

Mestrado em Engenharia Informática (MEI)

Mestrado Integrado em Engenharia Informática

(MiEI)

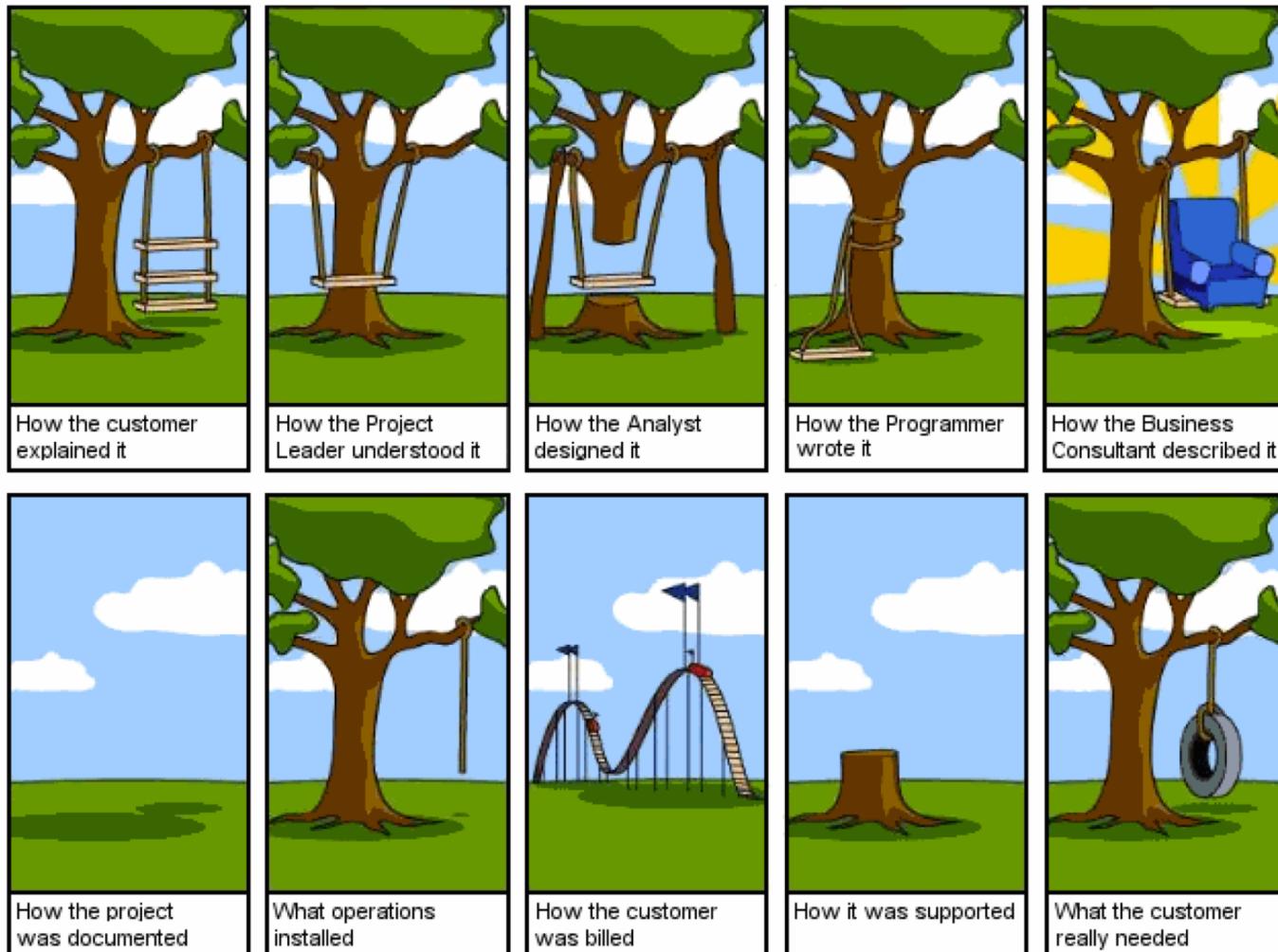
Perfil de Especialização **CSI** : Criptografia e Segurança da Informação

