



Software Engineering Method and Theory

Geraldo Xexéo

DCC/IM & PESC/COPPE

Universidade Federal do Rio de Janeiro

xexeo@cos.ufrj.br

<http://www.cos.ufrj.br/~xexeo>

<http://www.xexeo.net/>

Situação Atual

- Existe muito conhecimento sobre como desenvolver software
- Trocamos de método de desenvolvimento sem entender as consequências
 - Modismos
 - O que virá a seguir?
- Não há um acordo na comunidade em quais são as práticas adequadas para o desenvolvimento de software

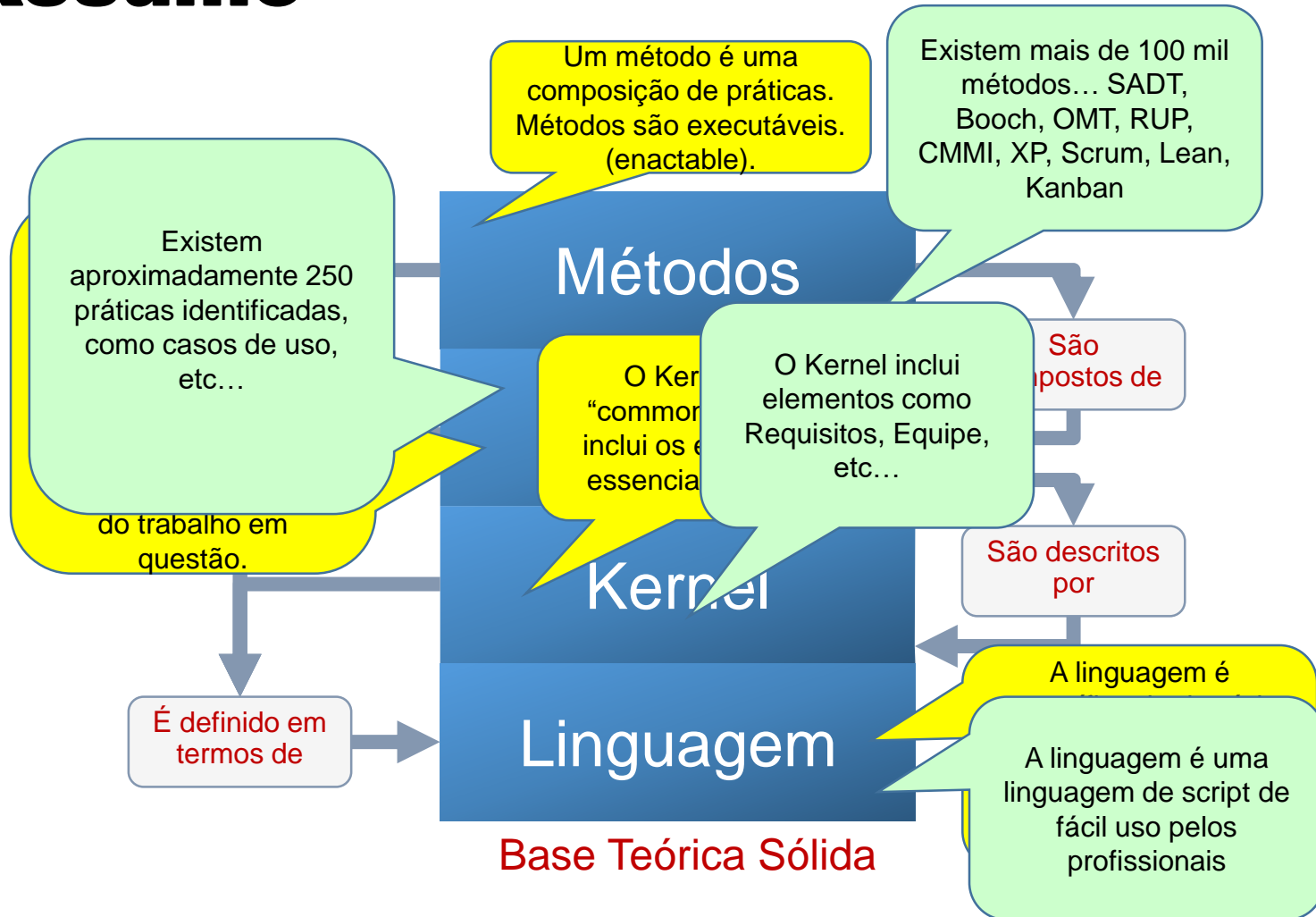
Práticas Imaturas

- Prevalência de modismos
- Falta de um base teórica sólida
- Grande número de métodos e suas variantes
 - Dificuldade de entender as diferenças
 - Diferenças aumentadas artificialmente
- Falta de avaliação experimento e validação
- Academia e indústria divididas

SEMAT

- Refundar a Engenharia de Software com base em
 - Uma teoria sólida
 - Princípios comprovados
 - Melhores Práticas
- Buscar Inicialmente
 - Base teórica sólida
 - “Kernel” de elementos amplamente aceitos

