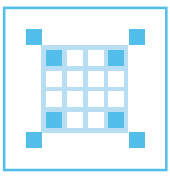


DEEC
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Projecto de Engenharia de Software
2017/2018

Etapa 1

Análise de Requisitos e Especificações em UML

Autoria:

Diogo M. T. Poço
2014205007
uc2014205007@student.uc.pt

João C. da C. Barreiros
2014196880
uc2014196880@student.uc.pt

João M. P. Agria
2010129833
uc2010129833@student.uc.pt

Grupo: 2

23 de Fevereiro de 2018

Conteúdo

1 Introdução

Quando um grupo de engenheiros fica encarregue de um projecto de desenvolvimento de software, o primeiro passo a tomar é marcar reuniões entre os analistas de requisitos do grupo e todos os diferentes grupos de *stakeholders* do sistema, tais como a organização empregadora, para determinar quais são os comportamentos e características desejadas no produto final. Apenas quando se chegar a um consenso poderá o grupo de *developers* passar para a análise do problema.

Qualquer engenheiro que tenha um problema para resolver saberá que tem de começar por perceber a natureza do mesmo. A investigação de um problema de tamanho considerável (como é normalmente o caso de um projecto de desenvolvimento de software) inicia-se com a análise do mesmo, ou seja, desconstroi-se o problema em pequenas parcelas que sejam facilmente percebidas e solucionadas. Chega-se à solução geral do problema estabelecendo conexões entre as soluções obtidas para cada uma das parcelas. Chama-se a este processo justamente síntese, pois trata-se da sintetização de uma solução com base na análise realizada.

A primeira parte do processo de análise é a identificação dos limites do sistema a ser desenvolvido, das suas partes constituintes e das relações entre estas. A identificação dos requisitos de sistema é seguida pela criação do design de um sistema que garanta o cumprimento dos mesmos. A organização empregadora (*o customer*) terá de ser consultado no fim deste passo para aprovar o trabalho. Fica então claro que o design do sistema é feito a partir da prespectiva do *customer*, é facilmente interpretável pelo mesmo e descreve apenas a aparência e funcionalidade do sistema.

O design do sistema é utilizado para gerar o design de cada um dos programas individuais envolvidos. Estes serão testados como peças individuais de código, na fase designada por teste de módulos, antes de serem conectados e testados conjuntamente de forma a verificar que trabalham apropriadamente em conjunto. Esta fase de teste é designada teste de integração, pois o sistema é construído peça a peça até a totalidade do sistema estar operacional. A última fase de teste é o teste do sistema e serve para verificar que as funcionalidades e interacções especificadas no design do sistema foram correctamente implementadas. Todos os *stakeholders* confirmam que o sistema serve o seu propósito e o produto final está feito e pronto para entrega. Em qualquer fase deste processo de desenvolvimento de software, os *stakeholders* podem verificar o progresso do trabalho e fornecer *feedback* aos *developers*. Salientamos também que fases que se pensavam ter sido terminadas com sucesso podem ser revistas para incluírem novas realidades só podiam ser consideradas em fases mais avançadas de desenvolvimento.

Tendo estabelecido os conceitos básicos de resolução de um problema genérico, falta apenas dar resposta às perguntas: Qual é o problema a ser resolvido? e Quem são os *stakeholders* presentes no projecto de desenvolvimento?

1.1 Definição do Problema

No âmbito da disciplina de Engenharia de Software pretende-se criar uma aplicação de software para ajudar um utilizador individual a arquivar adequadamente grandes quantidades de fotos em formato digital, de forma organizada, permitindo assim a sua posterior consulta e visualização de forma prática e eficiente.

1.2 Identificação dos *stakeholders*

Os quatro *stakeholders* presentes no desenvolvimento deste projecto são:

- o grupo de três alunos, que assumem o papel de *developers* e ficam incumbidos de estruturar e implementar o programa, tendo em conta os requisitos e restrições impostas pelo cliente
- o professor da disciplina de Engenharia de Software, que encarna vários papéis:

– o de *customer*, pois empregará (sem remuneração) os *developers* para criarem o sistema de acordo com as suas exigências.

– o de educador, que usará este projecto para aferir se os seus educandos, os *developers*, estão a aprender as estratégias necessárias para desenvolvimento de software.

2 Requisitos

2.1 Identificação de requisitos

Define-se um requisito como um comportamento que esperado do sistema. Durante a fase de análise de requisitos, o nosso grupo interpretará o documento [1] para registá-los e categorizá-los em quatro tipos distintos.

2.1.1 Requisitos Funcionais

Os requisitos funcionais definem que serviços devem ser fornecidos pelo sistema, que operações podem ser realizadas, quais as reacções a certos estímulos (entradas no sistema), e como deve o estado das entidades constituintes do sistema mudar devido à ocorrência de certas actividades.

Apresentamo-los organizados em forma de tabela, e a cada requisito funcional atribuimos um código de identificação numérica:

	Requisitos Funcionais	Prioridade
1	Gerir álbuns	1
1.1	Adicionar novos	1
1.2	Modificar dados	2
1.3	Eliminar	1
2	Pesquisar fotos	1
2.1	por palava-chave	1
2.2	por pessoas	1
2.3	por datas	1
3	Fazer browsing na árvore de diretórios	2
3.1	por miniaturas (thumbnails)	3
3.2	por nome	3
3.3	por lista detalhada	3
4	Gerir fotos	1
4.1	Adicionar	1
4.2	Copiar	3
4.3	Modificar dados	3
4.4	Eliminar	1
4.5	Mover entre páginas do mesmo álbum ou páginas de álbuns diferentes	3
5	Gerir uma lista de pessoas	1
5.1	Adicionar	1
5.2	Alterar dados	2
6	Gerir páginas	1
6.1	Adicionar novas	1
6.2	Mover entre álbuns	3
6.3	Copiar	3