Engenharia de Segurança



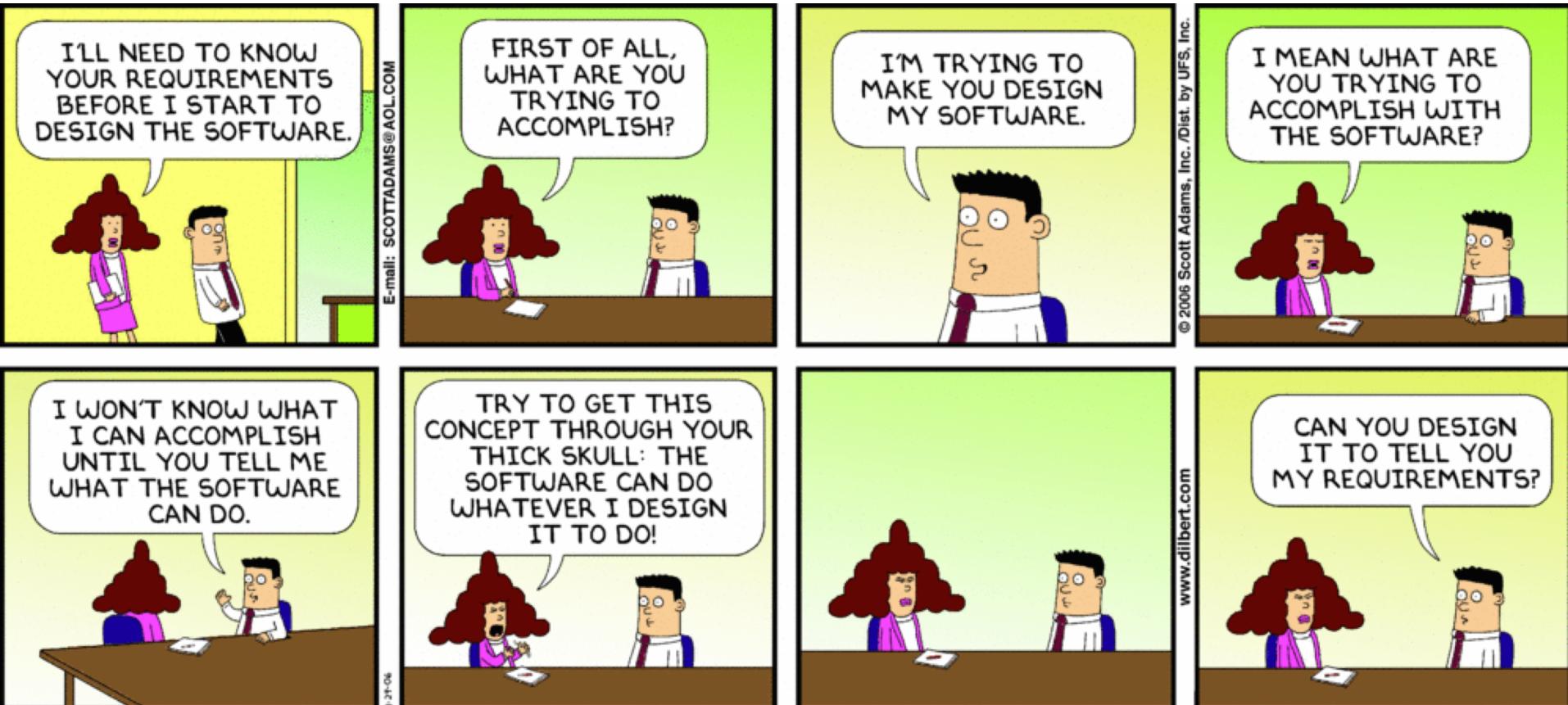
Tópicos de Segurança de Software

- *Secure Software Development Lifecycle (S-SDLC)*



Tópicos de Segurança de Software

- *Secure Software Development Lifecycle (S-SDLC)*



Tópicos de Segurança de Software

- *Secure Software Development Lifecycle (S-SDLC)*



Software Life Cycle Process

- O que é o *Software Life Cycle Process*?
 - Processo utilizado pela indústria de software para adquirir, fornecer, desenvolver, operar e manter software;
 - Tem como **objetivo** produzir um software de alta qualidade que vá de encontro ou supere as expetativas do cliente e, seja finalizado dentro da estimativa de tempo e custo;
 - **ISO/IEC/IEEE 12207:2017** (*Systems and software engineering -- Software life cycle processes*) define processos de Engenharia de Software, atividades e tarefas associados com o ciclo de vida do software desde sua conceção até a retirada/descontinuação do software, para serem adaptados de acordo com cada projeto de software.
 - **Processos fundamentais:** Aquisição, Fornecimento, Desenvolvimento, Operação, Manutenção.
 - **Processos de apoio:** Documentação, Gestão de configuração, Garantia da qualidade, Verificação, Validação, Revisão conjunta, Auditoria, Resolução de problemas, Usabilidade, Gestão de solicitação de mudanças, Avaliação do produto.
 - **Processos organizacionais:** Gestão, Infraestrutura, Melhoria, Recursos Humanos.
 - **Processos de reutilização de software:** Gestão de ativos, Gestão de programa de reutilização, Engenharia de manutenção.
 - **Processos de adaptação,** no âmbito de Projeto, Organização, Cultura, Modelo de ciclo de vida, métodos e técnicas, e linguagens.



Software Life Cycle Process

- **Processos fundamentais** (ISO/IEC/IEEE 12207:2017) do ciclo de vida do software
 - **Aquisição** – todas as atividades para obter o produto/serviço que satisfaça as necessidades da empresa contratante;
 - **Fornecimento** – desenvolvimento do plano de gestão de projeto, com as diferentes etapas/milestones, a ser utilizado no processo de desenvolvimento;
 - **Desenvolvimento** – desenho, criação e teste do produto/sistema software, resultando num produto/sistema software apto a ser entregue ao cliente;
 - **Operação** – atividades necessárias para a utilização do produto/sistema software;
 - **Manutenção** – atividades necessárias para manter o produto/sistema a funcionar.

Software Life Cycle Process

- Processos fundamentais (ISO/IEC/IEEE 12207:2017) do ciclo de vida do software
 - **Aquisição** – todas as atividades para obter o produto/serviço que satisfaça as necessidades da empresa contratante:
 - **Início e âmbito:** Descrição da necessidade de adquirir, desenvolver ou melhorar um produto/sistema software; Definição e aprovação dos requisitos do produto/sistema software; Avaliação de alternativas (compra de produto off-the-shelf, desenvolvimento específico ou melhoria de produto existente); Desenvolvimento de plano de aquisição; Definição dos critérios de aceitação do produto/sistema software;
 - **Preparação do pedido de proposta:** Contém requisitos do produto/sistema software e restrições técnicas (interoperabilidade com outros sistemas, ambiente em que vai ser utilizado, ...), etapas/milestones de entrega e critérios de aceitação;
 - **Preparação do contrato:** Desenvolvimento de procedimento de seleção dos fornecedores; Seleção dos fornecedores, baseado no procedimento de seleção; Versão inicial do contrato;
 - **Negociação de alterações:** Negociação com os fornecedores selecionados;
 - **Atualização do contrato:** Contrato é atualizado, como resultado das negociações da atividade anterior;
 - **Monitorização dos fornecedores:** Monitorização da atividade dos fornecedores, de acordo com as etapas/milestones contratadas (se necessário, trabalhar diretamente com fornecedores para garantir entregas atempadas);
 - **Aceitação e finalização:** Desenvolvimento dos testes e procedimentos de aceitação; Realização dos testes de aceitação; Realização da gestão da configuração do produto/sistema.



