pode ser realizada clicando com o botão direito do rato sobre o log e seleccionando a opção *Analyze* do menu de contexto, como demonstrado na figura B.7.

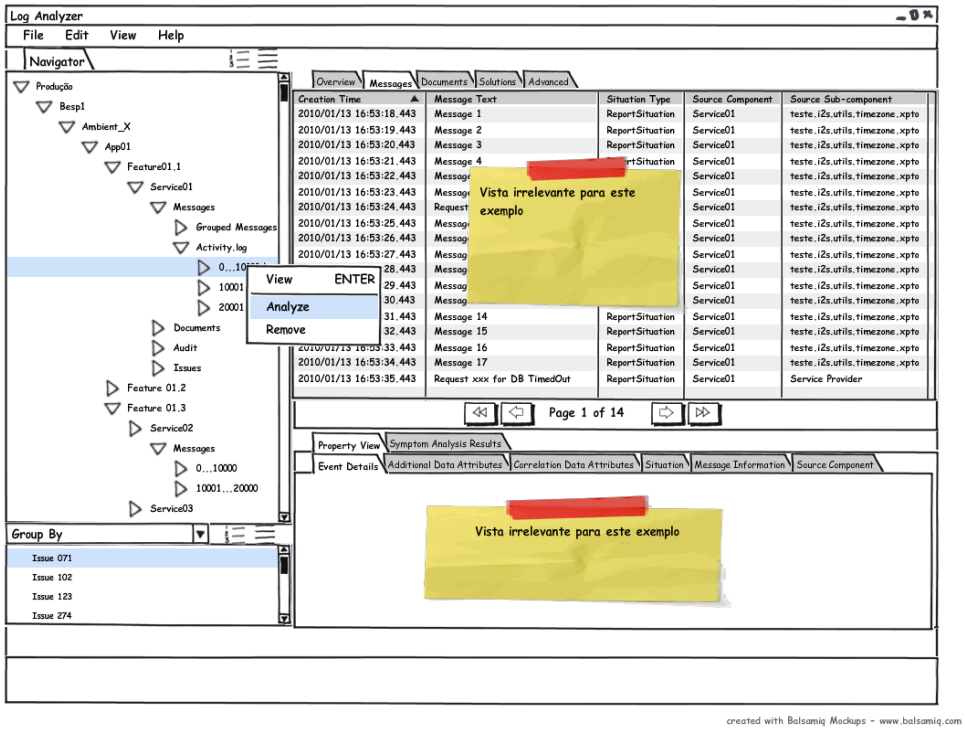

Figura B.7: Interacção homem-máquina - Analisar logs recorrendo a uma base de dados de sintomas (1º passo)

No caso de ser analisada apenas uma secção do ficheiro de log, é necessário seleccionar a secção a partir da vista *Messages* (usando a tecla CTRL + clique esquerdo nas entradas pretendidas) e depois, clicando sobre a selecção com o botão direito, seleccionar a opção *Analyze*, como ilustrado na figura B.8.

Após o final da análise terminar, o separador *Symptom Analysis Results* fica seleccionado apresentando os resultados da análise. A análise dos logs é feita com base nas bases de dados de sintomas escolhidas, no separador *Advanced* (caso de uso nº 6). Do lado esquerdo são apresentados os sintomas detectados e a data a que esses eventos ocorreram. Quando um desses sintomas for seleccionado, com um clique do rato, são apresentadas as possíveis soluções do lado direito no campo *Recommendations and Actions*. Esta interacção encontra-se ilustrada na figura B.9.

B.7 Gerir bases de dados de sintomas

A *user story* para gerir uma base de dados de sintomas encontra-se resumida na tabela B.13.

O caso de uso para gerir bases de dados de sintomas encontra-se resumido na tabela B.14.

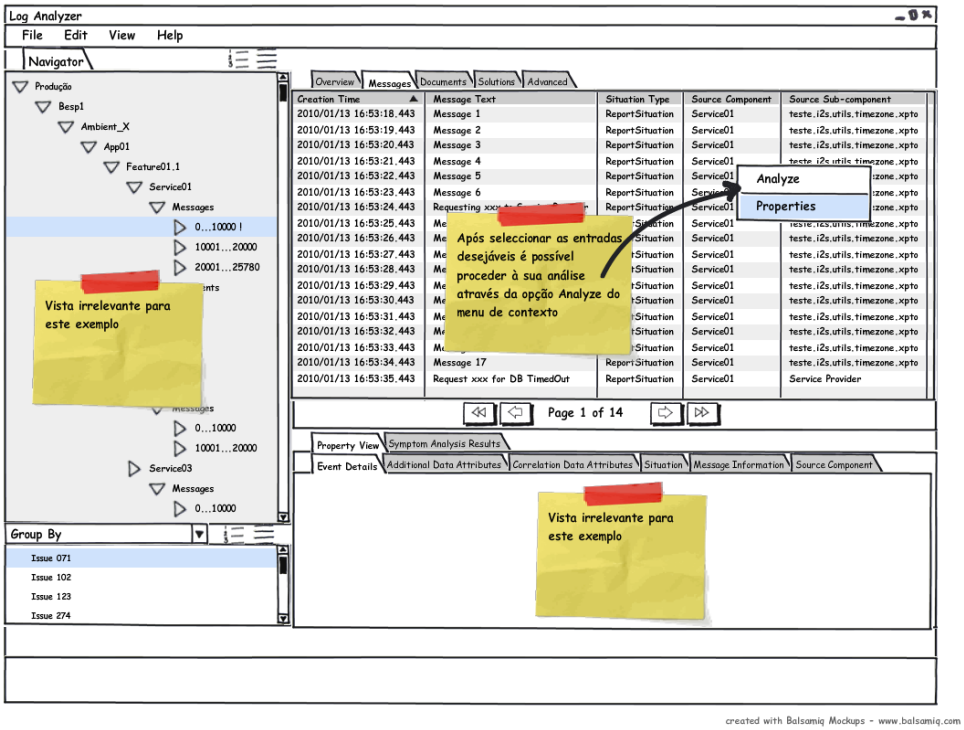

Figura B.8: Interacção homem-máquina - Analisar logs recorrendo a uma base de dados de sintomas (2º passo)

User Story - US06	
Nome	Gerir bases de dados de sintomas
Descrição	O utilizador acede à página de configuração das bases de dados de sintomas e adiciona ou remove entradas que correspondem a ficheiros de base de dados de sintomas conhecidos pela aplicação. Para cada uma delas pode ainda activar ou desactivá-las. No final confirma as alterações efectuadas e estas são aplicadas.
Casos de uso associados	CU06

Tabela B.13: Descrição da User Story - US06

B.7.1 Interacção homem-máquina - CU06

As bases de dados de sintomas podem ser geridas a partir do separador *Advanced*. É possível adicionar bases de dados de sintomas à lista através do botão *Add* seguido da selecção da localização da BD. É possível remover entradas da lista bastando seleccionar as entradas que se pretende eliminar e de seguida carregar no botão *Remove*. As BD de sintomas que serão usadas na análise de sintomas serão apenas aquelas que estiverem seleccionadas na lista (através da caixa de selecção). Esta interacção é ilustrada na figura B.10.