Resumo

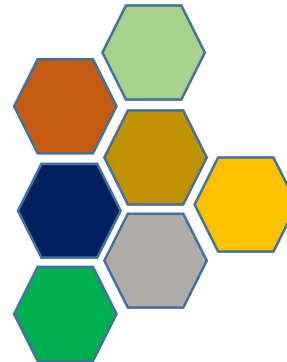
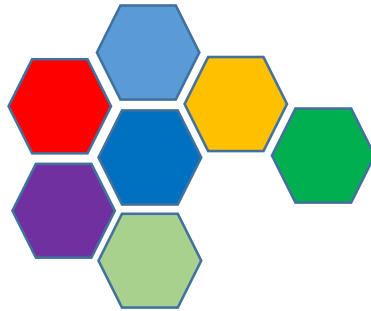


O Kernel

- Alphas
 - Representações das coisas essenciais com que trabalhamos
 - Descrições das coisas que o time gerência, produz e usa no processo de desenvolver, manter e suportar um bom software.
 - Âncoras para sub-alphas adicionais e produtos de trabalho exigidos pelas práticas de engenharia de software
- Espaços de atividade
 - Representações das coisas essenciais a fazer.
 - Fornecem descrições dos desafios que os times encontram enquanto desenvolvendo, mantendo e suportando sistemas de software, e
 - tipos de coisas que os times vão fazer para atingi-los.
- Competências
 - Habilidades Necessárias

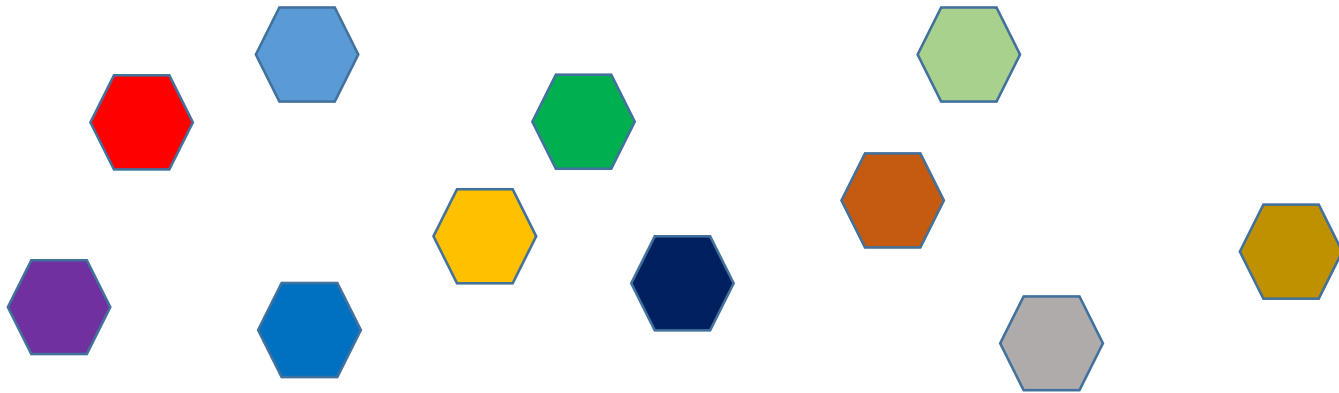
Método

- Uma composição de Práticas
- São executáveis
 - Enactable
- Mais de 100.000 encontrados
 - Incluindo SADT, RUP, Booch, OMT, CMMI, XP, Scrum, Lean, Kanban



Prática

- Uma abordagem repetitível para fazer algo com um objetivo específico em mente
- Fornece uma maneira sistemática e verificável de abordar um aspect particular do trabalho sendo feito
- Pelo menos 250 identificadas
 - Casos de Uso, Histórias do Usuário, Componentes, etc...



Kernel

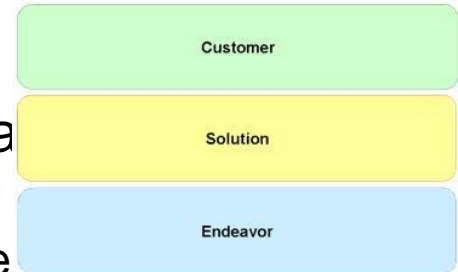
- É um campo comum que inclui os elementos essenciais da Engenharia de Software
- Elementos essenciais incluem:
 - Partes interessados, oportunidades, equipe,...

Linguagem

- Definem métodos, práticas e os elementos essenciais do Kernel
- Define as coisas essenciais para as quais um projeto de desenvolvimento precisa prestar atenção
- Linguagem de Script

O Kernel Essencial

- Define uma base comum para definição de prática desenvolvimento de software
 - De forma que possam ser definidas e aplicadas indepe
- Estruturado em 3 áreas
 - Cliente
 - Contém tudo relacionado com o uso e exploração do software sendo produzido
 - Solução
 - Contém tudo em relação a especificação e desenvolvimento do sistema de software
 - Empreendimento
 - Contém tudo em relação ao time e a forma como aborda o trabalho



O Kernel Essencial (II)

- Cada área contém
 - Alphas
 - Espaços de Atividades
 - Competências
- As áreas fornecem
 - Coisas essenciais a serem produzidas, gerenciadas e controladas
 - Coisas essenciais a serem feitas
 - Competências essenciais

