Tópicos Especiais em Arquitetura de Software

Por Sérgio Crespo
Email: screspo@id.uff.br
http://www.professores.uff.br/screspo
Twitter: @screspo

Objetivos

Esta disciplina tem por objetivo estudar a teoria que envolve a criação de arquiteturas de software, enfatizando modelos, artefatos e linguagens de descrição de arquiteturas e fazer experimentos teóricos usando a UML-Components como forma de representação de arquiteturas.

Ementa

Conceitos e modelos de arquitetura de software, revisão de UML, reuso de software, pipes and filters, layers, blackboards, broker, model-view-controller, visões de arquitetura, pipelines, componentes, Linguagens de descrição de arquiteturas, model drive archtecture, service oriented architecture, sistemas móveis, agentes, interpretadores, Agile Software Architecture e arquiteturas baseadas em ontologias.

Bibliografia

- An Introduction to Software Architecture, David Garlan and Mary Shaw, 1994.

(http://www.cs.cmu.edu/afs/cs/project/able/ftp/intro_softarch/intro_softarch.pdf)

- Architecting for Large Scale Agile Software Development: A Risk-Driven Approach, Ipek Ozkaya, Michael Gagliardi, Robert L. Nord, 2013. (http://resources.sei.cmu.edu/asset_files/Article/2013_101_001_87865.pdf)

- Quality Attributes and Service-Oriented Architectures, Liam O'Brien,Len Bass and Paulo Merson, 2005.

(http://resources.sei.cmu.edu/asset_files/TechnicalNote/2005_004_001_14489.pdf)

Bibliografia

- Evaluating a Service-Oriented Architecture, Software Engineering Institute, 2007.

(http://resources.sei.cmu.edu/asset_files/TechnicalReport/2007_005_001_14894.pdf)

- Agile at Scale (AAS) - SPRUCE / SEI

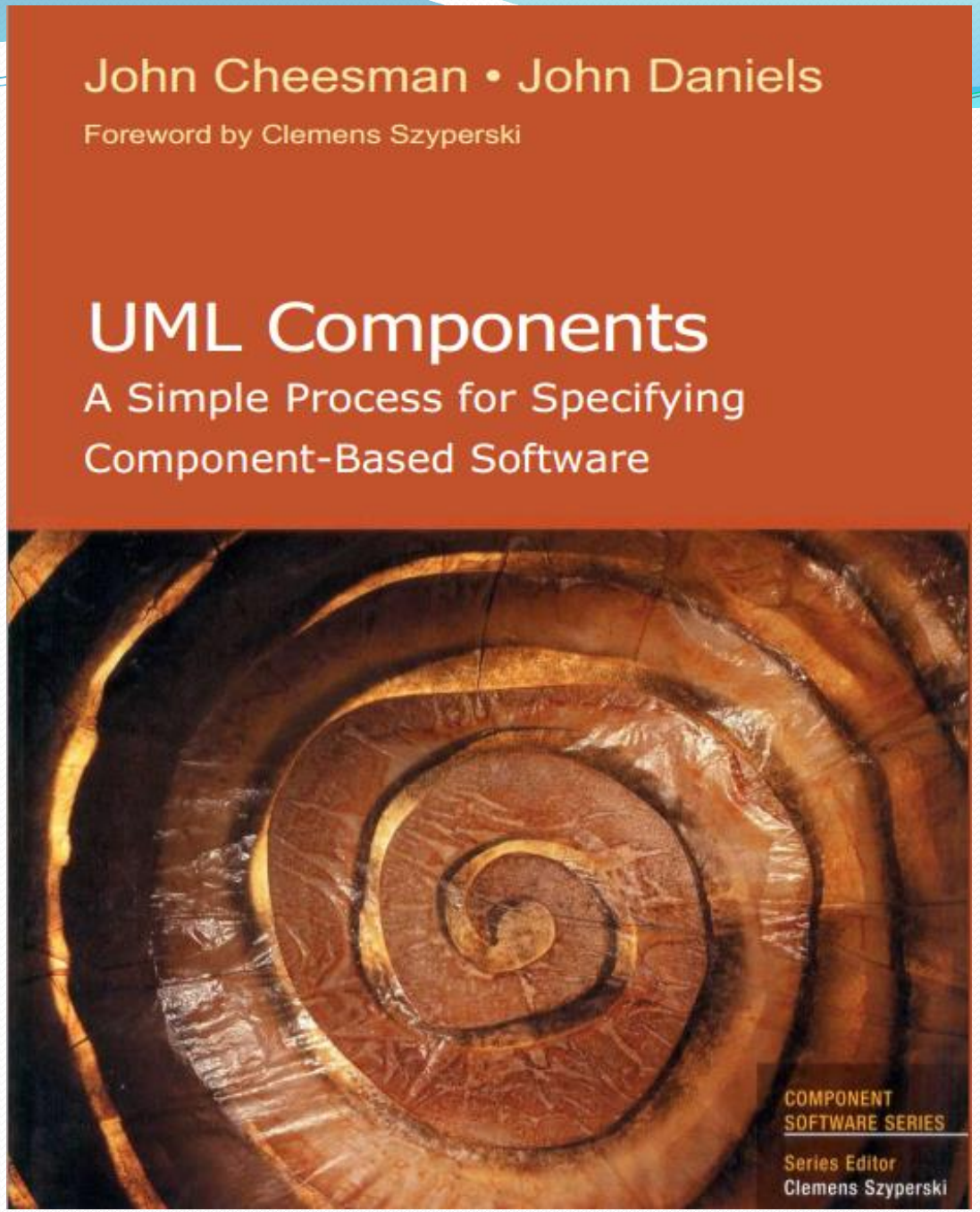
(https://www.csiac.org/spruce/resources/ref_documents/agile-scale-aas-spruce-sei)

- Making Architecture Visible to Improve Flow Management in Lean Software Development, Robert L. Nord and Ipek, zkaya, Raghvinder S. Sangwan, 2012.

(http://resources.sei.cmu.edu/asset_files/Article/2012_101_001_58828.pdf)

Bibliografia

- JOHN Chessman, John Daniels. UML Components – A Simple Process For Specifyng Component-Based Software, Addison-Wesley, 2000.
- MENDES, Antônio. Arquitetura de Software. Campus-Elsevier, 2002, (ISBN 853521013X)
- BASS, Len; CLEMENTS, Paul. Software Architecture in Practice. 2ed., 2003. (ISBN 0321154959)
- SHANK, Mary; GULAN, David. Software Architecture. 1996. (ISBN 0131829572)


John Cheesman • John Daniels
Foreword by Clemens Szyperski
UML Components
A Simple Process for Specifying
Component-Based Software
COMPONENT
SOFTWARE SERIES
Series Editor
Clemens Szyperski

Introduction MOTIVAÇÃO

Software architecture is an area of growing importance to practitioners and researchers in government, industry, and academia. The April 1995 issue of *IEEE Transactions on Software Engineering* and the November 1995 issue of *IEEE Software* were devoted to software architecture. Industry and government working groups on software architecture are becoming more frequent. Workshops and presentations on software architecture are beginning to populate software engineering conferences. There is an emerging software architecture research community, meeting and collaborating at special-purpose workshops such as the February 1995 International Workshop on Software Architectures held in Dagstuhl, Germany, or the April 1995 International Workshop on Architectures for Software Systems held in Seattle, Washington. The October 1996 ACM Symposium on the Foundations of Software Engineering will focus on software architecture. Textbooks devoted entirely to software architecture are appearing, such as the one by Shaw and Garlan [Shaw 95b]. The government is investing in the development of software architectures as core products in their own right; the Technical Architecture Framework for Information Management (TAFIM) is an example. The Common Object Request Broker Architecture (CORBA) and other computer-assisted software engineering environments with emphasis on architecture-based development are entering the marketplace with profound effect.