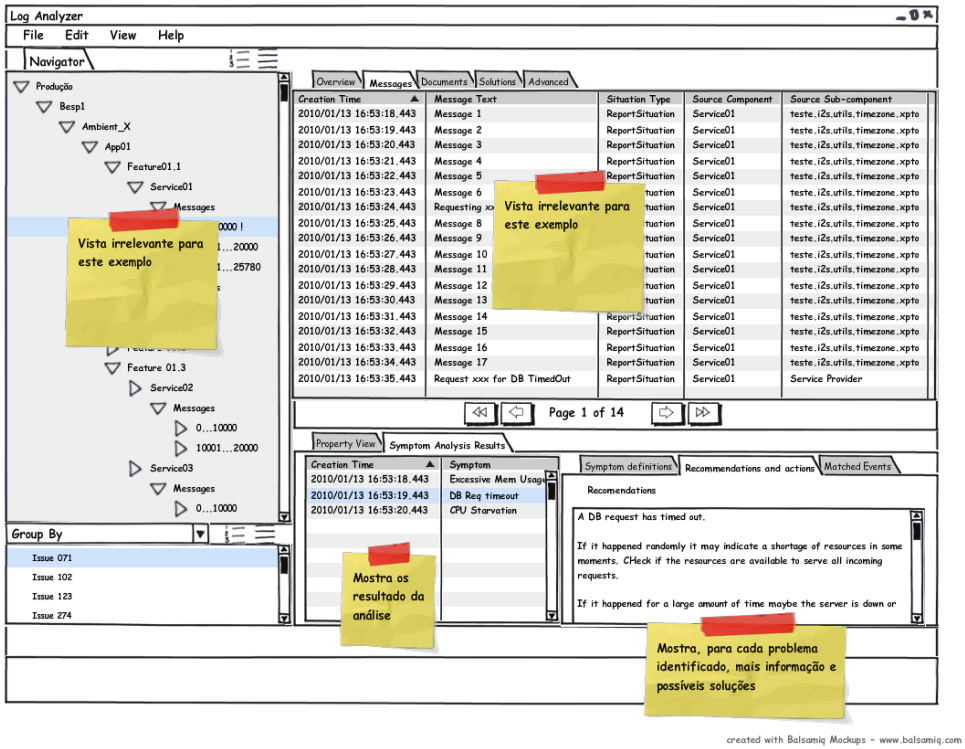

Figura B.9: Interacção homem-máquina - Analisar logs recorrendo a uma base de dados de sintomas (3º passo)

Caso de Uso - CU06	
Nome	Gerir bases de dados de sintomas
Descrição	O utilizador gere as bases de dados de sintomas usadas pela aplicação, usadas para efectuar análise de problemas
Prioridade	Alta
Pré-condição	A aplicação foi iniciada correctamente
Gatilhos	Confirmação das alterações efectuadas
Cenário Principal	O utilizador modifica as opções das bases de dados de sintomas. No final confirma as alterações efectuadas
Cenários Alternativos	O utilizador pode cancelar as alterações efectuadas antes de confirmar as alterações.
User story associada	US06

Tabela B.14: Descrição do Caso de Uso - CU06

B.8 Aplicar filtro às entradas dos ficheiros carregados

A *user story* para aplicar um filtro às entradas dos ficheiros carregados encontra-se resumida na tabela B.15.

O caso de uso para aplicar um filtro às entradas dos ficheiros carregados encontra-se resumido na tabela B.16.

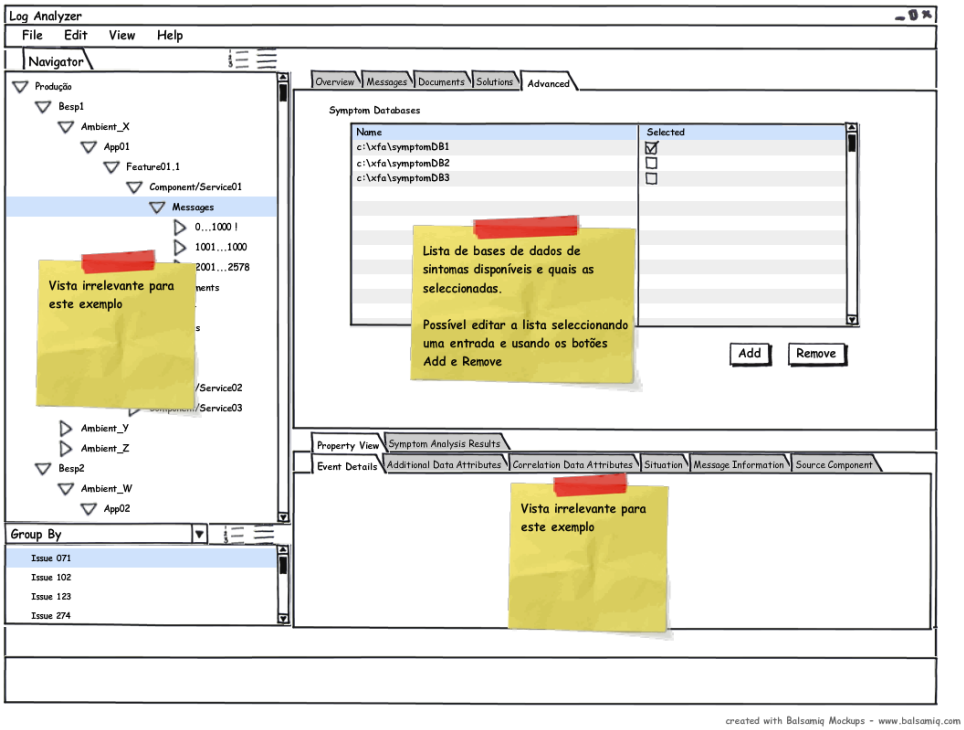

Figura B.10: Interacção homem-máquina - Gerir bases de dados de sintomas

User Story - US07	
Nome	Aplicar filtro às entradas dos ficheiros carregados
Descrição	O utilizador selecciona a opção para aplicar um filtro. Após clicar na opção são apresentados os filtros disponíveis e cabe ao utilizador seleccionar o pretendido. Após seleccionar são mostradas apenas as mensagens que cumprem as condições especificadas no filtro que foi aplicado.
Casos de uso associados	CU07 e CU13

Tabela B.15: Descrição da User Story - US07

B.8.1 Interacção homem-máquina - CU07

Para aplicar um filtro sobre um log, ou conjunto de logs correlacionados, é utilizado o botão *Manage Filters...* da barra de ferramentas da vista de logs (visíveis na figura seguinte). Após clicar neste botão é apresentado um menu onde é possível criar um novo filtro, especificando o nome e a lógica do filtro, através de um conjunto de operadores relacionais aplicados a campos do CBE e valores, como demonstrado na figura B.11.

Clicando na seta ao lado deste botão (não está visível na figura) é possível escolher qual o filtro a aplicar ou escolher ainda gerir os filtros, com possibilidade de apagar, editar ou criar novos, bastando seleccionar o filtro da lista e clicar no botão correspondente, acção exemplificada na figura B.12.

Caso de Uso - CU07	
Nome	Aplicar filtro às entradas dos ficheiros carregados
Descrição	As entradas dos ficheiros são filtradas e apenas aquelas que cumprem as regras dos filtros são visualizadas
Prioridade	Normal
Pré-condição	Alguma informação das aplicações monitorizadas já foi carregada para o programa e existe pelo menos uma vista para um ficheiro de log aberta e um filtro definido
Gatilhos	O filtro é aplicado
Cenário Principal	O utilizado aplica um filtro a um conjunto de dados já disponível. O resultado é dado pelas entradas do conjunto de dados que havia sido carregado previamente e que cumprem as regras especificadas no filtro.
Cenários Alternativos	O utilizador pode cancelar a operação de activação do filtro antes de efectivar a acção.
User story associada	US07