Kernel: Áreas de Interesse

- Cliente
 - Tudo que tem relação com o uso e exploração do sistema de software sendo produzido
- Solução
 - Tudo para real sistema de software
- Empreendimento
 - Tudo que tem dam o trabalho

Cliente

Solução

Empreendimento

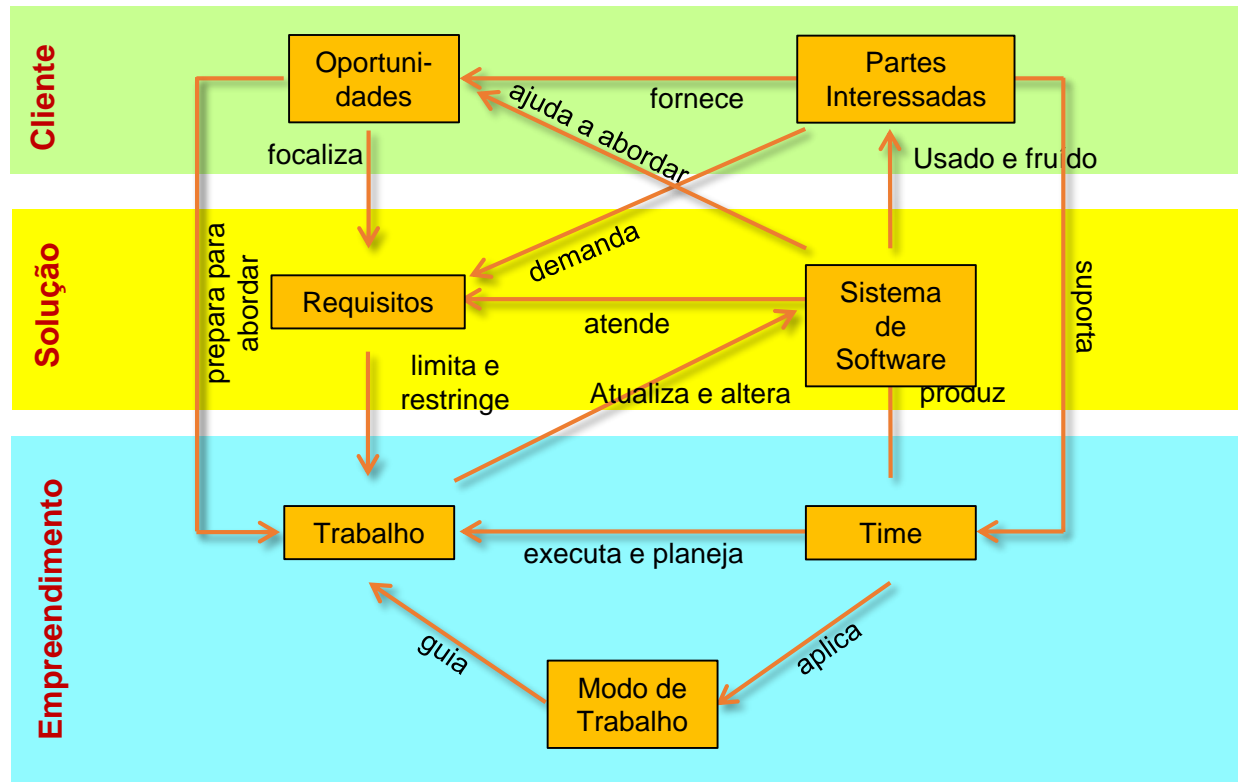
Alphas



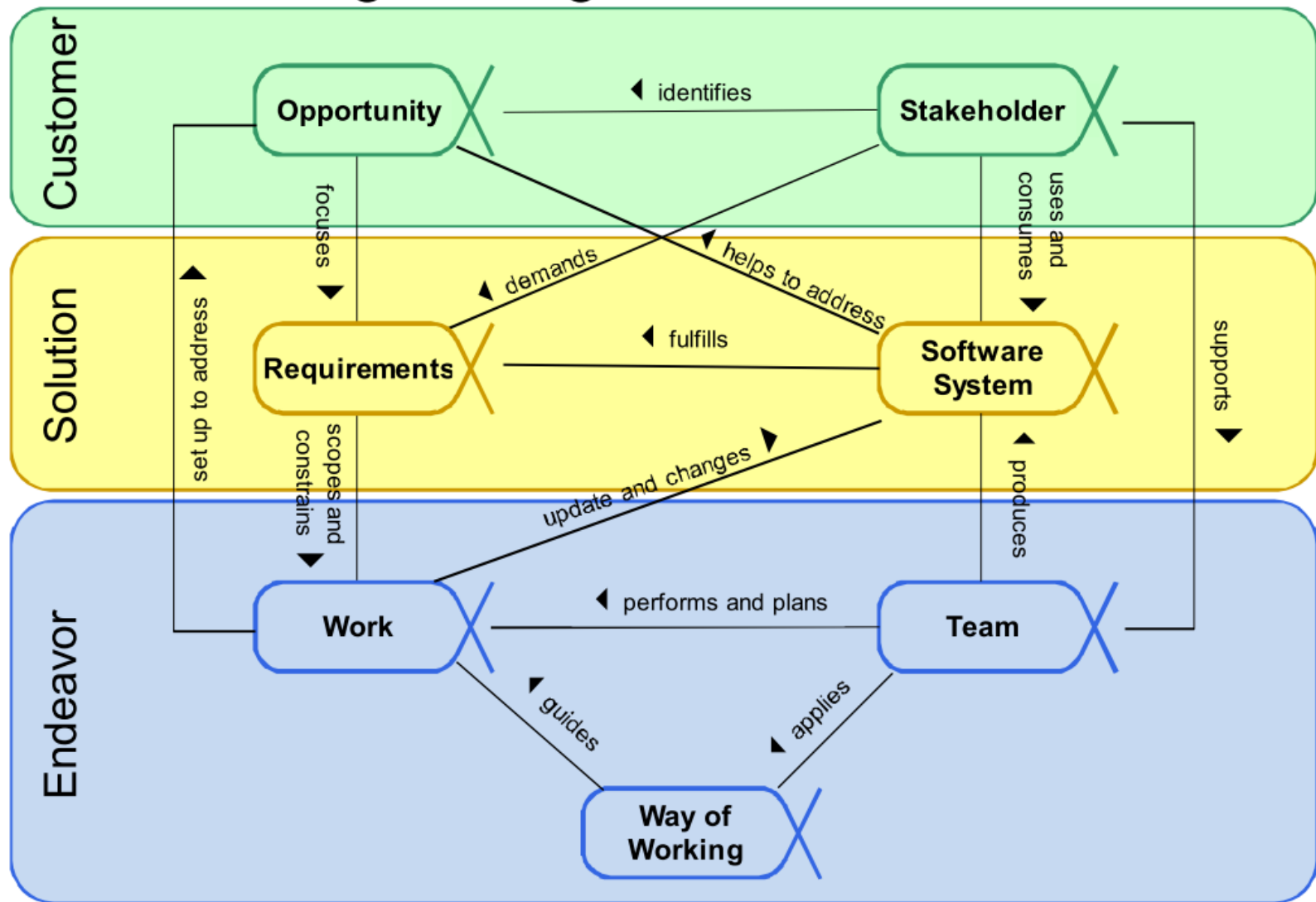
- Representações das coisas essenciais com que trabalhamos
- Descrições das coisas que o time gerência, produz e usa no processo de desenvolver, manter e suportar um bom software.
- Âncoras para sub-alphas adicionais e produtos de trabalho exigidos pelas práticas de engenharia de software