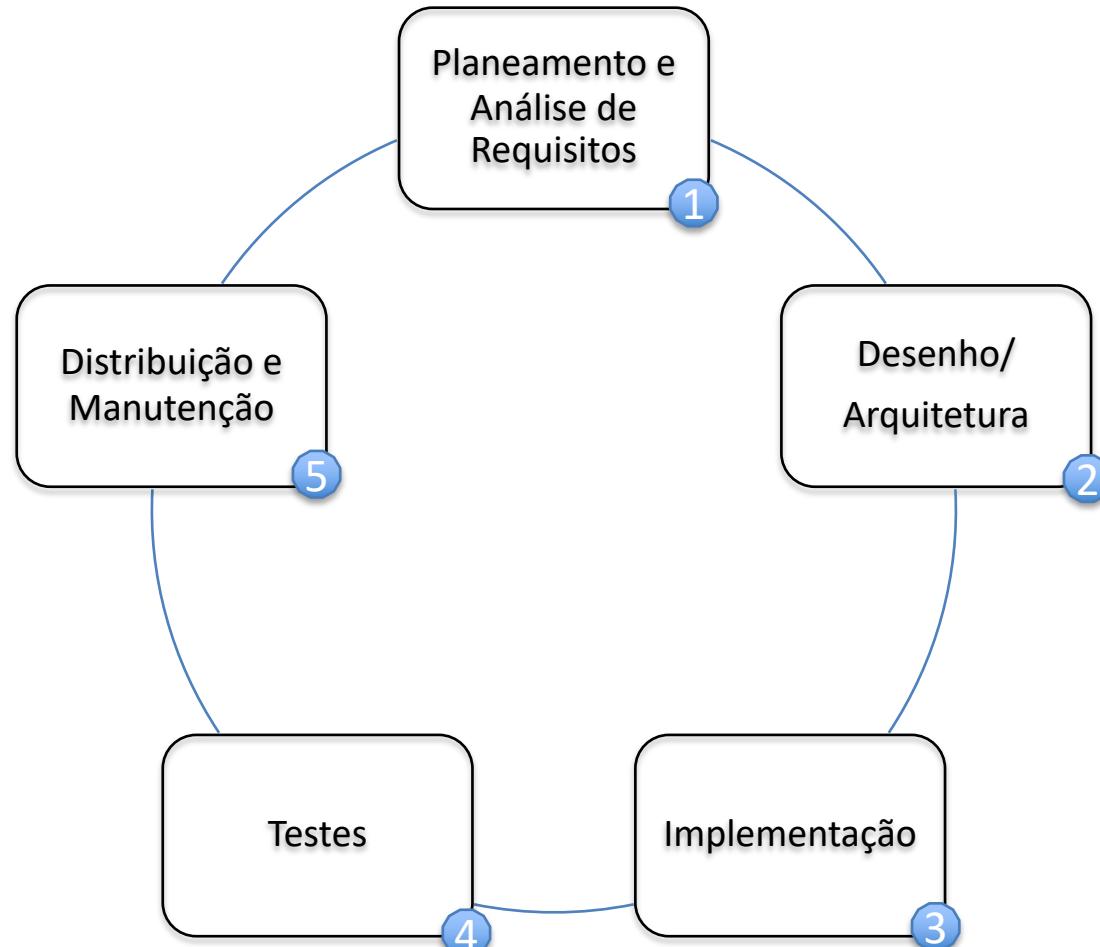
Software Life Cycle Process

- Processos fundamentais (ISO/IEC/IEEE 12207:2017) do ciclo de vida do software
 - **Desenvolvimento** – desenho, criação e teste do produto/sistema software, resultando num produto/sistema software apto a ser entregue ao cliente:
 - **Obtenção dos requisitos** – Obter e definir os requisitos e solicitações do cliente através da sua solicitação direta ou através do pedido de proposta. É imprescindível entender as expectativas do cliente e assegurar que tanto o cliente quanto o fornecedor entendam os requisitos da mesma forma. É importante gerir todas as mudanças feitas nos requisitos do cliente em relação à linha-base definida assegurando que o resultado de mudanças tecnológicas e de necessidades do cliente são identificados e os impactos de introdução dessas mudanças são avaliados.
 - **Especificação de requisitos funcionais** – Definição dos requisitos funcionais do produto/sistema software a criar (Funções e capacidades do sistema; Requisitos de negócio, organizacionais e dos utilizadores; Requisitos de segurança, de ergonomia, de interface, de operação e de manutenção; Restrições de projeto e requisitos de certificação/credenciação);
 - **Arquitetura de alto nível** – Desenho da organização básica do produto/sistema, i.e., identificação dos diferentes módulos (hardware, software e operações manuais) e como comunicam entre si (sem grande detalhe dos módulos);
 - **Desenho dos módulos** – Os diferentes módulos são desenhados separadamente, com o máximo detalhe possível;
 - **Codificação/Implementação** – Criação do código de acordo com a arquitetura de alto nível e desenho dos módulos; Execução do controle de alterações; Documentação e resolução de não-conformidades e problemas encontrados; Seleção, adaptação e utilização de standards, métodos, ferramentas e linguagens de programação.
 - **Implementação**
 - **Testes**



Software Development Lifecycle (SDLC)

Fases usuais do *Software Development Lifecycle (SDLC)*



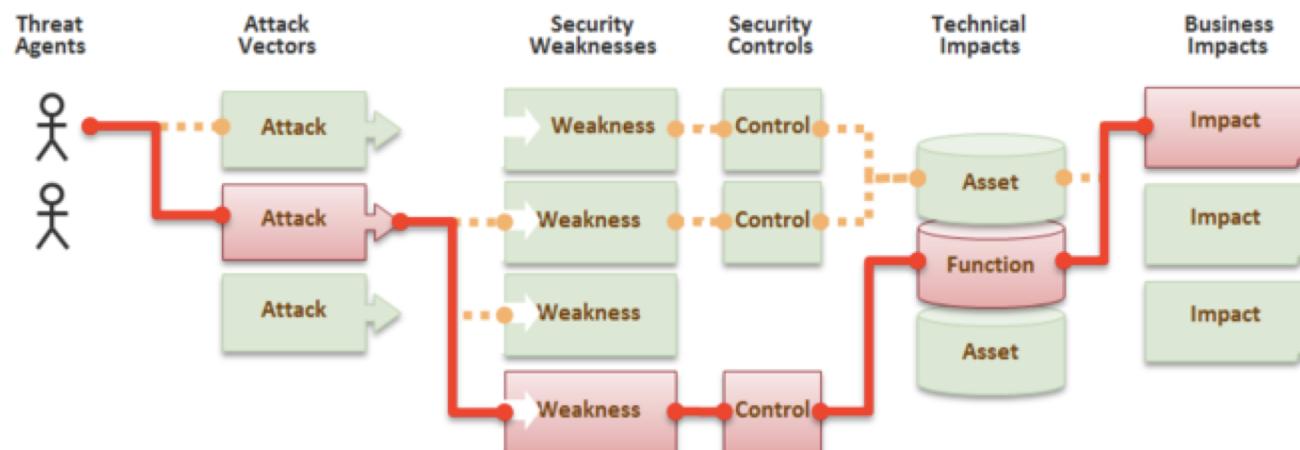
Software Development Lifecycle (SDLC)

- As falhas de segurança de software podem ser introduzidas em qualquer fase do ciclo de desenvolvimento:
 - No início, ao não identificar as necessidades de segurança
 - Na criação de arquiteturas conceptuais que possuam erros de lógica
 - No uso de más práticas de programação que introduzam vulnerabilidades técnicas
 - Na implementação do software de modo inapropriado
 - Na inserção de falhas durante a manutenção ou a atualização
- As vulnerabilidades de software podem ser resultantes de um âmbito muito maior do que o do próprio software. Entre outros:
 - As APIs de terceiros utilizadas pelo software
 - Os servidores aplicacionais (e outros) utilizados
 - O sistema operativo dos servidores associados
 - A base de dados do backend
 - Outras aplicações num ambiente partilhado
 - O sistema do utilizador
 - Outros softwares com os quais o utilizador interage

Software Development Lifecycle (SDLC)

- Factos

- Uma equipa de desenvolvimento desenvolve uma aplicação para executar tarefas específicas baseadas em requisitos funcionais e casos de uso documentados.
- Um atacante está mais interessado no que a aplicação pode ser levada a fazer e parte do princípio que "qualquer ação não expressamente proibida, é permitida".
- Os atacantes podem utilizar muitos “caminhos” para atacar o negócio ou a organização.
- Os “caminhos” podem ser triviais de encontrar e explorar ou podem ser extremamente difíceis. O dano causado pode ser nulo ou dar cabo do negócio da empresa.

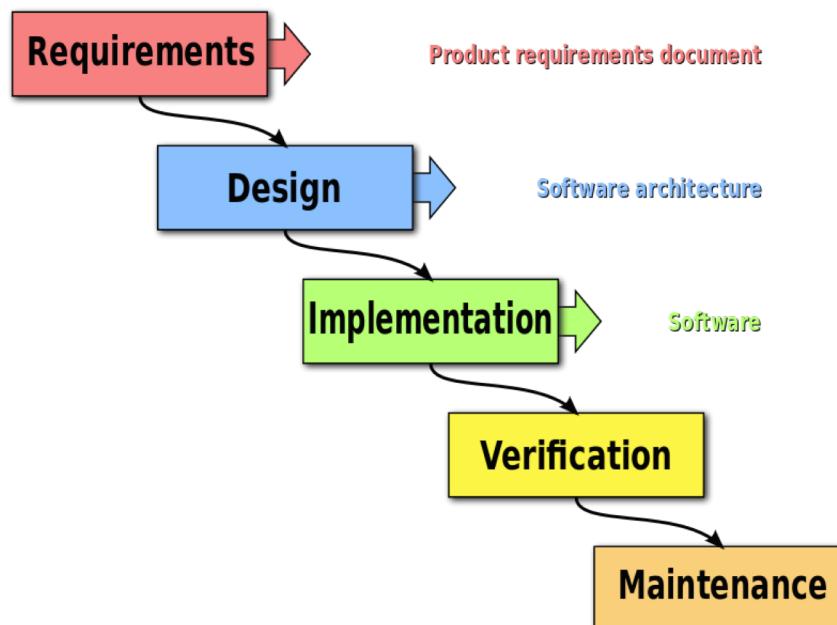


Secure Software Development Lifecycle (S-SDLC)

- Vários modos de introduzir segurança no SDLC:
 - Introduzir segurança nos modelos convencionais de desenvolvimento de software;
 - Utilizar um modelo de desenvolvimento de software seguro.
 - Através da adição de segurança utilizando um Modelo de Maturidade.

