Universidade do Minho
Escola de Engenharia

Departamento de Informática

Mestrado em Engenharia Informática

Relatório de Actividade Profissional- Utilização de Programação Linear na Gestão de Risco e na Eficiência dos Mercados Financeiros

Paulo Jorge Ribeiro Moreira

Nº 24157

Introdução

A circular EEUM-CC-02/2012 formaliza a implementação do despacho RT-38/2011 ao nível da Escola de Engenharia da Universidade do Minho, apresentando a regulamentação para a creditação da formação adquirida por licenciados pré-Bolonha.

Concluí a licenciatura em Engenharia de Sistemas e Informática em 2001, e tenho 13 anos de experiência profissional na área de desenvolvimento de Software. Inscrevi-me no Mestrado de Engenharia Informática no ano lectivo de 2013/2014. Por conseguinte, e em conformidade com a circular supracitada, o presente relatório de actividade profissional tem como objectivo fornecer uma descrição detalhada das tarefas que fui desempenhando ao longo do meu percurso profissional, de forma a apresentar as experiências e competências adquiridas, enquadrando a relevância dessas mesmas competências no âmbito do Mestrado de Engenharia Informática, demonstrando assim que a minha licenciatura pré-Bolonha, em conjunto com o meu percurso profissional, justificam a obtenção do grau de mestre em Engenharia Informática.

Tal como referido, possuo uma experiência profissional de 13 anos, focada maioritariamente no desenvolvimento de software. Apresentar em detalhe todos os projectos e actividades desempenhadas durante um período desta duração é uma tarefa que excede os moldes dum relatório desta natureza. Por conseguinte, decidi fornecer uma visão geral dos diferentes desafios profissionais por que passei, e explorar em detalhe um desses desafios. O relatório tem assim a seguinte estrutura:

- Apresentação do percurso profissional, com uma explicação detalhada das tarefas desempenhadas nas diferentes empresas por que fui passando.
- Descrição de artigos, certificações e formações que adquiri e que são de alguma forma relevantes para o contexto do mestrado.
- Apresentação em detalhe dum projecto desenvolvido como parte da minha actividade profissional, e que consiste no desenvolvimento dum motor de gestão de risco ao nível dos mercados financeiros.

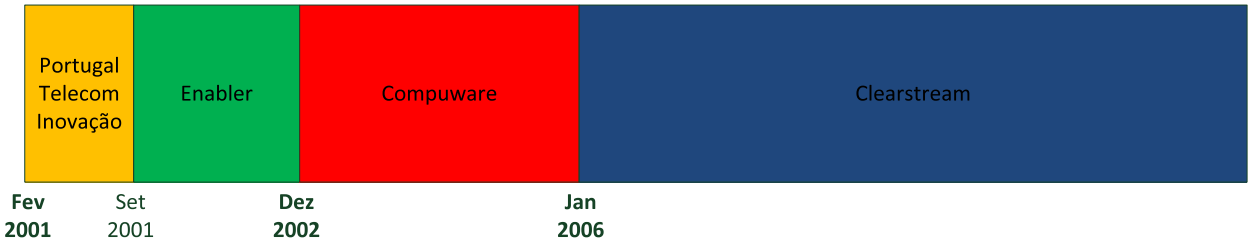
Percurso Profissional

O meu percurso profissional começou em Fevereiro de 2001, quando iniciei o meu estágio curricular na Portugal Telecom Inovação, e continuou ininterruptamente até aos dias de hoje. Durante este período trabalhei em quatro empresas e em três países diferentes. Estas experiências foram desempenhadas em equipas multiculturais e expostas a uma multitude de mercados em todos os pontos do globo. Apesar de ter passado por muitas realidades profissionais, a função que desempenhei durante toda a minha vida laboral foi sempre a de Engenheiro de Software, e as tarefas que pus em prática andaram invariavelmente à volta do ciclo de vida do desenvolvimento de software:

- Análise das necessidades ao nível das áreas de negócio e correspondente mapeamento em requisitos funcionais que exprimam essas mesmas necessidades num formato que possa ser aplicado ao sistema em mãos. Esta é uma das tarefas mais delicadas, que quando negligenciada pode facilmente condenar um projecto ao insucesso. Obriga a que o Engenheiro Informático saia da sua área de conforto e explore outras áreas de conhecimento:
    - Exige o conhecimento do domínio de negócio servido pelo software. O desenvolvimento de software é um processo que surge para resolver uma necessidade numa área que na maioria das vezes não tem nada a ver com informática. Desenvolver software sem ter uma visão detalhada da área em questão torna-se numa tarefa complicada e propícia a erros. Durante a minha actividade profissional adquiri competências nas áreas da Banca, Retalho e Telecomunicações.
    - Apesar de muito importante, o conhecimento da área de negócio não é suficiente. Cada organização tem uma cultura muito específica (processos, mercados, clientes, regulamentos internos, etc.) e essa cultura tem que ser bem assimilada pelo Engenheiro Informático. Temos assim que dominar não apenas a área de negócio em questão, mas também a forma como a empresa para que trabalhamos aborda essa área de negócio, e os processos que se relacionam.

- Planeamento da implementação. A este nível desempenhei variadas tarefas tais como arquitectura de alto nível, *design* de componentes específicos, selecção de tecnologias e construção de planos de desenvolvimento.
- Tarefas de implementação tais como desenvolvimento de código e testes unitários.
- Definição de boas práticas ao nível do desenvolvimento.
- Mentor de outros elementos e revisão de código.
- Suporte a ciclos de teste e à produção.
- Interacção com clientes.

Os 13 anos de experiência profissional dividem-se por 4 empresas. Seguidamente será apresentada uma descrição detalhada do perfil profissional que desempenhei em cada uma das empresas.



Portugal Telecom Inovação

O meu percurso profissional começou em Fevereiro de 2001, quando iniciei o meu estágio curricular na Portugal Telecom Inovação. Visto o estágio ter sido parte da licenciatura, farei uma breve descrição das responsabilidades que tive (a descrição detalhada do mesmo foi feita no âmbito da Licenciatura).

Em 2001, a Java ME (*Java Platform, Micro Edition*) dava os primeiros passos ao nível dos dispositivos móveis. A tecnologia tinha potencial elevado, mas a aplicabilidade era ainda algo incerta. Tal como muitas outras empresas na área das telecomunicações, a Portugal Telecom fez uma análise da tecnologia de forma a perceber como é que a mesma poderia ser utilizada em proveito da qualidade dos serviços fornecidos pelas empresas do grupo. O estágio que fiz decorreu no âmbito desta iniciativa, consistindo em fazer uma análise profunda à Java ME a vários níveis:

- Estudo detalhado da especificação. Nessa altura a Java ME definia uma série de perfis e configurações que tinham como alvo diferentes famílias de dispositivos. Analisei a especificação, os diferentes perfis de aplicabilidade em termos de dispositivos, e a forma como poderia ser aplicada à realidade dos produtos e serviços fornecidos pela Portugal Telecom.
- Análise das ferramentas de desenvolvimento existentes.
- Criei uma aplicação simples que pudesse ser executada num dispositivo móvel e que pudesse ser utilizada como protótipo.

Como resultado do estágio produzi um relatório detalhado que abordava os 3 pontos enumerados anteriormente. Esse relatório foi posteriormente utilizado como ponto de partida pela equipa de Serviços e Redes Móveis, para desenvolver uma série de serviços e aplicações baseadas na tecnologia Java ME.

Enabler

A Enabler foi uma empresa tecnológica focada no mercado de retalho, criada em 1997 e adquirida pela Wipro Technologies em Junho de 2006. Fazendo parte do Grupo Sonae, assumiu-se como o líder europeu em termos de implementação e suporte de sistemas destinados à área do retalho. Com sede no Porto, teve projectos espalhados em vários países.

O meu trabalho na Enabler começou em Setembro de 2001, tendo terminado em Dezembro de 2002. Durante esse período desempenhei a função de Analista, no escritório do Porto. O Analista tem uma participação activa em todo o ciclo de vida dum projecto - definição de requisitos funcionais, definição de requisitos técnicos, implementação e suporte. Em todas estas fases o Analista divide o seu tempo pelas instalações da Enabler e pelo escritório do cliente do projecto.

O tempo que passei na Enabler foi dedicado quase em exclusivo a um grande projecto de *Markdown Modelling* (MDM). Tive a oportunidade de participar no projecto desde o início até ao fim. O cliente do MDM foi uma das mais importantes cadeias de *department stores* britânicas, e consistiu na criação dum sistema que permitisse o planeamento inteligente de saldos e promoções, usando para tal dados históricos sobre as preferências dos clientes. Por

um lado, o sistema deveria fornecer mecanismos que permitissem aos utilizadores o planeamento das promoções de forma simples, mas eficiente. Por outro, o sistema deveria extrair conhecimento de outros sistemas (Sistema Operacional, *Data Warehouse*) de forma a auxiliar o utilizador no planeamento dessas mesmas promoções.

A definição de requisitos funcionais foi dividida em 3 partes- especificação de funcionalidades de negócio a implementar, integração com outros sistemas já existentes e definição do interface com o utilizador. Fui responsável pela integração com outros sistemas, elaborando um número de documentos que estabeleceram os interfaces entre o MDM e outros sistemas existentes, de forma a integrar a nova área de negócio com os processos vigentes. Participei também na discussão do interface com o utilizador, elaborando um protótipo em Visual Basic que proporcionou aos futuros utilizadores uma ideia sobre o *look and feel* da aplicação. Este protótipo mostrou-se muito útil, facilitando a discussão com o cliente e adaptação iterativa da oferta à vontade dos utilizadores. Como resultado produzi dois FRDs (*Functional Requirements Document*), um com a especificação dos interfaces entre o MDM e os sistemas existentes, e outro com uma especificação detalhada do interface com o utilizador.

A fase seguinte consistiu na escolha das soluções tecnológicas a utilizar, na qual participei elaborando um processo de análise. A J2EE *(Java 2 Enterprise Edition)* foi a tecnologia que melhor se encaixou no problema em causa. O sistema tinha um interface gráfico bastante complexo, o que impossibilitou a escolha duma solução Web (em 2001 as tecnologias Web estavam longe daquilo que são hoje em dia). Assim, Java Swing foi a tecnologia escolhida para a implementação do interface gráfico. Toda a lógica de negócio ficaria na camada do meio, implementada através duma série de EJBs (*Enterprise Java Beans*), e suportada por um Servidor Aplicacional WebLogic. O IBM DB2 daria suporte à camada de dados. A integração com o Sistema Operacional faria recurso ao IBM MOSeries. Teríamos assim um sistema em que os utilizadores trabalhariam no seu PC, utilizando para o tal uma aplicação desenvolvida em Java Swing. Esta aplicação interagia com os componentes de negócio (EJBs) situados ao nível do Servidor Aplicacional, que por sua vez interagia com uma Base de Dados DB2. A integração deste sistema far-se-ia a duas dimensões: 1) A Base de Dados seria alimentada a partir do *Data Warehouse*, utilizando-se para tal processos de fluxo suportados pelo DB2; 2) A integração com

o Sistema Operacional seria feita através do MQSeries, tendo que se desenvolver processos (em Java) de integração para o efeito. Todas estas escolhas foram apresentadas e justificadas no TRD- *Technical Requirements Definition*.

A fase de implementação correu de acordo com o previsto, havendo o cumprimento de todos os prazos delineados. Tive ao meu cargo o desenvolvimento de toda a integração com o Sistema Operacional, de parte do interface gráfico, e da lógica de negócio responsável pelas políticas de preço. Durante este período continuou a existir uma constante troca de informação com o cliente de forma a fazer pequenos ajustes.

Por fim, decorreu a fase de instalação da aplicação no cliente, e o suporte da mesma. Esta fase obrigou a constantes deslocações às instalações do cliente, na Inglaterra. Existiram alguns problemas iniciais que foram solucionados rapidamente. A estabilização do sistema deu-se de forma bastante rápida. Podemos dizer que este projecto teve um grau de sucesso bastante elevado. O nível de satisfação dos utilizadores foi muito positivo, e o proveito da implementação deste sistema tornou-se visível muito rapidamente. Desempenhei um papel bastante activo em todo o projecto. Visto as tecnologias adoptadas serem, à data, bastante recentes, os conhecimentos que trazia da licenciatura foram determinantes para a escolha e implementação da solução técnica a utilizar. Como os prazos foram bastante apertados, este projecto obrigou a uma constante dedicação. Envolveu a utilização de variadas tecnologias, o que implicou um estudo atento das mesmas. Devido à minha inexperiência na área do retalho, tive também cuidados especiais em dedicar algum tempo de estudo a esta área, adquirindo conhecimentos que se manifestaram muito proveitosos no desempenho da minha função. O facto de ser um cliente inglês obrigou também a um cuidado especial. Foi necessária uma adaptação cultural de forma a melhor entender as necessidades do cliente, e de forma a gerar um grau de confiança confortável. O MDM continua a ser utilizado presentemente.

Compuware

A Compuware é uma empresa Americana (com sede em Detroit) que fornece Software e Serviços Profissionais na área tecnológica. Com cerca de três décadas de história, a Compuware

já forneceu Software/Serviços a 25 000 das maiores e mais respeitadas organizações mundiais. Com escritórios e centros de desenvolvimento espalhados por todo mundo, agrega cerca de 12 000 colaboradores. A Compuware conta com um centro de desenvolvimento em Amesterdão, com cerca de 250 colaboradores. À data, este centro era responsável pelo desenvolvimento de três produtos- Uniface, OptimalJ e OptimalFlow.

A minha função na Compuware começou em Dezembro de 2002, prolongando-se até ao início de 2006. A minha posição inicial foi de Engenheiro de Software, pertencendo ao departamento de Integração do OptimalJ. O Engenheiro de Software tem como missão a participação no ciclo de vida completo duma *release* dum produto. Este ciclo de vida é composto por várias fases:

- Definições de Requisitos- Nesta fase definem-se todos os requisitos que a nova *release* deve satisfazer. Tem uma forte componente de investigação, onde se procuram novas soluções para dar resposta a problemas/necessidades existentes.
- Planeamento- São estabelecidos os planos de desenvolvimento. É planeada a gestão de tempo e de recursos para o resto do projecto.
- Desenvolvimento- As novas funcionalidades são desenvolvidas de forma a satisfazer os requisitos definidos.
- Resolução de Problemas- Os problemas detectados pelas equipas de teste são resolvidos nesta fase.

Todas estas responsabilidades estão directamente relacionadas com o ciclo de vida do produto. Existem outras funções que o Engenheiro de Software tem que desempenhar tais como a deslocação às instalações do cliente ou a gestão tecnológica de parcerias efectuadas com outras empresas.

A partir de Julho de 2003 passei também a acumular a função de *Build Manager*. O produto em que trabalhei tem uma equipa composta por cerca de 100 Engenheiros. Todos os dias existe uma série de novas funcionalidades que são adicionadas ao produto. Para fazer a gestão desta evolução foi criada uma infra-estrutura de *build* que funciona 24 horas por dia, 7 dias por semana, e que é controlada por um *Build Manager*. De forma geral, o *Build Manager* tem as seguintes responsabilidades:

- Proceder à compilação e criação de todos os executáveis do produto, incluindo as novas funcionalidades que estão disponíveis. No caso de este processo não ser bem-sucedido, deve encontrar os responsáveis e notificá-los do sucedido.
- Responsável pela monitorização da qualidade do produto, mais especificamente pelos testes QA (*Quality Acceptance*). Cada *build* é sujeito a mais de 100 testes de qualidade. Caso algum destes testes seja quebrado, o *Build Manager* tem a responsabilidade de descobrir o responsável e proceder à correspondente notificação.
- Monitorização de novas funcionalidades que são adicionadas ao produto. O *Build Manager* é responsável por estar atento ao evoluir do produto, podendo em determinadas situações recusar que certas funcionalidades façam parte do produto, se vierem a pôr em causa a qualidade ou a estabilidade do mesmo.

Acumulei as funções de Engenheiro de Software e de *Build Manager*, dedicando 90% do meu tempo à primeira e 10% à segunda.

Tal como referido anteriormente, o meu trabalho foi feito num produto chamado OptimalJ. O OptimalJ é uma ferramenta para desenvolvimento de aplicações J2EE. Baseia-se na MDA (*Model-Driven Architecture*), uma metodologia inovadora que centra o desenvolvimento de Software em modelos (de domínio e de aplicação, maioritariamente baseados em UML), recorrendo à geração automática de código.

Durante o período em que trabalhei nesta equipa, o OptimalJ tinha uma equipa de cerca de 100 Engenheiros, divididos por Amesterdão e Detroit. Esta equipa dividia-se em três departamentos- Arquitectura, *Core* e Integração. O departamento de Arquitectura tinha como objectivo o planeamento do produto a médio/longo prazo, definindo as direcções que o mesmo deve tomar de forma a dar resposta a necessidades futuras. O departamento de *Core* é responsável pelo desenvolvimento do núcleo central do OptimalJ. O departamento de Integração, tal como o nome indica, é responsável pela integração do OptimalJ com outros produtos existentes no mercado. A minha função foi desempenhada inicialmente no departamento de Integração, e posteriormente no departamento de *Core*. Tive a meu cargo as seguintes responsabilidades:

- Implementação do suporte para a integração com o *Mainframe*. De forma resumida, o objectivo seria permitir que o utilizador importasse um programa Cobol, e a partir desta informação o OptimalJ criava automaticamente modelos e a geração automática de código (via JCA- *Java Connector Architecture*) que permitisse o acesso ao programa Cobol no Mainframe por parte duma aplicação J2EE. Ao nível do *Mainframe*, implementou-se o suporte para CICS e para IMS. Posteriormente estendi este módulo de forma a suportar integração assíncrona com o Mainframe através do uso de JMS (*Java Message Service*) e MQSeries.
- Implementação do suporte à integração do OptimalJ com SAP. A filosofia é muito semelhante à do ponto anterior, mas desta vez visando o sistema SAP como alvo.
- Alteração do OptimalJ de forma a suportar J2EE 1.4. Tive que alterar o *core* do produto de forma a que os modelos e código gerado automaticamente pudessem reflectir as novas funcionalidades introduzidas pela J2EE 1.4- *Timer Service*, alterações na JCA e suporte mais flexível para *Web Services*.

A metodologia de desenvolvimento seguiu o manifesto *Agile*, com uma abordagem iterativa em que as diferentes funcionalidades foram sendo gradualmente desenvolvidas em ciclos pequenos, com um paradigma de trabalho muito orientado aos testes. Todo o processo de planificação, implementação e suporte dos módulos descritos anteriormente esteve à minha responsabilidade. Começou com o processamento de algum feedback de clientes acerca de determinadas necessidades que existiam e que não estavam a ser satisfeitas. Seguiu-se um processo de investigação sobre as possíveis soluções e a correspondente viabilidade técnica. Depois de analisadas as possíveis soluções e escolhida aquela que se afigurava mais adequada, procedi à planificação do desenvolvimento. Esta fase obriga não só à gestão de tempo e recursos, mas também a uma análise detalhada sobre a forma como as novas funcionalidades se vão integrar com o produto já existente, e quais os possíveis riscos. Seguidamente procedeu-se à implementação, e por fim a fase de estabilização onde se procede à correcção de problemas. No final do período de estabilização, quando o produto está pronto a ser distribuído, existe uma semana de formação. Durante este período, colaboradores de todos os escritórios, das áreas de vendas, marketing, serviços profissionais e serviços de consultoria

deslocam-se a Amsterdão de forma a receberem formação no produto que sairá em breve. No final da formação estes colaboradores são convidados a expressar a sua opinião sobre as novas funcionalidades, escolhendo aquelas que segundo o seu juízo terão mais saída em termos de mercado.

Clearstream

A Clearstream é uma empresa financeira que oferece serviços de *Settlement* e de Guarda de títulos financeiros. Sedeada no Luxemburgo, faz parte do grupo Deutsche Börse, e é uma das empresas dominantes nesta área de negócio, com volumes de 12 triliões de euros em termos de Guarda e cerca de 10 milhões de transacções financeiras por mês.