Tabela B.16: Descrição do Caso de Uso - CU07

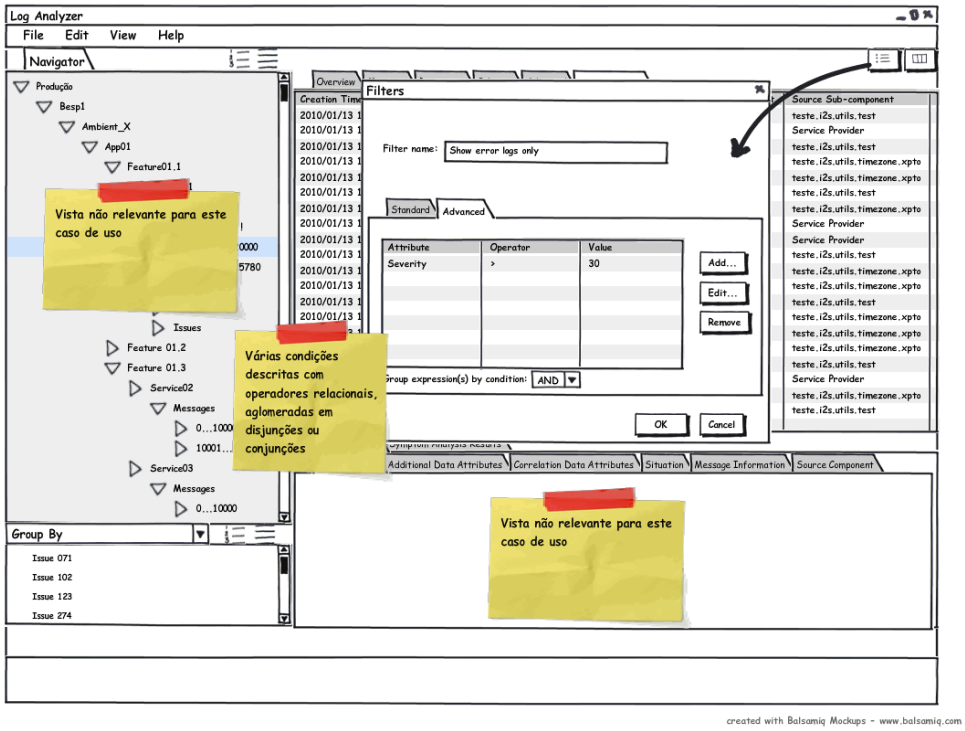

Figura B.11: Interacção homem-máquina - Aplicar filtro às entradas dos ficheiros carregados (1º passo)

B.9 *Color coding* de entradas de ficheiro de log

A *user story* para aplicar *color coding* às entradas dos ficheiros de *logs* encontra-se resumida na tabela B.17.

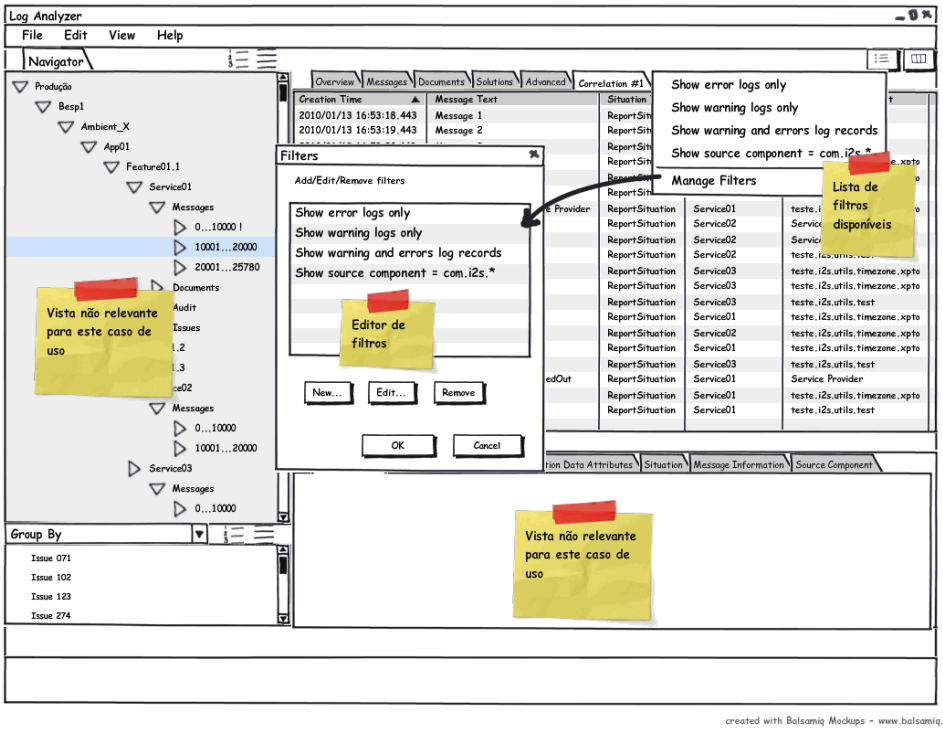

Figura B.12: Interacção homem-máquina - Aplicar filtro às entradas dos ficheiros carregados (2º passo)

User Story - US08	
Nome	*Color coding* de entradas de ficheiro de log
Descrição	O utilizador selecciona a opção para aplicar um filtro de *highlight* às mensagens. Após seleccionar essa opção é levado para um menu de opções onde pode escolher quais os filtros (de entre os que foram previamente definidos) a activar e quais as cores associadas a cada um dos filtros. Após confirmar estas opções é apresentada a vista das mensagens, na qual as entradas que verificam as condições especificadas nos filtros que se encontram activos passam a ter o fundo colorido com a cor associada a esse filtro.
Casos de uso associados	CU07 e CU08

Tabela B.17: Descrição da User Story - US08

O caso de uso para aplicar *color coding* às entradas de ficheiros de *log* encontra-se resumido na tabela B.18.

B.9.1 Interacção homem-máquina - CU08

Para fazer sobressair eventos usa-se o botão *Highlight Events* e depois escolhe-se o filtro, da lista de filtros disponíveis, e a cor a aplicar aos eventos que cumpram as condições especificadas no filtro, como exemplificado na figura B.13.

Caso de Uso - CU08	
Nome	*Color coding* de entradas de ficheiro de log
Descrição	São aplicadas cores às entradas dos ficheiros logs que permitem distinguir mais facilmente determinados eventos
Prioridade	Normal
Pré-condição	Alguma informação das aplicações monitorizadas já foi carregada para o programa existe pelo menos uma vista para um ficheiro de log aberta e um filtro definido
Gatilhos	É confirmada a aplicação de alguns filtros
Cenário Principal	O utilizador associa uma cor a um filtro, que será aplicado às mensagens.
Cenários Alternativos	O utilizador pode cancelar a operação de activação do filtro antes de efectivar a acção.
User story associada	US08

Tabela B.18: Descrição do Caso de Uso - CU08

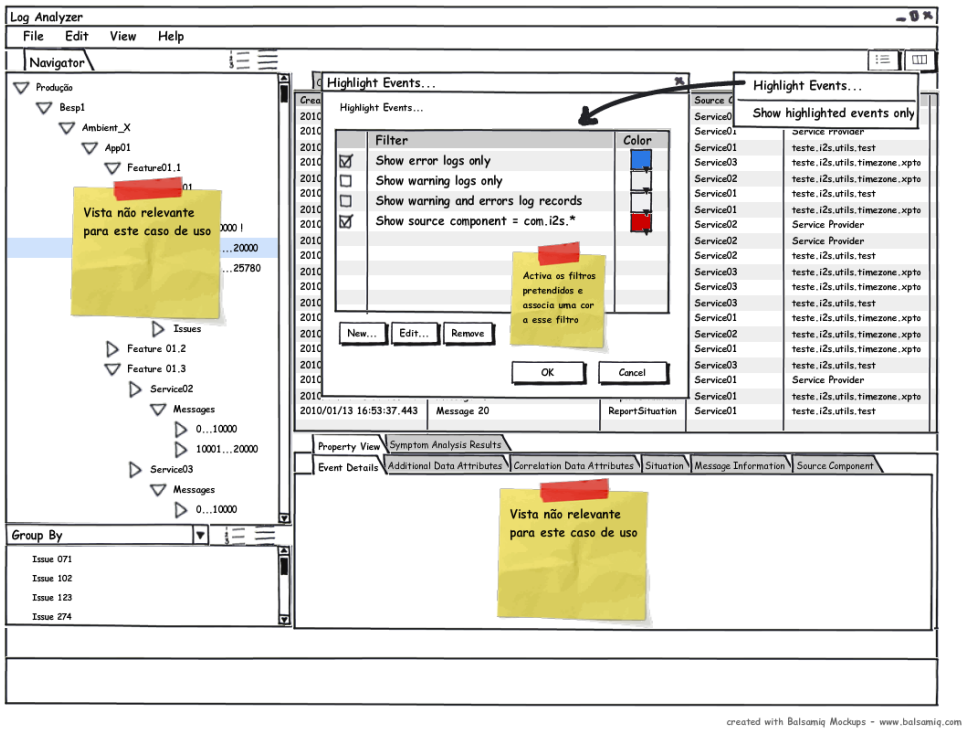

Figura B.13: Interacção homem-máquina - Color coding de entradas de ficheiro de log

B.10 Criar Issue

A *user story* para criar uma nova *issue* encontra-se resumida na tabela B.19.