Secure Software Development Lifecycle (S-SDLC)

- Um modo de introduzir segurança no SDLC, consiste em introduzir segurança nos modelos convencionais de desenvolvimento de software.
 - Modelo em cascata (waterfall)



https://en.wikipedia.org/wiki/Waterfall_model

Secure Software Development Lifecycle (S-SDLC)

- Um modo de introduzir segurança no SDLC, consiste em introduzir segurança nos modelos convencionais de desenvolvimento de software.
 - **Modelo em cascata (waterfall)**
 - Segurança deve ser considerada nas diferentes fases do modelo
 - **Fase de Requisitos**
 - Requisitos de segurança devem ter por base a legislação em vigor (e.g., Sarbanes-Oxley Act, Health Insurance Portability and Accountability Act (HIPAA), RGPD, ...) e as recomendações e normas internacionais (família ISO 27000), conforme sejam aplicáveis, devendo ser traduzidos em requisitos específicos para o software a desenvolver.
 - **Fase de Desenho**
 - Os requisitos de segurança definidos na fase anterior são traduzidos para mecanismos que os concretizem (mecanismos de autenticação, proteção de confidencialidade e integridade de dados, ...).
 - **Fase de Implementação**
 - Programação segura, para o que é necessário conhecer as boas práticas de codificação, e devem ser usadas ferramentas de análise estática de código (ambos os assuntos, já visto nos projectos).
 - **Fase de Testes**
 - Realizar validações e testes de segurança (já visto nos projetos).
 - **Fase de Manutenção**
 - Medidas de segurança na distribuição de patches, updates/upgrades do produto,

Vulnerabilidades
de Projeto

Erros nestas fases



Vulnerabilidades
de Codificação

Erros nesta
fase



Secure Software Development Lifecycle (S-SDLC)

- Vários modos de introduzir segurança no SDLC:
 - Introduzir segurança nos modelos convencionais de desenvolvimento de software;
 - **Utilizar um modelo de desenvolvimento de software seguro.**
 - Através da adição de segurança utilizando um Modelo de Maturidade.



Secure Software Development Lifecycle (S-SDLC)

- Modelo de desenvolvimento de software seguro
 - Microsoft Security Development Lifecycle
 - **(Pré)Fase de Formação**
 - Preparação dos membros da equipa de desenvolvimento de software para lidarem com os aspetos de segurança.
 - **Fase de Requisitos**
 - Identificar as pessoas responsáveis por coordenar as questões de segurança do projeto;
 - Identificar ferramenta de seguimento de bugs, adequada à gestão de vulnerabilidades;
 - Definir requisitos mínimos de segurança, de modo análogo ao que foi definido para a fase de requisitos do modelo em cascata.
 - **Fase de Desenho**
 - Projetar os mecanismos de segurança a implementar, seguindo recomendações e melhores práticas, e evitando vulnerabilidades de projeto;
 - Efetuar análise de risco de segurança, de modo a identificar os riscos mais críticos sob o ponto de vista de segurança e privacidade.
 - **Fase de Codificação**
 - Estabelecer e seguir boas práticas de codificação;
 - Documentar o modo como os utilizadores devem configurar o software de forma segura.
 - **Fase de Verificação**
 - Objetivo é garantir que o software detém as propriedades de segurança desejadas, através de revisão manual de código, teste e, análise estática do código.
 - **Fase de Publicação/Manutenção**
 - Planejar as ações a tomar quando forem descobertas vulnerabilidades;
 - Antes da publicação deve ser efetuada uma revisão de segurança final e independente.
 - **(Pós)Fase de Resposta**
 - Recolha de informação sobre incidentes de segurança e, criação de relatórios sobre vulnerabilidades e *patches*.

Secure Software Development Lifecycle (S-SDLC)

Em qualquer um dos S-SDLC é fundamental na fase de Requisitos:

- Identificar os Requisitos de segurança
 - Requisitos de software indicam o que o software deve fazer enquanto que os requisitos de segurança indicam as expectativas de segurança na execução/utilização do software;
 - Dois tipos de requisitos de segurança:
 - Objectivos relacionados com a segurança (politica de segurança):
 - Confidencialidade (e Privacidade e Anonimato)
 - » Exemplo Política: a conta do Banco é conhecida apenas pelo seu dono legítimo
 - Integridade
 - » Exemplo Política: as operações sobre a conta têm de ser autorizadas pelo dono da conta
 - Disponibilidade
 - » Exemplo Política: o utilizador pode aceder sempre ao balanço da sua conta

Secure Software Development Lifecycle (S-SDLC)

- Identificar os Requisitos de segurança

- Requisitos de software indicam o que o software deve fazer enquanto que os requisitos de segurança indicam as expectativas de segurança na execução/utilização do software;
- Dois tipos de requisitos de segurança:
 - Objectivos relacionados com a segurança (politica de segurança):
 - Mecanismos necessários para fazer cumprir os requisitos:
 - Autenticação
 - » O que o utilizador sabe (e.g., password),
 - » O que o utilizador é (e.g., impressão digital),
 - » O que o utilizador tem (e.g., cartão de cidadão)
 - Autorização
 - » De acordo com políticas que definem acções que podem ser autorizadas (controlo de acessos, ...)
 - Auditabilidade
 - » Guardar informação suficiente para determinar as circunstâncias de determinada acção (normalmente em logs, protegidos contra falsificação)

Secure Software Development Lifecycle (S-SDLC)

- Identificar os Requisitos de segurança
 - Quais os requisitos de segurança que têm de ser seguidos?
 - Baseados em legislação aplicável ao negócio;
 - Baseados em standards;
 - Baseados nos valores e/ou políticas da empresa;
 - Baseados nos modelos de risco;
 - Baseados nos casos de abuso.



Secure Software Development Lifecycle (S-SDLC)

Em qualquer um dos S-SDLC é fundamental na fase de Requisitos:

- Identificar os Casos de Abuso

- Ilustram Requisitos de Segurança;
- Descrevem o que **não** deve ser feito e/ou permitido;
- Partindo de padrões de ataque e cenários prováveis, construir casos de abuso nos quais um atacante pode violar requisitos de segurança.
 - Por exemplo, um atacante rouba o ficheiro de passwords e obtém todas as passwords dos utilizadores
 - Outro exemplo, um atacante repete uma mensagem de transferência de dinheiro

Secure Software Development Lifecycle (S-SDLC)

Em qualquer um dos S-SDLC é fundamental na fase de Desenho:

- Análise de Riscos da Arquitectura

- Processo de avaliação do Risco de uma falha de segurança, com base na probabilidade e custo de vários tipos de ataques, e fazendo o possível para minimizar esse risco;
- Baseada em modelos de risco/ameaça:
 - **Atacante na rede** – utilizador anónimo que pode aceder ao serviço através da rede – pode:
 - Medir o tamanho dos pedidos e o tempo das repostas;
 - Executar sessões paralelas;
 - Fornecer inputs e mensagens mal formatadas;
 - Remover ou enviar pacotes extra de rede;
 - **Atacante na mesma rede** que o utilizador de determinado serviço – por exemplo, ligado numa rede Wi-Fi insegura, num centro comercial –, que em adição pode:
 - Ler e medir mensagens de terceiros;
 - Interceptar, duplicar e modificar mensagens;