Trabalho na Clearstream desde Janeiro de 2006, como Engenheiro de Software. Em 2005 a Clearstream optou pela criação dum novo sistema dedicado à gestão de colateral (esta área de negócio será explicada em detalhe mais à frente neste relatório). Foi no âmbito desta iniciativa que me juntei à empresa, e lá continuei nos 8,5 anos que se seguiram. Participei na criação do sistema de gestão de colateral, sistema esse que cresceu a um ritmo muito elevado, hoje em dia representando uma área vital da empresa, sendo um dos sistemas de gestão de colateral mais avançados a nível global, e tendo clientes espalhados por todo o globo. Além dos nossos clientes, fazemos também o *outsourcing* deste tipo de serviço para vários mercados domésticos- Alemanha, Espanha, Brasil, Austrália, África do Sul, Singapura e Noruega.

A minha tarefa andou sempre à volta do desenvolvimento de software e todas as actividades que se relacionam:

- Análise dos requisitos de negócio. Temos uma equipa vasta de vendas e de gestão de produto que está constantemente a recolher as novas tendências do mercado. Daqui surgem novos requisitos de negócio que são depois discutidos com a equipa de desenvolvimento, onde estou inserido. São avaliadas as possibilidades de implementação, o impacto no sistema, e é feita a priorização dos requisitos.
- Os requisitos de negócio são mapeados em requisitos funcionais. Este processo é delicado e consiste na criação de documentos funcionais, que exprimem de forma

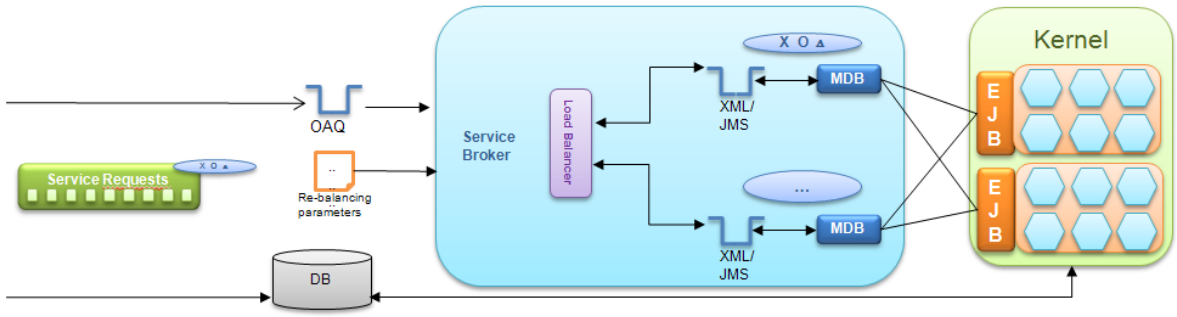
correcta as necessidades de negócio, e que servem como especificação para o desenvolvimento.

- Arquitectura de alto nível. O sistema de gestão de colateral tem hoje uma dimensão considerável e a evolução constante força a que a arquitectura seja frequentemente revista de forma a considerar as novas funcionalidades. Tenho um papel preponderante na arquitectura do sistema. Participo na definição da direcção a tomar quer em termos de arquitectura quer em termos de tecnologias a utilizar, para que o sistema funcione como um todo, que se reuse ao máximo o código, e que existam princípios gerais que sejam adoptados a todos os níveis do sistema.
- Desenho das variadas funcionalidades. O sistema divide-se num conjunto de componentes que gerem diferentes funcionalidades. A gestão destes componentes é feita de forma cuidada, para que: 1) o novo componente se integre de forma harmoniosa com o resto do sistema; 2) se acomode não apenas as necessidades presentes, mas que também se tente perceber as tendências futuras de forma a facilitar a evolução e adaptação a requisitos futuros; 3) o desenho seja escalável, quer em termos de performance, quer em termos de suporte (planear cenários de contingência, recuperação de dados, etc.). Tenho à minha responsabilidade o desenho dum grande número de funcionalidades. Sou também responsável por rever os desenhos feitos por outros engenheiros.
- Desenhei e implementei uma grande parte das *frameworks* que dão suporte base ao sistema, tais como mecanismos de excepções, de testes unitários, de persistência de dados, de gestão de transacções, de cache, etc.
- Uma parte significativa do meu trabalho consiste na implementação de código. O processo de implementação utiliza os documentos funcionais como ponto de partida. Dependendo do tipo de projecto, a metodologia varia entre Scrum e metodologias mais formais e pesadas. No entanto, é sempre dada ênfase aos testes. Assim, a implementação consiste no desenvolvimento do código, na criação de testes unitários e de testes *End-To-End*.

- Sou responsável por tarefas tais como a delineação de boas práticas em termos de desenvolvimento, mecanismos de integração contínua, revisão de código, e definição de metodologia de desenvolvimento. Trabalhamos com integração contínua, em que sempre que existem alterações ao código se despoleta a compilação do código, a execução dum número elevado de testes unitários e o novo código é automaticamente auditado de forma a verificar se respeita uma série de regras básicas. Além disso temos vários ambientes de teste automáticos que estão 24 horas por dia a executar uma bateria de 2000 cenários *End-To-End,* e que detectam qualquer regressão. Fui responsável pelo estabelecimento desta metodologia de trabalho e faço a gestão destes ambientes com o auxílio de alguns elementos mais juniores.
- Devido à dimensão do sistema, e ao facto de termos clientes espalhados por todo o globo, temos actividade elevada durante todo o dia. A equipa de desenvolvimento tem uma estrutura de suporte à produção da qual faço parte. É uma actividade que exige uma boa capacidade de gestão de stress, devido à urgência na resolução de alguns dos problemas em mãos.
- Desde à muito que sou responsável pela performance do sistema. Em cada projecto temos um ciclo de testes dedicado exclusivamente à medição da performance. O sistema tem uma série de critérios de aceitação em termos de desempenho. Nesse ciclo de testes verificamos se os critérios são satisfeitos. Muitas vezes é necessário efectuar alterações ao código de forma a fazê-lo mais robusto. Essa tarefa cabe-me a mim. Tenho também que monitorizar o sistema em produção de forma a verificar que não existem desvios relativamente ao comportamento esperado.
- Devido à criticidade deste tipo de sistemas (os erros têm por norma um preço muito elevado) as fases de teste são muito longas. Uma das minhas responsabilidades passa por dar suporte a esses ciclos de teste. Chegamos a ter cinco projectos a serem testados em paralelo, e por conseguinte esta tarefa pode ser muito desgastante.
- Tenho a responsabilidade de servir como mentor a vários colegas. Esta tarefa aplica-se a 3 níveis diferentes: 1) Auxiliar os colegas na adaptação à empresa e aos processos. Normalmente aplica-se aos primeiros meses de trabalho de colegas que chegam à

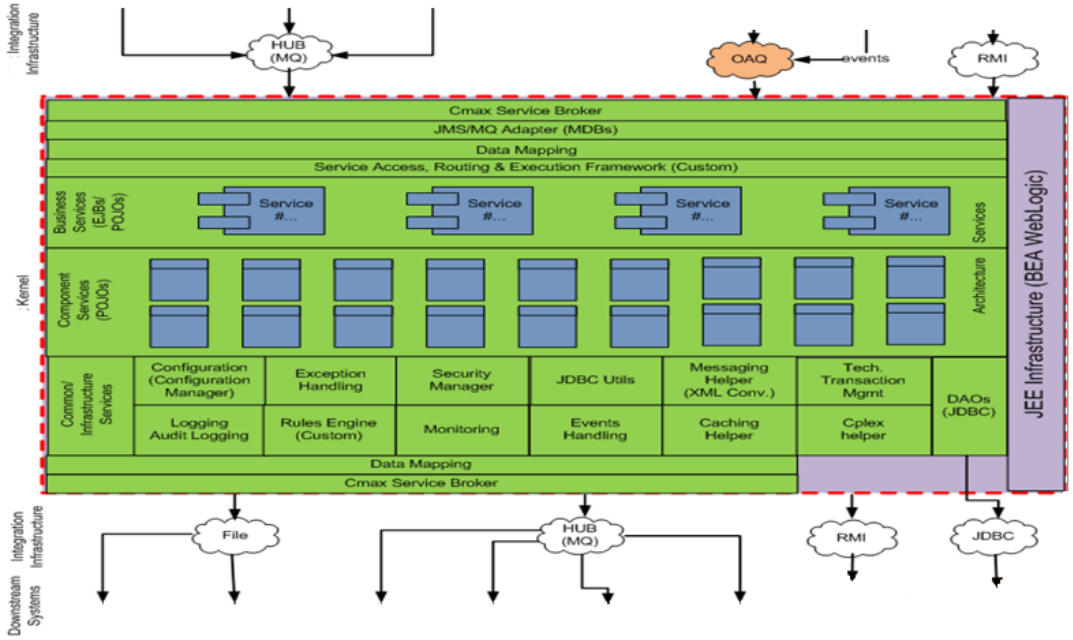
empresa; 2) Dar apoio no que toca a compreender a área de actividade em que trabalhamos; 3) Suporte ao nível técnico, explicando o sistema, auxiliando e aconselhando nos desafios por que passam, e revendo o trabalho feito por eles.

O sistema em que trabalho começou a ser desenhado no começo de 2006. É um sistema baseado na SOA (*Service-Oriented Architecture*), constituído por um número elevado de serviços e *workflows* que fazem uso dos serviços para implementar cenários de negócio. Esta abordagem permitiu-nos criar uma arquitectura modular que facilmente acomoda novas funcionalidades, e que promove a reutilização de código. O sistema é constituído por várias aplicações, com mais de 300 *workflows* diferentes. Tem um núcleo central responsável pela gestão de colateral, com toda a lógica inerente (avaliação, gestão de perfis de risco, gestão de saldos de conta, etc.). A execução de *workflows* e a coordenação entre serviços é feita através duma aplicação de encaminhamento de mensagens (*Service Broker*) também ela implementada em Java.

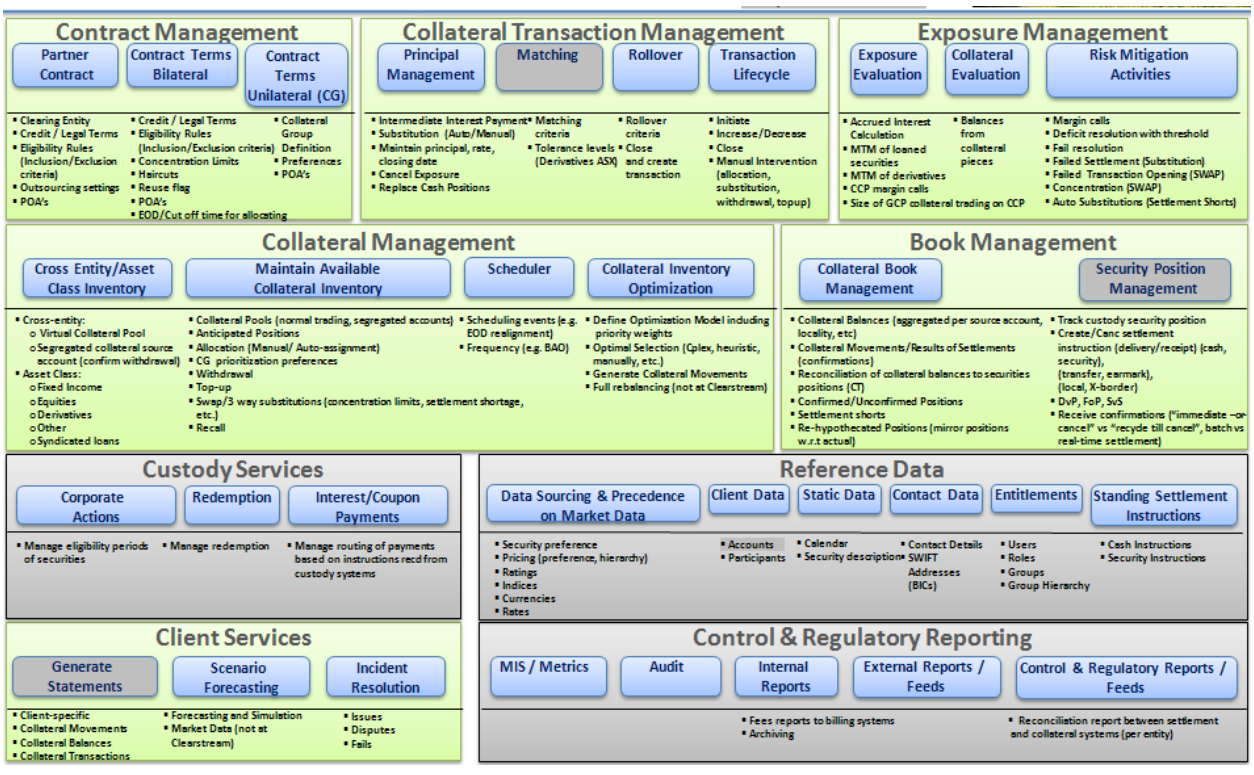


Arquitectura ao nível da coordenação de mensagens enviadas aos serviços

O sistema tem também uma série de outras aplicações que implementam variadas funcionalidades que vão desde o interface Web para clientes, a geração de relatórios para reguladores e clientes, a integração com mais de 30 sistemas, etc.



Arquitectura de alto nível (sistemas de entrada e saída foram removidos por questões de confidencialidade)

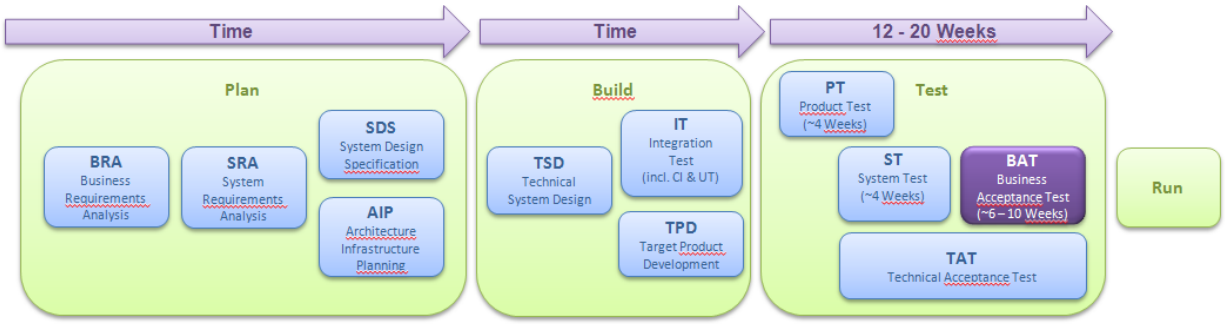


Modelo funcional do sistema de gestão de colateral

Em termos de tecnologias trabalhamos com a plataforma Java, usando uma grande parte das tecnologias Java EE, quer para a lógica de negócio, quer para lógica de apresentação, recorrendo a um Servidor Aplicacional Weblogic. Desde a criação do sistema que sempre acompanhamos a evolução das últimas versões das tecnologias em causa, de momento usando Java 7 e Java EE 6. Em termos de Bases de Dados usamos Oracle e recorremos bastante às tecnologias que se relacionam- uma parte importante do nosso código é PL-SQL e usamos *Oracle Advanced Queueing* para troca de mensagens. Existe mais uma série de outras tecnologias que usamos (Perl, MQSeries, etc.) mas que têm menos relevância no contexto do sistema.

O sistema que temos utiliza uma Base de Dados com cerca de 5 *Terabytes* e a gestão de dados é central ao desempenho do sistema. Temos várias dezenas de interfaces com mais de 30 sistemas diferentes.

As metodologias variam de projecto para projecto. A área financeira é muito regulada e por vezes estamos condicionados nas abordagens que adoptamos, dependendo dos parceiros, dos reguladores e das exigências internas. Sempre que possível adoptamos uma metodologia *agile*, com iterações múltiplas em que cada iteração consiste numa série de requisitos informais, que são implementados e testados em ciclos de curta duração. Neste caso existem equipas multidisciplinares (negócio, desenvolvimento e testes) que trabalham em conjunto nas iterações. Outros projectos têm abordagens mais formais e pesadas, em que se definem requisitos formais de negócio, passando-se depois aos requisitos funcionais, posteriormente aos documentos técnicos, seguidos da implementação e dos testes. São processos mais morosos e menos eficientes, mas que se adequam melhor à realidade de determinados projectos de complexidade mais elevada. Este ciclo mais formal tem normalmente as fases apresentadas abaixo:



Independentemente da abordagem que se adopte, existe sempre uma série de princípios que têm de ser imperiosamente respeitados pela equipa de desenvolvimento:

- Antes da implementação existem sempre discussões técnicas em que os elementos da equipa são chamados a analisar o problema e a opinar sobre a solução a seguir.
- A implementação é muito orientada ao teste. Todo código que é criado e/ou alterado tem que ter testes unitários (testes básicos que testam unidades granulares do código) e testes *End-To-End* (testes num ambiente real que simulam cenários de execução do sistema). Para tal utiliza-se as *frameworks* que desenvolvi para testes. Neste momento temos 2000 testes *End-To-End*, que cobrem uma grande parte dos cenários de execução. Os testes são executados regularmente (2 vezes por dia) e os resultados são analisados e publicados automaticamente de forma a aferir a qualidade dum certo projecto.
- O código alterado é sempre revisto. Esta revisão consiste em duas partes:
    - Auditoria automática do código durante o processo de compilação. Temos um sistema de integração contínua que detecta alterações no código e procede à respectiva compilação e auditoria automática. Para tal usamos ferramentas standard (checkstyle, findbugs, PMD, etc.) que verificam se o código respeita os critérios definidos ao nível da equipa, além de detectarem a presença de alguns erros mais evidentes.
    - Temos também um processo de revisão manual de código. Alterações nas partes mais sensíveis do sistema têm que ser revistas por um Engenheiro diferente daquele que as fez.

- Existem reuniões informais diárias (*daily standup meetings*) em que cada Engenheiro faz um sumário rápido daquilo que está a fazer de forma a que os elementos da equipa estejam familiarizados com todos os fluxos de actividade que estão a decorrer.

Tal como referido anteriormente, é difícil resumir todas as competências e experiências vivenciadas numa empresa durante mais de oito anos. Foram muitos os projectos por que passei, o sistema sofreu inúmeras mudanças e lidei com variadas tecnologias diferentes. Mais tarde neste relatório será apresentado um projecto em específico, onde apresentarei em detalhe a área em que trabalho, e darei uma explicação elaborada sobre um dos problemas que tive em mãos e a forma como o resolvi.

É relevante salientar os diferentes aspectos do meu perfil profissional que fui desenvolvendo durante esta caminhada profissional. Um dos pontos em que investi bastante esforço e tempo foi em adquirir conhecimento na área bancária e nas especificidades da gestão de colateral. Quando cheguei à Clearstream não tinha mais do que um punhado de conhecimentos básicos, que tive que rapidamente desenvolver. Ao longo destes anos aprendi bastante sobre esta matéria, quer com colegas, quer através da leitura de livros técnicos, quer com a frequência de formações e mais recentemente com o MBA. Este conhecimento mostrou-se precioso na execução da minha tarefa como Engenheiro de software. Posso melhor compreender os requisitos que me são dados, consigo comunicar de forma mais eficiente com os colegas da área de negócio, e consigo colocar questões pertinentes e sugestões. É-me também mais fácil antecipar evoluções, de forma a planear o meu trabalho duma forma mais flexível e adaptável a necessidades futuras. Não partilho da filosofia defendida por algumas metodologias de desenvolvimento, que argumentam que o desenvolvimento de software pode ser feito sem qualquer conhecimento da área de actividade em causa, e que, logo que os requisitos sejam estabelecidos de forma clara e detalhada, o Engenheiro de software pode simplesmente implementá-los sem se preocupar em perceber o propósito por trás do trabalho que desenvolve. Com a experiência profissional que tenho fui aprendendo que, um bom conhecimento do domínio a que o software se aplica, é uma qualidade que nos proporciona vantagens significativas a vários níveis:

- A cooperação com os colegas do negócio funciona muito melhor porque existe uma atmosfera de respeito e compreensão mútua (falamos a mesma "língua"), o que proporciona um melhor entendimento.
- O resultado do trabalho de desenvolvimento é muito mais acertado, na medida em que se reduz a falta de alinhamento entre as expectativas do negócio e o resultado da implementação. Para tal contribui o facto de quem implementa perceba o porquê daquilo que está a implementar.
- Mesmo em termos técnicos a tarefa fica facilitada. Por exemplo, a revisão do código feita a outros colegas resulta muitas vezes em algumas sugestões de optimização no que toca à forma como o código põe em prática uma certa funcionalidade. O mesmo objectivo pode ser implementado de várias formas, e muito trabalho de optimização anda à volta de implementar uma certa funcionalidade de negócio de forma mais simples, ou seja, as optimizações funcionais têm tanto ou mais relevância que as optimizações técnicas, e advêm dum bom conhecimento da área de negócio.