O caso de uso para criar uma nova *issue* encontra-se resumido na tabela B.20.

User Story - US09	
Nome	Criar Issue
Descrição	O utilizador selecciona a opção para criar uma nova Issue. É levado para o editor de *Issues*, onde pode preencher os atributos com toda a informação que seja adequada. Após preencher todos os campos desejáveis invoca a opção guardar. A partir desse momento um novo registo de uma *Issue* foi criado.
Casos de uso associados	CU09

Tabela B.19: Descrição da User Story - US09

Caso de Uso - CU09	
Nome	Criar Issue
Descrição	Uma nova *Issue* é criada com os atributos especificados
Prioridade	Alta
Pré-condição	A aplicação foi iniciada correctamente
Gatilhos	Acção "guardar" é invocada
Cenário Principal	O utilizador selecciona a opção de criar nova Issue preenchendo de seguida a *Issue*. No final a *Issue* é guardada.
Cenários Alternativos	O utilizador, após seleccionar a opção de criar uma nova Issue, pode cancelar a qualquer momento a *Issue*. Nesse caso nenhum registo da *Issue* é guardado
User story associada	US09

Tabela B.20: Descrição do Caso de Uso - CU09

B.11 Agrupamento de *Issues*

A *user story* para agrupamento de *issues* encontra-se resumida na tabela B.21.

User Story - US10	
Nome	Agrupamento de *Issues*
Descrição	O utilizador acede à vista das *Issues* e selecciona a opção de "Agrupar". Após seleccionar essa opção são apresentados os tipos de agrupamento possíveis e cabe ao utilizador seleccionar o pretendido. Após efectuar a selecção a lista de *Issues* é actualizada para mostrar a informação agrupada, de acordo com a opção seleccionada.
Casos de uso associados	CU10

Tabela B.21: Descrição da User Story - US10

O caso de uso para agrupar *issues* encontra-se resumido na tabela B.22.

Caso de Uso - CU10	
Nome	Agrupamento de *Issues*
Descrição	A lista de *Issues* é agrupada em conjuntos, de acordo com diferentes parâmetros
Prioridade	Baixa
Pré-condição	A aplicação foi iniciada correctamente e existem algumas entradas distintas na lista de *Issues*
Gatilhos	Clique com o botão direito do rato no tipo de ordenação a aplicar
Cenário Principal	O utilizador selecciona um tipo de ordenação a aplicar à lista de *Issues*. Após efectuar esta acção a lista de *Issues* é mostrada com as suas entradas agrupadas de acordo com a opção seleccionada.
Cenários Alternativos	Se o actor seleccionar um método de agrupamento que já se encontra aplicado nenhuma acção é executada
User story associada	US10

Tabela B.22: Descrição do Caso de Uso - CU10

B.12 Criar uma nova base de dados de sintomas

A *user story* para criar uma nova base de dados de sintomas encontra-se resumida na tabela B.23.

User Story - US11	
Nome	Criar uma nova base de dados de sintomas
Descrição	O utilizador selecciona a opção para criar uma nova base de dados de sintomas. Define a localização do ficheiro, o nome e o formato e após confirmar os dados inseridos a base de dados de sintomas é criada e guardada. De seguida é levado para o editor da base de dados de sintomas. Aí poderá adicionar sintomas, regras e recomendações. No final as alterações são aplicadas utilizando a opção "Guardar"
Casos de uso associados	CU11

Tabela B.23: Descrição da User Story - US11

O caso de uso para criar uma nova base de dados de sintomas encontra-se resumido na tabela B.24.

B.13 Editar uma nova base de dados de sintomas

A *user story* para editor uma base de dados de sintomas encontra-se resumida na tabela B.25.

O caso de uso para editar uma base de dados de sintomas encontra-se resumido na tabela B.26.

Caso de Uso - CU11	
Nome	Criar uma nova base de dados de sintomas
Descrição	É criada uma nova base de dados de sintomas
Prioridade	Baixa
Pré-condição	A aplicação foi iniciada correctamente
Gatilhos	Seleccionar a opção para criar uma nova base de dados de sintomas
Cenário Principal	O utilizador selecciona a opção de criar uma nova base de dados de sintomas. De seguida preenche as opões necessárias e confirma a criação da base de dados de sintomas, guardando-a
Cenários Alternativos	O actor pode cancelar a criação de uma nova base de dados de sintomas, usando a opção "cancelar", disponível aquando da definição das opções necessárias. Neste caso nenhuma alteração é efectivada.
User story associada	US11

Tabela B.24: Descrição do Caso de Uso - CU11

User Story - US12	
Nome	Editar uma nova base de dados de sintomas
Descrição	O utilizador acede ao editor da base de dados de sintomas seleccionando a opção respectiva. Seguidamente selecciona o separador "Details" para visualizar os sintomas definidos na base de dados. Cada sintoma pode ter uma ou várias regras definidas para identificar o sintoma e também recomendações de resolução. Estes sintomas, regras e recomendações são definidos através de campos que correspondem a atributos, os quais podem ser editados. Após efectuar a edição dos aspectos pretendidos é aplicada a opção "Guardar" para salvar as alterações efectuadas.
Casos de uso associados	CU12

Tabela B.25: Descrição da User Story - US12

B.14 Criar um novo filtro

A *user story* para criar um novo filtro encontra-se resumida na tabela B.27.
O caso de uso para criar um novo filtro de *logs* encontra-se resumido na tabela B.2.

Caso de Uso - CU12	
Nome	Editar uma nova base de dados de sintomas
Descrição	Uma base de dados de sintomas é editada
Prioridade	Baixa
Pré-condição	A aplicação foi iniciada correctamente e uma base de dados de sintomas foi criada
Gatilhos	Guardar a base de dados de sintomas
Cenário Principal	O utilizador abre a base de dados de sintomas para edição. Após alterar os atributos pretendidos salva as alterações.
Cenários Alternativos	Após abrir a base de dados de sintomas para edição o actor pode cancelar as alterações efectuadas, fechando a vista de edição da base de dados de sintomas sem guardar as alterações.
User story associada	US12

Tabela B.26: Descrição do Caso de Uso - CU12

User Story - US13	
Nome	Criar um novo filtro
Descrição	O utilizador selecciona a opção para gerir os filtros. Na lista de filtros disponíveis usa a opção para adicionar um novo filtro. Após preencher as opções necessárias, incluindo o nome e as expressões lógicas que vão definir o filtro, o utilizador confirma os dados introduzidos e é criado o novo filtro.
Casos de uso associados	CU13

Tabela B.27: Descrição da User Story - US13

Caso de Uso - CU13	
Nome	Criar um novo filtro
Descrição	É criada um novo filtro
Prioridade	Normal
Pré-condição	A aplicação foi iniciada correctamente
Gatilhos	Seleccionar a opção para criar um novo filtro
Cenário Principal	O utilizador selecciona a opção de criar um novo filtro. De seguida preenche as opões necessárias e confirma a criação de um novo filtro
Cenários Alternativos	O actor pode cancelar a criação de um novo filtro, usando a opção “cancelar”, disponível aquando da definição das características do filtro. Neste caso nenhuma alteração é efectivada.
User story associada	US13

Tabela B.28: Descrição do Caso de Uso - CU13

Anexo C

Exemplo de criação de um adaptador para o *Generic Log Analyzer*

É necessário fornecer adaptadores para cada tipo de ficheiro que se pretende importar com *Generic Log Analyzer*. O processo de criação desses adaptadores é apresentado de forma simplificada de seguida.