- Capturam conceitos chave da ES
- Permitem rastrear e avaliar a saúde e o progresso de qualquer empreendimento de ES
- Fornecem uma base comum para a definição de práticas e métodos da ES

Alphas

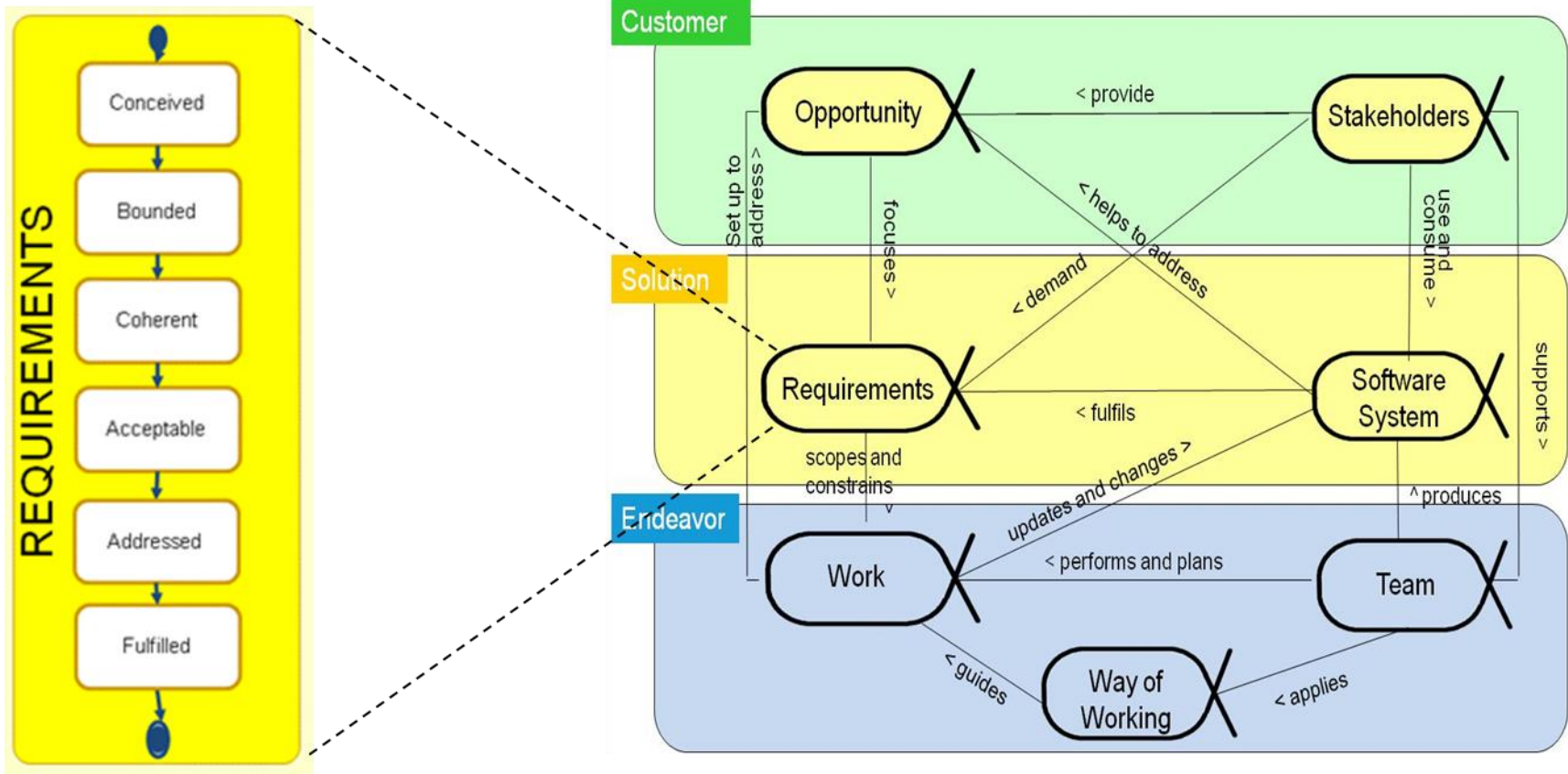
Práticas e ferramentas escolhidos sob medida e utilizados pelo time para guiar e apoiar seu trabalho. O time desenvolve seu MdT junto com o entendimento de sua missão e do seu ambiente de trabalho. No decorrer do trabalho o time continuamente reflete sobre seu modo de trabalho e o adapta como for necessário para o contexto corrente.