6.4	Eliminar	1
6.5	Modificar dados	2
7	Visualizar	1
7.1	Álbuns	1
7.2	Páginas	1
7.3	Fotos	1
7.4	Pessoas e dados dos mesmos	1
8	Criar Slideshows	4
8.1	Com possibilidade de escolha de música	5
8.2	Com configuração do tempo entre duas fotos	5
8.3	Com possibilidade de apresentação organizada de fotos	5
8.3.1	Por ordem alfabética do nome	5
8.3.2	Por ordem alfabética inversa do nome	5
8.3.3	Da data mais antiga para a data mais recente	5
8.4.4	Da data mais recente para a data mais antiga	5
9	Imprimir fotos	2
10	Confirmar para apagar fotos, páginas ou álbuns	3
10.1	Confirmação dupla para álbuns e páginas	3
10.2	Confirmação simples para fotos e pessoas	3
11	Organizar fotos de uma página	4
11.1	Por ordem alfabética do nome	5
11.2	Por ordem alfabética inversa do nome	5
11.3	Da data mais antiga para a data mais recente	5
11.4	Da data mais recente para a data mais antiga	5

A ordem hierárquica das funcionalidades foi definida com a atribuição das prioridades na tabela acima. A escala é de 1, que representa a prioridade mais elevada, a 5, que representa a prioridade mais baixa.

2.1.2 Requisitos Qualitativos ou Não Funcionais

Um requisito qualitativo ou não funcional descreve alguma característica qualitativa que a solução do problema deve ter. Estes requisitos estão relacionados com diversos aspectos, da performance do software à sua segurança, capacidade de utilização, manutenibilidade, precisão, exactidão, custo de desenvolvimento, meta temporal para apresentação do produto final, entre outros.

No documento [1], estes são os requisitos não funcionais descritos:

- Interface gráfica intuitiva, com menus e barras de ferramentas.
- Minimização do uso do teclado, preferindo o uso do rato.
- Uso de teclas de atalho.
- O projeto tem de estar acabado pelo dia 20 de Maio de 2018.

2.1.3 Restrições de Design

Uma restrição de design é uma limitação imposta às possíveis soluções do problema que cumprem os requisitos funcionais e não funcionais. Pode referir-se ao tipo de arquitectura de software, sistema

operativo, linguagem de programação, tipo de utilizadores que podem usar o sistema, formatação das interfaces do sistema, meio físico de localização do sistema, consumo de energia, configuração de hardware, entre outros.

Aqui estão as restrições de design que encontrámos no documento [1]:

- Utilização da linguagem de programação C++.
- A aplicação de software tem de ser compativel com um dos seguintes sistemas operativos: MS Windows, Ubuntu Linux ou MAC OS.
- A aplicação será executada por um único utilizador individual, num computador pessoal.
- Os álbuns têm alguns atributos bem definidos, nomeadamente o nome, descrição, tipo de páginas e pasta onde serão guardadas as fotos.
- As páginas têm alguns atributos bem definidos, nomeadamente o tipo.
- As fotos têm alguns atributos bem definidos, nomeadamente a resolução, a data em que foi tirada e as pessoas associadas.
- A vista por defeito quando se faz browsing é miniaturas (thumbnails).
- As pessoas têm alguns atributos bem definidos, nomeadamente o nome, a data de nascimento, género e relação (ou grau de parentesco) com o utilizador.
- Só se podem mover páginas inteiras entre álbuns se houver compatibilidade entre o tipo de páginas dos dois álbuns.
- As fotos copiadas de uma pasta para outra são automaticamente organizadas na pasta destino.
- Cada álbum corresponde a uma pasta no disco rígido, e cada uma das suas páginas corresponde a uma subpasta dessa pasta.

2.1.4 Restrições de Processo

Uma restrição de processo designa qualquer limitação nas técnicas, métodos, recursos e standards que podem ser utilizados na construção do sistema.

A descrição informal de requisitos fornecida[1] apresenta estas restrições de processo:

- O grupo só pode ter 3 elementos
- O processo de desenvolvimento tem de ser incremental
- Teremos que fazer várias submissões ao longo do projeto, com milestones bem definidos a atingir e que descrevemos mais à frente

3 Especificação de Requisitos

3.1 Diagrama de Casos de Uso

Após a identificação dos requisitos funcionais, construímos os diagramas UML.

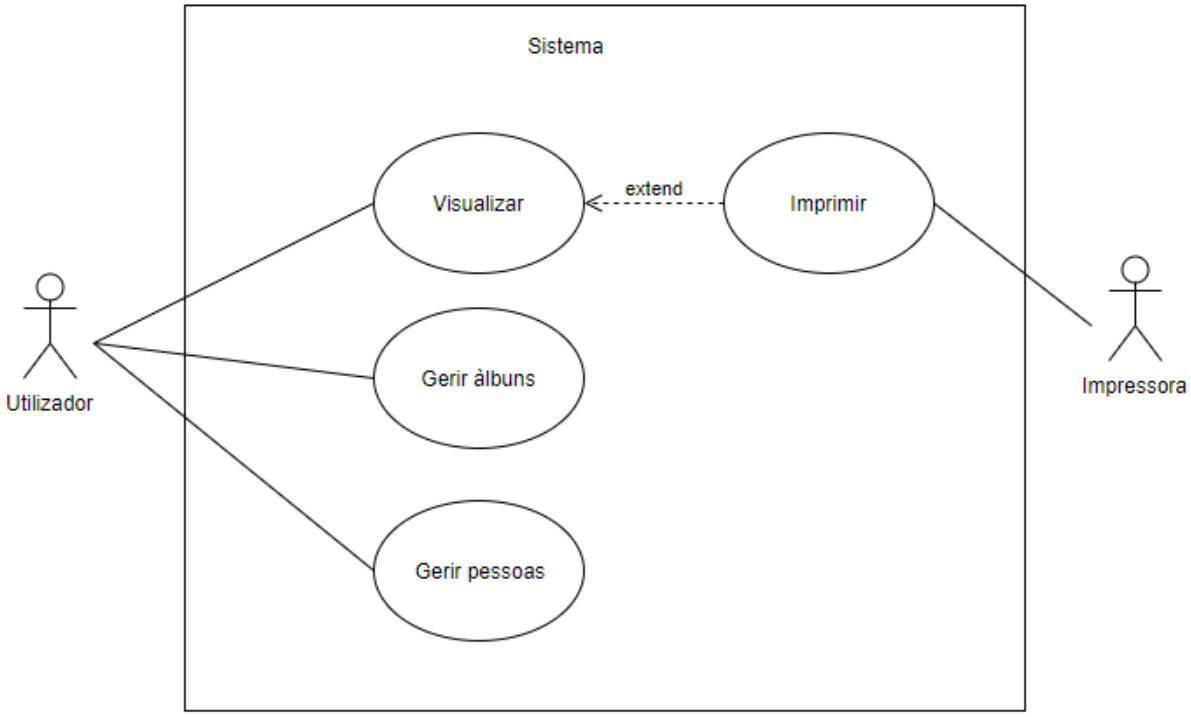


Fig 1: Diagrama UML de casos de uso geral (kite level)