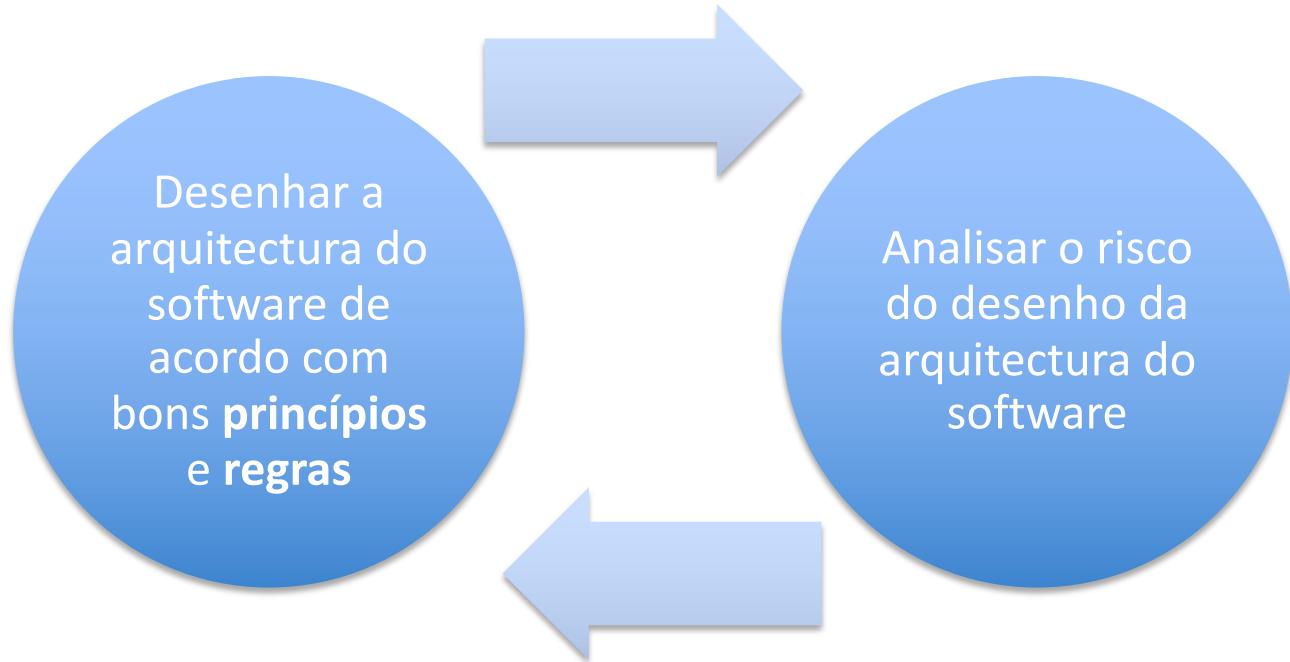
Secure Software Development Lifecycle (S-SDLC)

- Análise de Riscos da Arquitectura
 - Baseada em modelos de risco/ameaça:
 - **Atacante na mesma máquina** que o utilizador de determinado serviço – por exemplo, malware instalado no portátil –, que em adição pode:
 - Ler e escrever ficheiros (incluindo cookies) e memória da máquina do utilizador;
 - Espiar o que é escrito e outros eventos;
 - Ler e escrever no ecrã do utilizador.
 - Diferente modelo de risco considerado leva a diferente tipo de respostas na análise e desenho do software;
 - A **essência da análise de risco** é ter em consideração os **custos** potenciais do desenvolvimento da resposta a determinado tipo de risco, os **custos** de uma quebra de segurança com sucesso e, pesar a **probabilidade** de cada.

Secure Software Development Lifecycle (S-SDLC)

Em qualquer um dos S-SDLC é fundamental na fase de Desenho:

- Desenho orientado à segurança (*security by design*)
 - Processo iterativo.



Secure Software Development Lifecycle (S-SDLC)

- Desenho orientado à segurança

- Categorias de Princípios:

- Prevenção – eliminar totalmente os defeitos do software;
 - Diminuição de danos – diminuir os danos da exploração de defeitos desconhecidos;
 - Detecção (e recuperação) – identificar e perceber um ataque (e desfazer os danos).

- Princípios:

- Favorecer a simplicidade
 - Utilizar escolhas por omissão à prova de falhas
 - Não esperar que os clientes sejam especialistas
 - Confiança com relutância
 - Conceder o privilégio mínimo (“*need to know*”)
 - Defesa em profundidade (*Defend in depth*)
 - Evitar a dependência de apenas um mecanismo de defesa (*security by diversity*);
 - No security by obscurity
 - Manter-se actualizado em relação às ameaças e pesquisas mais recentes
 - Monitorização e Rastreabilidade
 - Detectar e diagnosticar violações e/ou ataques de segurança

Secure Software Development Lifecycle (S-SDLC)

- Vários modos de introduzir segurança no SDLC:
 - Introduzir segurança nos modelos convencionais de desenvolvimento de software;
 - Utilizar um modelo de desenvolvimento de software seguro (S-SDLC).
 - **Através da adição de segurança utilizando um Modelo de Maturidade.**



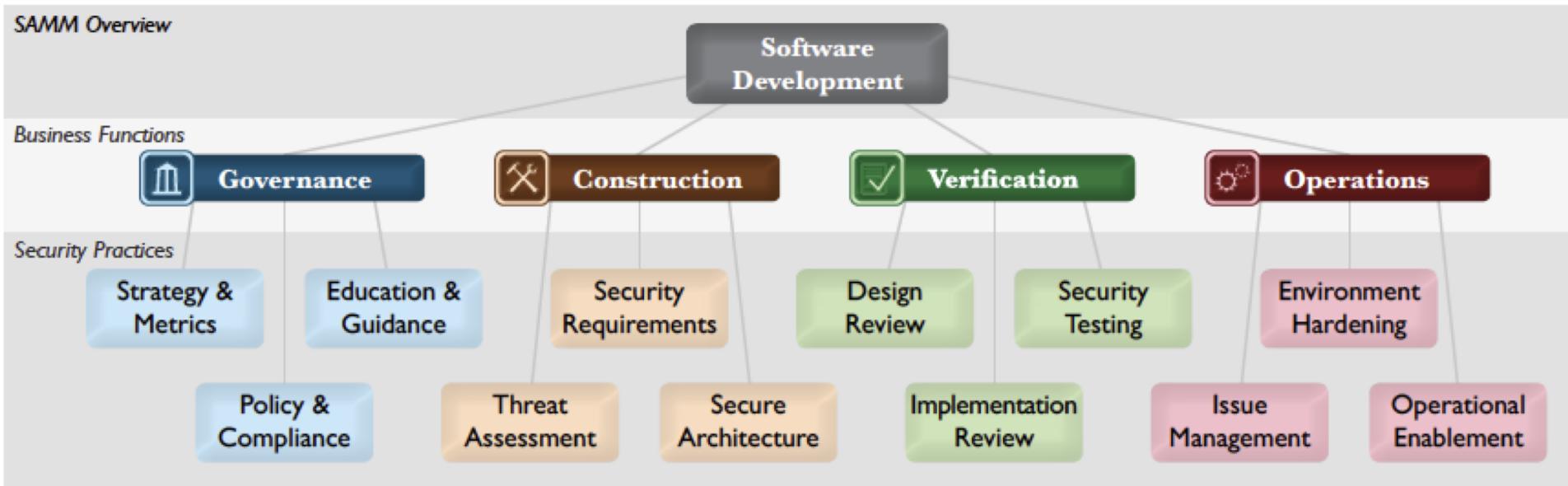
Secure Software Development Lifecycle (S-SDLC)

- Outro modo ainda, é através da adição de segurança utilizando um Modelo de Maturidade.
 - O comportamento das organizações muda lentamente, pelo que as mudanças devem ser iterativas, de acordo com os objetivos de longo prazo (desenvolvimento de software seguro);
 - Não existe uma receita que funcione para todas as organizações, pelo que a organização deve poder escolher os riscos que pretende correr;
 - Necessário avaliar as práticas de segurança de software existentes na organização e, criar um programa de segurança de software que permita atingir os seus objetivos em iterações bem definidas;
 - O guia das atividades de segurança a considerar no desenvolvimento de software deve fornecer detalhes suficientes para pessoas externas à área da segurança;
 - De um modo geral, o modelo a utilizar para o desenvolvimento de software seguro deve ser simples, bem definido e mensurável, assim como integrável no SDLC da organização.
- Existem vários Modelos de Maturidade:
 - *Microsoft SDL Optimization Model*
 - *Building Security In Maturity Model (BSIMM)*
 - *Software Assurance Maturity Model (SAMM)*,