O Kernel - Alphas



Detalhamento em Estados



Cliente

- Partes Interessadas

- Pessoas, grupos ou organizações que afetam ou são afetadas pelo sistema de software. As P.I. fornecem oportunidades e são a fonte de requisitos e financiamento do sistema de software. Elas precisam ser envolvidas atrás do empreendimento para apoiar o time e garantir que um sistema de software aceitável é produzido.

- Oportunidades

- Circunstâncias que tornam apropriado o desen. ou mudança de um sistema de software. Articula a razão da criação de um sistema novo ou modificado. Representa o entendimento compartilhado pelo time sobre as necessidades das P.I., ajudam a formar os requisitos fornecendo justificativas para seu desenvolvimento.

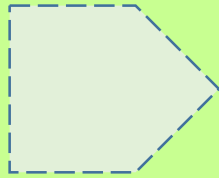
Solução

- Requisitos
 - O que o software precisa fazer para atender as oportunidades e satisfazer as P.I.. É importante descobrir o que é necessário para o SdS, compartilhar esse entendimento entre as P.I. e o time e usar para dirigir o desenvolvimento e teste do sistema
- Sistema de Software (SdS)
 - Um sistema feito de software, hardware e dados que fornece valor primário pela execução de software. O produto primário de qualquer empreendimento de engenharia de software, um SdS pode ser parte de um software maior, hardware ou solução de negócio.

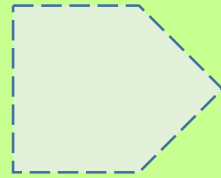
Empreendimento

- Time
 - Um grupo de pessoas ativamente engajadas no desenvolvimento, manutenção, entrega e suporte de um SdS específico. O time planeja e realiza o trabalho feito para atualizar e mudar o SdS
- Trabalho
 - Atividade envolvendo esforço físico ou mental feito com o propósito de alcançar um resultado. No contexto da ES, trabalho é tudo que o time faz para atingir seus objetivos e de produzir um SdS atendendo aos Req. e endereçando as Oport. Apresentadas pelo cliente. O trabalho é guiado por práticas que fazem o modo de trabalho.
- Modo de Trabalho
 - Práticas e ferramenta escolhidos sob medida e utilizados pelo time para guiar e apoiar seu trabalho. O time desenvolve seu MdT junto com o entendimento de sua missão e do seu ambiente de trabalho. No decorrer do trabalho o time continuamente reflete sobre seu modo de trabalho e o adapta como for necessário para o contexto corrente.