Em realidade…

Roots of Software Architecture

The study of software architecture is in large part a study of software structure that began in 1968 when Edsger Dijkstra pointed out that it pays to be concerned with how software is partitioned and structured, as opposed to simply programming so as to produce a correct result [Dijkstra 68]. Dijkstra was writing about an operating system, and first put forth the notion of a layered structure, in which programs were grouped into layers, and programs in one layer could only communicate with programs in adjoining layers. Dijkstra pointed out the elegant conceptual integrity exhibited by such an organization, with the resulting gains in development and maintenance ease.

David Parnas pressed this line of observation with his contributions concerning information-hiding modules [Parnas 72], software structures [Parnas 74], and program families [Parnas 76].

Arquitetura…

- Um ponto crítico no projeto e na construção de todo o sistema de software é sua arquitetura: isto é, sua organização bruta como uma coleção de componentes de interação.

- Uma boa arquitetura pode permitir que um sistema satisfaça às exigências chaves em áreas como: o desempenho, a confiabilidade, a portabilidade, a escalabilidade, e a sua interoperabilidade.

No passado...

- No Passado, a arquitetura recebeu uma crescente atenção como uma sub-área importante da Engenharia de Software. Os especialistas pregam que uma boa arquitetura é um fator crítico de sucesso para o projeto e o desenvolvimento do sistema. Começaram a reconhecer o valor de fazer escolhas arquiteturais explícitas.

The Many Roles of Software Architecture

People often make analogies to other uses of the word architecture about which they have some intuition. They commonly associate architecture with physical structure (building, streets, hardware) and physical arrangement. A building architect has a perspective of architecture that is driven by the need to design a building that as an entity addresses needs and requirements including accessibility, aesthetics, light, maintainability, etc. [Alexander 77]. A software architect has a perspective that is driven by the need to design a system that addresses needs such as concurrency, portability, evolvability, usability, security, etc. Analogies between buildings and software systems should not be taken literally—they break down fairly soon—but rather used to help understand that perspective is important and structure can have different meanings depending upon the motivation for examining structure. What to glean from this discussion is that a precise definition of software architecture is not nearly as important as the concept and what its investigation allows us to do.

Software architecture usually refers to some combination of structural views of a system, with each view a legitimate abstraction of the system with respect to certain criteria, that facilitates a particular type of planning or analysis. This relatively simple concept has been co-opted by a wide variety of stakeholders and participants in software development; architecture has become a concept that represents many things to many people. In subsequent sections, we will explore some of these views and viewpoints.

Fases

Fase1- Pesquisa básica: 1985-1994

- Nesta fase os projetistas descreviam suas estruturas utilizando-se de **diagrams compostos de caixa-e-setas e notas informais**. Experientes projetistas reconheciam a **fragilidade** da metodologia utilizada de forma **ad hoc**. Estas estruturas algumas vezes eram chamadas de **arquitetura** embora nem sempre tivessem o seu **processo realizado** de forma **sistemática**.
- Em meados dos anos 80, várias ideias se firmaram tais como: ocultamento da informação, tipos abstratos de dados, orientação a objetos e herança. Estas técnicas contribuiram para a ideia de considerar softwares como caixas pretas. A orientação a objetos foi importante na fixaçao deste conceito.
- Outras qualidades como dependabilidade e manutenibilidade foram discutidas.

Fases

Fase1- Pesquisa básica: 1985-1994

- Desenvolvedores iniciavam a exploração das ideias de **especialização do software** para **problemas específicos**. Alguns trabalhos focaram em *estruturas de software* para produtos particulares com: aviação, osciloscópios e controle de misseis.
- Paralelamente, iniciaram a catalogação de soluções estruturais comuns a diversas áreas e a ideia de arquitetura tomou forma. Estruturas como pipe-filter, repositórios, invocação implícita, processos cooperativos ajudaram a identificar arquiteturas para classes de problemas específicos e desta forma permitir uma melhor formalização da solução.

Fases

Fase2: Formalização do conceito: 1992-1996

- Modelos básicos foram elaborados e explorados largamente através da descrição de linguagens de arquiteturas, recentemente formalizadas e classificadas. A idéia principal estava centrada nas estruturas do software que eram comumente encontradas em outros sistemas. Os estudos nesta época ajudaram a organizar os princípios das boas práticas.
- As Principais linguagens de descrição de arquiteturas (ADLs) eram:
  - **Aesop** (explorava propriedades específicas de estilos arquiteturais),
  - **C2** (explorava o poder de estilos baseados em eventos),
  - **Darwin** (projeto e especificação de sistemas distribuidos),
  - **Meta-H** (projetos de controles em tempo real para aviação),
  - **Rapide** (simulação e analise de sistemas com comportamento dinamico),
  - **UniCon** (uso de conectores), e
  - **Wright** (interação entre componentes).
- O importante desta fase foi a conscientização do conceito de **visão arquitetural** como um conceito de trabalho.

Fases

Fase3: Development and extension phase: 1995-2000

- Durante esta fase, ocorre a troca do foco para a unificação e refinamento dos conceitos iniciais. O refinamento das taxonomias dos elementos arquiteturais e linguagens também ocorreu.
- Ocorre uma maturidade dos institutos de pesquisa, pesquisadores e desenvolvedores. O IEEE - *Transactions on Software Engineering, em 1995 lança um número especial no tema software architecture*.

Fases

Fase4: Aumento do uso interno do conceito e exploraçaõ:1996-2003

- O conceito *estilos arquiteturais*, nesta fase, passou a ser conhecido como ***padrões arquiteturais***.
- **Avaliação e análise de arquiteturas** tomam corpo como tópicos de pesquisa.
- Trabalha-se na ideia de construção de *ferramentas case* para **automatizar** o processo de **contrução** de arquiteturas.