Assim sendo, investi e continuo a investir uma parte significativa do meu tempo livre na aprendizagem de competências na área financeira porque, na minha opinião, me permite desempenhar de forma mais eficiente o meu trabalho. Esta tarefa vai além de conhecer o negócio em profundidade, consiste também em fazer a ponte entre o negócio e as tecnologias em específico. É uma tarefa que alguma bibliografia designa de Engenharia Financeira. O livro "*Java Methods for Financial Engineering*" é um bom exemplo deste tipo de empreitada- assume que o leitor domina uma série de conceitos financeiros e os cálculos matemáticos que se relacionam, assume que o leitor tem um conhecimento profundo de Java, e faz a ponte entre ambos.

Outra das áreas em que fiz um investimento contínuo foi em manter actualizadas as competências técnicas que adquiri na licenciatura. A área da programação está em mutação constante e obriga a um estudo permanente. Mantive-me sempre actualizado com as últimas tendências, principalmente nas tecnologias que uso mais frequentemente. O nosso sistema usa uma plataforma Java EE com uma Base de Dados Oracle e um Servidor Aplicacional Weblogic. Por conseguinte estive sempre ao corrente do evoluir destas tecnologias. Fiz várias certificações

em Java e na Base de Dados Oracle. Adquiri conhecimento profundo em Weblogic, o qual aplico na administração dos ambientes que temos. Estudei bastante a área de SQL em Oracle, com ênfase na forma como as *queries* são executadas, de forma a fazer optimização no acesso à camada de dados que é uma parte fulcral do nosso sistema, e que exige sistematicamente atenção redobrada.

O alvo deste processo de aprendizagem não foram apenas as tecnologias per se, mas também tudo aquilo que anda à volta do ecossistema de programação tal como ferramentas de desenvolvimento, ferramentas de monitorização e *profiling*, bibliotecas, etc. Além do estudo de artigos e livros, fiz também várias formações e certificações, e escrevi alguns artigos. Tentei ter um papel activo na comunidade, passando as minhas opiniões às equipas que elaboram as especificações, do qual destaco alguns comentários que enviei ao líder da especificação para EJB 3.1 (*Enterprise Java Beans*) relativamente à versão *draft* que lançaram e que resultou em algumas alterações à mesma.

Adquiri também competências em áreas mais genéricas como *design patterns*, princípios de arquitectura e integração de sistemas, e fui acompanhando novas tecnologias, e sempre que justificado, fui adoptando essas tecnologias ao sistema em que trabalho, como por exemplo Bases de Dados NoSQL. Este processo não se focou apenas em dominar as tecnologias, mas também adquirir o bom senso para perceber em que condições se devem aplicar, e como as aplicar. Para tal aprendi muito com a experiência de outras instituições, através do desenvolvimento duma rede de contactos. Contribuí também através da partilha das minhas experiências, estimulando um espírito de comunidade e partilha de conhecimento e vivências profissionais. Durante alguns períodos tive participações activas em fóruns da Oracle e da IBM com o intuito de esclarecer dúvidas em determinadas tecnologias.

A licenciatura forneceu-me uma série de conhecimentos vastos que vão muito além das tecnologias da informação, e que também uso frequentemente. A área de gestão de risco financeiro obriga a determinados cálculos que envolvem alguma complexidade (estatística avançada, integrais, derivadas, transformadas de Laplace, etc.). De forma a lidar com estes cálculos tive que fazer uso dos conhecimentos da licenciatura, e durante os últimos anos tentei

alargar o meu conhecimento nesta área, desenvolvendo algumas competências na área da análise quantitativa.

Outra das áreas em que trabalho activamente é na implementação de estratégias de optimização (descrito em detalhe mais tarde neste relatório). Para tal fiz uso do conhecimento de investigação operacional que adquiri na licenciatura. Desenvolvi competências na área de Programação Linear com inteiros, as quais foram aplicadas no trabalho que desenvolvo. Desenvolvi os modelos matemáticos de optimização que usamos na gestão de colateral, usando Programação Linear para tal. Cooperei com equipas da IBM e da Gurobi de forma a apurar estes modelos e adaptá-los à plataforma técnica que temos.

Para desempenhar o papel de Engenheiro Informático de forma bem-sucedida é preciso dominar as tecnologias de informação, as ciências base como Matemática ou Física, e a área de actividade em que a tarefa se desenvolve. Mas tudo isto não é suficiente. São raros aqueles que desenvolvem a sua função em isolamento. Na vasta maioria dos casos o trabalho é desenvolvido em equipa, com interacções com colegas, chefias, clientes, etc. Para tal é preciso amadurecer competências de comunicação e interacção. Por melhor que seja o nosso trabalho, de nada serve se não for correctamente comunicado e documentado. Por mais competências que um Engenheiro tenha, não chegará longe se não conseguir interagir de forma eficiente, motivando os outros a cooperar e a remar no mesmo sentido. Assim, nos últimos anos desenvolvi algumas competências em termos de *soft skills*, de forma a comunicar de forma clara, a saber ouvir, motivar e conseguir o melhor que cada individuo tem a dar. Os ambientes de trabalho mudaram muito nos últimos anos, tornaram-se bastante mais diversificados. Algumas áreas como a informática estão especialmente expostas a estas mudanças e hoje em dia é difícil encontrar uma equipa de dimensão média que não tenha pessoas de diferentes nacionalidades e com culturas distintas. Por conseguinte, as competências de liderança, interacção e motivação no local de trabalho exigiram a uma adaptação de forma a criar uma atmosfera em que todos são respeitados, tirando proveito daquilo que cada um tem de melhor, beneficiando da diversidade. Durante a minha carreira trabalhei em vários países, sempre em equipas com heterogeneidade de culturas. Além disso interagi com clientes espalhados pelos vários continentes, com mentalidades muito distintas. No escritório em que trabalho temos

perto de 30 nacionalidades diferentes. Consequentemente durante a minha carreira fui amadurecendo o conhecimento sobre as diferentes sensibilidades e culturas de forma a poder interagir de forma respeitosa e proveitosa.

Actividade Extraprofissional

Durante o meu percurso profissional exerci uma série de actividades extralaborais que contribuíram para o meu desenvolvimento curricular. Apresento aqui as mais relevantes. Os comprovativos das diferentes actividades são apresentados em anexo (algumas das formações não forneceram comprovativos).

Formações

- 2002- BEA- "*Developing Enterprise Applications with BEA WebLogic Server Using EJB and JMS*"- Lisboa, Portugal
- 2002- Unicenter- "Curso de Formação de Formadores"- Porto, Portugal.
- 2003- *British Language Training Centre*- "*Business English – level C*"- Amesterdão, Países Baixos.
- 2004- Charles Hamilton Associates- "*Powers of Communication*" "- Amesterdão, Países Baixos.
- 2004- Info2People - "*Object Oriented Design for Java*"- Amesterdão, Países Baixos.
- 2005- Info2People - "*Designing Java Web Services applications with J2EE 1.4*"- Amesterdão, Países Baixos.
- 2005- IBM - "*WebSphere Application Server 6.0 – deployment and management*"- Amesterdão, Países Baixos.

Certificações

- Sun Certified Java Programmer for Java 2 Platform 1.4. Exame concluído com 92%.
- Sun Certified Business Component Developer for J2EE 1.3. Exame concluído com 90%.
- Compuware UML 2.0 Certified.
- Sun Certified Java Programmer for Java SE 6.0. Exame concluído com 97%.

- Sun Certified Business Component Developer for Java EE 5. Exame concluído com 90%.
- Oracle Database 11g SQL Fundamentals I.

Artigos

- Autor do artigo "*From PersonalJava to J2ME: Some Introductory Ideas*" escrito em 2001 a convite do editor do site www.microjava.com. Este artigo foi também publicado pelo MovilForum, o site da Telefonica dedicado à indústria *wireless*.
- Participação no artigo "Moving to SOA in J2EE 1.4", publicado pelo JDJ (*Java Developers Journal*) em Fevereiro de 2006.
- Autor do artigo "*EJB 3.1- A Significant Step Towards Maturity*" publicado em Abril de 2009 no TheServerSide, um dos sites mais respeitados no que toca à tecnologia Java- http://www.theserverside.com/tt/articles/article.tss?l=EJB3-1Maturity.

Prémio Jovem Engenheiro

Tive duas participações no concurso "Prémio Inovação Jovem Engenheiro" organizado pela Ordem dos Engenheiros:

- 2003- Autor do trabalho "Novas Metodologias para o Desenvolvimento de Software", apresentando a MDA (*Model-Driven Architecture)* como novo paradigma para o desenvolvimento de aplicações. Este trabalho teve como base a nova abordagem de desenvolvimento introduzida pelo OptimalJ, o produto em que trabalhei enquanto desempenhei funções na Compuware. Em poucas palavras, a MDA foca o desenvolvimento na criação dum vasto conjunto de modelos que representam a especificação do sistema. Divide os modelos em duas famílias- os PIM (*Platform Independent Models*) e os PSM (*Platform Specific Models*). Os PIM usam uma DSL (*Domain-Specific Language*) que neste caso específico era baseada maioritariamente no UML. Uma ferramenta de MDA (como o OptimalJ) permite a geração automática de PSMs a partir de PIMs, e de código a partir dos PSMs. Assim sendo, o desenvolvimento

foca-se maioritariamente na criação de modelos de alto nível, deixando os detalhes específicos da tecnologia a cargo da ferramenta, consequentemente facilitando a migração de tecnologia. A cada momento o Engenheiro pode tomar o controlo quer dos PSMs quer do código gerado, de forma a introduzir um comportamento específico não previsto pela ferramenta (ou que não seja possível de especificar ao nível dos PIMs). A ferramenta detecta as alterações efectuadas e mantém o fluxo PIM/PSM/Código em sintonia. Este trabalho documentou este novo paradigma e a forma como pode ser utilizado para desenvolver aplicações de complexidade elevada, reduzindo os custos e o risco dum projecto.

- 2007- Autor do trabalho "Automação de Projectos de Software", apresentando novas abordagens para integração contínua e a automatização da auditoria da qualidade do código duma aplicação. Este trabalho baseou-se numa das tarefas iniciais que coloquei em prática na Clearstream. A empresa não tinha qualquer tradição em termos de automação de projectos, quer ao nível de testes, quer ao nível integração contínua, quer ao nível da análise automática de código. Introduzi uma série de práticas que mudou radicalmente a forma como as equipas de desenvolvimento trabalham em aplicações Java. Adoptei a utilização de sistemas de integração contínua (inicialmente com CruiseControl e posteriormente com Jenkins). Criei uma *framework* para testes unitários que executa todos os testes durante o processo de *build*. Posteriormente desenvolvi uma ferramenta de execução de testes *End-To-End*- permite a criação de cenários de teste (em XML), gerando todos os dados no sistema de forma a replicar o contexto em que o cenário é executado, executando os passos do cenário, e comparando os resultados com as expectativas. Esta ferramenta é parte fundamental do processo de desenvolvimento e teste, sendo utilizada por um número elevado de Engenheiros e Analistas, cobrindo uma parte significativa dos testes que executamos. Adoptei também uma série de ferramentas que analisam automaticamente o código (análise sintáctica, semântica, etc.) e defini um grupo de regras de boas práticas que servem como referência a essas mesmas ferramentas. O trabalho submetido à Ordem

dos Engenheiros apresentou todo este processo duma forma genérica e aplicável a qualquer organização.

MBA

Encontro-me presentemente a fazer um MBA com especialização na área financeira, na *Sacred Heart University*, na cidade do Luxemburgo. O objectivo deste desafio é complementar as minhas competências técnicas com uma visão mais aprofundada do negócio, de forma a melhor desempenhar a minha função. Adicionalmente, tenho adquirido novas competências no que toca à gestão de equipas e liderança, desenvolvimento profissional, e ética profissional.

Até ao momento concluí 9 disciplinas, estando sensivelmente a meio. O MBA é feito em horário pós-laboral e em paralelo com a minha actividade profissional.

Associações Profissionais

- Membro da Ordem dos Engenheiros.
- Membro do IEEE.

Programação Linear no Financiamento com Cobertura de Risco

Este capítulo apresenta um dos projectos em que tenho vindo a trabalhar durante alguns anos. Fornece uma introdução à área de negócio e parte para a descrição detalhada dum dos desafios com que me deparei, e a solução adoptada. Faz uma introdução teórica à Programação Linear e à Programação Inteira, e a sua aplicabilidade no projecto em causa. Apresenta também uma série de recomendações e conhecimentos que foram adquiridos durante este projecto. É sempre mantido um nível de abstracção elevado e os conceitos estão simplificados, porque por um lado o trabalho efectuado é extenso e a sua apresentação em detalhe exigiria um relatório de várias centenas de páginas, e por outro porque muita da informação é confidencial e é propriedade intelectual da empresa para que trabalho. Apesar do nível de abstracção elevada, é apresentada muita informação relevante, e por conseguinte **este capítulo está sujeito a confidencialidade e o seu conteúdo não deverá ser utilizado para outros propósitos que não o processo de creditação do mestrado**.

Adicionalmente, é dada mais enfâse à abordagem teórica do problema e às experiências adquiridas, não sendo explorado em detalhe a implementação dos conceitos feita ao nível do sistema de informação. No entanto, todos os conceitos apresentados seguidamente foram implementados pelo autor e servem esta área de negócio desde os últimos 6 anos.

Financiamento com Cobertura de Risco

Crédito

Em termos teóricos, crédito é a confiança que permite a uma entidade fornecer recursos a outra entidade, sendo esses mesmos recursos (ou recursos de valor equivalente) reembolsados numa data posterior. Normalmente o recurso ao crédito pressupõe o pagamento duma taxa de juro de forma a compensar o mutuante, quer pela perca temporária dos recursos, quer pelo risco que assume.

O crédito tem um papel central na sociedade. Por um lado permite estimular o consumo das famílias, por outro permite financiar a actividade das empresas. Podemos assim concluir que o crédito é imprescindível para o desempenho sustentado da actividade económica.

É imperativo que a concessão de crédito seja cuidadosamente monitorizada de forma a evitar desvios que poderão ter consequências nefastas para a sociedade:

- Crédito fácil estimula o abuso no consumo, causando bolhas especulativas que mais cedo ou mais tarde irão resultar em crises financeiras.
- Crédito difícil funciona com um entrave à vida das famílias e das organizações, podendo paralisar a actividade dos agentes económicos, tendo efeitos desastrosos.

Assim sendo, cabe aos reguladores estimular um mercado em que exista confiança, para que mutuantes se sintam tentados a conceder crédito. Este mercado tem que ser também regulamentado de forma a evitar que mutuantes concedam crédito arriscado e que mutuários façam uso abusivo de crédito.

Crise do Crédito

Uma crise de crédito caracteriza-se por uma redução significativa do financiamento a crédito, normalmente causada por uma diminuição nos fundos disponíveis para empréstimo, ou por uma alteração súbita nas condições exigidas a quem requer crédito.

A recente crise do *subprime* é um bom exemplo duma crise de crédito. Durante um período de tempo houve um grupo de instituições financeiras que concederam crédito hipotecário de alto risco, criando uma bolha especulativa no mercado imobiliário. Esta bolha alimentou-se à custa de crédito fácil e taxas de juro reduzidas. O aumento dos juros fez com que os mutuários de alto risco perdessem a capacidade de honrar as suas responsabilidades de crédito, fazendo com que um número elevado de imóveis fosse penhorado. A bolha imobiliária rebentou, os preços desceram a pique, arrastando outras famílias para uma situação de incumprimento.

Esta situação atirou várias instituições financeiras para a insolvência. Muitas destas instituições eram sobejamente respeitadas e eram tidas como referências sólidas. Inevitavelmente gerou-se um clima de falta de confiança. A grande maioria das instituições financeiras deixou de conceder crédito. Os bancos deixaram de se poder financiar com outros bancos, o crédito às

empresas bloqueou quase por completo (o pouco crédito que continuou a ser disponibilizado foi exercido a taxas proibitivas) e as famílias deixaram de poder recorrer ao crédito. Este clima de instabilidade levou à paralisia quase total da actividade económica. As consequências não foram mais desastrosas devido à rápida reacção que foi posta em prática pelos diferentes bancos centrais.

A crise do *subprime* deixou a descoberto algumas vulnerabilidades na forma como o sistema de crédito funciona:

- A falta de regulamentação na forma como o crédito é concedido. O epicentro do problema resume-se a todo o crédito de alto risco que foi concedido. Tal só foi possível devido à falta de regulamentação adequada.
- Um sistema de crédito entre instituições financeiras que se baseava na suposição de que tais instituições nunca ficariam insolventes. Antes da crise do *subprime*, a grande maioria do crédito entre bancos era feito sem qualquer garantia. O bom nome da instituição era garantia suficiente. A crise do *subprime* trouxe uma nova realidade, em que instituições respeitadas entraram rapidamente em colapso. Consequentemente os fluxos de crédito congelaram, passou-se dum ambiente de confiança abundante para uma atmosfera de receio constante.

O caos financeiro que se viveu obrigou à rápida adaptação das práticas de crédito. Muitas das soluções encontradas já existiam desde à muito, mas nunca foram adoptadas de forma significativa devido ao ambiente de confiança que se vivia.

Colateral como Cobertura de Risco

Tal como visto anteriormente, a crise do *subprime* deixou a descoberto algumas das fraquezas no sistema de crédito. De forma a que empresas e famílias pudessem recorrer ao financiamento necessário, foi preciso estimular o crédito entre bancos.

Devido à quebra de confiança teve que se encontrar formas de crédito em que mutantes pudessem ter a garantia de que receberiam a quantia emprestada, mesmo no caso de o mutuário entrar em falência. Só assim seria possível restituir a calma necessária para que o sistema financeiro pudesse continuar a funcionar com normalidade.

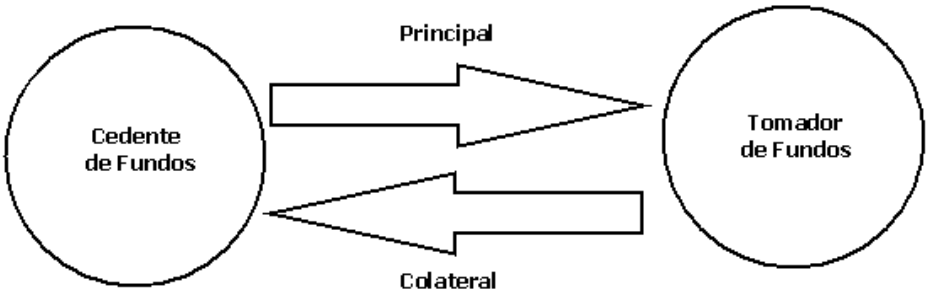
Uma das soluções encontradas foi a utilização de colateral. O recurso a colateral como garantia para cobrir o risco dum crédito é uma ferramenta que existe desde há muito tempo, mas nunca foi aplicada de forma significativa ao crédito entre instituições financeiras.

O termo colateral é usado para descrever qualquer tipo de activo que seja utilizado para garantir o empréstimo dum determinado recurso. No caso de o crédito não poder ser honrado, o colateral é utilizado como substituto. Por conseguinte, é imperativo que o valor do colateral seja o mais aproximado possível do montante emprestado, devendo também reflectir o juro correspondente ao período decorrido. Só assim se garante que, por um lado, o mutuante será justamente compensado no caso de o mutuário entrar em incumprimento, e por outro, que o mutuante não receberá mais do que aquilo que lhe é devido.