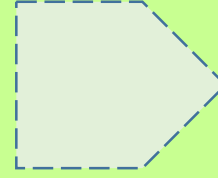
Kernel: Espaços de Atividades



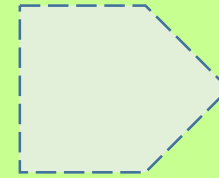
Explorar
Possibilidades



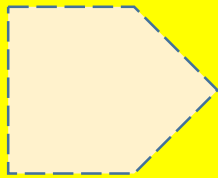
Envolver as
Partes Interessadas



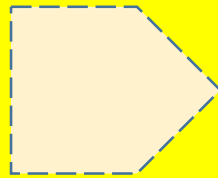
Garantir Satisfação
das Partes Interessadas



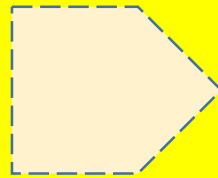
Usar o Sistema



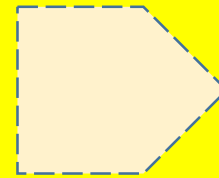
Entender os
Requisitos



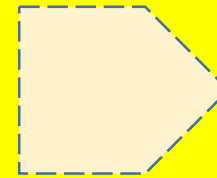
Dar Forma ao
Sistema



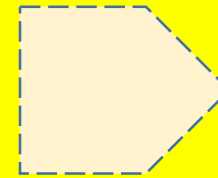
Implementar
o Sistema



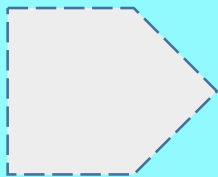
Testar
o Sistema



Implantar
o Sistema



Operar
o Sistema



Preparar o
Trabalho



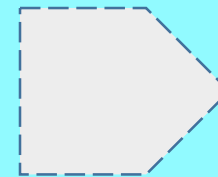
Coordenar
Atividades



Apoiar
o Time

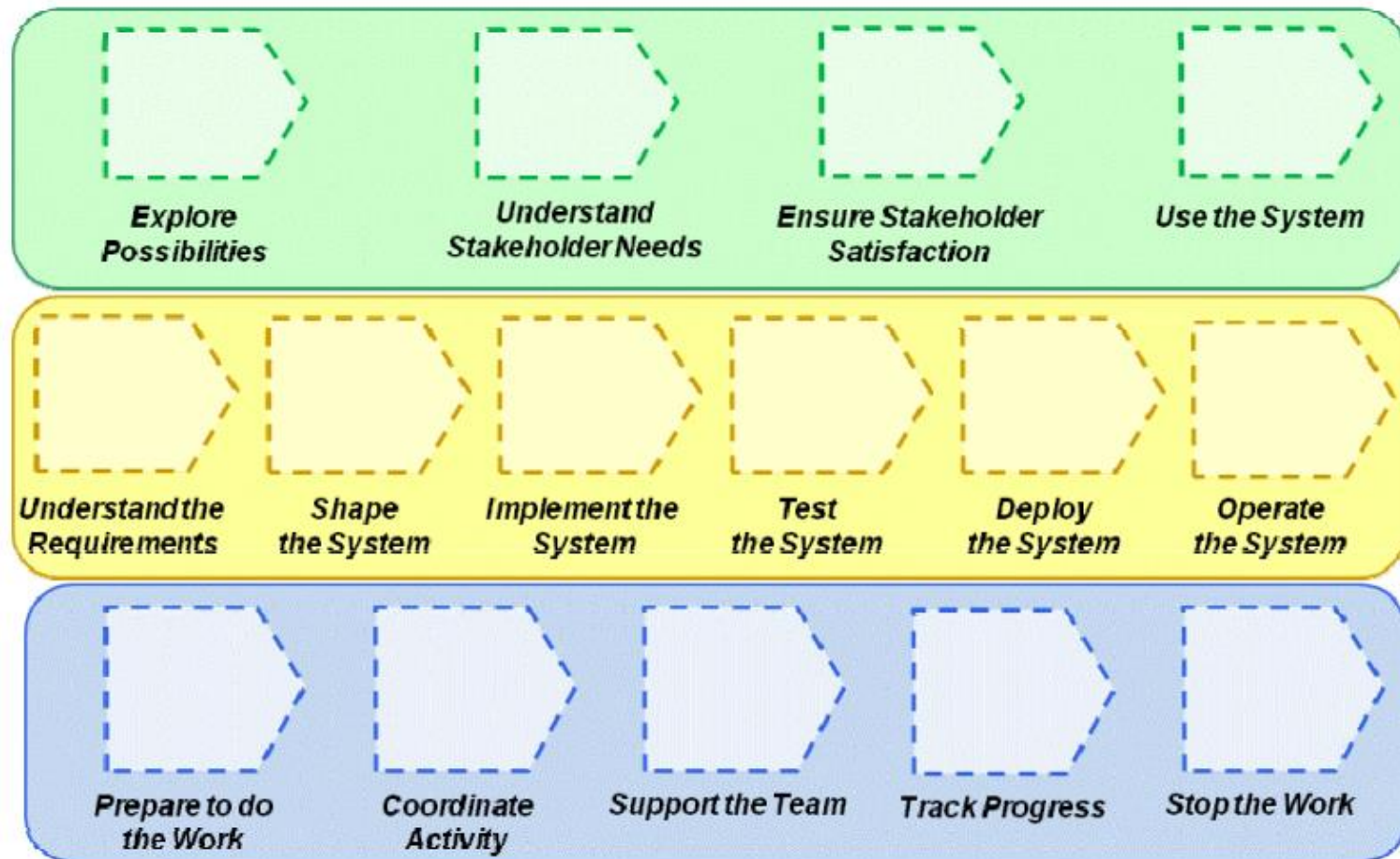


Rastrear o
Progresso



Parar o
Trabalho

Kernel: Espaços de Atividades



Competências