Um diagrama de casos de uso UML descreve funcionalidades iniciadas pelo(s) actor(es) em termos de interações entre o sistema e o meio onde este está localizado. Os limites do sistema são representados como uma caixa, e fora desta estão os actores e sistemas externos.

No diagrama da figura 1, observamos uma representação das funcionalidades do sistema no nível de abstração mais elevado. O utilizador, um dos actores presentes, quererá ver as fotos que tem na aplicação, bem como os dados destas, logo justifica-se a existência do caso de uso visualizar. Quererá também gerir os álbuns na aplicação e a lista de pessoas mantida pelo programa.

A impressora é outro actor, e será usada apenas para imprimir fotos. Para imprimir uma foto é necessário visualizá-la, logo podemos considerar que visualizar é um subcaso de imprimir.

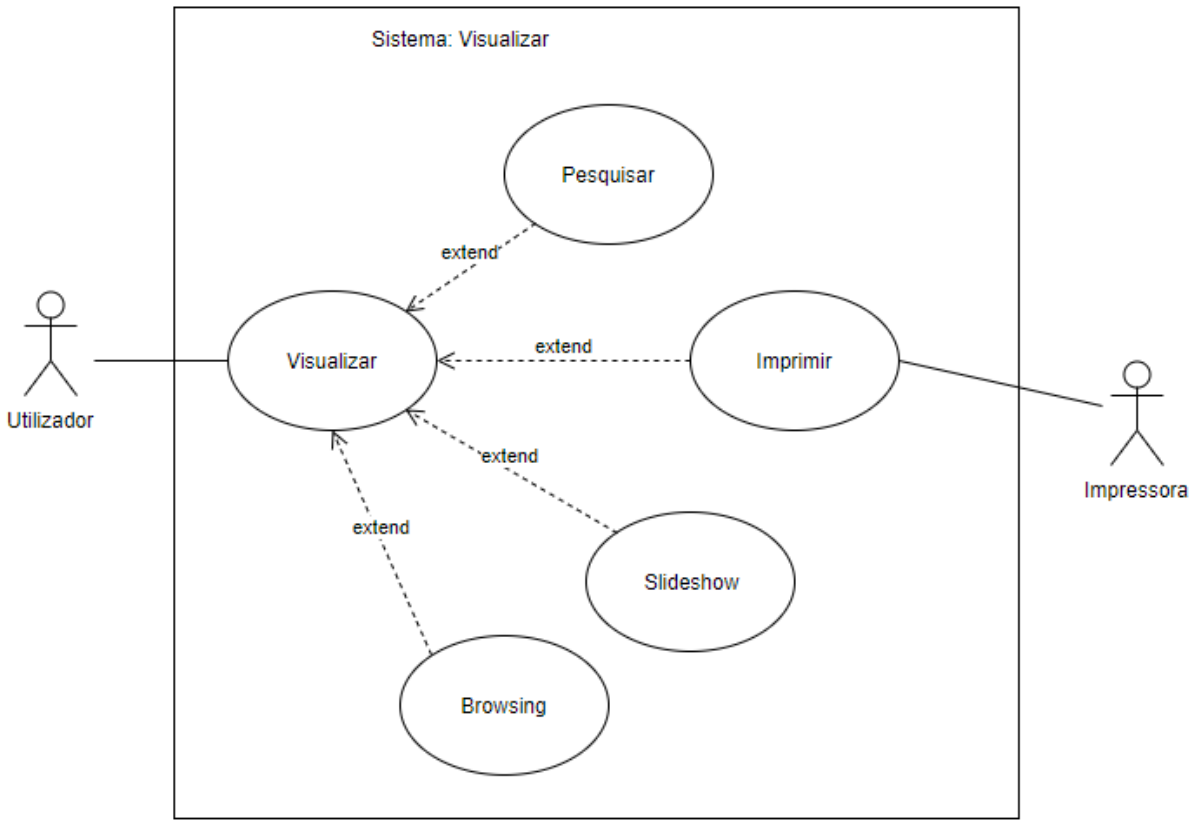


Fig 2: Diagrama UML de casos de uso Visualizar (sea level)

No caso de uso visualizar, o utilizador poderá querer pesquisar fotos, fazer browsing na árvore de directórios ou criar um slideshow com fotos.