O recurso a colateral, e a correspondente gestão de forma a que reflicta de forma precisa o valor do crédito, passou a ser uma técnica de gestão de risco adoptada por muitas instituições. Em 2001 estimava-se que existia um total de 200 mil milhões de dólares alocados como colateral, enquanto que no final de 2008 este volume atingiu um valor de 2,162 biliões, ou seja, mais de dez vezes superior. Os valores apresentados dizem respeito apenas a uso de colateral para garantir créditos entre instituições financeiras.

Modelo de negócio

Existem diversas variações de uso de colateral (reportes, empréstimo de títulos, etc.). Mas todas elas obedecem ao mesmo modelo de negócio. Existe um acordo entre duas partes mediante o qual um dos agentes (tomador de fundos) recorre ao crédito duma certa quantidade de recursos (chamado principal) concedido pela outra parte (cedente de fundos), fornecendo um número de activos como garantia, comprometendo-se a reembolsar os recursos, mais um juro acordado. Este modelo pode ser aplicado a qualquer tipo de recursos, mas para o efeito deste trabalho consideramos que quer os recursos cedidos quer os recursos usados como colateral se limitam a dinheiro ou a títulos financeiros (acções, obrigações, opções, etc.). As utilizações mais comuns são o empréstimo de dinheiro como principal, usando títulos como colateral, e o empréstimo de títulos usando outros títulos como colateral.



O processo de gestão dum crédito colateralizado poder ser desempenhado pelas partes envolvidas (bilateral) ou por uma entidade externa independente. A utilização duma instituição externa e neutra confere uma segurança acrescida ao processo, e os reguladores têm vindo a fazer exigências no sentido de que se utilize cada vez mais esta abordagem à gestão de colateral.

Os recursos envolvidos apresentam elevada volatilidade no que toca ao respectivo valor- tanto a cotação dos títulos como as taxas de câmbio estão em mutação constante. Por conseguinte, a utilização de colateral exige a constante avaliação quer de principal quer de colateral, de forma a fazer com que ambos estejam em sintonia:

= + é à

A utilização de colateral indiscriminado não é garantia suficiente. Suponhamos que se utiliza como colateral um activo cujo valor no mercado seja equivalente ao valor do principal, mas cuja liquidez seja reduzida. Isto quer dizer que no caso do tomador de fundos entrar em incumprimento, o cedente de fundos pode ter dificuldade em vender o activo usado como colateral, obrigando-o a baixar o preço, fazendo com que o valor do colateral não seja suficiente para cobrir o principal. Cabe assim ao cedente de fundos a definição dum perfil de risco que defina as características dos activos que são aceites como colateral. Cabe à entidade responsável por gerir o crédito a missão de acompanhar a evolução dos títulos no mercado, e

sempre que necessário adequar o colateral utilizado para que o perfil de risco do cedente de fundos continue a ser respeitado.

Em linhas gerais, temos um modelo de negócio cujo funcionamento assenta em duas condições obrigatórias:

- Os activos usados como colateral têm que cobrir o valor do crédito mais juros sobre o tempo de crédito decorrido. A cada momento o valor dos dois deve ser tão aproximado quanto possível.
- Os activos usados como colateral têm que respeitar o perfil de risco definido pelo cedente de fundos.

Cobertura de risco ao nível dos Sistemas de Informação

Desafio

Tal como apresentado anteriormente, a utilização de colateral sofreu um aumento exponencial nos últimos anos, e tornou-se uma ferramenta imprescindível para a gestão de risco no crédito e para a liquidez nos mercados. Os sistemas de informação que dão suporte a este tipo de negócio tiveram também que evoluir e fornecer a resposta para os novos desafios que foram surgindo:

- Constante avaliação dos créditos existentes tanto ao nível do principal como do colateral.
- Acompanhamento dos mercados de forma a reflectir quer a evolução dos preços quer a evolução das características dos títulos. A volatilidade elevada dos mercados dificulta esta tarefa.
- Adaptação em tempo real do colateral usado de forma a que os perfis de risco sejam respeitados.

As tarefas descritas acima são de implementação relativamente fácil. Requerem bastante trabalho mas as tarefas em si não são de complexidade relevante, centram-se no processo de obtenção de dados de referência atempadamente e na respectiva utilização.

Mas com o tempo o mercado foi exigindo um nível adicional de serviço na gestão de colateral- a optimização da utilização de activos de forma a que um tomador de fundos que contraia créditos com variados cedentes possa maximizar a utilização dos seus activos como colateral. O verdadeiro desafio não está na utilização de activos para a cobertura de risco mas sim na utilização optimizada desses activos. Uma instituição financeira é normalmente tanto cedente como tomador, e tem inúmeros créditos em simultâneo. É de importância imperativa que um sistema de gestão de colateral faça uma alocação eficiente de activos de forma a permitir a maximização de fundos que uma determinada instituição pode tomar em paralelo.

Durante muitos anos esta tarefa foi desempenhada manualmente. Os sistemas de informação eram responsáveis pela gestão de colateral, mas a tarefa de optimização era deixada a cargo dos analistas que adaptavam o uso desse mesmo colateral de forma a maximizar a respectiva utilização. Nos finais da década de noventa começaram a surgir alguns algoritmos para o efeito. Mas a maioria destes algoritmos era muito simplista e pouco flexível, obrigando a que o sistema seguisse o modelo de negócio assumido pelo algoritmo. Adicionalmente, estes algoritmos eram pouco eficientes e o processo de optimização era executado pouco regularmente. Esta era a situação quando a crise do *subprime* chegou. Tornou-se urgente encontrar uma solução que respeitasse as seguintes condições:

- Optimizar a utilização de colateral.
- Executar rapidamente para que a optimização possa ser exercida numa base regular (o quão mais frequente, melhor).
- Ser flexível de forma a assimilar novos critérios no que toca à definição de perfis de risco.

Foi este o contexto em que começou o trabalho que aqui apresento.

Conceitos Base

De forma a modelar uma solução para o problema em mãos, é necessário compreender os conceitos base. Para efeitos deste trabalho, muita da terminologia foi simplificada:

- Agente- Instituição envolvida num crédito, como cedente ou tomador de Fundos.

- Cedente de Fundos- Parte responsável por fornecer o crédito, recebendo colateral em troca.
- Tomador de Fundos- Parte responsável por receber o crédito, fornecendo colateral de forma a garantir o cumprimento do crédito.
- Conta- Unidade legal pertencente a um agente, utilizada para acolher activos financeiros.
- Activo Financeiro- Os activos financeiros são activos intangíveis (isto é, que não têm existência física) que conferem ao respectivo detentor o direito ao recebimento de benefícios em data futura, cabendo a responsabilidade do seu pagamento à entidade que procedeu à sua emissão. Para efeito de simplificação consideramos que os activos financeiros se resumem a dinheiro, acções e obrigações.
- Instrumento Financeiro- Activo financeiro que se baseia em documentos representativos de situações jurídicas homogéneas que visem, directa ou indirectamente, o financiamento de entidades públicas ou privadas. No contexto deste trabalho consideram-se apenas acções e obrigações. Um instrumento financeiro tem uma série de características. Nos sistemas financeiros modernos existem centenas de atributos que são considerados relevantes. Para o efeito de trabalho usamos uma série de atributos simples:
    - Tipo de Instrumento Financeiro- Acção ou Obrigação.
    - Notação de Risco- Classificação de risco do valor mobiliário. Consideramos 3 níveis de risco- A, B, C.
    - Preço- Valor unitário. Na prática os preços das obrigações funcionam de forma mais complexa, mas vamos considerar um sistema simplificado em que cada tipo de instrumento financeiro tem uma unidade associada e consequentemente um preço unitário.
    - Volatilidade do Preço- Medida estatística das flutuações do preço.
    - Emissor- Entidade que emitiu o instrumento financeiro.
    - País da Entidade Emissora- País em que o emissor está registado.
    - Tipo de Emissor- Governo ou Empresa.

- Valor da Emissão- Capitalização bolsista no caso de acções, valor emitido no caso de obrigações.
- Moeda de Denominação- Moeda em que o instrumento foi emitido.

- Saldo - Quantidade dum activo financeiro disponível numa conta.
- Crédito- Actividade através da qual um cedente empresta uma quantidade de activos financeiros a um tomador. A quantidade de activos cedidos designa-se de principal. Um crédito tem uma data inicial, uma data de maturidade, e uma taxa de juro. Na data inicial, o principal é transferido do cedente para o tomador. Em troca, o tomador passa ao cedente uma série de activos financeiros que funcionam como colateral. Durante a duração do crédito o valor do colateral tem que ser sempre equivalente ao valor do principal mais o valor do juro relativo à duração decorrida. Na data de maturidade o principal e o juro são devolvidos ao cedente, e o colateral é devolvido ao tomador. Vamos considerar apenas créditos que usam dinheiro como principal e instrumentos financeiros como colateral.
- Perfil de Risco dum Cedente- Cada cedente de fundos pode descrever um perfil de risco. Este risco tem duas dimensões. Em primeiro, indica aquilo que é aceitável como colateral. Por exemplo, um cedente pode optar por apenas aceitar instrumentos financeiros com notação A. Esta dimensão é de fácil implementação e não tem grande relevância para o contexto deste trabalho. A segunda dimensão diz respeito ao perfil de risco ao nível de limites. Por exemplo, o tomador aceita instrumentos financeiros com qualquer notação, mas do total de colateral aceita no máximo 10% de instrumentos com notação C e 20% com notação B. Implicitamente, a cada momento, temos que ter no mínimo 70% do colateral com notação A.
- Perfil de Preferências dum Tomador- Cada tomador pode descrever um perfil de preferências no que toca aos activos usados como colateral. Por exemplo, o tomador pode definir um perfil que dá preferência à utilização de instrumentos com notação C, seguido dos com notação B, e por fim os de notação A. Sempre que possível (dependendo do perfil de risco do cedente), dever-se-á respeitar as preferências do tomador. Num cenário realista é normal que o perfil de risco dum cedente e as

preferências do tomador caminham em direcções opostas, isto é, o cedente tem tendência a exigir activos de alta qualidade, enquanto que o tomador tem tendência a preferir fornecer como colateral os activos de menor calibre.

Programação Linear

A área de gestão de colateral sofreu um crescimento significativo nos últimos 10 anos. No entanto, esta área de negócio já existe desde algumas décadas. Tal como mencionado anteriormente, as soluções existentes até ao passado recente assentaram sempre num processo de optimização manual e posteriormente em algoritmos simples feitos à medida de modelos de negócio existentes.

Foi este o meu ponto de partida, tendo como objectivo encontrar uma solução que permitisse automatizar por completo o processo de gestão de colateral de forma eficiente, rápida e assente num modelo flexível que se possa adaptar às alterações constantes no mundo financeiro.

A Programação Linear é uma disciplina da matemática que se centra na área de optimização. A Programação Linear inclui uma classe de modelos da Programação Matemática relacionados com a alocação eficiente de recursos limitados a actividades matemáticas, com a finalidade de atingir um determinado objectivo. Um modelo de Programação Linear tem uma função objectivo linear e uma série de condições lineares. Adicionalmente, também se assume a linearidade das combinações das equações referidas.

$$\max\ 5x + 12y$$

$$20x + 10y \leq 200$$

$$10x + 20y \leq 120$$

$$10x + 30y \leq 150$$

$$x \geq 0$$

$$y \geq 0$$

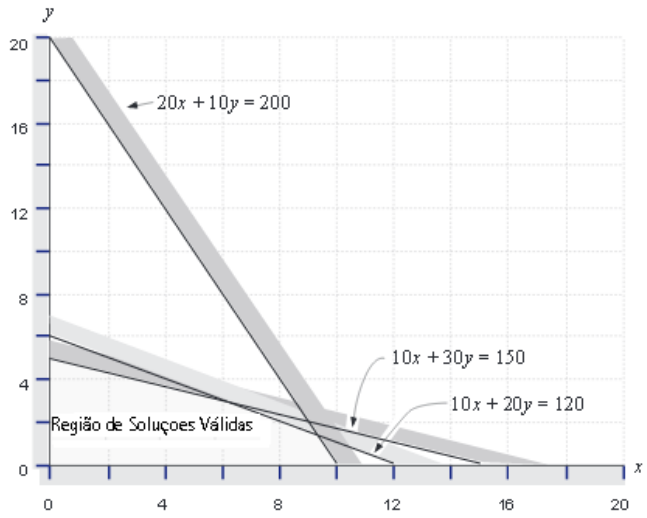
Exemplo dum problema de Programação Linear

Geometricamente, as restrições lineares definem uma região de soluções que consiste num poliedro convexo. Um problema poderá não ter solução em dois contextos:

1- Existem condições que se contradizem e por conseguinte a região de soluções é vazia.

2- A região de soluções é ilimitada na direcção da função objectivo, fazendo com que não haja uma solução óptima.

Caso as duas condições anteriores não se verifiquem, existe sempre uma solução óptima, e essa solução óptima é sempre alcançada num dos vértices do poliedro. A solução óptima não tem que ser única, podem existir várias soluções óptimas.

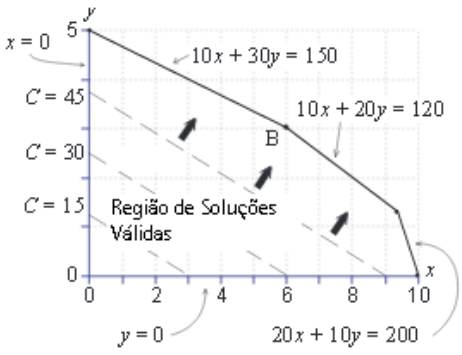


Região de Soluções Válidas para o problema apresentado

Graficamente podemos desenhar o grupo de rectas definidas pela função objectivo. À medida que estas rectas de deslocam para o exterior, aumenta o valor da função objectivo. A solução óptima é o último ponto da região de soluções tocado pelas rectas da função objectivo à medida que estas se deslocam para o exterior. No exemplo a solução óptima é dada pelo ponto B.

$C = 5x + 12y$

$C = 15,\ C = 30,\ C = 45.$



Solução Óptima através de análise gráfica

Como visto anteriormente, a Programação Linear exige que tanto a função objectivo como as condições sejam lineares. Recordemos que, para que seja linear, uma função deve possuir duas propriedades: aditividade e homogeneidade.

$$f(x + x') = f(x) + f(x')$$
$$f(ax) = af(x)$$

Condições para a linearidade duma função

Existe uma série de algoritmos que permitem determinar a solução óptima dum problema de Programação Linear. Estes algoritmos escalam de forma aceitável com o aumento do número de variáveis e/ou condições. Com as implementações modernas destes algoritmos é possível resolver rapidamente problemas que tenham vários milhares de variáveis e de condições.
Existem vários algoritmos para a resolução de modelos de Programação Linear. O simplex é um exemplo, tendo 65 anos, continua a ser um dos algoritmos mais eficientes. Os métodos dos pontos interiores são uma alternativa, assim como o método projectivo.

Formulação do Problema de Gestão de Colateral em termos de Programação Linear

A gestão de colateral encaixa-se perfeitamente na Programação Linear. É um problema típico de gestão de recursos, em que e alocação dos mesmos deve ser optimizada de forma a maximizar proveito:

- Ao nível dum crédito deve-se utilizar o colateral de forma a respeitar o perfil de risco do cedente, e a seguir tanto quanto possível as preferências definidas pelo tomador.
- Ao nível do sistema deve-se fazer reavaliação constante dos recursos disponíveis e recursos alocados como colateral, e sempre que possível, adaptá-los de forma a atingir dois objectivos extremamente importantes:
    - Gerir o risco do mercado, fazendo com que todos os tomadores tenham o seu perfil de risco respeitado. Não é demais salientar a importância deste ponto. Só assim se consegue fazer com que o mercado esteja preparado para eventos

inesperados- mal os primeiros sinais de crise são detectados, o sistema reage rapidamente.

- Fomentar a liquidez do mercado. O mercado de crédito é constituído por uma teia complexa de agentes que desempenham em simultâneo papéis de tomador e cedente. Cabe ao sistema analisar estes relacionamentos e tentar maximizar a utilização de colateral de forma a que cada agente possa tirar o melhor partido dos activos que tem.

Um modelo de Programação Linear é constituído por uma função objectivo, e uma série de condições. Podemos traduzir estes conceitos ao nível dum crédito em específico, sendo o modelo definido da seguinte forma:

- Função Objectivo- Minimizar o risco do crédito.
- Condições- Respeitar o perfil de risco do cedente.
- Custo- Os custos da função objectivo podem ser materializados numa unidade de valor à escolha. Para efeitos de simplificação utilizaremos valores em euros.

De forma a poder formular o nosso problema, precisamos de definir as seguintes funções:

- CustoUnitario(IFi) – Dá o valor em euros duma unidade do instrumento financeiro IFi.
  ex. Empresa A é transaccionada na bolsa ao preço de 3 euros por acção => CustoUnitario(A) = 3
- ValorCredito(C)- Dá o valor do crédito C em euros. Este valor inclui o valor dos fundos cedidos mais o valor dos juros para o período de crédito decorrido até à data presente.
- Saldo(Ai, IFj)- Saldo do instrumento financeiro j na conta i.
- Q(IFi, Cj)- Quantidade de instrumento financeiro i utilizado como colateral no crédito j.
- ContribuiParaLimite(IFi, Lj)- Devolve um valor booleano que é 1 caso o instrumento financeiro IFi contribua para o limite Lj, 0 em caso contrário. Um limite representa um critério no perfil de risco dum cedente de fundos.

Podemos assim definir a função objectivo, que consiste em minimizar a diferença entre o valor do crédito e o valor dos activos utilizados como colateral. As variáveis a determinar são Q(IFi, C), ou seja, a quantidade de cada instrumento financeiro utilizada como colateral num determinado crédito C.

$$- \quad =1 \qquad \times \ (\quad , \ )$$

C- Crédito em causa

IFi- Instrumento Financeiro i disponível para ser utilizado como colateral no crédito C

As condições são todas definidas ao nível da utilização do colateral e respectivos limites. A soma do valor de todos os activos usados como colateral, e que correspondem ao perfil dum determinado limite, não deverá ser superior ao valor imposto pelo limite. Por exemplo, se o cedente de fundos definir que entre os activos usados como colateral o valor dos activos emitidos em Portugal não poderá exceder um terço do valor total do crédito, então haverá uma condição que indica que a soma do valor dos activos cujo país de emissão é Portugal terá que ser inferior ao valor dum terço do crédito:

$$\forall \quad =1 \qquad \times \ (\quad , \ ) \times \qquad (\quad , \ ) <$$

$$(\ )$$

l – limite de utilização de instrumentos financeiros com determinadas características

n- número de activos utilizados como colateral no crédito C

Existe uma série de outras condições que são utilizadas para garantir as regras básicas. A primeira garante que o valor dos activos utilizados como colateral não excede o valor do crédito. Normalmente existem margens de tolerância que são utilizadas a este nível, mas no âmbito deste exercício vamos considerar que o valor do colateral não poderá nunca exceder o valor do crédito.

$$\geq \quad =1 \qquad \times \ (\quad , \ )$$

Outra das condições base é que, para cada activo alocado como colateral, a quantidade usada não poderá exceder o saldo existente. Assim sendo, no contexto dum tomador de fundos, para cada instrumento financeiro disponível nas suas contas, a quantidade desse instrumento

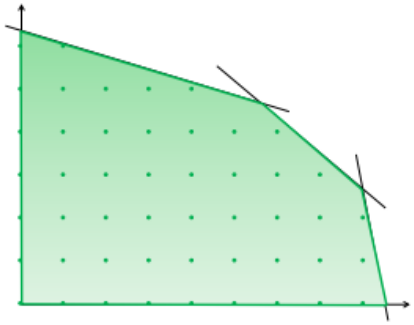
alocada como colateral em todos os créditos participados pelo tomador não poderá exceder a soma dos saldos existentes nas diferentes contas.

$$\forall \quad (\quad =1 \qquad , \quad \leq \quad =1 \qquad (\quad , \quad)$$

Temos assim um problema em que quer função objectivo, quer condições, são equações lineares. Perfila-se como um problema em que a utilização de Programação Linear se adapta perfeitamente.