Fases

Fase5: Aumento do uso externo do conceito e exploraçaõ:1998-present

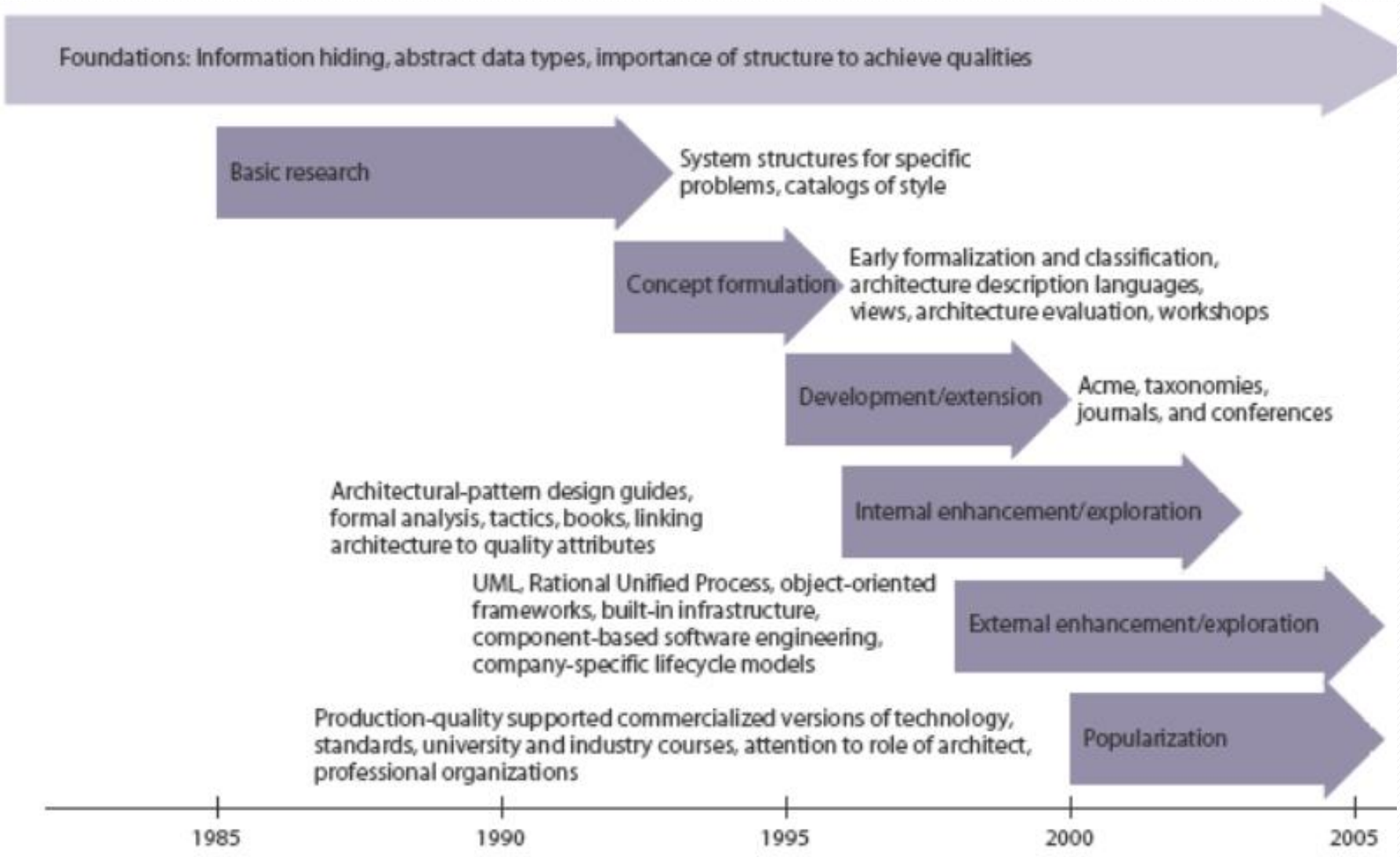
- Ocorre a maturidade do conceito e com isso a sua externalização (indústria, governo, etc).
- Uso da UML como um padrão para a descrição de arquiteturas, junto com o pacote da *Rational*.

Fases

Fase6: Popularização: 2000-present

- A popularização foi caracterizada pela produção qualificada, suporte,comercialização e marketing de produtos para a comunidade interna e externa.
- Padrões arquiteturias foram fortemente utilizados com a explosão da **W**orld **W**ide **W**eb e web-based e-commerce. Tecnologias como *N-tier client-server architectures*, *agent-based architectures*, e *Service-Oriented Architectures* ajudaram na fixação e uso do conceito.

Resumo: Fases



Definições

Sistema de Computação

- **UML 1.3**: A system is a collection of connected units that are organized to accomplish a specific purpose. A system can be described by one or more models, possibly from different viewpoints.
- **IEEE Std. 610.12-1990:** A system is a collection of components organized to accomplish a specific function or set of functions.

Arquitetura de software

- **UML 1.3:** Architecture is the organizational structure of a system. An architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems.

Bass, Clements, and Kazman. Software Architecture in Practice, Addison-Wesley 1997:

- 'The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.

Definições

Garlan and Perry, guest editorial to the *IEEE Transactions on Software Engineering*, April 1995:

- Software architecture is "the structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time."

Dewayne E. Perry and Alexander L. Wolf. "Foundations for the Study of Software Architecture". *ACM SIGSOFT Software Engineering Notes*, 17:4, October 1992:

- "... software architecture is a set of architectural (or, if you will, design) elements that have a particular form.
- We distinguish three different classes of architectural element:
  - processing elements;
  - data elements; and
  - connecting elements."

IEEE Std. 610.12-1990:

- Architecture is the organizational structure of a system.

Definições

- Apesar de existirem numerosas definições sobre arquitetura do software, no núcleo de tudo está a **noção de que a arquitetura de um sistema descreve sua estrutura bruta**.
- Esta estrutura ilumina as decisões de projeto de alto nível, incluindo coisas como:
  - o sistema é composto das peças de interação, onde estão os principais **caminhos de interação**
  - Adicionalmente, uma descrição arquitetural inclui informação suficiente para **permitir** a **análise de alto nível** e **crítica do sistema**.

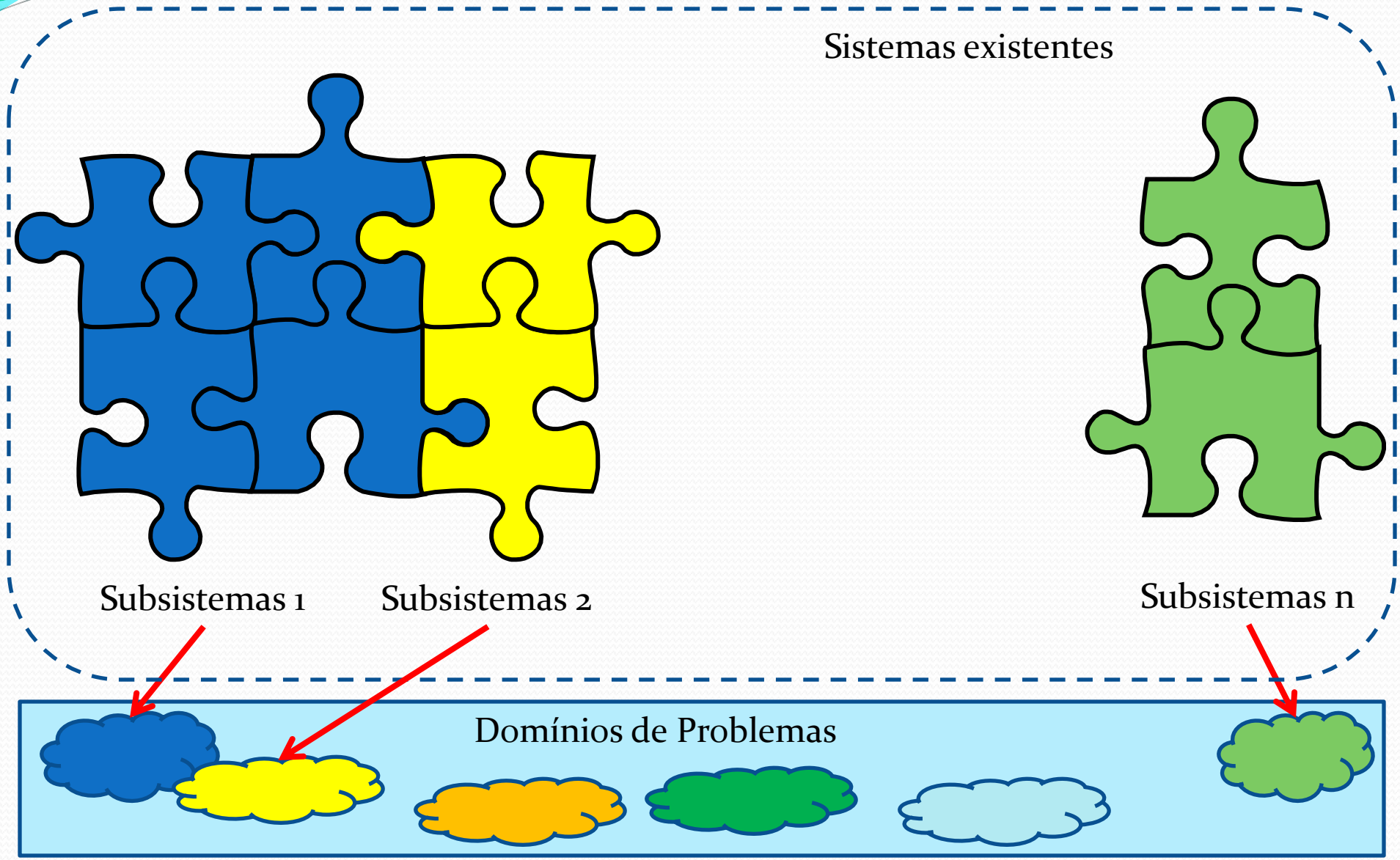
Why Is Software Architecture Important?