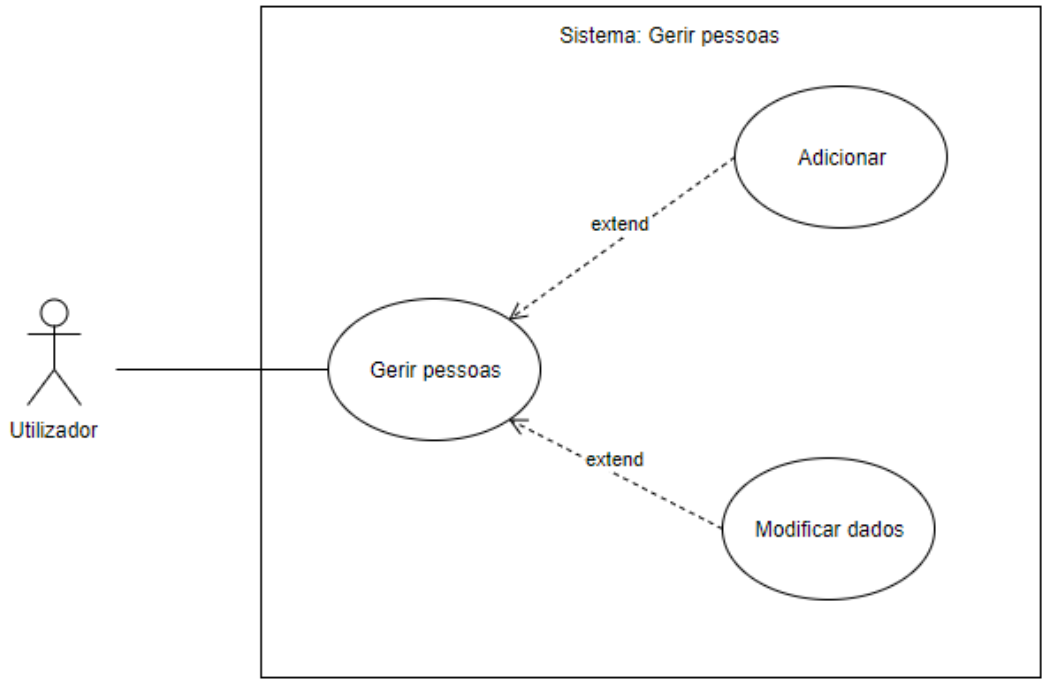


Fig 3: Diagrama UML de casos de uso Gerir pessoas (sea level)

No caso de uso gerir pessoas, o utilizador poderá querer adicionar novos elementos à lista de pessoas mantida pelo programa ou pode querer modificar dados destas.

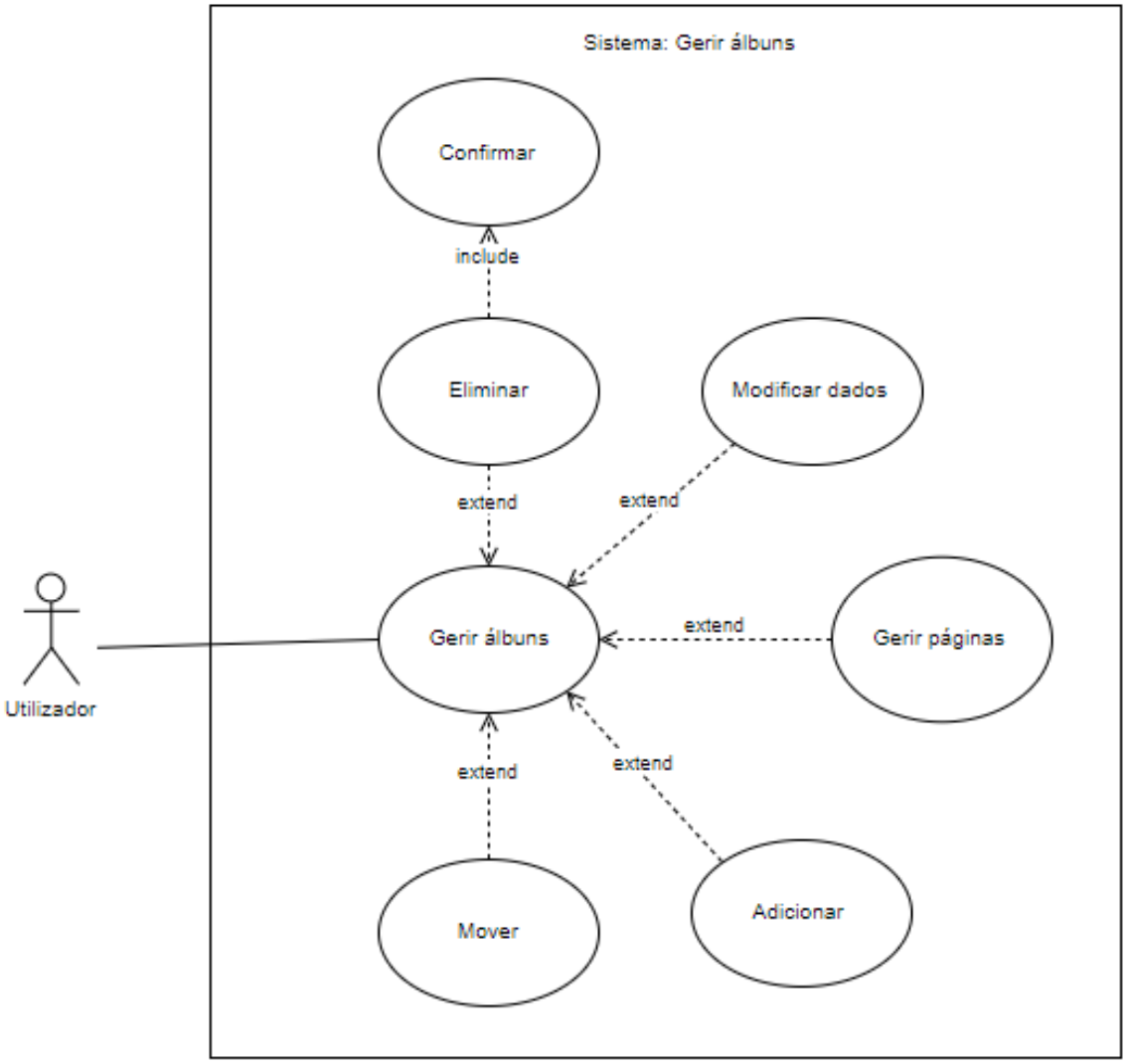


Fig 4: Diagrama UML de casos de uso Gerir álbuns (sea level)

No caso de uso gerir álbuns, o utilizador pode querer adicionar novos álbuns, modificar os dados referentes a um álbum em particular ou eliminar um ou mais álbuns. Para eliminar um álbum, é necessário introduzir confirmação. Gerir páginas é uma extensão da funcionalidade gerir álbuns, visto existir uma clara relação hierárquica entre álbuns e páginas (os primeiros pertencem a um nível hierárquico superior).