C.1 Configuração Comum

A criação de um adaptador para GLA é realizada através da plataforma Eclipse. É criado um novo projecto de *plug-in*, que utiliza o ponto de extensão `org.eclipse.hyades.lta.logging.parserslogParser`, com os campos necessários correctamente preenchidos. A maioria dos campos são de simples preenchimento, havendo no entanto um campo importante, "parserParameter", onde é definido qual o adaptador a usar, que não é mais do que um ficheiro no formato XML. Uma configuração exemplo para as definições do *plugin* é apresentada de seguida:

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.2"?>
<plugin>
 <extension
       point="org.eclipse.hyades.lta.logging.parsers.logParser">
    <parser
          class="mylogparser.MyParserExtension"
          icon="./icons/log_parser_obj.gif"
          id="myLogParser"
          name="Sample Rules Log Parser for MyApp"
          ui_name="JMeter log">
       <field
             browseType="*.log"
             defaultValue="c:\temp\sample.log"
             id="file_path"
             name="Directory"
             useBrowse="true">
       </field>
```

```
        <field
              defaultValue="JMeter 1.0(rules)"
              id="version"
              name="Supported versions"
              ui_type="combobox"
              useBrowse="false">
        </field>
        <parserParameter
              name="JMeter 1.0(rules)"
              value="./adapters/regex.adapter">
        </parserParameter>
    </parser>
 </extension>
</plugin>
```

A correcta interpretação pela aplicação de qualquer tipo de ficheiro está limitada pela existência de um adaptador capaz de o converter para o formato CBE. Esses adaptadores podem ser estáticos, utilizam classes JAVA para fazer o *parsing*[1] e criar o CBE correctamente, ou então através de adaptadores que usam simples expressões regulares para efectuar o *parsing* necessário. Um exemplo de cada um desses adaptadores é descrito nas próximas secções.

C.1.1 Adaptador estático

Ficheiro ParserWrapperExtension.java (Wrapper do Parser estático)

```
package staticparserproject;

import org.apache.commons.logging.Log;
import org.eclipse.hyades.logging.parsers.LogParserException;
import org.eclipse.hyades.logging.parsers.importer.ParserWrapper;

/**
 * @author developer
 * myLogParserWrapper class
 */
public class ParserWrapperExtension extends ParserWrapper {

public ParserWrapperExtension(){
/* The super class constructor must be called */
super();
currentPlugin = "StaticParserProject";
}

@Override
public void parse(Log argLogger) {
```

[1] Em ciências de computação refere-se à análise sintáctica de uma estrutura, geralmente sob a forma de texto

```
try{
super.parse(argLogger);
}
catch (LogParserException e) {
/*Nothing to catch*/
}
}

}
```

Ficheiro SParser.java (implementação do parser estático)

```
package staticparserproject;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;

import org.eclipse.hyades.logging.core.Guid;
import org.eclipse.hyades.logging.events.cbe.CommonBaseEvent;
import org.eclipse.hyades.logging.events.cbe.ComponentIdentification;
import org.eclipse.hyades.logging.events.cbe.ConfigureSituation;
import org.eclipse.hyades.logging.events.cbe.ConnectSituation;
import org.eclipse.hyades.logging.events.cbe.DestroySituation;
import org.eclipse.hyades.logging.events.cbe.FeatureSituation;
import org.eclipse.hyades.logging.events.cbe.ReportSituation;
import org.eclipse.hyades.logging.events.cbe.Situation;
import org.eclipse.hyades.logging.events.cbe.StartSituation;
import org.eclipse.hyades.logging.events.cbe.StopSituation;

import org.eclipse.hyades.logging.parsers.LogParserException;

import org.eclipse.hyades.logging.parsers.MonitoringParser;

/**
 * This class implements a parser for WAS serverout log files.
 *
 * @author Kholod
 */
public class SParser extends MonitoringParser {
// Parser name & version
private static String name    = "WAS Server Out Log Parser";
private static String version = "1.00";

// Time formats
private static final String LOG_TIME_FORMAT = "M/d/yy H:m:s:S z";

private static SimpleDateFormat logFormat = null;
```

```
static {
logFormat = new SimpleDateFormat(LOG_TIME_FORMAT);
}

/* (non-Javadoc)
 * @see org.eclipse.hyades.logging.parsers.IParser#parseNext()
 */
public CommonBaseEvent[] parseNext() throws LogParserException {
CommonBaseEvent[] temp = null;

arrayIndex = 0;

while (curLine != null) {
if (isContainLogEvent(curLine)) {
messages[arrayIndex].init();

String record = readSingleRecord();

// Source Component
messages[arrayIndex].setSourceComponentId(
getSourceComponentId(
getComponent(record), getThreadId(record)));

// Message severity
messages[arrayIndex].setSeverity(getSeverity(
getSeverityMark(record)));

// Message creation time
long creationTime = getCreationTime(getTimestamp(record));

Calendar c = Calendar.getInstance();
c.set(2000, 6, 20);

messages[arrayIndex].setCreationTimeAsLong(c.getTimeInMillis());

// Message text
String message = getMessage(record);
messages[arrayIndex].setMsg(message);

// MessageId
String messageID = getMessageID(message);
if (messageID != null) {
messages[arrayIndex].setMsgDataElement(
null, null, new String[]{}, null, messageID, "IBM4.4.1", null);
}

// Situation
```

```
messages[arrayIndex].setSituation(getSituation(messageID, message));

// Global Instance ID
messages[arrayIndex].setGlobalInstanceId(Guid.generate());

arrayIndex++;
if (arrayIndex == MessageArraySize) {
arrayIndex = 0;
recordCount++;

return messages;
}

recordCount++;
} else {
curLine = readALine();
}
}

// If we are not logging the message then null
// the array elements that weren't set on this call
if (arrayIndex == 0) {
temp = null;
setEndOfFile();
} else {
for (int i = arrayIndex; i < MessageArraySize; i++) {
messages[i] = null;
}
temp = messages;
//setEndOfFile();
}

// Throw an exception if no valid log records are parsed/logged:
if (recordCount == 0) {
throw new LogParserException("No records were found!");
}

return temp;
}

/* (non-Javadoc)
 * @see com.ibm.gla.sdf.TimeFilteringLogParser#
 *  isContainLogEvent(java.lang.String)
 */
protected boolean isContainLogEvent(String str) {
return str != null &&
       str.length() > 0 &&
```

```
        str.charAt(0) == '[' &&
        Character.isDigit(str.charAt(1));
}

private String readSingleRecord() {
String record = curLine;

curLine = readLine();
boolean flag = true;
while (curLine != null && flag) {
if (isContainLogEvent(curLine)) {
String message = getMessage(curLine);

if (message.startsWith("\t")) {
record = record.concat("\n").concat(message);
} else {
flag = false;
continue;
}
} else {
record = record.concat("\n").concat(curLine);
}

curLine = readLine();
}

return record;
}

// Record parsing functions

    protected String getTimestamp(String str) {
    return str.substring(1, str.indexOf("]"));
    }

    static String getMessage(String str) {
     String message = str.substring(str.indexOf("] ") + 27);

if (message.length() > 11 &&
     message.charAt(9) == ':' &&
Character.isLetter(message.charAt(8)) &&
Character.isDigit(message.charAt(7))) {
     message = message.substring(11);
     }

     return message;
}
```

```
    static char getSeverityMark(String str) {
     return str.charAt(str.indexOf("] ") + 25);
    }

    static String getComponent(String str) {
     int start = str.indexOf("] ") + 11;
     int end = start + 13;
     return str.substring(start, end).trim();
    }

    static String getThreadId(String str) {
     int start = str.indexOf("] ") + 2;
     int end = start + 8;

     return str.substring(start, end).trim();
    }

    static String getMessageID(String message) {
     String messageID = null;

     if (message.length() > 11 &&
     message.charAt(9) == ':' &&
Character.isLetter(message.charAt(8)) &&
Character.isDigit(message.charAt(7))) {
     messageID = message.substring(0, 9);
     }

     return messageID;
    }

// Data processing functions

protected long getCreationTime(String value) {
try {
return logFormat.parse(value).getTime();
} catch (ParseException e) {
return 0;
}
}