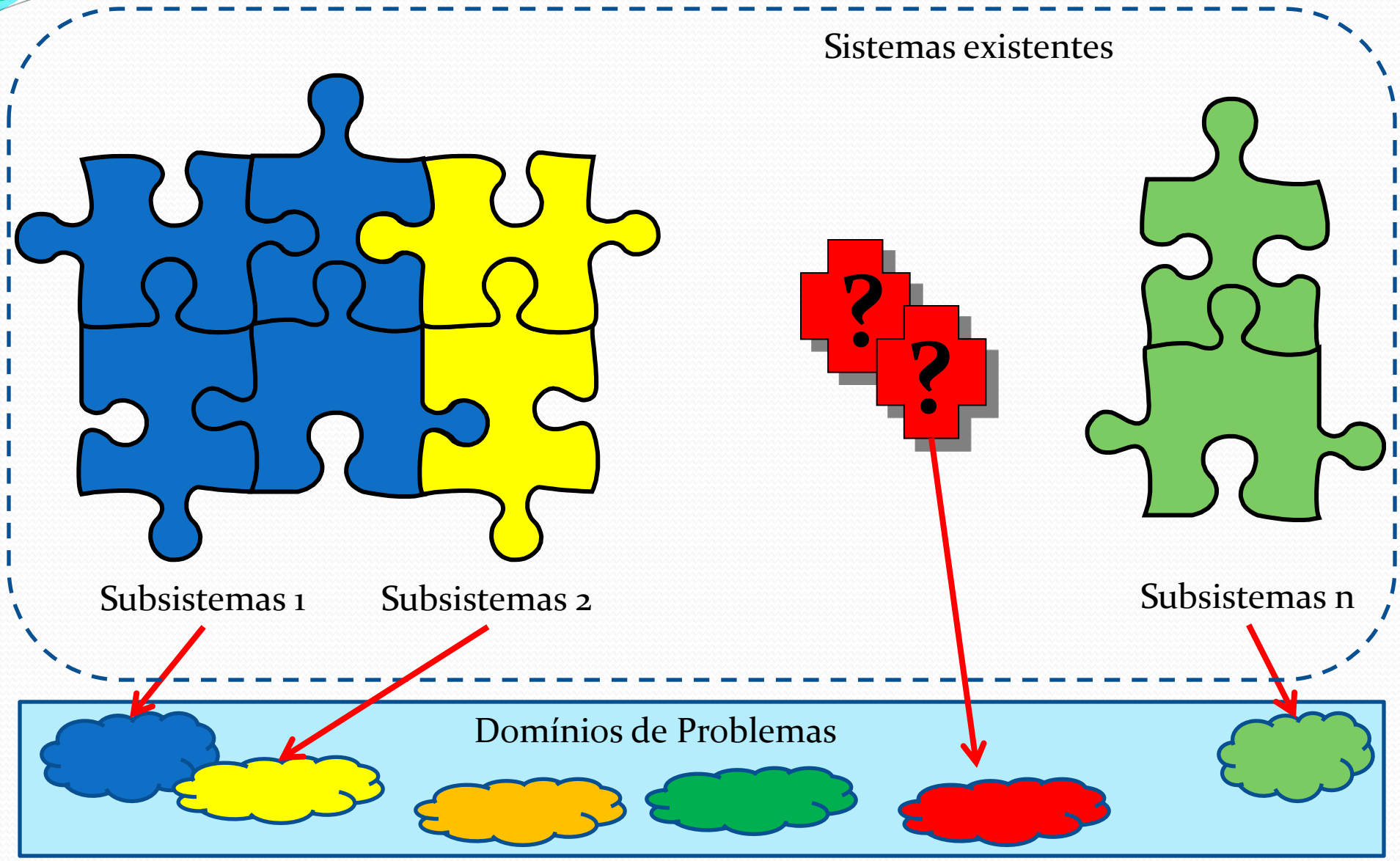
- If a project has not achieved a system architecture, including its rationale, the project should not proceed to full-scale system development. Specifying the architecture as a deliverable enables its use throughout the development and maintenance process.

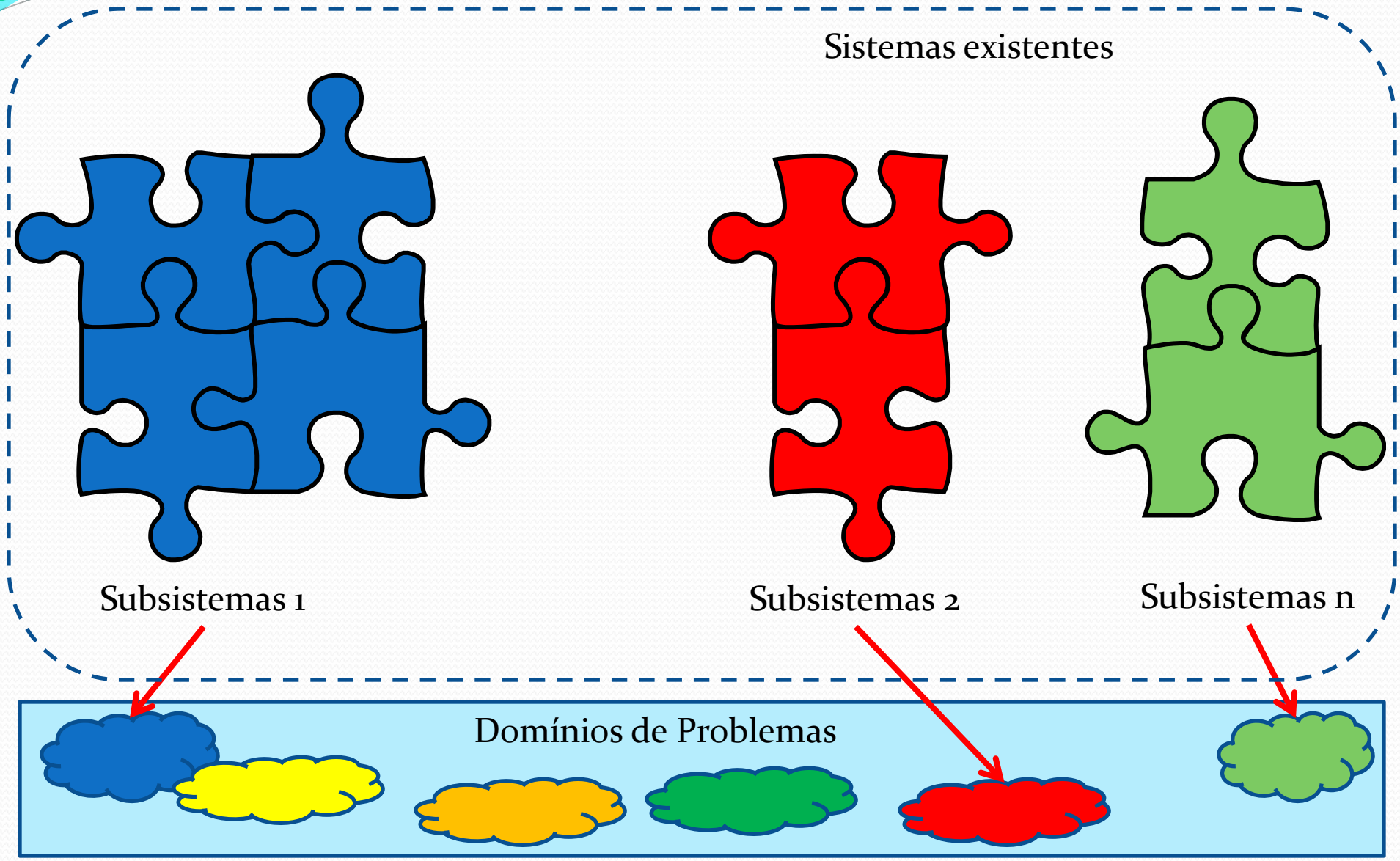
  — Barry Boehm [Boehm 95]