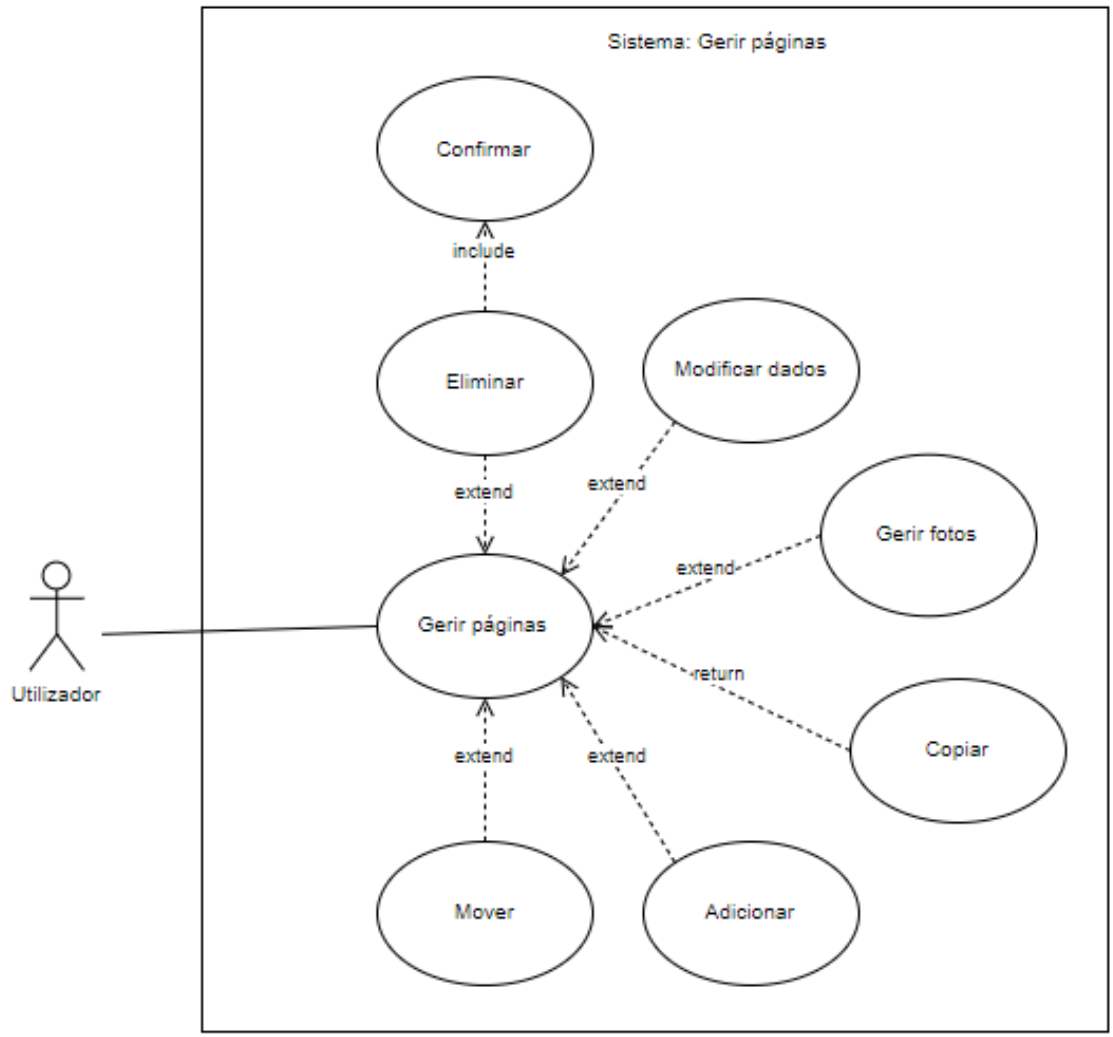


Fig 5: Diagrama UML de casos de uso Gerir páginas (underwater level)

Semelhantemente ao diagrama de casos de uso para gerir álbuns, no diagrama para gerir páginas temos incluídas as funcionalidades de eliminação de páginas, alteração de dados referentes a uma página, cópia de páginas para outras pastas, mover páginas de um álbum para outro e adicionar novas páginas a um álbum. Para ocorrer eliminação é preciso sempre confirmar com o utilizador. Gerir fotos é uma extensão de gerir páginas, devido à relação hierárquica entre páginas e fotos.