Situation getSituation(String messageID, String message) {
if (messageID == null)
return getReportSituation();

if ("WSVR0037I".equalsIgnoreCase(messageID) ||
"SRVE0169I".equalsIgnoreCase(messageID) ||
```

```
"WSVR0049I".equalsIgnoreCase(messageID)) {
return getStartSituation("START INITIATED", "SUCCESSFUL");
} else if ("WSVR0023I".equalsIgnoreCase(messageID) ||
   "WSVR0001I".equalsIgnoreCase(messageID) ||
   "ADMN0015I".equalsIgnoreCase(messageID)) {
return getStartSituation("START COMPLETED", "SUCCESSFUL");
} else if ("SRVE0091I".equalsIgnoreCase(messageID)
&& message.indexOf("init") >= 0) {
return getStartSituation("START INITIATED", "SUCCESSFUL");
} else if ("CNTR0031W".equalsIgnoreCase(messageID)) {
return getStartSituation("START COMPLETED", "UNSUCCESSFUL");
} else if ("WSVR0053I".equalsIgnoreCase(messageID) ||
   "ADMC0026I".equalsIgnoreCase(messageID)) {
return getFeatureSituation("AVAILABLE", "INTERNAL");
} else if ("SRVE0171I".equalsIgnoreCase(messageID)) {
return getFeatureSituation("AVAILABLE", "EXTERNAL");
} else if ("SRVE0172I".equalsIgnoreCase(messageID)) {
return getFeatureSituation("UNAVAILABLE", "EXTERNAL");
} else if ("SRVE0167I".equalsIgnoreCase(messageID)) {
return getConfigureSituation();
} else if ("SRVE0170I".equalsIgnoreCase(messageID)) {
return getStopSituation("STOP INITIATED", "SUCCESSFUL");
} else if ("SRVE0029E".equalsIgnoreCase(messageID)) {
return getStopSituation("STOP COMPLETED", "UNSUCCESSFUL");
} else if ("WSVR0024I".equalsIgnoreCase(messageID)) {
return getStopSituation("STOP COMPLETED", "SUCCESSFUL");
} else if ("CONM6009E".equalsIgnoreCase(messageID)) {
return getConnectSituation();
} else if ("CONM6007I".equalsIgnoreCase(messageID)) {
return getDestroySituation();
} else if ("SRVE0091I".equalsIgnoreCase(messageID)
&& message.indexOf("destroy") >= 0) {
return getDestroySituation();
} else {
return getReportSituation();
}
}

short getSeverity(char sevMark) {
if (sevMark == 'A' ||
sevMark == 'e' ||
sevMark == 'd' ||
sevMark == '>' ||
sevMark == '<' ||
sevMark == 'O' ||
sevMark == 'I') {
return (short)10;
```

```
} else if (sevMark == 'W' ||
   sevMark == 'R') {
return (short)30;
} else if (sevMark == 'm' ||
   sevMark == 'X' ||
   sevMark == 'E') {
return (short)50;
} else if (sevMark == 'F') {
return (short)60;
} else if (sevMark == 'Z' ||
   sevMark == 'U' ||
   sevMark == 'u') {
return (short)0;
} else {
return (short)0;
}
}

ComponentIdentification getSourceComponentId(
String subComponent, String threadId) {
ComponentIdentification sourceComponentId =
eventFactory.createComponentIdentification();
sourceComponentId.setLocation(localHostId);
sourceComponentId.setLocationType(localHostIdFormat);
sourceComponentId.setComponent("WebSphere Application Server");
sourceComponentId.setComponentIdType("ProductName");
sourceComponentId.setComponentType("WebSphereApplicationServer");
sourceComponentId.setSubComponent(subComponent);
sourceComponentId.setThreadId(threadId);

return sourceComponentId;
}

static private Situation getConnectSituation() {
Situation connectSituation = eventFactory.createSituation();

ConnectSituation connSituation =
eventFactory.createConnectSituation();
connSituation.setReasoningScope("EXTERNAL");
connSituation.setSituationDisposition("AVAILABLE");
connSituation.setSuccessDisposition("UNSUCCESSFUL");

connectSituation.setCategoryName("ConnectSituation");
connectSituation.setSituationType(connSituation);

return connectSituation;
}
```

```
static private Situation getConfigureSituation() {
Situation configureSituation = eventFactory.createSituation();

ConfigureSituation connSituation =
eventFactory.createConfigureSituation();
connSituation.setReasoningScope("INTERNAL");
connSituation.setSuccessDisposition("SUCCESSFUL");

configureSituation.setCategoryName("ConfigureSituation");
configureSituation.setSituationType(connSituation);

return configureSituation;
}

static private Situation getDestroySituation() {
Situation destroySituation = eventFactory.createSituation();

DestroySituation connSituation =
eventFactory.createDestroySituation();
connSituation.setReasoningScope("INTERNAL");
connSituation.setSuccessDisposition("SUCCESSFUL");

destroySituation.setCategoryName("DestroySituation");
destroySituation.setSituationType(connSituation);

return destroySituation;
}

private static Situation getFeatureSituation(
String featureDisposition, String scope) {
Situation featureSituation = eventFactory.createSituation();

FeatureSituation situationType = eventFactory.createFeatureSituation();
situationType.setFeatureDisposition(featureDisposition);
situationType.setReasoningScope(scope);

featureSituation.setCategoryName("FeatureSituation");
featureSituation.setSituationType(situationType);

return featureSituation;
}

static private Situation getStartSituation(String situationQualifier,
                          String successDisposition) {
Situation startSituation = eventFactory.createSituation();
```

```
StartSituation situationType = eventFactory.createStartSituation();
situationType.setSituationQualifier(situationQualifier);
situationType.setSuccessDisposition(successDisposition);
situationType.setReasoningScope("INTERNAL");

startSituation.setCategoryName("StartSituation");
startSituation.setSituationType(situationType);

return startSituation;
}

static private Situation getStopSituation(String situationQualifier,
  String successDisposition) {
Situation stopSituation = eventFactory.createSituation();

StopSituation situationType = eventFactory.createStopSituation();
situationType.setSituationQualifier(situationQualifier);
situationType.setSuccessDisposition(successDisposition);
situationType.setReasoningScope("INTERNAL");

stopSituation.setCategoryName("StopSituation");
stopSituation.setSituationType(situationType);

return stopSituation;
}

//  Situations
/**
 * Gets ReportSituation instance.
 *
 * @return
 */
static private Situation getReportSituation() {
Situation reportSituation = eventFactory.createSituation();

ReportSituation repSituation = eventFactory.createReportSituation();
repSituation.setReasoningScope("INTERNAL");
repSituation.setReportCategory("LOG");

reportSituation.setCategoryName("ReportSituation");
reportSituation.setSituationType(repSituation);

return reportSituation;
}

/**
 * @see org.eclipse.hyades.logging.parsers.IParser#getName()
```

Exemplo de criação de um adaptador para o *Generic Log Analyzer*