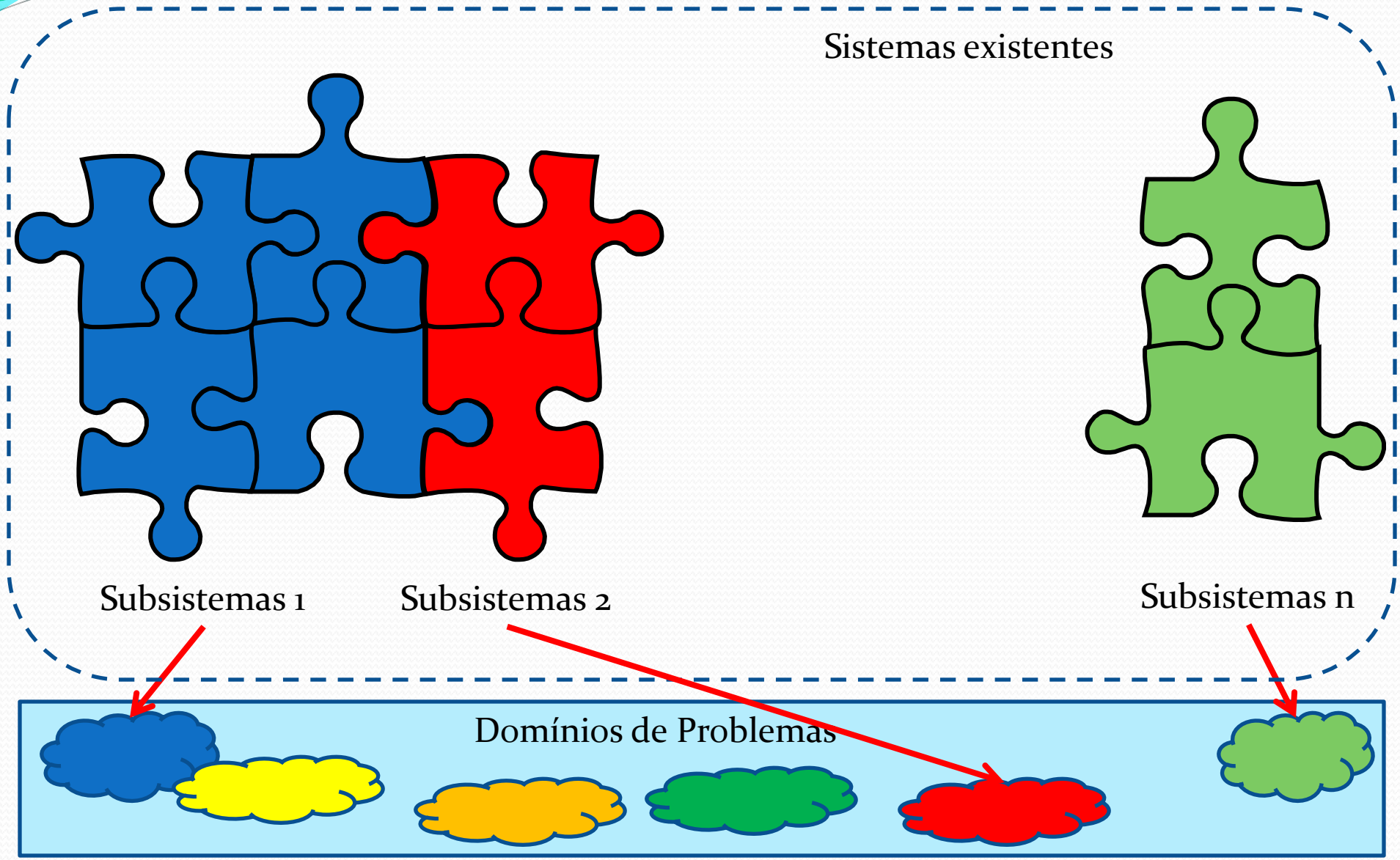
Arquitetura de Software, porquê é importante?

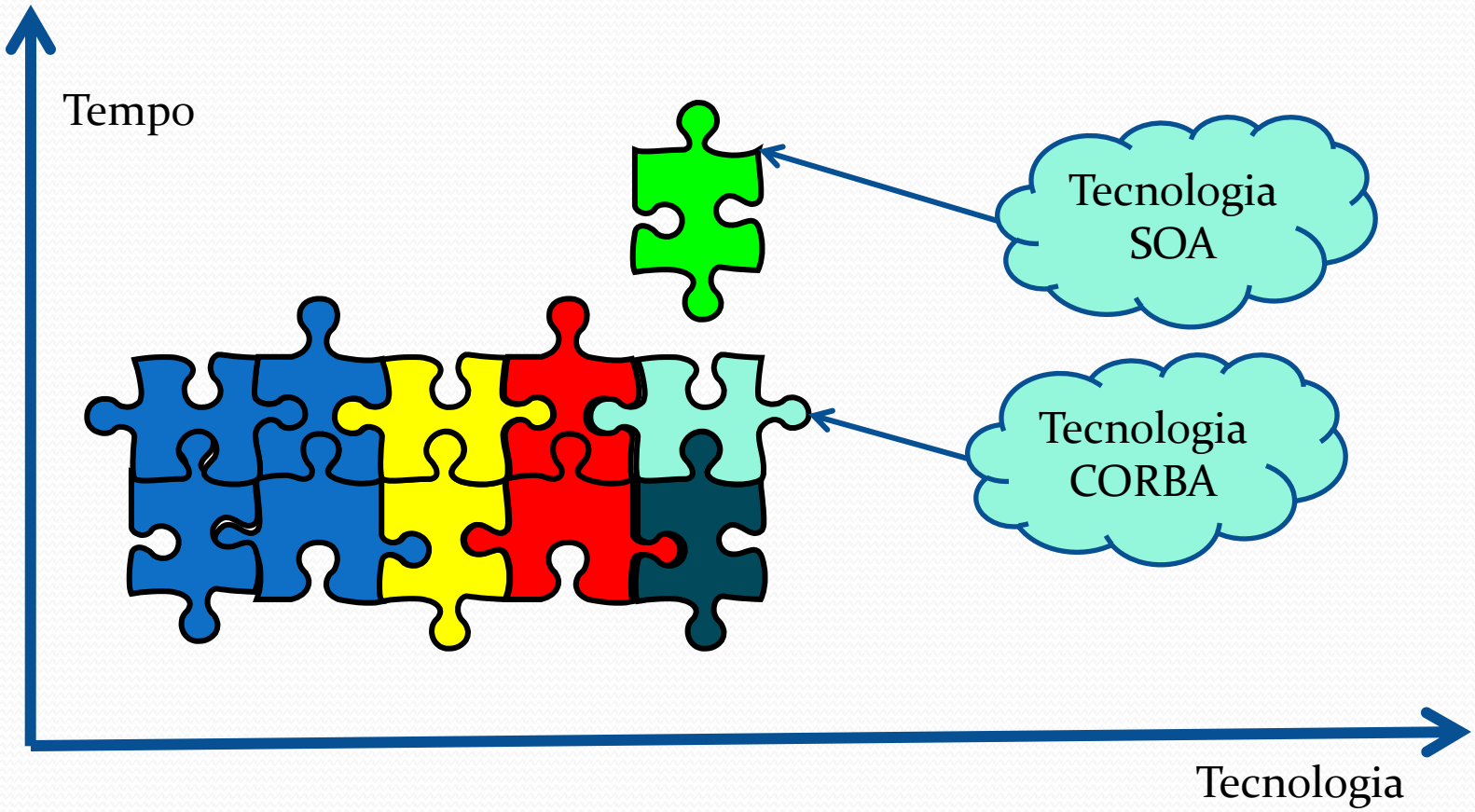
- Comunicação da equipe.
- Antecipação dos aspectos de decisões de infra-estrutura.
- Transferênicia da abstração de um sistema de software para outro.
- Importante veículo de comunicação com os *Stakeholder*.