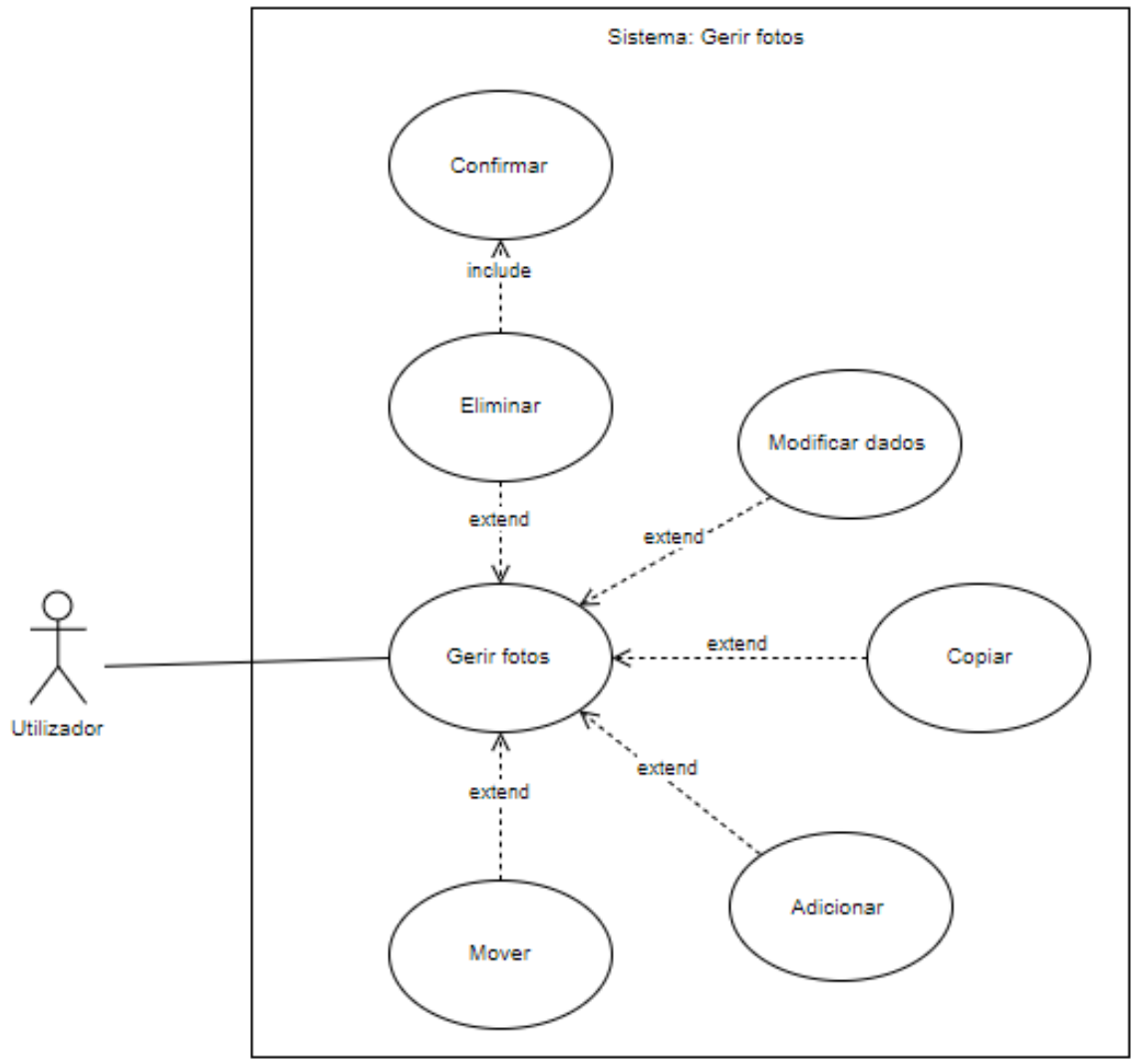


Fig 6: Diagrama UML de casos de uso Gerir fotos (clam level)

De forma análoga aos diagramas de casos de uso para gerir álbuns e gerir páginas, no diagrama para gerir fotos temos incluídas as funcionalidades de eliminação de fotos, alteração de dados referentes a uma foto, cópia de fotos para outras pastas, mover fotos de uma pasta para outra e adicionar novas fotos a um álbum ou página. Para ocorrer eliminação é preciso sempre confirmar com o utilizador.

3.2 Diagrama de Classes

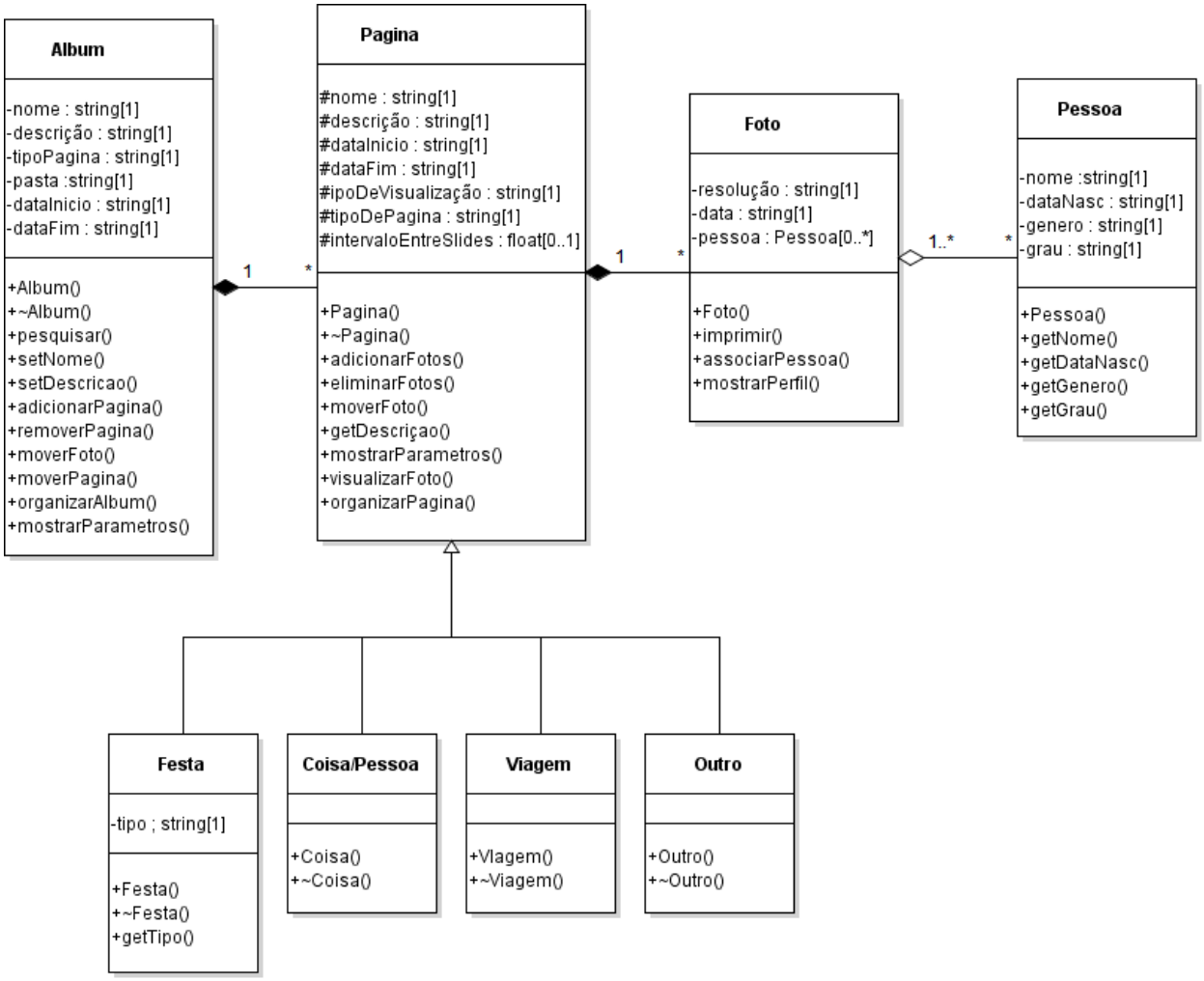


Fig 7: Diagrama UML de Classes

Neste diagrama UML de classes, verificamos que as classes álbum, página e foto têm uma relação muito próxima, de carácter hierárquico. Uma foto é parte de uma página e está contida totalmente nela. Uma página pode ter múltiplas fotos. Da mesma forma, uma página é parte de um álbum e está contida totalmente nele. Um álbum pode conter múltiplas páginas. Esta relação de composição entre as três é o cerne do diagrama.