Secure Software Development Lifecycle (S-SDLC)

- *Software Assurance Maturity Model (SAMM)*

- Objetivo é ajudar as organizações (quer desenvolva, faça outsourcing ou adquira software) a avaliar, formular e implementar uma estratégia para segurança de software, que possa ser integrada no SDLC utilizado.
- Descrição do Modelo de Maturidade:
 - Baseado em 12 **práticas de segurança**, que estão agrupadas em 4 **funções de negócio**;
 - Cada prática de segurança contém um conjunto de atividades, estruturadas em três **níveis de maturidade**;
 - As atividades num nível inferior de maturidade são mais facilmente executáveis e requerem menos formalização que as atividades num nível mais alto de maturidade.

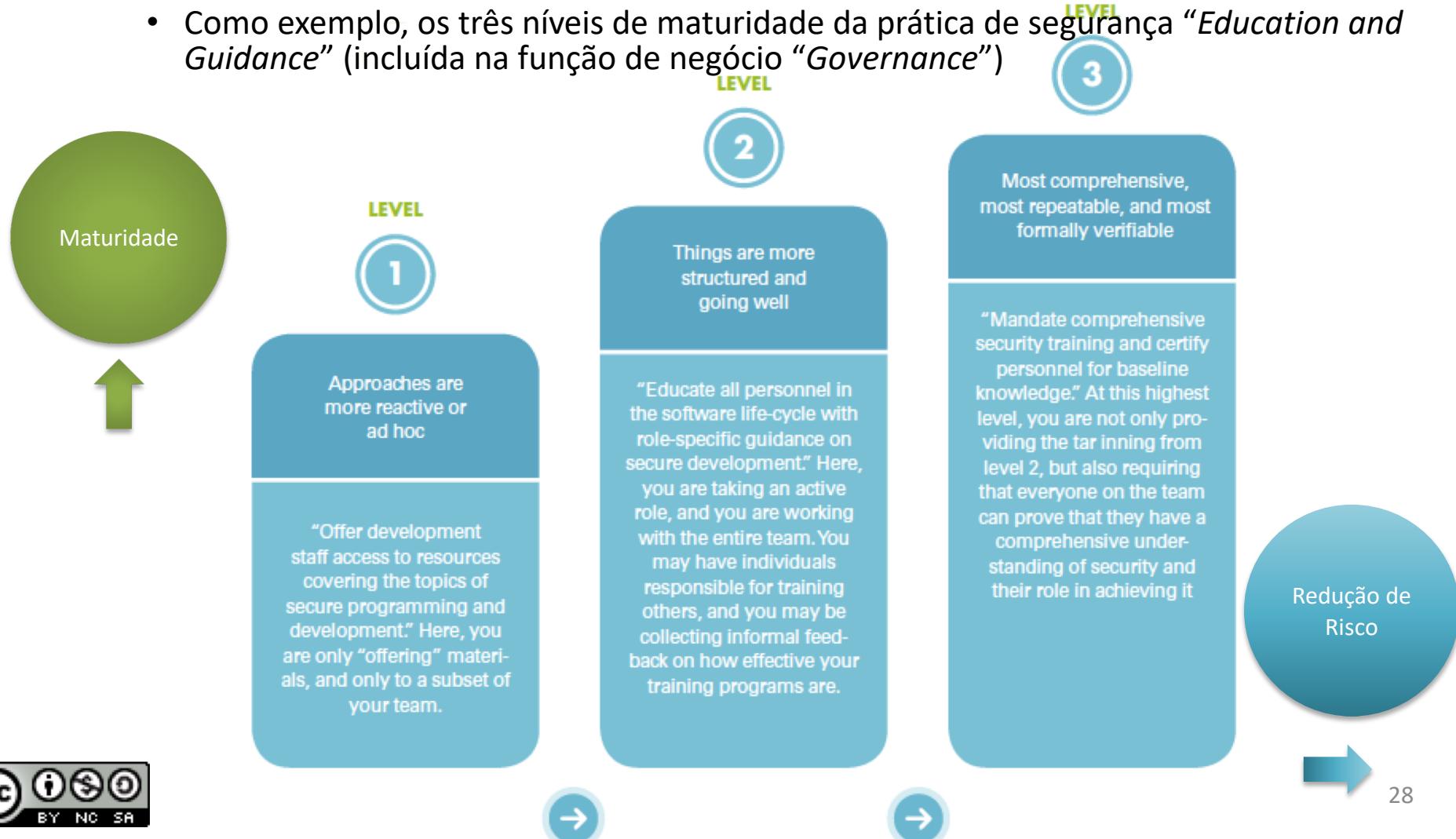


Secure Software Development Lifecycle (S-SDLC)

- *Software Assurance Maturity Model (SAMM)*

- Descrição do Modelo de Maturidade:

- Como exemplo, os três níveis de maturidade da prática de segurança “*Education and Guidance*” (incluída na função de negócio “*Governance*”)



Secure Software Development Lifecycle (S-SDLC)

- *Software Assurance Maturity Model (SAMM)*
 - Valor do SAMM está em fornecer um mecanismo para saber onde a organização se encontra no seu caminho em direção à segurança de software e, permitir entender o que é recomendado para passar para um nível de maturidade superior.
 - O SAMM não insiste em que todas as organizações alcancem o nível de maturidade 3 em todas as práticas de segurança. Cada organização determina o nível de maturidade de cada “Prática de segurança” que melhor se adapta às suas necessidades.
 - A configuração e o ciclo do modelo de maturidade do SAMM são feitas para permitir:
 - i. a avaliação das práticas atuais de segurança de software,
 - ii. a definição do objetivo que melhor se adapta à organização,
 - iii. a formulação do caminho para atingir o objetivo definido (de forma iterativa), e
 - iv. a implementação de actividades específicas das práticas de segurança na maturidade adequada para a empresa, seguindo medidas descriptivas.