```
 */
public String getName() {
return name;
}

/**
 * @see org.eclipse.hyades.logging.parsers.IParser#getVersion()
 */
public String getVersion() {
return version;
}
}
```

Adaptador que delega o processo de conversão na classe JAVA

```
<?xml version="1.0" encoding="UTF-8"?>
<adapter:Adapter xmlns:adapter="http://www.eclipse.org/hyades/schema/
Adapter.xsd" xmlns:cc="http://www.eclipse.org/hyades/schema/
ComponentConfiguration.xsd" xmlns:ex="http://www.eclipse.org/
hyades/schema/Extractor.xsd" xmlns:fmt="http://www.eclipse.org/
hyades/schema/Formatter.xsd" xmlns:hga="http://www.eclipse.org/
hyades/schema/Context.xsd" xmlns:op="http://www.eclipse.org/hyades/
schema/Outputter.xsd" xmlns:parser="http://www.eclipse.org/hyades/
schema/Parser.xsd" xmlns:pu="http://www.eclipse.org/hyades/schema/
ProcessUnit.xsd" xmlns:sensor="http://www.eclipse.org/hyades/schema/
Sensor.xsd">
  <hga:Contexts>
    <hga:Context description="Context Instance for the current
component" executableClass=
"org.eclipse.hyades.logging.adapter.impl.BasicContext"
implementationCreationDate="2010-04-27T16:09:40"
implementationVersion="1.0" loggingLevel="10" name=
"Basic Context Implementation" role="context" roleCreationDate=
"2010-04-27T16:09:40" roleVersion="1.0" uniqueID=
"N72F9801020E11DF8000C85FA12ADFC7">
      <hga:Component description="Operating System file sensor"
executableClass=
"org.eclipse.hyades.logging.parsers.adapter.sensors.StaticParserSensor"
implementationCreationDate="2010-04-27T16:09:40"
implementationVersion="1.0" loggingLevel="10" name=
"Static Parser Sensor" role="sensor" roleCreationDate=
"2010-04-27T16:09:40" roleVersion="1.0" uniqueID=
"N72FE621020E11DF8000C85FA12ADFC7"/>
      <hga:Component description="LogImportOutputter"
executableClass=
"org.eclipse.hyades.logging.parsers.adapter.outputters.LogImportOutputter"
implementationCreationDate="2010-04-27T16:09:41"
implementationVersion="1.0" loggingLevel="10" name=
```

```
"LogImportOutputter" role="outputter" roleCreationDate=
"2010-04-27T16:09:41" roleVersion="1.0" uniqueID=
"N7835D51020E11DF8000C85FA12ADFC7"/>
    </hga:Context>
  </hga:Contexts>
  <cc:Configuration description=
"The component level configurations for this Adaptor"
uniqueID="N72F9800020E11DF8000C85FA12ADFC7">
    <cc:ContextInstance continuousOperation="false"
description="Context Instance for the current component"
enableICU="false" maximumIdleTime="100000" pauseInterval="1000"
uniqueID="N72F9801020E11DF8000C85FA12ADFC7">
      <cc:Sensor description="A static parser sensor" uniqueID=
"N72FE621020E11DF8000C85FA12ADFC7" confidenceBufferSize="8"
maximumBlocking="5" type="StaticParserSensor">
        <pu:Property propertyName="directory" propertyValue=
"C:\Users\rfazevedo\Desktop"/>
        <pu:Property propertyName="fileName" propertyValue=
"example.log"/>
        <pu:Property propertyName="parserClassName" propertyValue=
"staticparserproject.SParser"/>
        <sensor:StaticParserSensor directory=
"C:\Users\rfazevedo\Desktop" fileName="example.log" parserClassName=
"staticparserproject.SParser"/>
      </cc:Sensor>
      <cc:Outputter description="Log Import Outputter"
uniqueID="N7835D51020E11DF8000C85FA12ADFC7" type=
"LoggingAgentOutputter"/>
    </cc:ContextInstance>
  </cc:Configuration>
</adapter:Adapter>
```

C.1.2 Adaptador por expressões regulares

Adaptador por expressões regulares - regexp.adapter

```
<?xml version="1.0" encoding="ASCII"?>
<adapter:Adapter xmlns:adapter="http://www.eclipse.org/hyades/schema/
Adapter.xsd" xmlns:cc="http://www.eclipse.org/hyades/schema/
ComponentConfiguration.xsd" xmlns:ex="http://www.eclipse.org/hyades/
schema/Extractor.xsd" xmlns:fmt="http://www.eclipse.org/hyades/schema/
Formatter.xsd" xmlns:hga="http://www.eclipse.org/hyades/schema/
Context.xsd" xmlns:op="http://www.eclipse.org/hyades/schema/
Outputter.xsd" xmlns:parser="http://www.eclipse.org/hyades/schema/
Parser.xsd" xmlns:pu="http://www.eclipse.org/hyades/schema/
ProcessUnit.xsd" xmlns:sensor="http://www.eclipse.org/hyades/schema/
Sensor.xsd">
  <hga:Contexts>
```

```
    <hga:Context description="Context Instance for the current
component" executableClass=
"org.eclipse.hyades.logging.adapter.impl.BasicContext"
implementationCreationDate="2003-10-07T12:00:00"
implementationVersion="1.0.0" loggingLevel="60"
name="Basic Context Implementation" role="context"
roleCreationDate="2003-10-07T12:00:00" roleVersion="1.0.0"
uniqueID="N35B6D910F8511D88000A31D7605087A">
      <hga:Component description="Operating System file sensor"
executableClass=
"org.eclipse.hyades.logging.adapter.sensors.SingleOSFileSensor"
implementationCreationDate="2003-10-07T12:00:00"
implementationVersion="1.0.0" loggingLevel="50" name="OS File Sensor"
role="sensor" roleCreationDate="2003-10-07T12:00:00" roleVersion="1.0"
uniqueID="N3604F900F8511D88000A31D7605087A"/>
      <hga:Component description=
"This extractor uses regular expression patterns to identify record
delimiters" executableClass=
"org.eclipse.hyades.logging.adapter.extractors.RegularExpressionExtractor"
implementationCreationDate="2003-10-07T12:00:00"
implementationVersion="1.0.0" loggingLevel="60"
name="Regular Expression Extractor" role="messageExtractor"
roleCreationDate="2003-10-07T12:00:00" roleVersion="1.0.0"
uniqueID="N36299800F8511D88000A31D7605087A"/>
      <hga:Component description="Regular expression parser"
executableClass="org.eclipse.hyades.logging.adapter.parsers.Parser"
implementationCreationDate="2003-10-07T12:00:00"
implementationVersion="1.0.0" loggingLevel="60" name="Generic Parser"
role="parser" roleCreationDate="2003-10-07T12:00:00"
roleVersion="1.0.0" uniqueID="N3650A800F8511D88000A31D7605087A"/>
      <hga:Component description="CBE Formatter" executableClass=
"org.eclipse.hyades.logging.adapter.formatters.CBEFormatter"
implementationCreationDate="2003-10-07T12:00:00"
implementationVersion="1.0.0" loggingLevel="60" name="CBE Formatter"
role="formatter"
roleCreationDate="2003-10-07T12:00:00" roleVersion="1.0.0" uniqueID=
"N36C36700F8511D88000A31D7605087A"/>
      <hga:Component description="Logging agent outputter"
executableClass=
"org.eclipse.hyades.logging.parsers.adapter.outputters.LogImportOutputter"
implementationCreationDate="2003-10-07T12:00:00" implementationVersion=
"1.0.0" implementationVersionDescription="" loggingLevel="60" name=
"Logging Agent Outputter" role="outputter" roleCreationDate=
"2003-10-07T12:00:00" roleVersion="1.0.0" uniqueID=
"N36E80600F8511D88000A31D7605087A"/>
    </hga:Context>
  </hga:Contexts>
```