A classe de foto tem uma relação de agregação com a classe pessoa. Uma foto pode ter várias pessoas associadas.

As classes festa, coisa/pessoa, viagem e outro evento, sendo tipos possíveis de página, são generalizadas na classe página.

Cada classe possui os atributos e métodos que considerámos essenciais para o sistema .Os nomes reflectem o propósito da sua existência.

4 Plano de Desenvolvimento

Num projecto de desenvolvimento de software, o *costumer* quererá que a equipa de *developers* lhe garanta quanto tempo demorarão a apresentar o produto final. Para responder a essa pergunta é necessário calendarizar pormenorizadamente o projecto, descrevendo o ciclo de desenvolvimento e enumerando as fases constituintes deste. Este mapa criado estimará quanto tempo cada estágio de desenvolvimento demorará e definirá *milestones*, tendo por isso a vertente de facilitar a divisão e atribuição de tarefas entre o grupo de *developers*.

4.1 Modelo de Processo de Desenvolvimento

Para além dos factores temporais, um grupo de *developers* tem de adoptar um modelo de processo de desenvolvimento que irá impor uma estrutura para combinar técnicas e ferramentas para se chegar ao produto final.

O sistema será construído recorrendo a um modelo de processo de desenvolvimento incremental, o que significa que se particiona o sistema em subsistemas definidos pela sua funcionalidade. Desta forma os *developers*, cada vez que lançam uma nova versão do software, adicionam um novo subsistema funcional que melhora a versão do último lançamento. Este desenvolvimento faseado permite aos utilizadores terem acesso a partes do software enquanto outras partes do mesmo ainda estão em desenvolvimento e a serem preparadas para substituirem as primeiras.

Uma das grandes vantagens associadas a lançamentos frequentes é que permitem aos *developers* receber feedback dos utilizadores e melhorar funcionalidades já implementadas. Outra vantagem inerente é que os *developers* podem focar-se em áreas de especialização distintas em lançamentos diferentes. Um exemplo pertinente à aplicação de software a ser desenvolvida por este grupo de *developers* é a possibilidade de alterar o sistema de uma interface que use o teclado e a linha de comandos para uma interface gráfica que privilegie o rato.

4.2 Fontes de risco

Apesar de nos encontrármos numa etapa inicial do projecto, temos desde já noção de que ao longo do nosso percurso vamos ter de lidar com diversos obstáculos e dificuldades, a que chamamos fatores ou fontes de risco. Nomeadamente:

- Existência de outras cadeiras em simultâneo

  O facto de estarmos a frequentar outras cadeiras aumenta de forma significativa a carga semanal e o esforço que é pedido a cada um de nós. Para além disso, estamos sujeitos a ter de realizar trabalhos práticos e/ou outros projetos cujas metas podem coincidir, por vezes, com as datas de submissão das etapas.

- A organização da equipa

  Envolve a gestão de tempo de cada um, a distribuição de tarefas e a discussão de ideias entre todos para que posso haver acordo.

- Inexperiência na implementação de Interfaces Gráficas

  Nós não estamos muito familiarizados na programação de aplicações que tomam proveito de interfaces gráficas, pelo que será necessário ter um maior cuidado durante a implementação de uma neste projeto.

- Falta de conhecimentos relativos à plataforma de desenvolvimento

  Uma vez que escolhemos a linguagem de programação C++, optámos também por utilizar a plataforma Qt para a criação do software, por já ter sido usada em cadeiras passadas e por

facilitar a construção de uma interface gráfica. No entanto existe o risco da implementação pedir o uso funções exclusivas da plataforma, desconhecidas a nós.

- Atividades extracurriculares dos membros

  Dois dos membros da equipa pertencem ao Núcleo de Estudantes de Engenharia Eletrotécnica e de Computadores (NEEEC/AAC), dessa forma, estão sujeitos a ajudar na organização de certas atividades e eventos ao longo do ano lectivo.

- Eventos indesejáveis

  Para além dos riscos já mencionados, ainda existem outros eventos que estão fora do nosso controlo, nomeadamente a doença de qualquer membro da equipa ou a falha dos computadores utilizados para a realização de este projeto.

4.3 Ferramentas para Seguir o Progresso

Para acompanhar o progresso do projecto e para estabelecer a cadência do trabalho, criámos quatro diagramas de Gantt (presentes em anexo na secção 6, figuras 8, 9, 10 e 11). Podemos observar que dividimos cada etapa em fases distintas e que estabelecemos milestones para cada uma dessas fases que nos permitam alcançar as metas temporais estabelecidas pelo professor. A elaboração destes mapas teve em conta vários elementos: desde os factores de risco descritos na secção anterior, como o carácter e capacidades de cada elemento do grupo. Uma vez que o projecto exigirá um esforço considerável, almejaremos estar sempre adiantados em relação a estes mapas para contornar quaisquer eventos imprevisíveis e, de certeza, inevitáveis que enfrentaremos ao longo do semestre.