Programação Inteira

Tal como visto anteriormente, o modelo de equações aplicado à gestão de colateral destina-se a calcular a quantidade de cada instrumento financeiro alocada para a cobertura dum crédito. Estas quantidades são de natureza inteira. O arredondamento duma solução óptima determinada por uma resolução pura de Programação Linear não garante uma solução óptima. Bem pelo contrário, a solução poderá ser ineficiente e até quebrar as condições. Os problemas de Programação Inteira são semelhantes aos problemas de Programação Linear, com a diferença de que parte ou a totalidade das variáveis de decisão apenas podem conter valores inteiros. Enquanto que na Programação Linear termos uma região de soluções válidas, na Programação Inteira o conjunto de soluções válidas é dado pelos pontos da região de soluções válidas em que as variáveis têm valores inteiros.



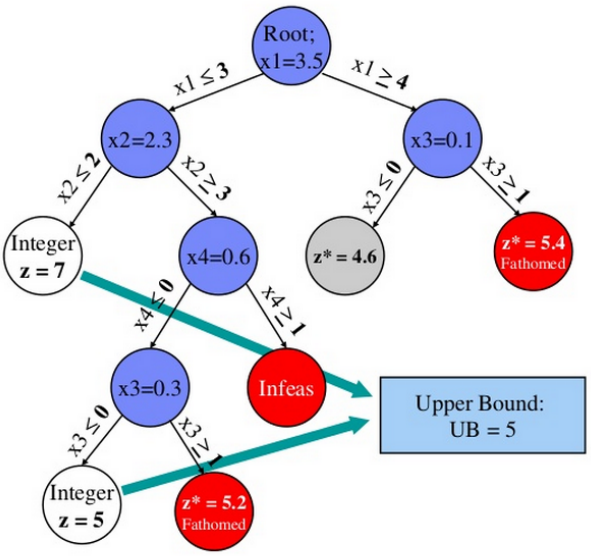
Região de soluções válidas para P.L. versus conjunto de soluções válidas para P.I.

A Programação Inteira tem maior aplicabilidade no que toca aos problemas de vida real, mas os métodos de resolução de modelos são muito mais complexos, principalmente quando se tem um número elevado de variáveis.

Existem várias metodologias para resolver um problema de Programação Inteira, mas todas elas consistem num método iterativo, onde se relaxa a condição inteira e se resolve o modelo usando Programação Linear. No caso da solução obtida ser inteira então pára-se. Caso não seja, reduz-se a região de soluções válidas, eliminando as zonas que não incluem nenhuma solução inteira. Ao novo problema relaxado é aplicada a metodologia habitual de problemas de Programação Linear, e continua-se até se obter uma solução inteira.

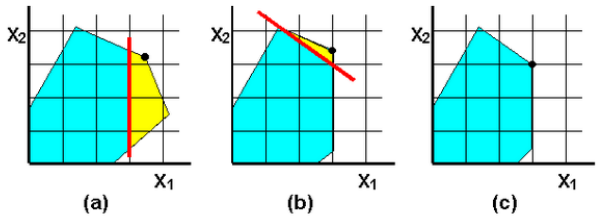
Um dos algoritmos mais utilizados é o B&B (*Branch & Bound*). Utiliza uma procura em árvore para ir eliminado as zonas de soluções onde não existem soluções inteiras. Começa por calcular uma solução para o problema relaxado. Caso a solução encontrada não seja inteira, selecciona uma das variáveis cujo valor encontrado não foi inteiro e divide o problema em dois subproblemas (por exemplo se x1 = 1.5, um dos subproblemas passa ter a condição x1 <=1 e outro passa a ter a condição x1 >= 2). O algoritmo consiste em 3 passos:

- *Branch*- Consiste na divisão que se vai fazendo dos problemas em subproblemas, e que vai eliminando as regiões com soluções não inteiras.
- *Bound*- Em cada ponto da árvore resolve o problema relaxado. Caso obtenha uma solução inteira, regista o valor da função objectivo. A melhor solução mantida até ao momento é designada de incumbente.
- *Prune*- Elimina os ramos da árvore que são desnecessários: 1) Aqueles em que os subproblemas não têm solução inteira; 2) Aqueles em que os subproblemas encontram uma solução cujo valor da função objectivo é pior que o valor da solução incumbente.



Exemplo da execução dum B&B

Uma alternativa consiste no método de geração de planos de corte. Esta abordagem resolve os problemas de Programação Inteira refinando o espaço de soluções até encontrar uma solução inteira. Mas em vez de dividir a região de soluções em subdivisões, tal como o B&B, vai adicionando condições (cortes) ao problema, o que reduz o espaço de soluções. Calcula-se a solução do problema relaxado e adiciona-se o corte (caso a solução não seja inteira). Os novos cortes têm que ser cuidadosamente adicionados de forma a não alterar a natureza do problema. Um corte deve satisfazer duas condições: 1) todas as soluções inteiras válidas continuam a ser válidas depois do corte; 2) a solução do problema relaxado torna-se uma solução inválida após o corte.



Exemplo da execução dum Plano de Corte

Na grande maioria das vezes o B&B superioriza-se ao algoritmo de planos de corte. No entanto, os planos de corte tiveram e continuam a ter um papel fundamental na evolução da Programação Inteira, tal com será visto mais adiante.

O problema de optimização de colateral é um problema de natureza inteira, e por conseguinte temos que fazer uso da Programação Inteira. Como lidamos com instituições financeiras, podemos facilmente encontrar agentes com dezenas de contas, centenas de créditos em paralelo, milhares de títulos financeiros, e com perfis de risco complexos. Deparamo-nos assim com um cenário em que quer o número de variáveis quer o número de condições pode ascender às centenas de milhares. Com este tipo de volumes, a árvore B&B vai inevitavelmente crescer de forma exponencial e a resolução do problema em tempo útil torna-se impossível.

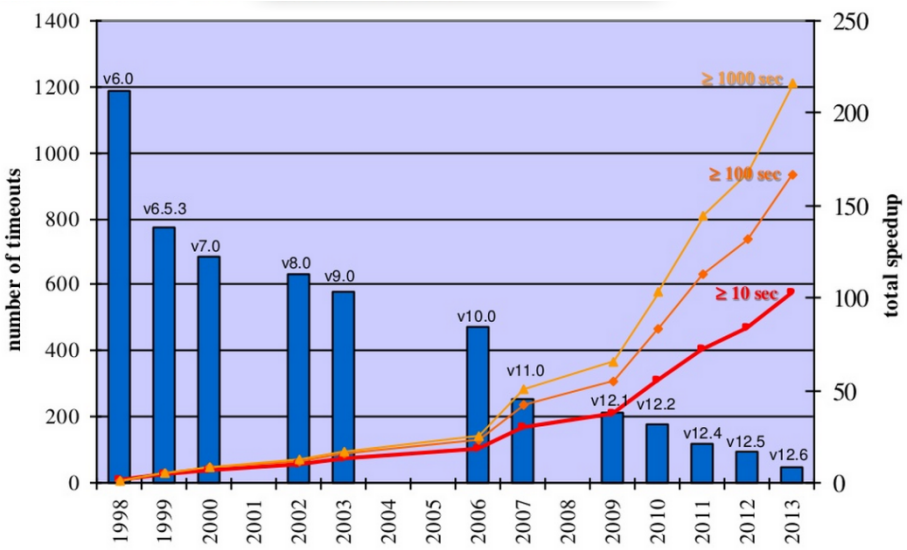
Nos últimos anos os algoritmos de Programação Inteira sofreram uma evolução tremenda. O B&B foi complementado com outras técnicas de forma a conseguir lidar com problemas de complexidade elevada:

- *Presolve*- Consiste na limpeza da formulação do problema antes da aplicação do B&B, removendo variáveis e/ou condições que são redundantes. Restringe os limites das variáveis de forma a fortalecer o modelo. Por exemplo, no caso de x1 e x2 serem variáveis inteiras positivas, e no caso de se ter a condição "2x1 + 2x2 <= 1", a fase de *presolve* pode imediatamente concluir que quer x1 quer x2 têm obrigatoriamente que ser 0, e a condição pode ser removida.
- Planos de corte- Os planos de corte, em conjunto com o B&B (*Branch & Cut*), têm tido um papel preponderante na melhoria dos tempos de resolução de modelos de Programação Inteira. O algoritmo começa por adicionar alguns cortes antes de começar o B&B. À medida que a árvore se vai expandido, novos cortes vão sendo adicionados.
- Heurísticas- Tal como visto, a progressão da árvore B&B depende em muito da qualidade das soluções já encontradas, caso já se tenha uma boa solução vão existir muitos ramos que vão ser abandonados em virtude de não conseguirem superar a solução já encontrada. Assim sendo, a maioria das implementações do B&B investe em sistemas de heurísticas para tentar encontrar uma boa solução inteira o mais rápido possível. Um exemplo consiste em arredondar o valor de varáveis que estejam muito perto dum valor inteiro, continuando com a resolução do modelo relaxado de forma a rapidamente encontrar uma solução inteira. Alguns dos problemas são de complexidade tal que é computacionalmente impossível encontrar a solução óptima. No entanto, ao

encontrar uma boa solução inicial, garante-se que após um determinado tempo limite o sistema vai encontrar uma solução com qualidade.

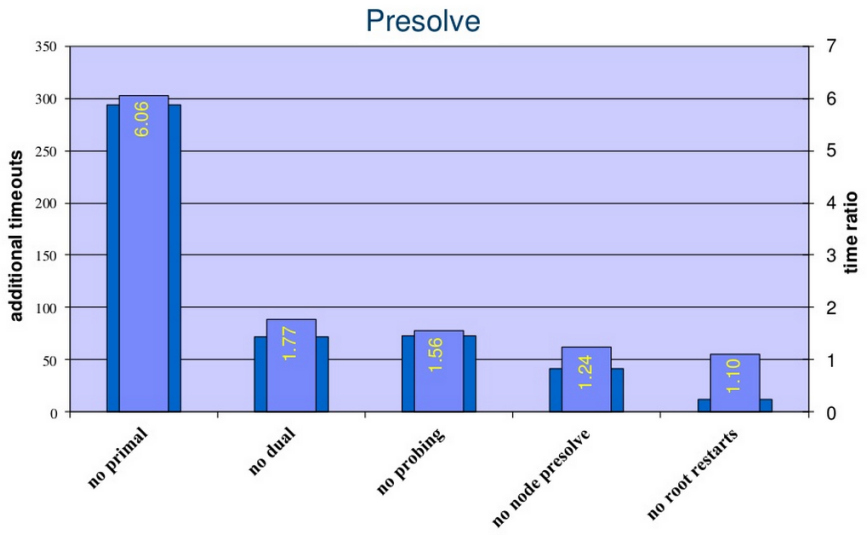
Estas três técnicas são aplicadas de forma mais intensa na raiz da árvore, e são depois aplicadas também, mas de forma menos intensa, nos restantes nós da árvore. Cada uma destas técnicas divide-se em diferentes abordagens e subtécnicas.

Existem dois produtos no mercado que se distinguem pela capacidade em termos de resolução deste tipo de problemas - Cplex da IBM e o Gurobi Optimizer. Estes produtos têm vindo a desenvolver as técnicas vistas anteriormente com o intuito de melhor resolverem modelos complexos. De forma a mostrar esta evolução é de relevância apresentar um estudo feito pela IBM que tem usado ao longo dos anos um conjunto de 1769 modelos de referência, com complexidade variada, que têm sido submetidos a uma série de testes com as diferentes versões do Cplex. No gráfico seguinte pode ver-se que nos últimos 15 anos o número de modelos em que uma solução óptima não foi encontrada diminuiu radicalmente, acompanhada duma diminuição no tempo de resolução.

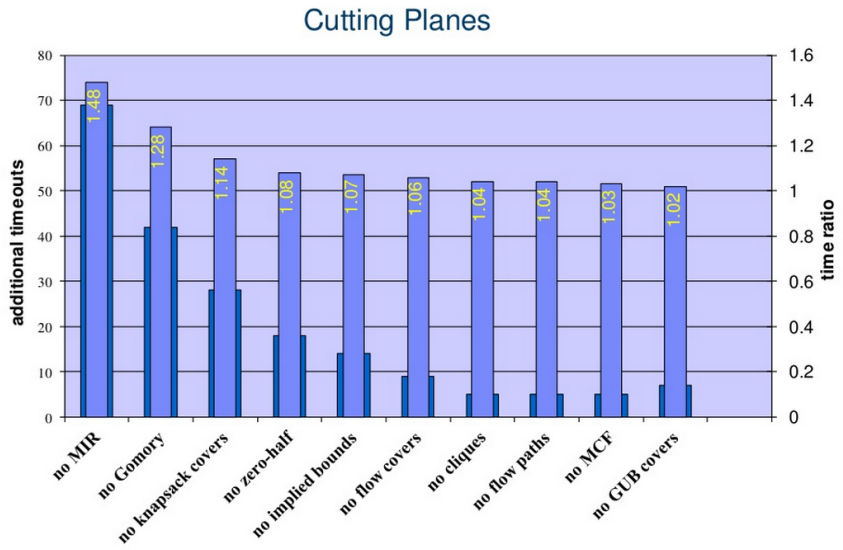


Evolução da resolução de modelos inteiros por parte do IBM Cplex

Os gráficos seguintes mostram o impacto na resolução no caso de não se utilizar as diferentes estratégias de *presolve* ou de planos de corte.



Impacto das técnicas de *presolve* na resolução de modelos inteiros



Impacto das técnicas de planos de corte na resolução de modelos inteiros

Aplicabilidade

Caso de Estudo

Regressando ao problema concreto que temos em mãos, vamos assumir a existência de três agentes- A1, A2 e A3- com as contas C1, C2, C3 respectivamente. Assuma-se um universo de 5 instrumentos financeiros- IF1, IF2, IF3, IF4, IF5. Para efeitos deste exercício assumimos que todos os créditos são feitos em Euros.

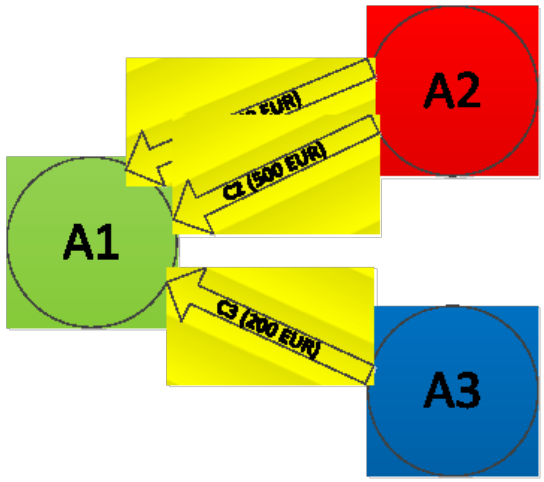
A função *Saldo(Conta, Activo Financeiro)* devolve o saldo dum determinado activo financeiro numa determinada conta.

A1- Conta C1	A2- Conta C2	A3- Conta C3
Saldo(C1, IF1) = 100	Saldo(C2, IF1) = 900	Saldo(C3, IF1) = 0
Saldo(C1, IF2) = 500	Saldo(C2, IF2) = 600	Saldo(C3, IF2) = 0
Saldo(C1, IF3) = 300	Saldo(C2, IF3) = 200	Saldo(C3, IF3) = 0
Saldo(C1, IF4) = 100	Saldo(C2, IF4) = 0	Saldo(C3, IF4) = 0
Saldo(C1, IF5) = 300	Saldo(C2, IF5) = 0	Saldo(C3, IF5) = 0
Saldo(C1, EUR) = 0	Saldo(C2, EUR) = 1000	Saldo(C3, EUR) = 5000

Característica	IF1	IF2	IF3	IF4	IF5
Tipo	Obrigação	Obrigação	Obrigação	Acção	Acção
Notação	A	B	B	B	C
Preço	5	1	10	5	3
Volatilidade	1%	10%	20%	5%	10%
Emissor	República Federal da Alemanha	Empresa E1	República Portuguesa	Empresa E2	Empresa E3
País da Entidade Emissora	Alemanha	EUA	Portugal	Alemanha	EUA
Tipo de Emissor	Governo	Empresa	Governo	Empresa	Empresa
Valor da Emissão	100 000 000 EUR	500 000 USD	1 000 000 EUR	3 000 000 EUR	100 000 EUR
Moeda de Denominação	EUR	USD	EUR	EUR	EUR

Vamos assumir que o agente A1 é tomador de fundos, A2 e A3 são ambos cedentes. Existem 3 créditos entre A1, A2 e A3:

Crédito	Cedente	Tomador	Valor	Data de Abertura	Data de Fecho	Taxa de Juro
C1	A2	A1	100 EUR	1 Jan 2014	31 Dez 2014	1%
C2	A2	A1	500 EUR	1 Jun 2014	31 Dez 2014	2%
C3	A3	A1	200 EUR	1 Jan 2010	31 Dez 2020	3%



Como tomador de fundos, A1 tem direito a definir as suas preferências, ou seja, definir os activos que preferencialmente devem ser utilizados. Como cedentes de fundos, A2 e A3 têm direito a definir um perfil de risco, ou seja, activos que não aceitam como colateral e activos que são aceites de forma condicionada. Dois agentes podem ter vários créditos entre eles, e o perfil de risco pode variar, dependendo do montante, da taxa de juro, do departamento que contraiu/concedeu o crédito, etc.

Perfil de Preferências para Tomadores (ordenadas por importância)
A1
• Usar activos com a pior notação possível • Usar acções em detrimento de obrigações • Usar obrigações de empresas em detrimento de obrigações governamentais

Perfil de Risco para Cedentes		
A2 (C1)	A2 (C2)	A3 (C3)
• Colateral dum crédito usa no máximo 10% de instrumentos financeiros como notação C. • Colateral dum crédito usa no máximo 40% de instrumentos financeiros como notação B. • Colateral dum crédito usa no máximo 20% de instrumentos financeiros emitidos por uma entidade não-governamental.	• Colateral dum crédito usa no máximo 30% de instrumentos financeiros como notação C. • Colateral dum crédito usa no máximo 5% de acções. • Colateral dum crédito usa no máximo 20% de instrumentos cuja moeda de denominação é USD. • Colateral dum crédito usa no máximo 1% de instrumentos cujo emissor é a empresa E3.	• Colateral dum crédito usa no máximo 10% de instrumentos financeiros emitidos em Portugal. • Colateral dum crédito usa no máximo 20% de instrumentos financeiros emitidos nos EUA. • Colateral dum crédito usa no máximo 10% de instrumentos financeiros com volatilidade superior a 10%. • O valor de colateral para cada instrumento financeiro não pode ultrapassar em 20% o total de valor de emissão desse mesmo instrumento.

Tipos de Optimização

Cobertura de Crédito

A cobertura dum crédito limita-se a um sistema simples, em que a função objectivo consiste em minimizar a diferença entre o valor do crédito e o valor de colateral utilizado. O perfil de risco do cedente deve ser sempre respeitado.

Vamos usar o crédito C1 entre A1 e A2. Ficamos assim com o seguinte sistema:

1− =15 × (, 1)

1, 1≤100 ; 2, 1≤500 ; 3, 1≤300; 4, 1≤100; 5, 1≤300

1, 1, 2, 1, 3, 1, 4, 1, 5, 1: á

çã : 5 × (5, 1)≤ 1 ×10%

çã : 2 × (2, 1)+ 3 × (3, 1)+ 4 × (4, 1)≤ 1 ×40%

ã – : 2 × 2, 1+ 4 × 4, 1 + 5 × (5, 1)≤ 1 ×20%

Até ao momento especificamos um sistema que permite a cobertura do crédito, respeitando o perfil de risco. O objectivo é sempre a cobertura exacta do total do crédito. No entanto, o sistema anterior tentará utilizar tanto colateral quanto possível. Temos assim que alterar a função objectivo de forma a indicar que deveremos aproximar o valor do colateral tanto quanto possível do valor do crédito. Para tal fazemos uso duma variável designada de DC (distância de colateralização):

(=15 × , 1)– 1≤

–(=15 × (, 1))+ 1≤

Desta forma fazemos com que o sistema tente minimizar o valor absoluto da diferença entre o valor do crédito e o valor do colateral a utilizar.