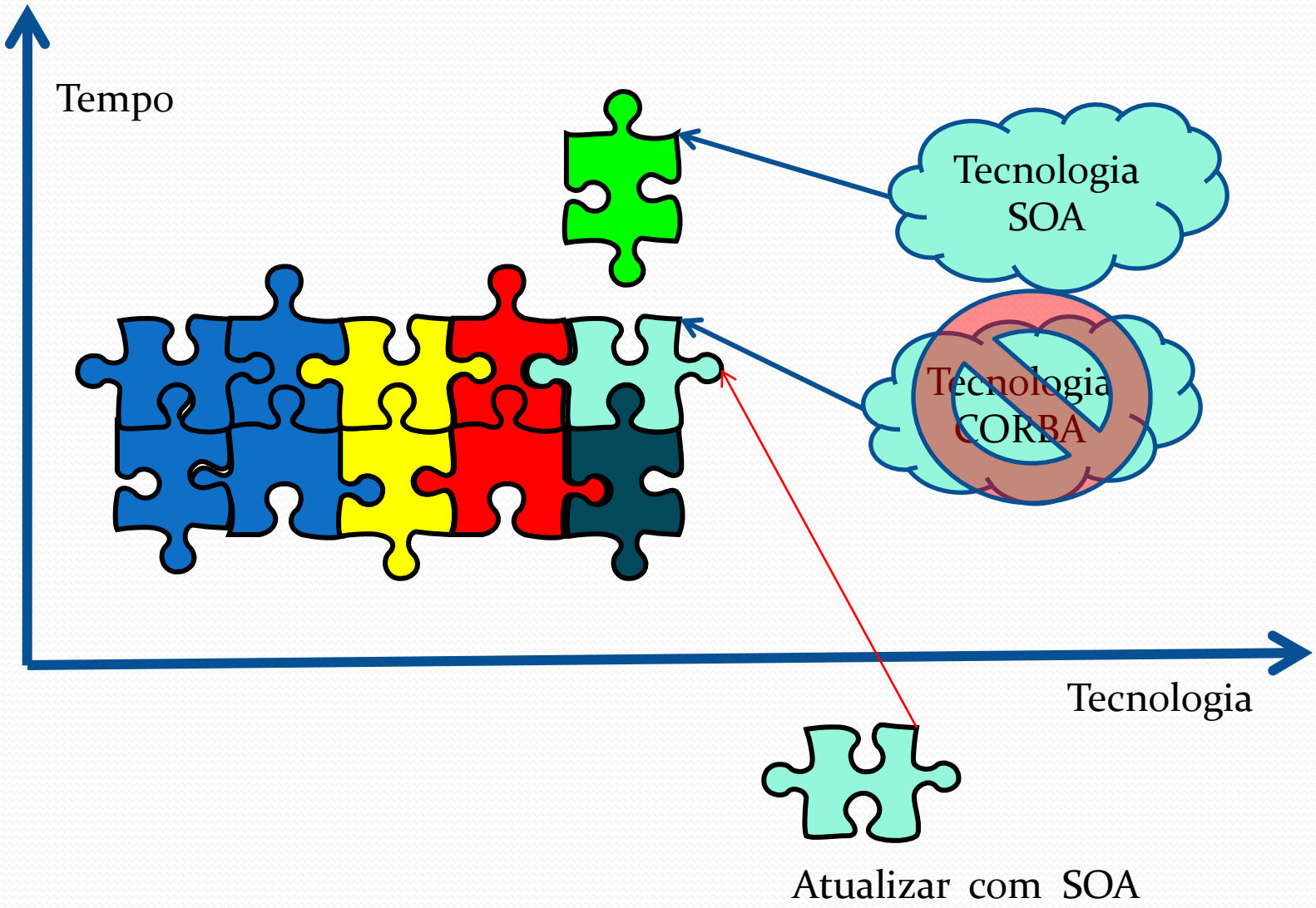

Sistemas existentes
Subsistemas 1
Subsistemas 2
Subsistemas n
Domínios de Problemas


Sistemas existentes
Subsistemas 1
Subsistemas 2
Subsistemas n
Domínios de Problemas


Sistemas existentes
Subsistemas 1
Subsistemas 2
Subsistemas n
Domínios de Problemas


Sistemas existentes
Subsistemas 1
Subsistemas 2
Subsistemas n
Domínios de Problemas