5 Conclusão

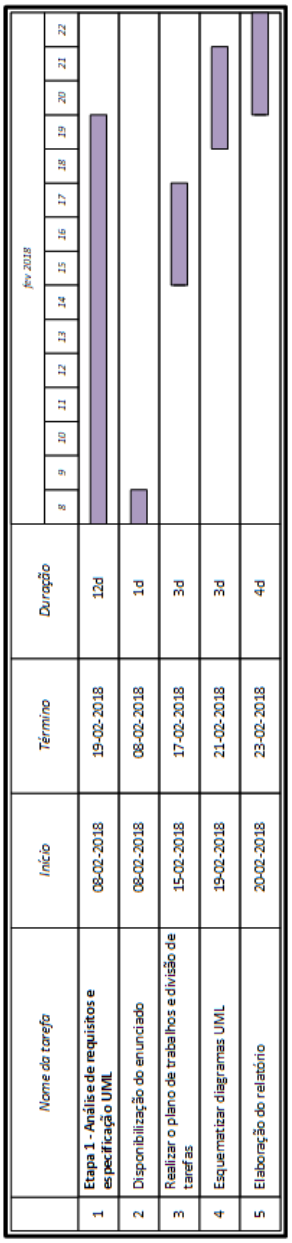
Uma das características vitais que nos permitirão desenvolver este projecto com sucesso e cumprir todas as metas estabelecidas é a capacidade de comunicação dos elementos do grupo. Outras são a flexibilidade e capacidade de compromisso que permitirão enfrentar os diversos obstáculos que se seguem. Esta etapa do trabalho foi muito útil porque permitiu conhecer a natureza do problema que nos propomos a resolver, estabelecer canais de comunicação e planear o processo de desenvolvimento.

No entanto, a planificação do projecto e a análise de problema estão sujeitas a revisões no futuro caso se decida que já não reflectem as nossas necessidades.

In preparing for battle, I have always found that plans are useless but planning is indispensable.

Dwight D. Eisenhower
retirado de Six Crises (1962) por Richard Nixon

6 Diagramas de Gantt



	Nome da tarefa	Início	Término	Duração
1	**Etapa 1 - Análise de requisitos e especificação UML**	08-02-2018	19-02-2018	12d
2	Disponibilização do enunciado	08-02-2018	08-02-2018	1d
3	Realizar o plano de trabalhos e divisão de tarefas	15-02-2018	17-02-2018	3d
4	Esquematizar diagramas UML	19-02-2018	21-02-2018	3d
5	Elaboração do relatório	20-02-2018	23-02-2018	4d

Figura 8: Diagrama de Gantt da primeira etapa

	Nome da tarefa	Início	Término	Duração
1	**Etapa 2 – Desenho da arquitetura em UML**	23-02-2018	10-03-2018	16d
2	Disponibilização do enunciado	23-02-2018	23-02-2018	1d
3	Realizar plano de trabalho e divisão de tarefas	23-02-2018	24-02-2018	2d
4	Elaboração do relatório	26-02-2018	09-03-2018	12d
5	Desenho da arquitetura em UML	26-02-2018	05-03-2018	8d
6	Revisão dos conceitos de interface gráfica	28-02-2018	07-03-2018	8d
7	Revisão da arquitetura em UML já realizada	14-03-2018	16-03-2018	3d
8	Protótipo da interface gráfica	09-03-2018	14-03-2018	6d

fev 2018: 23 24 25 26 27 28

mar 2018: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Figura 9: Diagrama de Gantt da segunda etapa

Figura 10: Diagrama de Gantt da terceira etapa

	Nome da tarefa	Início	Término	Duração
1	**Etapa 4 - Implementação completa, versão 2 do software**	13-04-2018	20-05-2018	38d
2	Disponibilização do enunciado	13-04-2018	13-04-2018	1d
3	Realizar plano de trabalho e divisão de tarefas	13-04-2018	15-04-2018	3d
4	Elaboração do relatório	16-04-2018	20-05-2018	35d
5	Implementação do código	18-04-2018	20-05-2018	33d

abr 2018: 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

mai 2018: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Figura 11: Diagrama de Gantt da quarta etapa

7 Glossário

- **Actividade**: Evento que ocorre num sistema.
- **Actor**: Elemento externo que interage com e inicia uma actividade noutro sistema.
- **Álbum**: Entidade da aplicação de software que agrega um conjunto de fotos com alguma característica em comum.
- **Costumer**: Empresa, organização ou pessoa que paga para desenvolver o sistema de software.
- **Developer**: Empresa, organização ou pessoa que constrói o software para o costumer.
- **Deliverable**: Itens que o *Costumer* espera ver para seguir o desenvolvimento do projecto.
- **Desenvolvimento incremental**: É uma estratégia de planeamento em que as várias partes são desenvolvidas separadamente e juntas no final, momento em que o projeto fica completo.
- **Diagrama de Gantt**: Gráfico que ilustra o avanço das várias etapas de um processo.
- **Entidade**: Também chamado objecto; elemento envolvido em actividades.
- **Fronteira do Sistema**: Delimitação entre o que está dentro do sistema e o que está fora deste.
- **Milestone**: Ponto no tempo em que se alcança a conclusão de uma actividade.
- **Objecto**: Ver entidade.
- **Página**: Entidade da aplicação de software que é contida por um álbum e arquiva fotos de um determinado tipo.
- **Sistema**: Um conjunto de actividades e um conjunto de objectos, complementados por uma descrição das relações entre os objectos e as actividades. Inclui uma definição da fronteira do sistema.
- **Stakeholder**: Grupo de pessoas envolvidas no desenvolvimento do projecto.
- **UML - Unified Modelling Language**: Colecção de notações usadas para documentar especificações e design de software.
- **Utilizador**: Pessoa ou pessoas que irão utilizar o sistema de software.

Referências

[1] R. P. Rocha, *Projecto de Engenharia de Software 2017/2018 - Descrição Informal de Requisitos*, Disciplina de Engenharia de Software, Departamento de Engenharia Eletrotécnica e de Computadores, Faculdade de Ciências e Tecnologia da Universidade de Coimbra, 2018

[2] R. P. Rocha, *Projecto de Engenharia de Software 2017/2018 - Etapa 1 - Análise de Requisitos e Especificação em UML*, Disciplina de Engenharia de Software, Departamento de Engenharia Eletrotécnica e de Computadores, Faculdade de Ciências e Tecnologia da Universidade de Coimbra, 2018

[3] Shari L. Pfleeger & Joanne M. Atlee, *Software Engineering: Theory and Practice, 4th edition*, Pearson Prentice Hall, ISBN10: 0136061699, ISBN13: 978-0136061694, 2010

[4] Martin Fowler, Addison-Wesley, *UML Distilled Third Edition: A Brief Guide to the Standard Object Modeling Language*, 3rd ed./5th printing, Pearson Education, 2003 (reimp. 2004). ISBN 0-321-19368-7.