Um ponto muito importante a ter em conta diz respeito à forma como os diferentes limites do perfil de risco são postos em prática. Considerando o enunciado, a definição dum perfil introduz uma condição em que o limite depende da solução em si. Em outras palavras, um limite de 10% no valor do colateral indica que a medida real desses 10% é um valor dinâmico que depende da solução. Esta situação poder-nos-ia levar a um modelo quadrático, que tem uma resolução muito mais complicada. Por conseguinte assume-se que o valor do limite se aplica ao valor do crédito e não ao valor do colateral alocado em cada momento. Desta forma conservamos a linearidade do modelo.

Falta-nos agora considerar as preferências do tomador de fundos no que toca à utilização de colateral. Começamos por ordenar os activos financeiros disponíveis, considerando as preferências do tomador. Considerando os critérios definidos anteriormente, IF5 tem a

preferência porque tem a pior notação. Depois vem IF4 porque dentro dos activos com notação B é a única acção. Depois vem IF2 seguido de IF3, ambos têm a mesma notação e ambos são obrigações, mas IF2 é uma obrigação duma empresa. Finalmente vem IF1, com notação A. Definimos a função PrefUtilização(IFi) que dá a preferência de utilização dum activo financeiro no contexto dum portefólio disponível para a utilização como colateral:

PrefUtilização(IF1) = 1; PrefUtilização(IF3) = 2; PrefUtilização(IF2) = 3;

PrefUtilização(IF4) = 4; PrefUtilização(IF5) = 5;

Precisamos agora de adicionar as preferências à função objectivo. É necessário estabelecer que a colateralização do crédito é a prioridade máxima, e o respeito do perfil do tomador não é mais do que uma preferência que apenas deve ser posta em prática no caso de existirem diferentes configurações de activos que permitam a colateralização do crédito e o respeito pelo perfil de risco do cedente. Para tal usa-se uma constante de peso de diferenciação, designada de PDif. A definição deste valor depende muito da amplitude de valores que se utiliza, mas um valor à volta de 100 000 é um bom ponto de começo.

× +

(=15 × (, 1))− 1≤

−(=15 × (, 1))+ 1≤

= =15 çã ( ) × (, 1)

As 5 variáveis que representam as quantidades usadas são definidas como variáveis inteiras positivas.

Ficamos assim com uma função objectivo que tenta maximizar a cobertura do crédito, respeitando tanto quanto possível as preferências do tomador. O modelo resultante é puramente linear e por conseguinte adapta-se facilmente a volumes elevados de variáveis e de condições.

O modelo final para este exemplo concreto é o apresentado seguidamente.

çã : × +

ç çã 1: (=15 × (, 1)− 1≤

ç çã 2: −(=15 × (, 1))+ 1≤

ê : = =15 çã × × (, 1)

í : (1, 1) ≤100 ; (2, 1) ≤500 ; (3, 1) ≤300; (4, 1) ≤100; (5, 1) ≤300

çõ : 5 × (5, 1)≤ 1 ×10%

çõ : 2 × 2, 1+ 3 × 3, 1+ 4 × (4, 1)≤ 1 ×40%

ã − : 2 × 2, 1+ 4 × 4, 1+ 5 × (5, 1)≤ 1 ×20%

Podemos aplicar o mesmo princípio para C2 e C3.

çã : × +

(=15 × (, 2)− 2≤

−(=15 × (, 2))+ 2≤

= =15 çã × × (, 2)

(1, 2) ≤100 ; (2, 2) ≤500 ; (3, 2) ≤300; (4, 2) ≤100; (5, 2) ≤300

5 × (5, 2)≤ 2 ×30%

4 × 4, 2+ 5 × (5, 2)≤ 2 ×5%

2 × 2, 2≤ 2 ×20%

3 × 3, 2≤ 2 ×1%

çã : × +

(=15 × (, 3)− 3≤

−(=15 × (, 3))+ 3≤

= =15 çã × × (, 3)

(1, 3) ≤100 ; (2, 3) ≤500 ; (3, 3) ≤300; (4, 3) ≤100; (5, 3) ≤300

3 × (3, 3)≤ 3 ×10%

2 × 2, 3+ 5 × (5, 3)≤ 3 ×20%

3 × 3, 3≤ 3 ×10%

1 × 1, 3≤ 100 000 ×20%

2 × 2, 3≤ 500 000 ×20%

3 × 3, 3≤ 1 000 000 ×20%

4 × 4, 3≤ 3 000 000 ×20%

5 × 5, 3≤ 100 000 ×20%

Podemos agora definir o nosso modelo de forma genérica para um crédito Cx:

çã : × +

ç çã 1: (=1 × (,))− ≤

ç çã 2: −(=1 × (,))+ ≤

ê : = =1 çã × × (,)

í : ∀ (,) ≤ (,)

: ∀ =1 × (,) × (,) <

( )

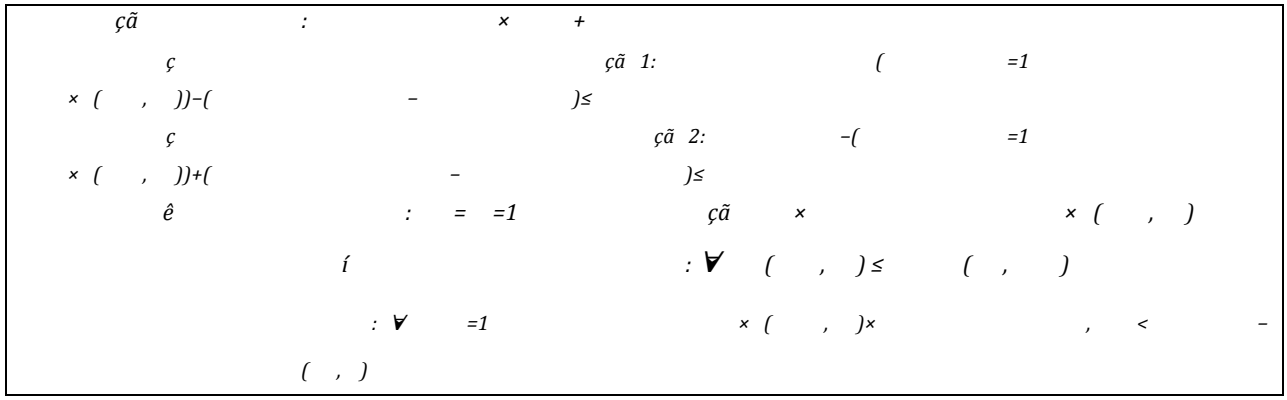
A função *Contribui*(IFi, l) indica se o instrumento financeiro IFi contribui para o limite l definido , retornando 1 em caso afirmativo e 0 em caso negativo.

Evolução da Cobertura de Crédito

A cobertura inicial do crédito é o primeiro passo da gestão de colateral. Depois de colateralizado, o crédito tem que ser acompanhado de maneira a validar que o valor de colateralização é suficiente. A este nível existe uma volatilidade elevada de preços. Assim, cabe

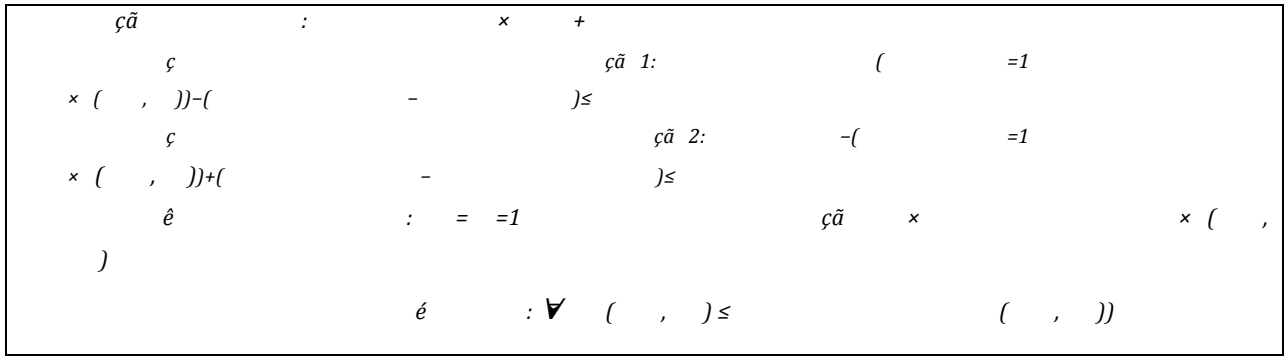
a um sistema de gestão de colateral avaliar regularmente o valor de crédito e o valor dos activos usados como colateral, e reagir de forma apropriada.

No caso do valor do colateral diminuir (descida dos preços dos activos financeiros) o sistema deve reagir e subir o nível de colateralização. A tarefa pode ser facilmente atingida com as seguintes equações:



O modelo é muito semelhante ao que vimos anteriormente, com uma pequena alteração causada pelo facto de já existir colateral alocado. A função *ValorColateral(Ci)* dá o valor do colateral utilizado no crédito Ci, e a função *ValorColateralLimite(Ci, Lj)* dá o valor do colateral no crédito Ci que contribui para limite j do perfil de risco do cedente. At é a conta do tomador. Podemos substituir o modelo de novos créditos com este modelo, sabendo que quer *ValorColateral* quer *ValorColateralLimite* vão devolver 0.

No caso do valor alocado ser superior ao valor do crédito, o sistema de equações é ligeiramente diferente, e mais simples.



Neste caso *Q(IFi, Cx)* representa a quantidade de IFi a remover do colateral no crédito Cx. *PrefInversaUtilização* é a função que funciona de forma inversa a *PrefUtilização*, ou seja, ordena

os activos disponíveis para ser removidos pela ordem inversa da preferência definida pelo tomador. Desta forma faz-se com que os activos removidos sejam aqueles que o tomador menos quer utilizar. A função *QuantidadeAlocada* é utilizada para garantir que não se remove mais do que aquilo que existe. As equações relativas ao perfil de risco são desnecessárias porque a diminuição do valor de colateral não tem impacto nos limites.

Maximização da Liquidez

Até ao momento utilizamos a Programação Linear para optimizar a alocação de activos financeiros no contexto dum crédito. Apesar de este ser o objectivo principal ao nível dum crédito, um sistema de gestão de colateral deve tentar maximizar a liquidez e a utilização de activos em diferentes créditos. Além disso, os modelos vistos acima não podem ser executados em paralelo visto todos eles fazerem uso dos mesmos activos nas mesmas contas. De forma a melhorar o resultado quer em termos de optimização, quer em termos de performance, agregam-se os créditos por (tipo de optimização/cedente) e executa-se um modelo para cada.
No caso de novos créditos ou do reforço do colateral em créditos existentes ficaríamos com o seguinte para um cedente com n créditos e com m instrumentos:

çã : =1 × +
∀ (=1 × (,))−(−)≤
∀ −(=1 × (,))+(−)≤
∀ = =1 çã × × (,)
∀ =1 (,) ≤ (,)
∀ ∀ =1 × (,)× , < − (,)

Usando o exemplo do caso de estudo, ficaríamos com o modelo apresentado seguidamente. Assume-se que os diferentes perfis de risco são designados de l1 a l10.

çã : 1 × + 1+ 2 × + 2+ 3 × + 3

(=15 × , 1− (1− 1)≤ 1
−(=15 × (, 1))+ (1− 1≤ 1
(=15 × , 2− (2− 1)≤ 2
−(=15 × (, 2))+ (2− 2≤ 2
(=15 × , 3− (3− 3)≤ 3

−(=15 × (, 3))+ (3− 3≤ 3

1= =15 çã × × (, 1)
2= =15 çã × × (, 2)
3= =15 çã × × (, 3)

1, 1+ 1, 2+ (1, 3) ≤100
2, 1+ 2, 2+ (2, 3) ≤500
3, 1+ 3, 2+ (3, 3) ≤300
4, 1+ 4, 2+ (4, 3) ≤1000
5, 1+ 5, 2+ (5, 3) ≤300

5 × 5, 1≤ 1 ×10% − (1, 1)

2 × 2, 1+ 3 × 3, 1 4 × 4, 1≤ 1 ×40%− (1, 2)

2 × 2, 1+ 4 × 4, 1+ 5 × 5, 1≤ 1 ×20%− (1, 3)

5 × 5, 2≤ 2 ×30%− (2, 4)
4 × 4, 2+ 5 × 5, 2≤ 2 ×5%− (2, 5)

2 × 2, 2≤ 2 ×20%− (2, 6)

3 × 3, 2≤ 2 ×1%− (2, 7)

3 × 3, 3≤ 3 ×10%− (3, 8)

2 × 2, 3+ 5 × 5, 3≤ 3 ×20%− (3, 9)

3 × 3, 3≤ 3 ×10%− (3, 10)

1 × 1, 3≤ 100 000 ×20%
2 × 2, 3≤ 500 000 ×20%
3 × 3, 3≤ 1 000 000 ×20%
4 × 4, 3≤ 3 000 000 ×20%
5 × 5, 3≤ 100 000 ×20%

Flexibilidade

Os mercados estão em variação constante. Cada instrumento financeiro tem um número elevado de características, que podem ganhar relevância a qualquer momento. Por conseguinte, é muito importante ter um sistema que se possa adaptar rapidamente às novas necessidades. Esta é uma das grandes limitações de muitos dos sistemas existentes- baseiam-se num conjunto de regras estáticas que consideram um número limitado de atributos de instrumentos financeiros. Sempre que existe uma alteração no mercado, os algoritmos têm que ser adaptados. Esta reacção é por normal lenta, expondo os créditos a risco mais elevado, consequentemente aumentando as taxas cobradas e reduzindo a liquidez.

A Programação Linear, e os modelos apresentados anteriormente, adaptam-se de forma dinâmica a novos critérios, logo que o sistema respeite as seguintes condições:

- Processamento de novos atributos de forma dinâmica. As entidades que fornecem dados de referência expõem interfaces que permitem a adição dinâmica de novos atributos, sendo necessário que o sistema se adapte as esses interfaces e possa processar novos atributos que vão sendo adicionados.
- Permitir que os novos atributos sejam utilizados na definição das condições do modelo linear.

A flexibilidade da Programação Linear permite a adaptação em tempo-real às evoluções dos mercados, e às necessidades tanto de tomadores como de cedentes. Desta forma consegue-se minimizar o risco e aumentar a liquidez.

Os modelos vistos anteriormente são simples. O sistema pode agora ser estendido com outros modelos matemáticos que implementem funcionalidades tais como a reorganização do colateral alocado de forma a melhor atender às preferências dos tomadores ou com o objectivo de corrigir perfis de risco que possam passar a ser desrespeitados devido a alterações nos dados de referência (por exemplo um activo que tem uma mudança na correspondente notação).

Implementação

Tal como indicado, a resolução dum modelo de Programação Inteira é um problema de complexidade elevada. A dimensão dos modelos associados a alguns dos cenários de gestão de risco dificulta ainda mais a tarefa. Em casos extremos chegamos a ter modelos com perto dum milhão de variáveis e um milhão de condições. A implementação da criação, execução, e análise de resultados tem que ser feita de maneira muito cuidadosa de forma a garantir a obtenção de resultados em tempo útil. Esta tarefa depende de três pontos fundamentais:

- Preparação dos modelos- a definição das condições é determinante para a qualidade do modelo. Apesar da fase de *presolve* fazer a limpeza de muitas incoerências, as especificidades das equações matemáticas e a forma como se adaptam ao problema são uma peça fundamental no que toca à performance do modelo.
- Execução dos modelos- o algoritmo escolhido para a resolução do modelo e todos os parâmetros associados. A performance com que um modelo linear é executado depende muito da forma como a implementação da biblioteca escolhida é posta em prática.
- Análise dos resultados- nem todos os modelos convergem para uma solução. A falta de convergência ocorre na maioria dos casos com modelos de dimensão elevada, mas esporadicamente existem modelos de tamanho muito reduzido que não chegam a convergir. É preciso criar um sistema que seja robusto e se adapte dinamicamente a estas situações.

Cada uma destas situações será analisada em detalhe nas páginas que se seguem.

Preparação dos Modelos

A delineação das equações a utilizar é provavelmente o passo mais importante no que toca à implementação. Em primeiro deve-se definir uma função objectivo que traduza correctamente o propósito do modelo. Em segundo, as equações utilizadas como condições devem também reflectir as características do problema de forma sucinta e restrita. Apesar do processo de *presolve* proceder à limpeza de muitas das impuridades do modelo, é determinante que as

equações sejam tão optimizadas quanto possível. Deve-se verificar que a linearidade do modelo é respeitada.

Um ponto muito importante relaciona-se com o facto de algumas das bibliotecas nesta área oferecerem funcionalidades extra que vão além da linearidade pura. Estas funcionalidades permitem simplificar a criação de modelos, mas recorrem a técnicas complexas que têm um impacto tremendo na performance da resolução dos modelos. Através da experiência pessoal posso testemunhar que a grande maioria destas funcionalidades extra trazem mais custos do que benefícios, e devem ser evitadas. Por conseguinte, para que se consiga uma resolução tão rápida quanto possível, deve-se restringir o modelo à formulação tradicionalmente linear.

Deve-se também introduzir uma fase de limpeza de dados em termos funcionais. Para tal é necessário fazer um julgamento baseado na natureza do tipo de valores relacionados com o domínio específico de negócio. Este é um ponto muitas vezes negligenciado, mas de grande importância. Um bom exemplo diz respeito à natureza das variáveis e ao facto delas apenas poderem ter valores inteiros, e consequentemente tem que se aplicar restrições no que toca aos respectivos ranges de valores possíveis. O tempo de resolução de modelos lineares sofre uma degradação significativa quando se utiliza variáveis inteiras. Por conseguinte, sempre que possível, deve-se tentar evitar o recurso a variáveis inteiras. Adicionalmente, o tempo de resolução dum modelo linear aumenta sempre que as amplitudes das diferentes variáveis são muito diferentes. Por exemplo, se existirem duas variáveis, x1 e x2, em que $0 \leq x1 \leq 10$ e $1\,000\,000 \leq x2 \leq 1\,000\,000\,000$, a resolução do modelo será lenta, a grande maioria das implementações de algoritmos de resolução de modelos lineares lida mal com amplitudes de variáveis muito diferentes. Muitos destes problemas podem ser optimizados na fase da formulação através do uso de regras simples. Por exemplo, no caso da gestão de colateral, se tivermos um crédito dum determinado montante, se para alguns instrumentos financeiros tivermos saldos elevados, e para outros tivermos saldos reduzidos cuja contribuição é pouco relevante para o valor total do crédito, podemos facilmente ignorar os instrumentos financeiros com saldos reduzidos, diminuindo-se o tamanho do modelo, facilitando a resolução (a remoção da existência de variáveis com amplitudes de valores muito diferentes faz com a resolução seja muito mais estável), sem que se sacrifique de forma relevante a qualidade da solução.

Como será visto mais à frente, existe uma série de critérios que determinam a forma como a resolução dos modelos é feita. Não existe uma solução ideal para todos os modelos, os critérios aplicados dependem em muito do modelo em mãos. De forma a se poder seleccionar esses mesmos critérios em tempo de execução, a fase de preparação de modelos deve criar uma camada de metadados com um sumário do modelo. Informações como número de varáveis, número de condições, amplitude de variáveis, são exemplos de conteúdo relevante. Estes metadados são depois utilizados para tomar decisões ao nível da resolução com o intuito de tornarem o processo mais eficiente.