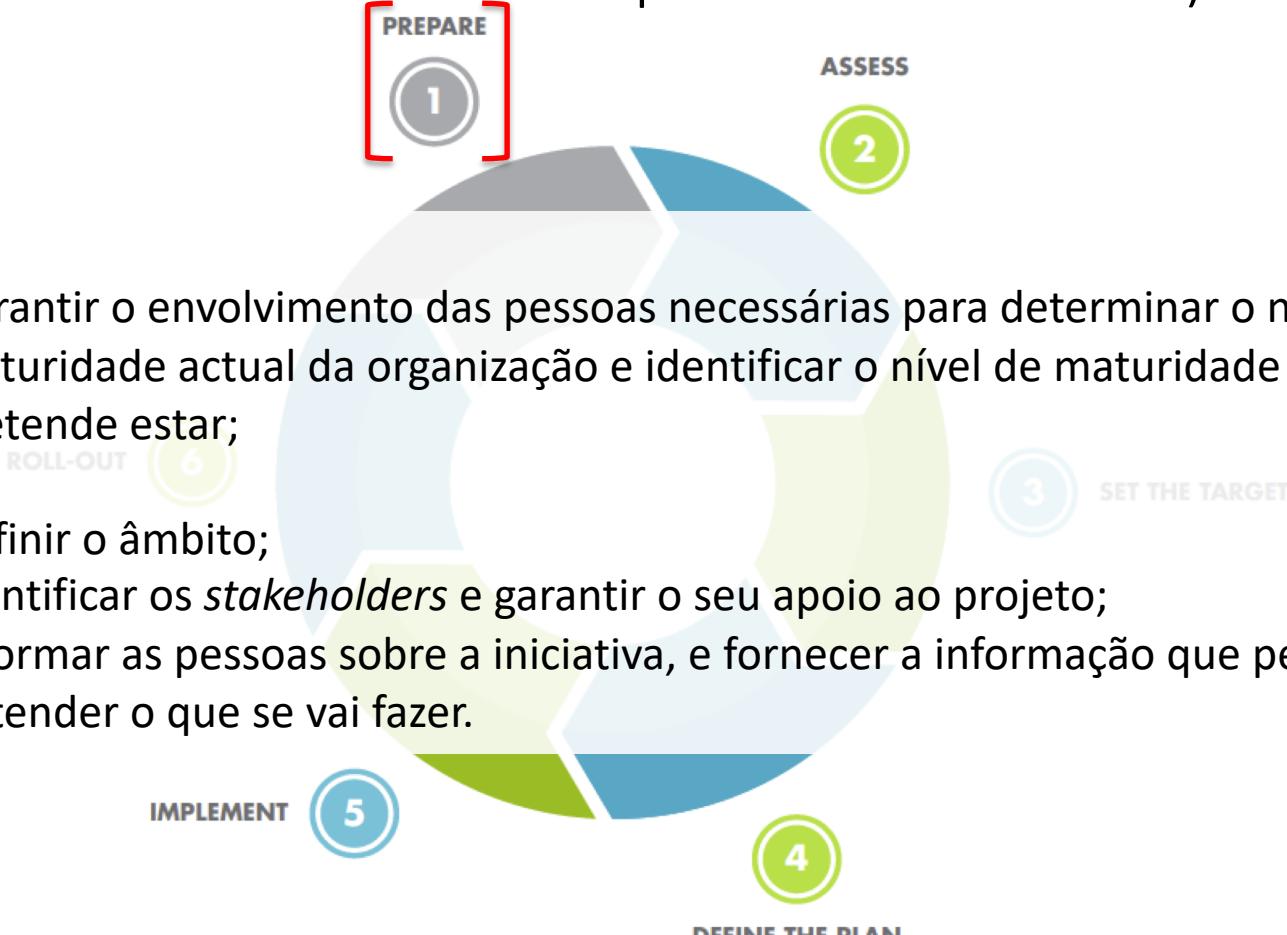
Secure Software Development Lifecycle (S-SDLC)

- *Software Assurance Maturity Model (SAMM)*
 - Ciclo do SAMM, adequado a melhoria contínua (caso em que o ciclo deve ser executado continuamente em períodos de 3 a 12 meses)



Secure Software Development Lifecycle (S-SDLC)

- *Software Assurance Maturity Model (SAMM)*
 - Ciclo do SAMM, adequado a melhoria contínua (caso em que o ciclo deve ser executado continuamente em períodos de 3 a 12 meses)



Objetivo:

- Garantir o envolvimento das pessoas necessárias para determinar o nível de maturidade actual da organização e identificar o nível de maturidade em que pretende estar;

Atividades:

- Definir o âmbito;
- Identificar os *stakeholders* e garantir o seu apoio ao projeto;
- Informar as pessoas sobre a iniciativa, e fornecer a informação que permita entender o que se vai fazer.

Secure Software Development Lifecycle (S-SDLC)

- *Software Assurance Maturity Model (SAMM)*
 - Ciclo do SAMM, adequado a melhoria contínua (caso em que o ciclo deve ser executado continuamente em períodos de 3 a 12 meses)

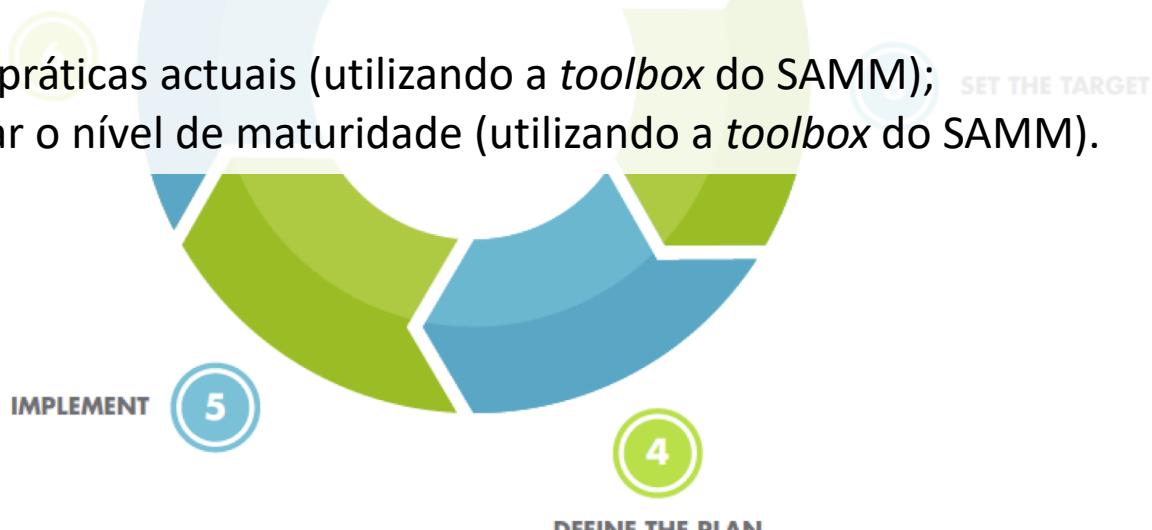


Objetivo:

- Identificar e compreender a maturidade da organização em cada uma das 12 práticas de segurança;

Atividades:

- Avaliar as práticas actuais (utilizando a *toolbox* do SAMM);
- Determinar o nível de maturidade (utilizando a *toolbox* do SAMM).



Secure Software Development Lifecycle (S-SDLC)

- *Software Assurance Maturity Model (SAMM)*
 - Ciclo do SAMM, adequado a melhoria contínua (caso em que o ciclo deve ser executado continuamente em períodos de 3 a 12 meses)



Objetivo:

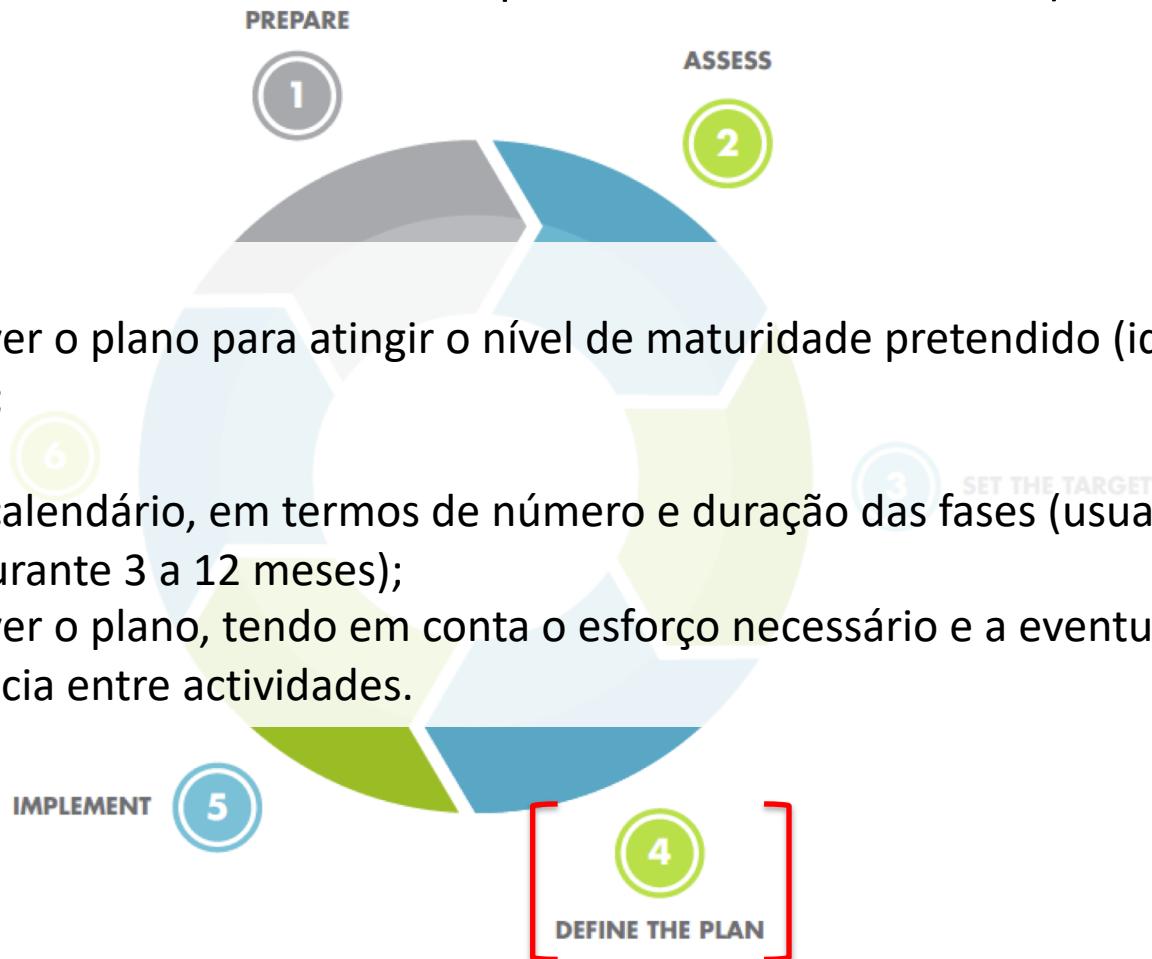
- Identificar um *score* objetivo para cada uma das 12 práticas de segurança, que sirva de guia para atuar sobre as actividades mais importantes;

Atividades:

- Definir o objetivo (utilizando a *toolbox* do SAMM);
- Estimar o impacto do objetivo na organização (em termos financeiros, se possível).

Secure Software Development Lifecycle (S-SDLC)

- *Software Assurance Maturity Model (SAMM)*
 - Ciclo do SAMM, adequado a melhoria contínua (caso em que o ciclo deve ser executado continuamente em períodos de 3 a 12 meses)



Objetivo:

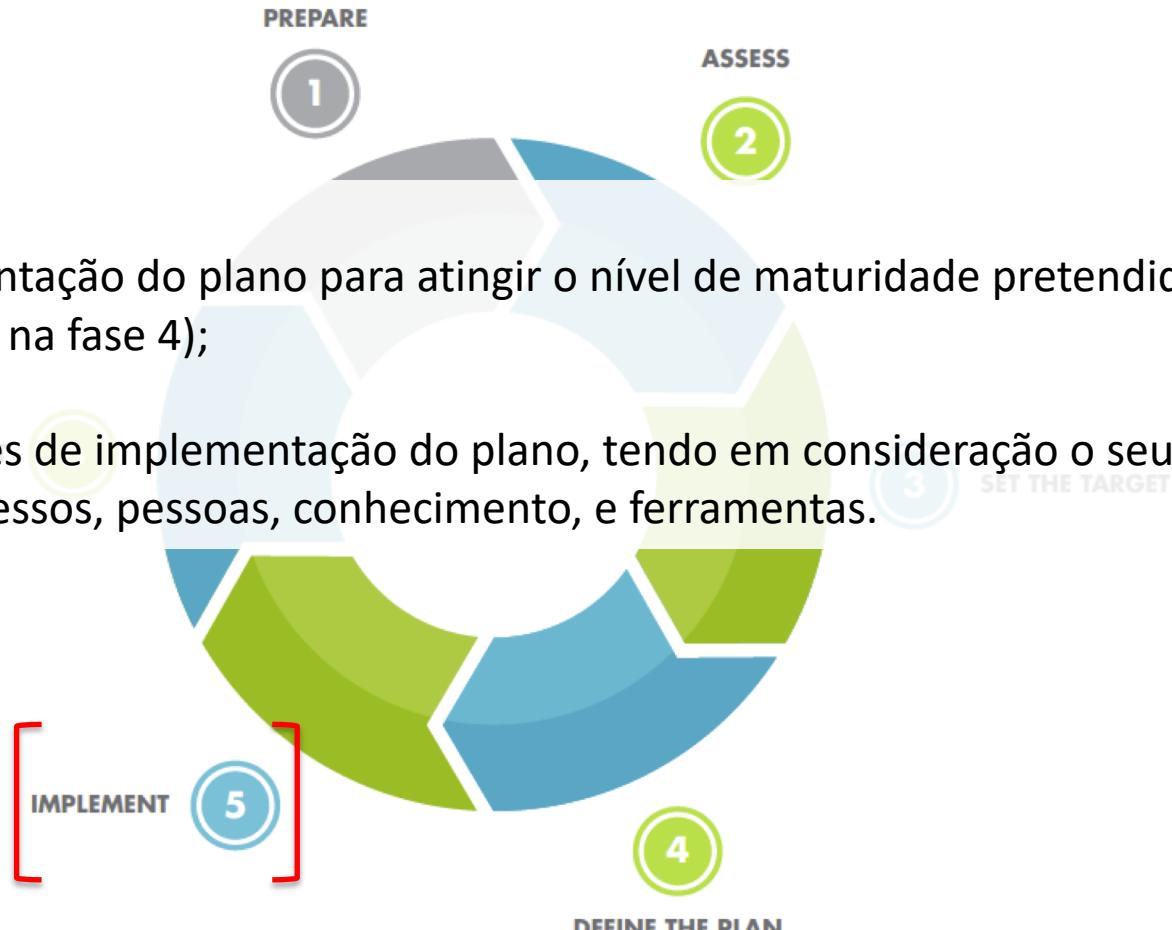
- Desenvolver o plano para atingir o nível de maturidade pretendido (identificado na fase 3);

Atividades: **ROLL-OUT**

- Definir o calendário, em termos de número e duração das fases (usualmente 4 a 6 fases, durante 3 a 12 meses);
- Desenvolver o plano, tendo em conta o esforço necessário e a eventual dependência entre actividades.

Secure Software Development Lifecycle (S-SDLC)

- *Software Assurance Maturity Model (SAMM)*
 - Ciclo do SAMM, adequado a melhoria contínua (caso em que o ciclo deve ser executado continuamente em períodos de 3 a 12 meses)



Objetivo:

- Implementação do plano para atingir o nível de maturidade pretendido (definido na fase 4);

Atividades:

- Atividades de implementação do plano, tendo em consideração o seu impacto nos processos, pessoas, conhecimento, e ferramentas.

Secure Software Development Lifecycle (S-SDLC)

- *Software Assurance Maturity Model (SAMM)*
 - Ciclo do SAMM, adequado a melhoria contínua (caso em que o ciclo deve ser executado continuamente em períodos de 3 a 12 meses)