```
  <cc:Configuration description="The component level configurations
for this adapter" uniqueID="N35B6D900F8511D88000A31D7605087A">
    <cc:ContextInstance continuousOperation="false" description=
"Context Instance for the current component" maximumIdleTime="500000"
pauseInterval="10" uniqueID="N35B6D910F8511D88000A31D7605087A">
      <cc:Sensor description="A single file sensor" uniqueID=
"N3604F900F8511D88000A31D7605087A" maximumBlocking="5"
type="SingleFileSensor">
        <pu:Property propertyName="directory" propertyValue=
"c:\temp"/>
        <pu:Property propertyName="fileName" propertyValue=
"sample.log"/>
        <sensor:SingleFileSensor directory="c:\temp" fileName=
"sample.log"/>
      </cc:Sensor>
      <ex:Extractor description="This extractor uses regular
expression patterns to identify record delimiters" uniqueID=
"N36299800F8511D88000A31D7605087A" containsLineBreaks="false"
endPattern="$" includeEndPattern="false" includeStartPattern="true"
lineBreakSymbol=
"" replaceLineBreaks="false" startPattern="^"/>
      <cc:Parser designationToken=":" separatorToken="[ ]{2}"
uniqueID="N3650A800F8511D88000A31D7605087A">
        <parser:RuleElement index="N3650A810F8511D88000A31D7605087A"
name="CommonBaseEvent">
          <parser:RuleElement index="N8CA7C54002A11D88000C6FD28070181"
name="sourceComponentId">
            <parser:RuleAttribute index=
"ND66C0C1002A11D88000C6FD28070181" name="component"
usePreviousMatchSubstitutionAsDefault="false">
 <SubstitutionRule substitute="HTTP Server"/> </parser:RuleAttribute>
            <parser:RuleAttribute index=
"N77CD869002A11D88000C6FD28070181" name="subComponent"
usePreviousMatchSubstitutionAsDefault="false">
<SubstitutionRule substitute="Unknown"/>
            </parser:RuleAttribute>
            <parser:RuleAttribute index=
"N12069D2002A11D88000C6FD28070181" name="componentIdType"
usePreviousMatchSubstitutionAsDefault="false">
              <SubstitutionRule substitute="ProductName"/>
            </parser:RuleAttribute>
            <parser:RuleAttribute index=
"N3AB0CD6010F11D88000873B061CB0B3" name="location"
usePreviousMatchSubstitutionAsDefault="false">
<SubstitutionRule substitute="" useBuiltInFunction="true"/>
            </parser:RuleAttribute>
            <parser:RuleAttribute index=
```

```
"NCD2CCD7010F11D88000873B061CB0B3" name="locationType"
usePreviousMatchSubstitutionAsDefault="false">
              <SubstitutionRule substitute=""
useBuiltInFunction="true"/> </parser:RuleAttribute>
            <parser:RuleAttribute index=
"NDE661030DAB11D88000930FA6625271" name="componentType"
usePreviousMatchSubstitutionAsDefault="false">
              <SubstitutionRule substitute="HTTPServer"
useBuiltInFunction="false"/>
            </parser:RuleAttribute>
          </parser:RuleElement>
          <parser:RuleElement index=
"N8273E56002C11D88000C6FD28070181" name="situation">
            <parser:RuleElement index=
"NA8AA990002E11D88000C6FD28070181" name="situationType">
              <parser:RuleElement index=
"NF8AFFE9002F11D88000C6FD28070181" name="RequestSituation">
                <parser:RuleAttribute index=
"N2815562002F11D88000C6FD28070181" name="successDisposition"
usePreviousMatchSubstitutionAsDefault="false">
                  <SubstitutionRule match="" positions="" substitute=
"SUCCESSFUL"/>
                </parser:RuleAttribute>
                <parser:RuleAttribute index=
"N30BA5C1002F11D88000C6FD28070181" name="situationQualifier"
usePreviousMatchSubstitutionAsDefault="false">
                  <SubstitutionRule match="" positions="" substitute=
"REQUEST INITIATED"/>
                </parser:RuleAttribute>
                <parser:RuleAttribute index=
"NCCBFC40002F11D88000C6FD28070181" name="reasoningScope"
usePreviousMatchSubstitutionAsDefault="false"> <SubstitutionRule
match="" positions="" substitute="EXTERNAL"/>
                </parser:RuleAttribute>
              </parser:RuleElement>
              <parser:RuleElement index=
"N224169A084711D8800091856AF4A65C" name="ReportSituation">
                <parser:RuleAttribute index=
"N224169B084711D8800091856AF4A65C" name="reasoningScope"
usePreviousMatchSubstitutionAsDefault="false">
<SubstitutionRule substitute="INTERNAL" useBuiltInFunction="false"/>
                </parser:RuleAttribute>
                <parser:RuleAttribute index=
"N224169C084711D8800091856AF4A65C" name="reportCategory"
usePreviousMatchSubstitutionAsDefault="false">
<SubstitutionRule substitute="LOG" useBuiltInFunction="false"/>
                </parser:RuleAttribute>
```

```
              </parser:RuleElement>
            </parser:RuleElement>
            <parser:RuleAttribute index=
"ND4A1091002F11D88000C6FD28070181" name="categoryName"
usePreviousMatchSubstitutionAsDefault="false">
              <SubstitutionRule match="" positions="" substitute=
"RequestSituation"/>
              <SubstitutionRule substitute="ReportSituation"
useBuiltInFunction="false"/>
            </parser:RuleAttribute> </parser:RuleElement>
          <parser:RuleAttribute index=
"N3650A820F8511D88000A31D7605087A" name="creationTime"
usePreviousMatchSubstitutionAsDefault="true">
            <SubstitutionRule match=
"^(\d{4}/\d{2}/\d{2}\s\d{2}:\d{2}:\d{2})\s" substitute="$1"
timeFormat="yyyy/MM/dd hh:mm:ss" useBuiltInFunction="false"/>
          </parser:RuleAttribute> <parser:RuleAttribute index=
"NA9EB40208A811DF8000CBC9E948A33C" name="severity"
usePreviousMatchSubstitutionAsDefault="false">
 <SubstitutionRule match=
"^(\d{4}/\d{2}/\d{2}\s\d{2}:\d{2}:\d{2})\sINFO\s{1,2}-(.*)$"
substitute="10" useBuiltInFunction="false"/>
            <SubstitutionRule match=
"^(\d{4}/\d{2}/\d{2}\s\d{2}:\d{2}:\d{2})\sWARN\s{1,2}-(.*)$"
substitute="30" useBuiltInFunction="false"/>
            <SubstitutionRule match=
"^(\d{4}/\d{2}/\d{2}\s\d{2}:\d{2}:\d{2})\sERROR\s{1,2}-(.*)$"
substitute="50" useBuiltInFunction="false"/>
          </parser:RuleAttribute>
          <parser:RuleAttribute index=
"N66A0AFD08A811DF8000CBC9E948A33C" name="msg"
usePreviousMatchSubstitutionAsDefault="false">
            <SubstitutionRule match=
"^(\d{4}/\d{2}/\d{2}\s\d{2}:\d{2}:\d{2})\s(\w{4,5})\s{1,2}-(.*)$"
substitute="$3" useBuiltInFunction="false"/>
          </parser:RuleAttribute>
        </parser:RuleElement>
      </cc:Parser>
      <fmt:Formatter description="CBE Formatter" uniqueID=
"N36C36700F8511D88000A31D7605087A"/>
      <cc:Outputter description="Default Logging Agent" uniqueID=
"N36E80600F8511D88000A31D7605087A" type="LoggingAgentOutputter">
        <pu:Property propertyName="agentName" propertyValue=
"Sample Log logging agent"/>
        <op:LoggingAgentOutputterType agentName=
"Sample Log logging agent"/>
      </cc:Outputter>
```

```
    </cc:ContextInstance>
  </cc:Configuration>
</adapter:Adapter>
```