Resolução dos Modelos

Até ao momento não foram abordadas tecnologias em específico, os princípios enumerados são suficientemente genéricos para poderem ser aplicados em plataformas diferentes. Neste caso em concreto utiliza-se a linguagem Java mas poder-se-ia utilizar outra linguagem qualquer.
A resolução de modelos em específico entra num domínio mais técnico. Em primeiro lugar é necessário decidir como implementar essa mesma resolução. O espectro de possibilidades vai desde a implementação dum algoritmo, à utilização de bibliotecas *open-source*, ou à utilização duma biblioteca comercial. Existem no mercado algumas bibliotecas *open-source* de qualidade que se perfilam como a melhor opção para problemas de pequena/média dimensão. Para problemas de grande dimensão, tal como a gestão de colateral, existe ainda um fosso significativo entre *open-source* e bibliotecas comerciais. O IBM Cplex e o Gurobi Optimizer são os dois produtos que se destacam. Ambos são implementados em C++, mas ambos dispõem de interfaces Java que acedem à implementação via JNI (*Java Native Interface*).
A JNI permite que uma aplicação Java possa utilizar código nativo. Podemos assim usufruir do melhor dos dois mundos- por um lado temos um sistema em Java que beneficia de todas as vantagens duma tecnologia madura que fornece um conjunto de funcionalidades que se adaptam perfeitamente às necessidades que temos, por outro podemos optimizar a resolução dos modelos através da utilização de código nativo. No entanto, ao utilizar a JNI abre-se o sistema a vulnerabilidades que podem advir de problemas na execução do código nativo. Por exemplo, no caso de existir um problema no código nativo, existe o risco da JVM (*Java Virtual*

Machine) ser terminada, algo que não poderá acontecer no caso de se executar uma aplicação Java pura. Assim sendo, a opção de utilizar JNI tem que ser tomada com consciência do risco adicional que se assume, e por conseguinte deve ser complementada com mecanismos eficientes de detecção de erros, para que os problemas sejam rapidamente identificados, e com mecanismos de resiliência, tais como *clustering*, para que a ocorrência dum problema não ponha em causa o serviço.

Mais atrás vimos algumas das técnicas que são utilizadas para melhorar o tempo de resolução de modelos. Existe uma técnica adicional que é de vertente puramente técnica, e que diz respeito à utilização de paralelismo. O paralelismo pode melhorar a execução do B&B mas deve ser executado com cautela. À medida que a árvore se vai expandindo existem benefícios em recorrer a paralelismo, os diferentes ramos podem ser executados em paralelo. Mas quando se dá enfâse à procura duma solução inicial, existe bastante trabalho feito ao nível da raiz da árvore. Nesta fase o paralelismo não trás valor acrescentado, e pode até atrasar o processo devido ao *overhead*. No caso de se poder beneficiar com paralelismo, é necessário controlar a implementação e limitar o número de *threads* em paralelo para que a execução dum modelo não consuma demasiados recursos e tenha impacto negativo noutras execuções que estejam a decorrer no mesmo servidor. Algumas arquitecturas optam por ter servidores dedicados à resolução de modelos de forma a optimizar o *hardware* para esse efeito.

Exemplo de pontos que podem ser resolvidos em paralelo

A utilização de paralelismo levanta uma questão que afecta o determinismo da solução encontrada. Quando se usam múltiplas *threads* em paralelo para executar um modelo, e no caso de existirem várias soluções com a mesma qualidade, o resultado de resolver várias vezes o mesmo modelo pode acabar em resultados diferentes (com a mesma qualidade). Isto é um factor importante a ter em conta nos testes, principalmente quando se tem testes automáticos

que validam números elevados de modelos, e que podem ter resultados variáveis. Assim sendo, este comportamento deve ser acautelado- ou se criam modelos que têm soluções únicas, ou se criam mecanismos de teste que avaliam a qualidade das soluções em vez de as compararem com um resultado específico esperado. Algumas bibliotecas de Programação Inteira permitem usar paralelismo, garantindo o determinismo do resultado, tendo a vantagem de garantir que a execução repetida do mesmo modelo gera a mesma solução, mas perdendo parte do valor do paralelismo na medida em que esse mesmo paralelismo é controlado e sacrificado em determinados pontos-chave do algoritmo.

Existe uma série de critérios que devem ser indicados aquando da resolução do modelo, que são determinantes na forma como a resolução de adapta às especificidades do modelo em causa:

- Tempo limite de resolução. Alguns modelos podem demorar muito tempo a convergir para uma solução. É importante definir um tempo limite para a resolução, caso contrário a execução pode continuar por períodos indefinidos. Este tempo limite deve estar relacionado com a complexidade do modelo. Para modelos simples deve-se estabelecer tempos limite de poucos segundos. Para modelos complexos não se dever ir além de poucos minutos. A experiência do passado permitiu-nos construir um conjunto de centenas de modelos e testá-los sob diferentes condições de execução. No que toca aos modelos complexos, a grande maioria converge antes do final do terceiro minuto, e daqueles que não convergem, são muito poucos aqueles que convergem com mais um punhado de minutos. Em outras palavras, no caso dum modelo complexo não convergir em 3 minutos, é muito improvável que o faça num intervalo de tempo superior, a maior parte desses modelos demorará horas (ou até dias) a convergir. Consequentemente, não existe benefício relevante em ter tempo limites superiores a 2/3 minutos.
- Quando se começa a resolver um modelo, existe uma decisão importante a tomar e que diz respeito à enfâse da resolução. No caso de o modelo ser de complexidade reduzida deve-se dar prioridade à procura da solução óptima, isto é, temos algum grau de confiança que o modelo pode ser resolvido em tempo útil e por conseguinte focamos a resolução na obtenção da melhor solução. No caso de o modelo ser complexo, a ênfase

deve ser canalizada na obtenção duma solução, mesmo que não óptima- a resolução deve investir energia na aplicação de heurísticas que facilitem a obtenção duma solução inicial com alguma qualidade. Só então se investe na obtenção da solução óptima. Em resumo, caso seja previsível que a solução óptima não seja obtida antes do tempo limite, dever-se-á direccionar a execução para a obtenção duma solução. Caso seja provável que a obtenção duma solução óptima seja possível, então a energia deve ser imediatamente investida na obtenção dessa solução óptima.

- Na maioria das vezes a obtenção duma solução óptima é uma tarefa exigente que pode demorar muito tempo. No entanto é possível fornecer a nossa própria definição de solução óptima, indicando o quão longe pode estar a solução inteira da solução óptima não-inteira de forma a ser considerada óptima. Este é um parâmetro muito importante para modelos complexos, em que a solução óptima é difícil de obter mas uma solução de qualidade pode ser obtida facilmente. Assim sendo é recomendável definir sempre este critério (muitas vezes designado de *mip gap tolerance*) para modelos de média/elevada complexidade de forma a poupar tempo na resolução. A distância depende muito do modelo em mãos, mas uma solução que diste 5% a 10% da solução óptima é por norma uma solução com qualidade que pode ser utilizada.
- É possível indicar o quão longe o valor duma variável pode estar dum valor inteiro para ser considerado inteiro. Este critério costuma ser designado de tolerância de integralidade. Na maior parte dos casos pode-se definir alguma margem de manobra (por exemplo 1e-5) sem se perder muito. Através deste parâmetro pode-se evitar novos subproblemas desnecessários e o desperdício de tempo. No entanto é preciso muito cuidado, no caso dos resultados destas variáveis serem associados a coeficientes de magnitude elevada, a violação da integralidade pode atingir proporções elevadas, criando soluções que na prática são inválidas.
- Número máximo de *threads* que pode ser utilizado na resolução dum modelo. Este parâmetro depende do ambiente específico de execução. Caso se opte por um valor maior do que 1, tem também que se indicar se o determinismo das soluções deve ser garantido.

- Algoritmo de Programação Linear utilizado para a resolução dos modelos relaxados. A maioria das bibliotecas suporta *Primal Simplex*, *Dual Simplex*, *Network Simplex*, *Barrier* e *Sifting*. Cada um destes mecanismos de optimização tem vantagens e desvantagens. A escolha automática por parte da biblioteca é na grande maioria das vezes a decisão mais acertada.

Nos critérios acima mencionados é recorrentemente referida a complexidade dos modelos. O critério mais óbvio para definir a complexidade dum modelo é o tamanho (número de variáveis e de condições). Mas existem outros elementos determinantes. O tipo de função objectivo é também importante. Uma função objectivo com um número elevado de critérios que oscilam em direcções opostas introduz uma certa instabilidade na convergência e aumenta a complexidade. Tal como visto, a existência de variáveis com grandes diferenças em termos de amplitudes de valores é também um factor de instabilidade. Considerando todas as fonte possíveis de complexidade, um passo fundamental a implementar consiste em utilizar os metadados construídos na preparação do modelo e produzir um perfil de complexidade do modelo. Este perfil pode ir duma escala básica (simples, normal, complexo) até uma escala muito mais elaborada. Para domínios de complexidade elevada recomenda-se uma escala estruturada que traduza as diferentes dimensões de complexidade. Para cada perfil existe depois uma configuração em termos de resolução de modelos. É esta a abordagem que temos na gestão de colateral.

Em termos de projecto, o desenvolvimento do componente do sistema que trata da resolução de modelos complexos de gestão de colateral exige um processo de análise dos requisitos funcionais e correspondente formulação dos modelos. Este trabalho é feito de forma iterativa durante as fases de análise e desenvolvimento, e consiste em vários passos:

- Definição das equações matemáticas que servem de base aos modelos.
- Definição de potenciais simplificações que possam ser adoptadas ao nível da execução e cujo impacto é aceitável dum ponto de vista do negócio. Por exemplo, como foi visto anteriormente, saldos pequenos podem ser ignorados quando na presença de créditos elevados e de saldos elevados de outros instrumentos. Os impactos destas optimizações ao nível do negócio têm que ser aceites por todos.

- Definição dos graus de complexidade dum modelo.
- Mapeamento dos graus de complexidade em configurações de execução. Um grau de complexidade pode ser mapeado em mais do que uma configuração, ordenadas por graus de preferência. O sistema tenta gradualmente as diferentes configurações.

O desenvolvimento consiste em por em prática todas estas opções. Combinando esta abordagem com uma biblioteca eficiente consegue-se uma solução que resolve o problema em mãos. No caso concreto da gestão de colateral, temos picos de dez mil modelos por dia, com graus de complexidade muito variada. Destes modelos, 98% são resolvidos à primeira tentativa. Em termos de plataforma técnica temos um componente Java a executar em Linux, que faz uso da libraria Cplex, cuja funcionalidade é implementada em código nativo e acessível via JNI.

Análise de Resultados

O último passo consiste em analisar os resultados da execução. Caso seja obtido um resultado, não existe muito a fazer além de o pôr em prática. No caso de não existir solução o processo fica também concluído, é indicativo de que o tomador de fundos não tem activos que possam ser usados como colateral e que satisfaçam o perfil de risco do cedente. Neste caso o sistema deve automaticamente indicar uma condição de alarme e notificar as entidades relevantes.

No contexto dos modelos matemáticos, o resultado mais interessante é quando a resolução do modelo termina devido ao tempo limite de resolução ter sido atingido. Esta é a situação mais preocupante na medida em que potencialmente existe uma solução mas o sistema não a consegue atingir. Foi visto que tal acontece numa pequena minoria dos casos, no entanto é necessário criar mecanismos de reacção sob pena de criar uma situação de exposição de risco.

Considerando que certos problemas não têm resolução em tempo útil, a melhor abordagem é sacrificar a solução óptima em detrimento duma solução de qualidade inferior que possa manter a colateralização. Assim sendo, o sistema deve reagir automaticamente reduzindo a complexidade dos modelos. Existem duas estratégias que o conseguem fazer de forma eficiente:

- Processar os créditos um a um.

- Reduzir o número de instrumentos financeiros considerados. Procede-se a iterações em que o número de instrumentos financeiros vai sendo sucessivamente reduzido. Isto permite diminuir o número de variáveis e condições, simplificando o modelo. Reduzir em 50% o número de instrumentos (por cada iteração) permite a resolução rápida do problema.

A solução obtida não é óptima, mas quando a decisão é entre não ter uma solução ou ter uma solução subóptima, a escolha é evidente. O objectivo principal é que os créditos estejam cobertos em termos de colateralização, e que os perfis de risco sejam respeitados. Este processo é depois complementado com processos de correcção que reorganizam o colateral de forma a respeitar, tanto quanto possível, as preferências do tomador.

No nosso caso concreto, e considerando que cada resolução tem um máximo de 2 minutos, e que 99.99% dos modelos fica resolvido após a sexta tentativa, na grande maioria dos casos o sistema reage num máximo de 12 minutos.

De forma a optimizar ao máximo a utilização dos recursos dum tomador de fundos, o sistema deve criar um modelo que considere todos os créditos por ele contraídos. Só desta forma se consegue maximizar a utilização de colateral, conseguindo cobrir tantos créditos quanto possíveis. Como consequência desta abordagem temos que todos os créditos são abordados no mesmo processo, havendo o risco de falta de escalabilidade. Em casos específicos pode-se optar por processar os diferentes modelos de forma independente, em paralelo, fazendo com que o sistema se adapte melhor a tomadores com números muito elevados de créditos. Novamente, esta é uma decisão que resulta da preparação de dados e que é tomada antes da resolução do modelo, dependendo do correspondente perfil de complexidade.

Conclusão

O sistema financeiro tem passado por diferentes crises de crédito. A falta de confiança entre instituições bancárias aumentou rapidamente, e o financiamento interbancário atingiu níveis muito baixos, colocando em cheque toda a forma como as economias modernas estão estruturadas. De forma a reagir, foram criados mecanismos para proporcionar garantias a quem concede crédito, aumentando a liquidez no mercado.

Uma das ferramentas mais utilizada é a gestão de colateral. Através da colateralização de créditos, os cedentes de fundos passam a usufruir de garantias que os protegem contra o incumprimento dos tomadores. Nos últimos tempos tornou-se uma parte imprescindível da actividade de crédito, os volumes de crédito garantidos com colateral subiu exponencialmente.

Este crescimento trouxe novos desafios para os sistemas de gestão de colateral, que tradicionalmente se baseavam em algoritmos rudimentares que lidavam com volumes de crédito limitados. Tornou-se urgente introduzir uma nova geração de algoritmos que atingisse três objectivos: 1) Acorrer ao modelo de negócio existente e satisfazer as necessidades existentes; 2) Execução rápida; 3) Ser flexível e acompanhar a evolução do mercado, adaptando-se dinamicamente aos novos requisitos das entidades financeiras.

A Programação Linear é o candidato perfeito para uma nova geração de algoritmos de gestão de colateral. A gestão de colateral é um problema típico de alocação de recursos, com objectivos e condições lineares. A natureza da gestão de colateral obriga a que se usem valores inteiros, e consequentemente envereda-se pela Programação Inteira.

Contrariamente à Programação Linear, em que os algoritmos existentes permitem a resolução rápida dum modelo complexo, a Programação Inteira é menos eficiente e, apesar das melhorias significativas dos últimos anos, continua a exigir um cuidado especial. A dimensão dos modelos de gestão de colateral pode chegar a valores muito elevados. Assim sendo, de forma a se poder utilizar a Programação Inteira neste domínio é necessário desenhar o sistema de maneira a que se possa adequar a abordagem de resolução à complexidade do modelo em causa. Para tal é necessário considerar cuidadosamente quatro áreas:

- O domínio de negócio servido pelos modelos e todas as especificidades que se relacionam de forma a maximizar a simplificação das condições.
- A teoria por trás da Programação Linear e Programação Inteira, de forma a conhecer os algoritmos de resolução e a forma como se adaptam aos diferentes tipos de modelo. Só assim se podem criar modelos optimizados que tiram partido dos pontos fortes dos algoritmos existentes.
- As bibliotecas existentes e todas as funcionalidades que lhe são específicas. Cada biblioteca tem um número de técnicas que podem ser adoptadas para melhorar a

resolução dos modelos. O domínio destas técnicas é imprescindível para tirar partido do melhor que a biblioteca tem a dar.

- A plataforma e as tecnologias que suportam o sistema. O conhecimento detalhado do hardware, sistema operativo, plataforma de desenvolvimento e execução, é vital para tirar o melhor partido das tecnologias em mão.

Recorrendo a estes quatro pontos pode-se criar um sistema que seja bem-sucedido na aplicação da Programação Inteira à resolução de modelos complexos, que consegue resolver a grande maioria dos problemas, e que reage dinamicamente aos problemas não convergentes, simplificando-os automaticamente de forma a obter uma solução.

Muitas das condições que são parte dos modelos, servem requisitos que evoluem constantemente. Assim sendo, desenhar um sistema que seja eficiente na resolução de modelos não chega, é preciso criar um sistema que permita que a criação desses mesmos modelos seja feita de forma dinâmica, que se vá adaptando à progressão das necessidades. Para tal é necessário conceber um conjunto de configurações flexíveis que estão preparadas para acomodar novos critérios, mas que continuem a criar modelos de forma estável sem pôr em causa a resolução dos mesmos. Só assim se consegue um sistema adaptável, mas cuja eficiência é previsível.

Foi este o desafio com que me deparei no sistema de gestão de colateral. Trabalhei neste sentido durantes os 6 anos passados, pondo em prática a abordagem que foi apresentada neste capítulo. Temos um sistema que gere perfis de risco complexos e que permite a definição dinâmica desses perfis de risco. Um perfil de risco baseia-se numa série de critérios complexos e estes critérios dependem duma série de atributos (características do título financeiro, do mercado, do país ou contexto político, etc.). A criação e utilização de novos atributos é feita de forma dinâmica, assim como o mapeamento desses mesmos atributos em modelos matemáticos que reflectem correctamente os critérios de risco que os usam. Só assim se consegue ter uma solução que se adapta à evolução diária e à incansável volatilidade dos mercados financeiros. Só assim se consegue uma solução que garanta uma reacção eficiente, apaziguando os intervenientes no mercado de crédito, e aumentando a liquidez dos mercados

financeiros, tarefa de importância extrema e de que tanto depende o bem-estar da nossa economia.

Temos o sistema que por muitos é considerado o sistema com o nível de optimização mais avançado do mercado, e que tem vindo a gerir centenas de milhares de milhões de euros, que diariamente produz resultados consistentes e rápidos, e que foi provado em alguns momentos de crises financeiras sem que os resultados fossem de alguma forma afectados. Ao fazê-lo reestabelecemos alguma confiança no mercado, e os reguladores têm vindo cada vez mais a adoptar este tipo de abordagem. O facto de fazermos o *outsourcing* deste serviço para uma série de mercados domésticos em países com economias tão significativas como Austrália, Brasil, Espanha ou África do Sul, mostra que esta abordagem e os resultados obtidos têm tido um papel preponderante no crescimento deste tipo de filosofia ao crédito.

Experiência Profissional e o Mestrado de Engenharia Informática

"*O objectivo do mestrado é preparar Engenheiros Informáticos com uma formação simultaneamente amadurecida e vasta, capazes de contribuírem de forma crítica, criativa e informada em todas as fases do processo de análise, projecto, concepção e desenvolvimento de sistemas informáticos.*". "*O Mestrado de Engenharia Informática visa proporcionar suporte para uma formação integral em Engenharia, integrando um sólido conhecimento científico, uma criteriosa formação metodológica, um conhecimento amplo e crítico das tecnologias emergentes e uma consciência deontológica e cívica apurada.*"

Durante o período profissional que percorri, passei por uma série de vivências e experiências que desenvolveram as competências que adquiri na licenciatura. Desempenhando sempre o papel de Engenheiro Informático, tive a oportunidade de participar em dezenas de projectos de diferentes dimensões, que contribuíram para a criação e manutenção de sistemas informáticos de complexidade elevada. Passei por uma heterogeneidade de contextos profissionais, com diferentes metodologias de desenvolvimento, variadas áreas de actividade (Telecomunicações, Retalho, Banca, etc.) e diferentes localizações geográficas (Portugal, Inglaterra, Países Baixos e Luxemburgo). Em todos eles fui bem-sucedido, e dei um contributo activo para o sucesso dos projectos.