Tempo
Tecnologia
SOA
Tecnologia
CORBA
Tecnologia


Tempo
Tecnologia SOA
Tecnologia CORBA
Tecnologia
Atualizar com SOA

Ciclo de vida

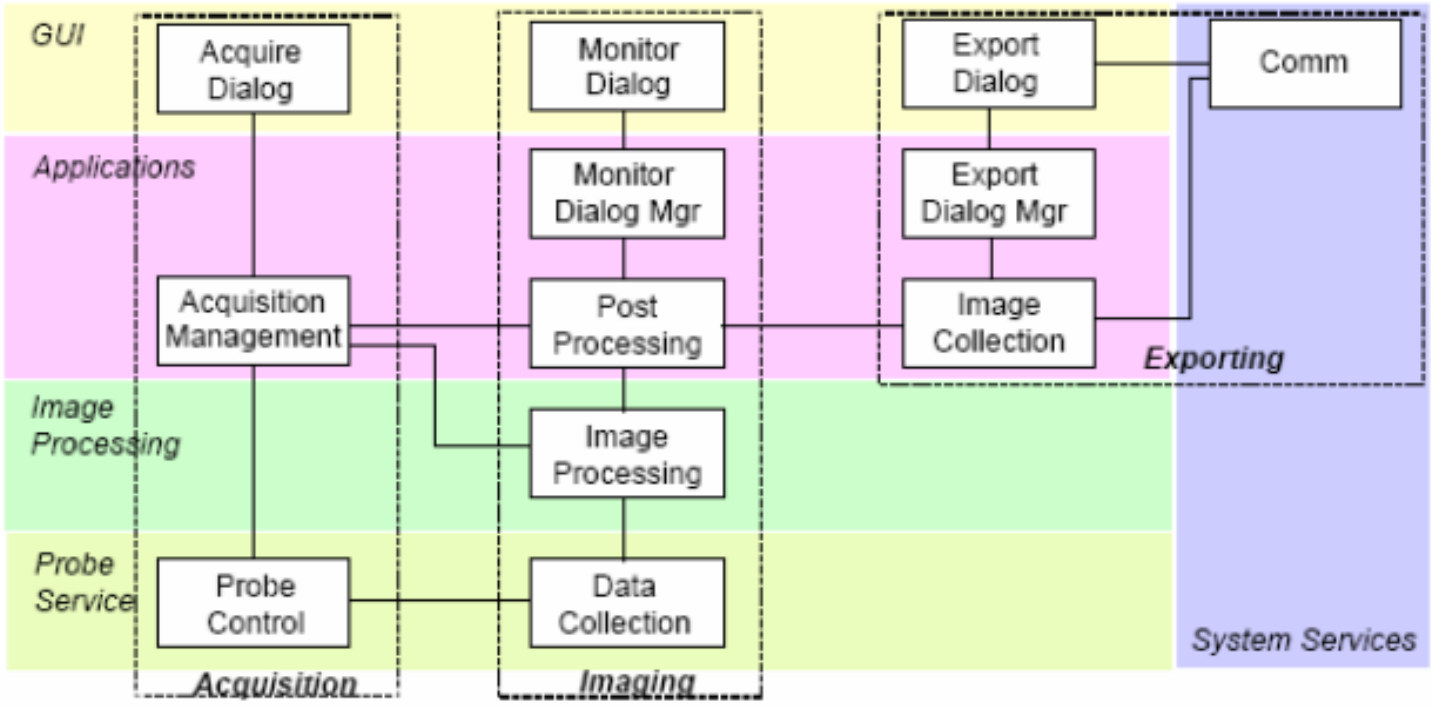
O clico de vida da arquitetura de software compreende:

- Criar a arquitetura
  - Usando padroes arquiteturais, design patterns e experiências passadas
- Avaliar a Arquitetura
- Refinar, atualizar e refatorar a arquitetura dentro do ciclo de vida do produto
- Usar a arquitetura como um guia para a implementação
- Tentar enfatizar a arquitetura durante a implementação

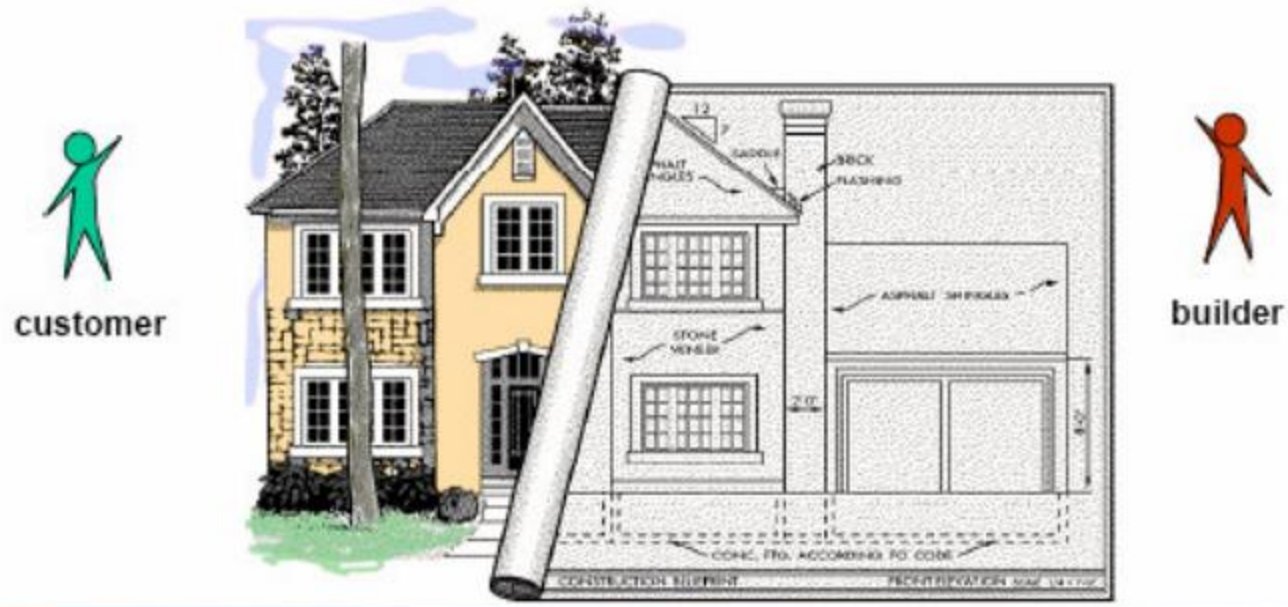
O que e o seu cliente entendem por isso?

```
interface ContextManager {

exception UnknownParticipant { long unknownParticipant; }
exception TransactionInProgress { string instigatorName; }
exception InvalidTransaction { string reason; }
exception InvalidContextCoupon { }
exception ChangesNoteEnded { }
exception AcceptNotPossible { }
...
StartContextChanges (in long participantCoupon, out long contextCoupon) raises
(UnknownParticipant, TransactionInProgress, InvalidTransaction)

EndContextChanges (in long contextCoupon, out boolean noContinue, out string[ ]
responses) raises (InvalidContextCoupon, NotInTransaction, InvalidTransaction)

PublishChangesDecision (in long contextCoupon, in string decision) raises
(NotInTransaction, InvalidContextCoupon, ChangesNotEnded, AcceptNotPossible)
... }
```

O que você e seu cliente entendem por isso?



Visões diferentes para cada público!

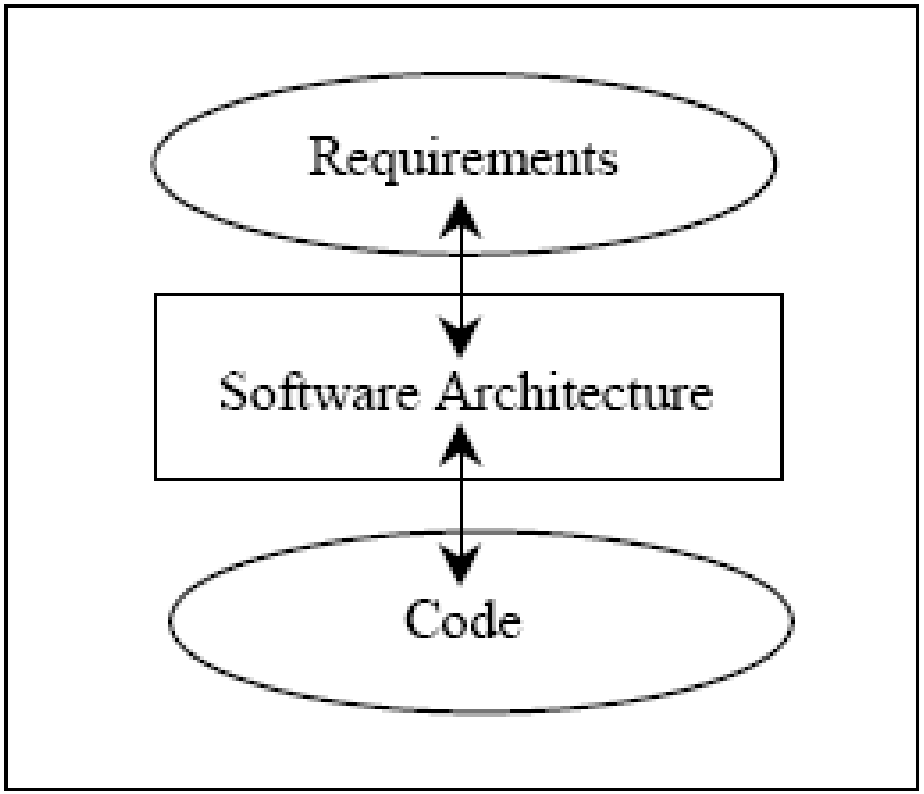


Papéis

- Apesar de existirem numerosas definições sobre arquitetura do software, no núcleo de tudo está a **noção de que a arquitetura de um sistema descreve sua estrutura bruta**.
- Esta estrutura ilumina as decisões de projeto de alto nível, incluindo coisas como:
  - o sistema é composto das peças de interação, onde estão os principais caminhos de interação
  - Adicionalmente, uma descrição arquitetural inclui informação suficiente para permitir a análise de alto nível e crítica do sistema.

Papéis

- A arquitetura do software desempenha tipicamente um papel chave como uma ponte entre requisitos e código.



Papéis

- Fornecendo uma descrição abstrata do sistema, a arquitetura **expõe determinadas propriedades**, ao mesmo tempo que esconde outras.
- Idealmente, esta representação fornece um **guia de entendimento intelectual do sistema** como um todo, permitindo aos projetistas *raciocinar sobre a possibilidade do sistema satisfazer a determinadas exigências*, e sugere um *blueprint* para a construção do sistema e a sua composição.

Papéis

- Por exemplo: uma arquitetura para um aplicação de processamento de sinal, pôde ser construída como uma rede do fluxo de dados em que os nós lêem vetores de entrada dos dados, transformam esses dados, e os escrevem aos vetores da saída.
- E aqui, os projetistas devem pensar em estruturas para este tipo de processamento.

Papéis

- Na elaboração da Arquitetura de um software, 6 papéis são desempenhados:
  - Compreensão,
  - Reuso,
  - Construção,
  - Evolução,
  - Análise e
  - Gerência.

Papéis: Compreensão

- A arquitetura do software simplifica nossa habilidade de entender grandes sistemas apresentando-os em um alto nível do abstração de forma a ser facilmente compreendido.
- Além disso, **expõe** as principais **restrições** do projeto do sistema, tanto quanto o **aspecto racional** para fazer escolhas arquiteturais específicas.

Papéis: Reuso

- Descrições de arquiteturas suportam reuso em múltiplos níveis. Trabalhos correntes focam geralmente em bibliotecas de componentes.
- Projetos de Arquiteturas, promovem reuso em grandes componentes e em Frameworks, onde seus componentes possam ser integrados.

Papéis: Construção

- A Construção de uma arquiterua oferece um norte para os desenvolvedores, indicando ou apresentando os maiors componentes e suas dependências.

- Por exemplo: as camadas de uma arquitetura tipicamente documentam as fronteiras da implementação, expondo os maiores componentes e suas interfaces, bem como as suas principais restrições.

Papéis: Evolução

- A arquitetura do software pode expor as dimensões na linha do tempo da possível evolução de um sistema.
- Deixa explícito "as paredes internas" de um sistema, os desenvolvedores do sistema podem melhor compreender as ramificações das mudanças, e estimar desse modo mais preciso o custo das modificações.
- Além disso, as descrições arquiteturais separam interesses sobre as funcionalidade de um componente da forma com que esse componente é conectado (interage) a outros componentes.
- Esta separação permite mais facilmente aos mecanismos da conexão obter um maior reuso e interoperabilidade.

Papéis: Análise

- As descrições arquiteturais fornecem novas facilidades para a tarefa de análise tais como:
  - a consistência do sistema,
  - o conformidade dos atributos da qualidade,
  - a análise da dependência e
  - A análises específicas de domínio para identificar arquiteturas semelhantes.

Papéis: Gerência

- A avaliação crítica de uma arquitetura conduz tipicamente a uma grande compreensão das exigências, das estratégias da execução, e de riscos potenciais a serem enfrentados na implementação de um sistema de software.

Mas o que é e não é Arquitetura de software?

Arquitetura define elementos de software:

- Foco no comportamento externamente observável e interações
- Responsabilidades das partes
- Abstrações de elementos de software

Todo sistema tem uma arquitetura

- Indica como requisitos são satisfeitos
- Detalhes de implementação **não** entram
- Algoritmos normalmente **não afetam** a arquitetura
- Tudo que está lá **tem uma razão!**

http://twiki.cin.ufpe.br/twiki/bin/viewfile/TAES/ArquiteturaPos200902?rev=2;filename=01_introducao_arq_sw.pdf

Arquitetura de Software

Lida com?

Lida com...

A arquitetura de software lida com os requisitos do sistema!

http://twiki.cin.ufpe.br/twiki/bin/viewfile/TAES/ArquiteturaPos200902?rev=2;filename=01_introducao_arq_sw.pdf

Lida com requisitos

Com ênfase em?

Ênfase nos...

Grande ênfase nos

não-funcionais!

http://twiki.cin.ufpe.br/twiki/bin/viewfile/TAES/ArquiteturaPos200902?rev=2;filename=01_introducao_arq_sw.pdf

Questões que uma arquitetura pode responder...

- Como representar o mapa em um sistema que traça rotas percorridas por ônibus de modo a minimizar o trabalho da equipe?
- Como garantir a confiabilidade de um servidor a um baixo custo?
- Qual a maneira mais eficiente de se construir uma grade de horários levando-se em conta as várias restrições impostas por professores, diretores e regras departamentais?
- Como tornar o sistema capaz de lidar com 1.000.000 de usuários simultâneos sem sobrecarregar a rede?

Arquitetura de referência

Alguns autores utilizam os termos estilos arquiteturais e arquitetura de referência como sinônimos.

Contudo, arquitetura de referência é:

Uma estrutura que provê uma caracterização das funcionalidades dos sistemas de software de um dado domínio de aplicação.

Estilos Arquiteturais

- Capturam conjuntos de escolhas de projeto comuns a vários sistemas
- Podem representar interações como elementos de primeira ordem
  - Conectores

Exemplos comuns:

- Decomposição Modular
- Cliente/Servidor
- Camadas

Associados à representação da arquitetura

http://twiki.cin.ufpe.br/twiki/bin/viewfile/TAES/ArquiteturaPos200902?rev=2;filename=01_introducao_arq_sw.pdf

Estilo Arquitetural

Estilo arquitetural:

- Permite que um profissional (projetista, arquiteto ou engenheiro) determine a classe a qual pertence a organização de um sistema.
- O que ajudam a identificar o estilo que retrata a arquitetura de software do sistema?
  - Características dos componentes (subsistemas) e conectores do sistema
  - Topologia da arquitetura
  - Restrições semânticas
  - Mecanismos de interação entre os componentes.

Para a proxima aula.

- Fazer uma apresentação de 3 estilos arquiteturais. A apresentaçaõ deve ter:
- PPT, prós e contra do uso da arquitetura, de onde foi criada esta arquitetura, para que propósito é mais aconselhavel o seu uso, exemplos, referencias e pesquisas em termos de artigos padrao IEEE e ACM falam desta arquitetura. (isto configura o mínimo de detalhes que a apresentação deve ter).
- Deve-se entregar um documento impresso em formato de artigo descrevendo as três arquiteturas selecionadas.

- 1. Layered
- 2. Pipes and filters
- 3. Message Bus Architecture
- 4. Component Based Architecture
- 5. Blackboard
- 6. Data-centric
- 7. Microkernel
- 8. Domain specific languages
- 9. Client-server (2-tier, 3-tier, n-tier exhibit this style)
- 10. MVC
- 11. Peer-to-peer
- 12. Service-oriented
- 13. Repository Style