Fiz um investimento constante na expansão do meu conhecimento científico. Aprofundei as minhas competências tecnológicas e acompanhei as novas tendências. Adoptei novas áreas de saber de forma a melhor desempenhar a minha função. Exerci sempre as minhas funções num ambiente de respeito e cooperação mútuas com aqueles com quem interagi. Auxiliei e motivei aqueles que dependeram de mim, tentei aprender ao máximo com aqueles de quem dependi.

Assim sendo, acredito que o meu perfil profissional se coaduna com as expectativas delineadas para um mestre em Engenharia de Informática e por conseguinte solicito a creditação do grau de mestre em Engenharia Informática.

Referências

- Guimarães Rodrigues, A. J. M., “Investigação Operacional”, Universidade do Minho
- Tramontani, A., “Recent MIP Performance Improvements in IBM ILOG Cplex Optimization Studio”,IBM, 2014
- Klotz, E., “Practical Guidelines for Solving Difficult Mixed Integer Programs”, IBM, 2014
- Larrosa, J., Oliveras A. M., “Mixed Integer Linear Programming”
- Smith, J., Taskm Z., “A Tutorial Guide to Mixed-Integer Programming Models and Solution Techniques”, University of Florida, 2007
- “Recent MIP Performance Improvements in IBM ILOG Cplex Optimization Studio”, IBM
- “Practical Guidelines for Solving Difficult Mixed Integer Programs”, IBM, 2013
- “Concert Technology for Java Users”, IBM, 2014
- “Optimizing MIP Problems”, Gurobi, 2014
- Albert S., “Solving Mixed Integer Linear Programs Using Branch and Cut Algorithm”, North Carolina State University, 2007
- Borghoff J., “Mixed Integer Programming: Algorithms and Applications”, 1995
- Cornuejols G., Trick M., “A Tutorials on Integer Programming”, 1995
- Bien A., “Real World Java EE Patterns-Rethinking Best Practices”, Lulu, 2012
- Hunt C., “Java Performance”, Addison-Wesley, 2011
- Liu H, “Java Performance and Scalability: A Quantitative Approach”, Create Space, 2013
- Barker P, “Java Methods for Financial Engineering: Applications in Finance and Investment”, Springer, 2007
- Bolsa de Valores do Porto, “Um Mercado de Repos em Portugal”, Instituto do Mercado de Capitais, 1996
- Bolsa de Valores do Porto, “Introdução aos Mercados de Futuros e Opções.”, Instituto do Mercado de Capitais, 1994.
- Fabozzi F., “Securities Finance: Securities Lending and Repurchase Agreements”, Wiley, 2005
- Hull J., “Risk Management and Financial Institutions”, Wiley, 2012

Anexos

2 July, 2001

Paulo Moreira
Urbanizacao Vila Campos, Lote 10
5000 Vila Real
Portugal

Dear Paulo,

I would like to thank you for submitting your article entitled *From Personal Java to J2ME: Some Introductory Ideas.*

After reviewing your article we are pleased to inform you that it has been accepted for publication on the Micro Java Network. We will run your article in the Perspective Articles section and will be carried on the front page for four weeks starting on 5 July 2001.

Your article has provided the community with a good overview of the different J2ME Profiles and how they are related. This is an area that a number of readers have expressed some confusion and so we are pleased to be able to showcase your article in response to their needs. We would like to encourage you to consider future submissions to the Micro Java Network.

Please accept our congratulations and we look forward to any future submissions you would like to provide for our consideration.

Yours truly,

Angus D. Muir
President & CEO
MicroDevGroup

Sun Certified Programmer for the Java 2 Platform 1.4
Examination Score Report

CANDIDATE: PAULO MOREIRA
ADDRESS: HOOGPOORTDREEF 5
AMSTERDAM, 1101 BA
CANDIDATE ID: SR1596796 **DATE:** March 29, 2004
REGISTRATION NUMBER: K21DUS50F2 **SITE NUMBER:** NL48
EXAM: Sun Certified Programmer for the Java 2 Platform 1.4
SERIES: 035

Exam Results
Your percentage score was determined by taking the overall number of correct answers divided by the total number of questions.

There are 61 questions in this exam. You answered 55 questions correctly which gives you a score of 90%.

PASSING SCORE: 52% **YOUR SCORE:** 90% **TEST STATUS:** Pass

Assessment Section
This report shows the percentage of items in each section you answered correctly for the Sun Certified Programmer for the Java 2 Platform 1.4 exam. The following information is provided to give you feedback on your relative strengths on a per section basis.

Section Analysis	Score %
Declarations and Access Control	100 %
Flow Control, Assertions, and Exception Handling	100 %
Garbage Collection	66 %
Language Fundamentals	90 %
Operators and Assignments	100 %
Overloading, Overriding, Runtime Type and Object Orientation	100 %
Threads	62 %
Fundamental Classes in the java.lang package	100 %
The Collections Framework	66 %

Certification testing is a means of measuring your knowledge and skill level. It can also be used to identify areas that need improvement and areas of strength. It can be used as a tool for further learning and future achievement.

Sun Educational Services offers a variety of tools to help you in your career. By logging on to our certification database, you can find scores for examinations you have taken, confirm or update your address, track your certification progress, and submit questions to the Sun Educational Services' Certification Department.

This assistance is available online at http://www.certmanager.net/sun/.

This examination was delivered at an Authorized Prometric Testing Center. To register for another Sun Microsystems exam, logon to http://www.register.prometric.com. To learn more about Sun Microsystems certification programs visit www.prometric.com/sun.

Upgrade Exam for the Sun Certified Programmer for the Java Platform, Standard Edition 6.0
Examination Score Report

CANDIDATE: PAULO MOREIRA
ADDRESS: RUE DE ASTRID 19
LUXEMBOURG VILLE, L-1143
CANDIDATE ID: SR1596796 **DATE:** May 22, 2009
REGISTRATION NUMBER: R7APAR5005 **SITE NUMBER:** PT5
EXAM: Upgrade Exam for the Sun Certified Programmer for the Java Platform, Standard Edition 6.0
SERIES: 066

Exam Results
Your percentage score was determined by taking the overall number of correct answers divided by the total number of questions.

There are 48 questions in this exam. You answered 47 questions correctly which gives you a score of 97%

PASSING SCORE: 66% **YOUR SCORE:** 97% **TEST STATUS:** Pass

Assessment Section
This report shows the percentage of items in each section you answered correctly for the Upgrade Exam for the Sun Certified Programmer for the Java Platform, Standard Edition 6.0 exam. The following information is provided to give you feedback on your relative strengths on a per section basis.

Section Analysis	Score %
Declarations, Initialization and Scoping	100 %
Flow Control	100 %
API Contents	83 %
Concurrency	100 %
OO Concepts	100 %
Collections / Generics	100 %
Fundamentals	100 %

Certification testing is a means of measuring your knowledge and skill level. It can also be used to identify areas that need improvement and areas of strength. It can be used as a tool for further learning and future achievement.

Sun Educational Services offers a variety of tools to help you in your career. By logging on to our certification database, you can find scores for examinations you have taken, confirm or update your address, track your certification progress, and submit questions to the Sun Educational Services' Certification Department.

This assistance is available online at http://www.certmanager.net/sun/.

To help you further your growth and achievement in the Java language, Sun Educational Services offers

This examination was delivered at an Authorized Prometric Testing Center. To register for another Sun Microsystems exam logon to http://www.register.prometric.com. To learn more about Sun Microsystems certification programs visit www.prometric.com/sun.

RUMOS, S. A.
Campo Grande, 56-4º
1700-093 Lisboa
Cont. n.º 260 268 039
Registo Comercial nº 43.632
Capital Social: 500.000 Euros

Sun Certified Business Component Developer for J2EE 1.3
Examination Score Report

CANDIDATE: PAULO MOREIRA
ADDRESS: HOOGPOORTDREEF 5
AMSTERDAM, 1101 BA
CANDIDATE ID: SR1596796 **DATE:** December 13, 2004
REGISTRATION NUMBER: L54DUS5038 **SITE NUMBER:** NL48
EXAM: Sun Certified Business Component Developer for J2EE 1.3
SERIES: 090

Exam Results
Your percentage score was determined by taking the overall number of correct answers divided by the total number of questions

There are 70 questions in this exam. You answered 65 questions correctly which gives you a score of 92%.

PASSING SCORE: 64% **YOUR SCORE:** 92% **TEST STATUS:** Pass

Assessment Section
This report shows the percentage of items in each section you answered correctly for the Sun Certified Business Component Developer for J2EE 1.3 exam. The following information is provided to give you feedback on your relative strengths on a per section basis.

Section Analysis	Score %
EJB Overview	100 %
Client View of a Session Bean	100 %
Session Bean Component Contract	100 %
Session Bean Lifecycle	100 %
Client View of an Entity	100 %
Component Contract for Container-Managed Persistence (CMP)	66 %
CMP Entity Bean Lifecycle	100 %
Entity Beans	100 %
EJB-QL	80 %
Message-Driven Bean Component Contract	100 %
Transactions	87 %
Exceptions	100 %
Enterprise Bean Environment	83 %
Security Management	100 %

Certification testing is a means of measuring your knowledge and skill level. It can also be used to identify areas that need improvement and areas of strength. It can be used as a tool for further learning and future achievement.

Sun Educational Services offers a variety of tools to help you in your career. By logging on to our certification database, you can find scores for examinations you have taken, confirm or update your address, track your certification progress, and submit questions to the Sun Educational Services' Certification Department.

This examination was delivered at an Authorized Prometric Testing Center. To register for another Sun Microsystems exam logon to http://www.register.prometric.com. To learn more about Sun Microsystems certification programs visit www.prometric.com/sun.

Sun Certified Business Component Developer Java EE Platform 5 Upgrade Exam
Examination Score Report

CANDIDATE: PAULO MCREIRA
ADDRESS: RUE DE ASTRID 19
LUXEMBOURG VILLE. L1143
CANDIDATE ID: SR1596796 **DATE:** December 31, 2008
REGISTRATION NUMBER: QE8PAR507D **SITE NUMBER:** PT5
EXAM: Sun Certified Business Component Developer Java EE Platform 5 Upgrade Exam
SERIES: 092

Exam Results
Your percentage score was determined by taking the overall number of correct answers divided by the total number of questions.

There are 50 questions in this exam. You answered 46 questions correctly which gives you a score of 92%.

PASSING SCORE: 60% **YOUR SCORE:** 92% **TEST STATUS:** Pass

Assessment Section
This report shows the percentage of items in each section you answered correctly for the Sun Certified Business Component Developer Java EE Platform 5 Upgrade Exam exam. The following information is provided to give you feedback on your relative strengths on a per section basis.

Section Analysis	Score %
EJB 3.0 Overview	100 %
General EJB 3.0 Enterprise Bean Knowledge	80 %
EJB 3.0 Session Bean Component Contract & Lifecycle	100 %
EJB 3.0 Message-Driven Bean Component Contract	100 %
Java Persistence API Entities	71 %
Java Persistence Entity Operations	100 %
Persistence Units and Persistence Contexts	83 %
Java Persistence Query Language	100 %
Transactions	100 %
Exceptions	100 %
Security Management	100 %

Certification testing is a means of measuring your knowledge and skill level. It can also be used to identify areas that need improvement and areas of strength. It can be used as a tool for further learning and future achievement.

Sun Educational Services offers a variety of tools to help you in your career. By logging on to our certification database, you can find scores for examinations you have taken, confirm or update your address, track your certification progress, and submit questions to the Sun Educational Services' Certification Department.

This assistance is available online at http://www.certmanager.net/sun/.

This examination was delivered at an Authorized Prometric Testing Center. To register for another Sun Microsystems exam logon to http://www.register.prometric.com. To learn more about Sun Microsystems certification programs visit www.prometric.com/sun.

British Language Training Centre

Certificate

This is to certify that	*Paulo Jorge Ribeiro Moreira*
has attended BLTC's	*Business Fluency C*
an	*English language course*
consisting of	*28 hours*
held between	*16 September – 16 December 2003*

Individual Comment

This course is for students who already have a highly proficient level of English and who are used to operating in English in business and social contexts. The aim of the course is to deepen the students' awareness of the language in specific fields of the business world.

Course Tutor

Director of Studies

REPÚBLICA PORTUGUESA

MINISTÉRIO DO TRABALHO E DA SOLIDARIEDADE

SNCP

SISTEMA NACIONAL DE CERTIFICAÇÃO PROFISSIONAL

CERTIFICADO DE APTIDÃO PROFISSIONAL

(Decreto-Lei n.º 95/92, de 23 de Maio e Decreto-Regulamentar n.º 66/94, de 26 de Novembro)

Certifica-se que **PAULO JORGE RIBEIRO MOREIRA** nascido em 12-11-1978, natural de Vila Real (S. Dinis) - Vila Real, portador do B.I. nº 11294482 emitido pelo Arquivo de Identificação de Lisboa, em 16-03-2000, possui competências pedagógicas para exercer a profissão de **FORMADOR** (M/F), conforme as que são definidas no respectivo perfil profissional

IEFP

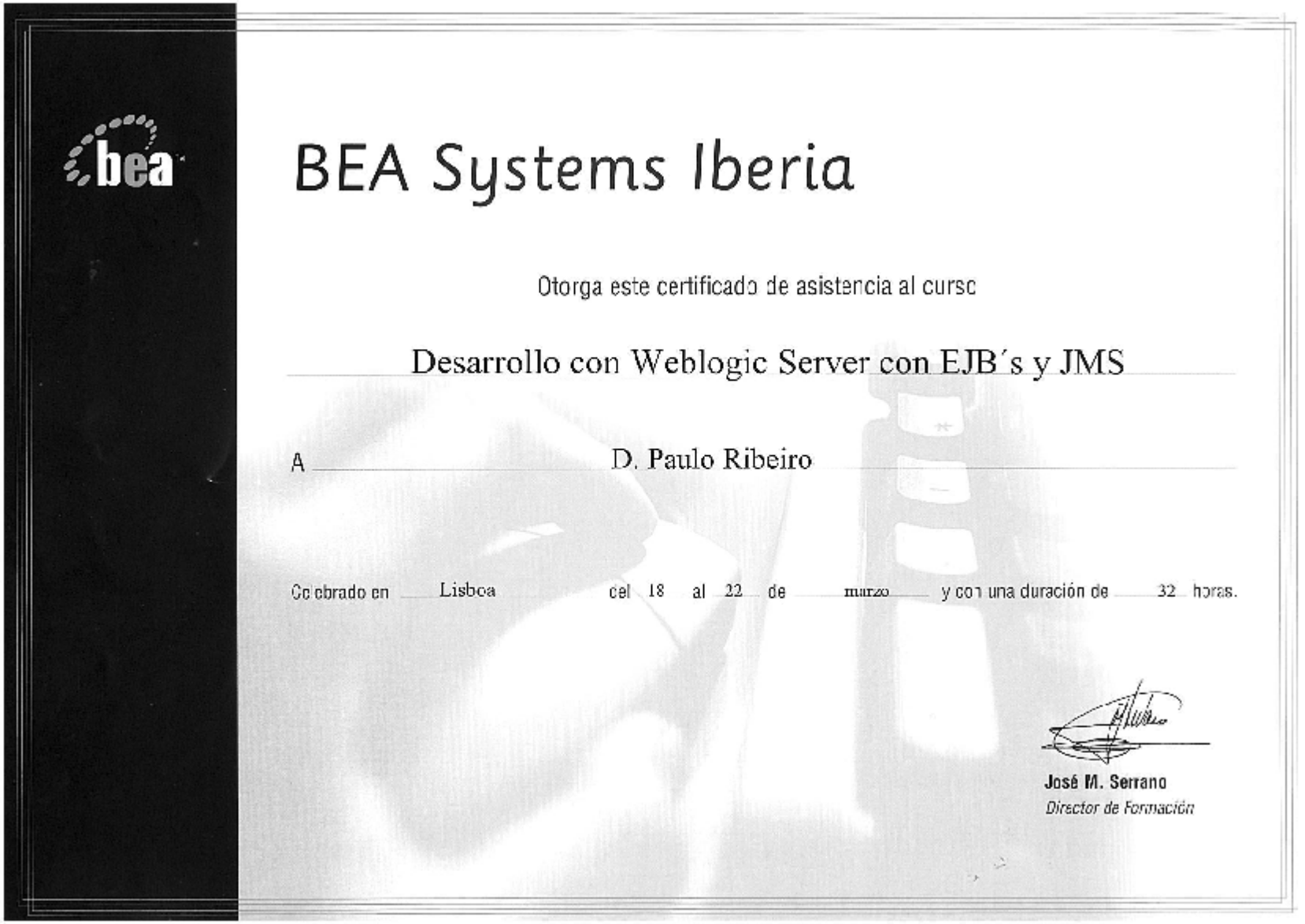
Instituto do Emprego e Formação Profissional, entidade certificadora competente ao abrigo dos Decretos Regulamentares 66/94 de 18 de Novembro e 26/97 de 18 de Junho.

Porto 04 de Junho de 2002

O Delegado Regional

(Carlos Boticas)

Certificado nº EDF 34025/2002 CN Válido até 04-06-2007



BEA Systems Iberia

Otorga este certificado de asistencia al curso

Desarrollo con Weblogic Server con EJB´s y JMS

A D. Paulo Ribeiro

Celebrado en Lisboa del 18 al 22 de marzo y con una duración de 32 horas.

José M. Serrano
Director de Formación

wapeton

nuevas tecnologías

Madrid, 30 de octubre de 2001

Wapeton Nuevas Tecnologías, compañía dedicada a la consultoría y el desarrollo de proyectos en el ámbito de las nuevas tecnologías, con domicilio en Madrid, C/ Fuencarral 121, certifica que:

El artículo titulado "From Personal Java2ME: some introductory ideas", realizado por el el Sr. Don Paulo Moreira, ha sido publicado en la web MovilForum (www.movilforum.com) perteneciente a Telefónica Móviles y de la que Wapeton Nuevas Tecnologías es proveedor de contenidos.

Covadonga Aznar Arias
Responsable de Explotación de Contenidos
Wapeton Nuevas Tecnologías

C/ Fuencarral 121 · 28010 Madrid · Tel.: 91 591 20 88 - 626 457 390 · Fax: 91 591 34 41

SACRED HEART UNIVERSITY
LUXEMBOURG
John F. Welch COLLEGE OF BUSINESS

Luxembourg, 8 August 2014

Certification of Student Status
Master of Business Administration Program
Student Name: Paulo Jorge Ribeiro Moreira

To Whom It May Concern:

This letter is to confirm that Mr. Paulo Jorge Ribeiro Moreira is an active student in the Master of Business Administration program at Sacred Heart University Luxembourg. This program enables working professionals to participate in a graduate level educational experience that enhances their career opportunities by developing their managerial skills and abilities. Below is a list of courses taken by Mr. Moreira:

Course Title	Grade	Term
WGB 600 Professional Planning	A-	Summer 2013
WGB 520 Introduction to Economics & Statistics	A	Summer 2013
WGB 603 Corporate Financial Management	A	Fall 2013
WGB 612 Leading & Influencing with Integrity	A	Fall 2013
FN 662 Corporate Finance	A	Fall 2013
FN 673 Mathematics for Finance Practitioners	A	Spring 2014
FN 674 Advanced Risk Management	A-	Spring 2014
FN 663 Global Investments	B+	Late Spring 2014
WGB 614 Social & Legal Responsibilities in Business	A	Summer 2014

Mr. Moreira has completed a total of 480 hours in the MBA program at Sacred Heart University Luxembourg.

Below is a list of tuition fees paid by Mr. Moreira:

Payment	Date	Amount
Installment 3 of 4	4-Jan-00	[redacted]
Installment 2 of 4	5-Nov-13	[redacted]
Installment 1 of 4	12-Jul-13	[redacted]
Admission Fee	12-Jul-13	[redacted]
Application Fee	12-Jul-13	[redacted]
Total	[redacted]	[redacted]

Sacred Heart University is a U.S. accredited and U.S. based university with its main campus in Fairfield, Connecticut. The university was founded in 1963 and has been offering the Master of Business Administration and Graduate Professional Certificate programs in Luxembourg since 1991.

If you have any questions or require any further information, please do not hesitate to contact me.

Sincerely,

Alexandra Lewis
Office Manager