- Níveis
 - Auxilia
 - Aplica
 - Domina
 - Adapta
 - Inova



Representar
Partes Interessadas



Análise



Desenvolvimento



Teste

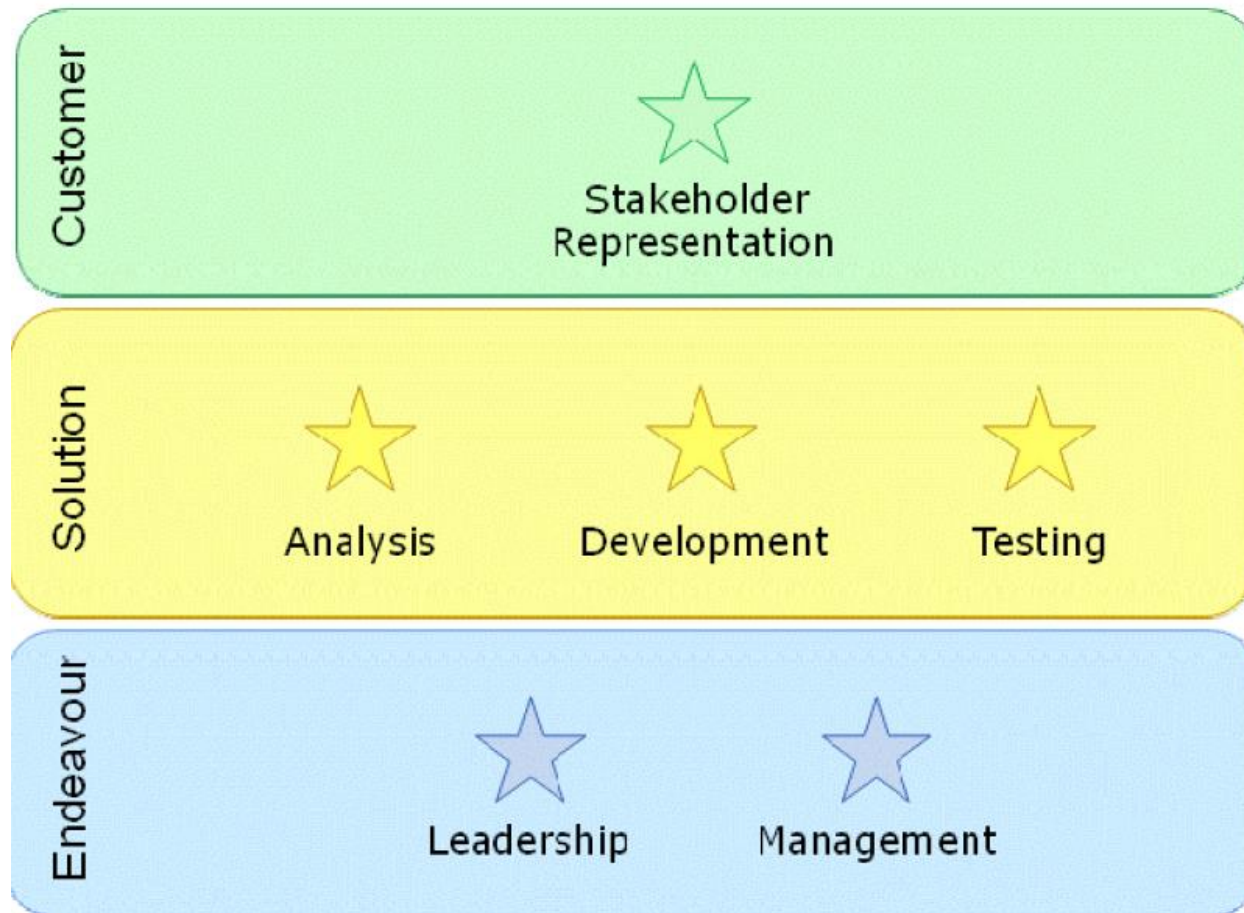


Liderança

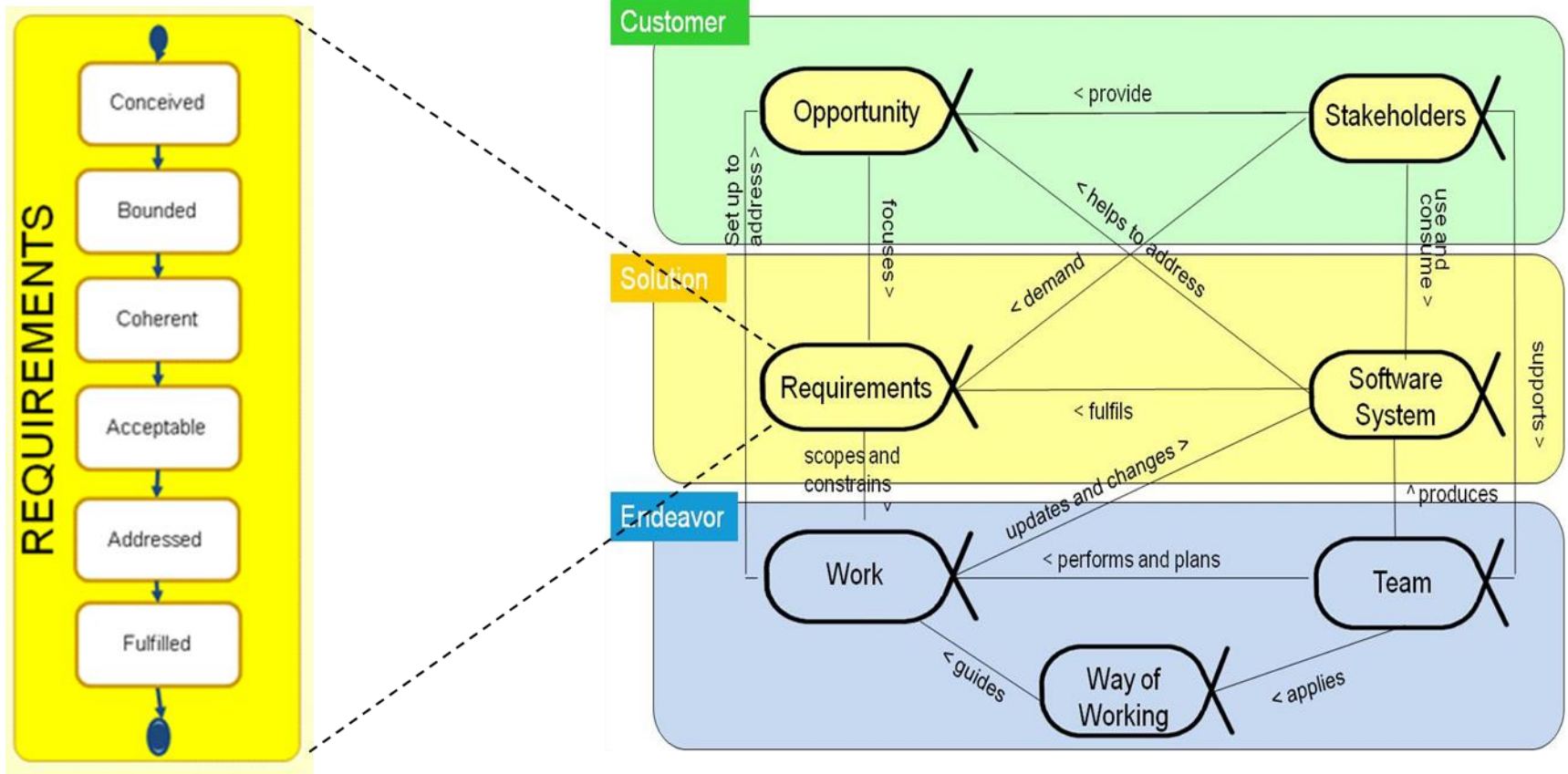


Gerência

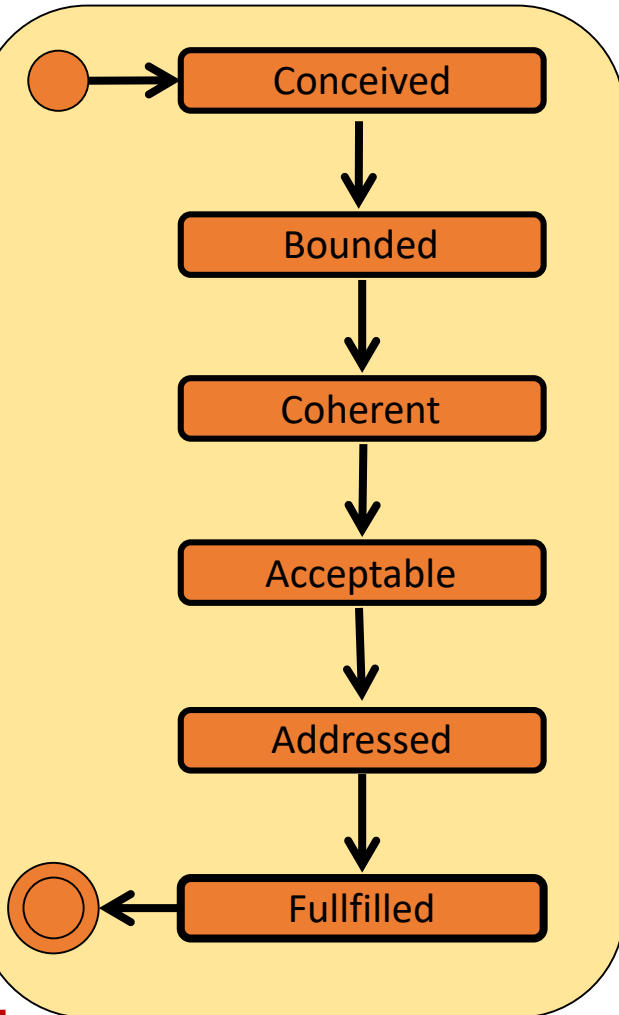
Kernel: Competências



Detalhamento em Estados



Requirements – states



The need for a new system has been agreed.

The purpose and theme of the new system are clear.

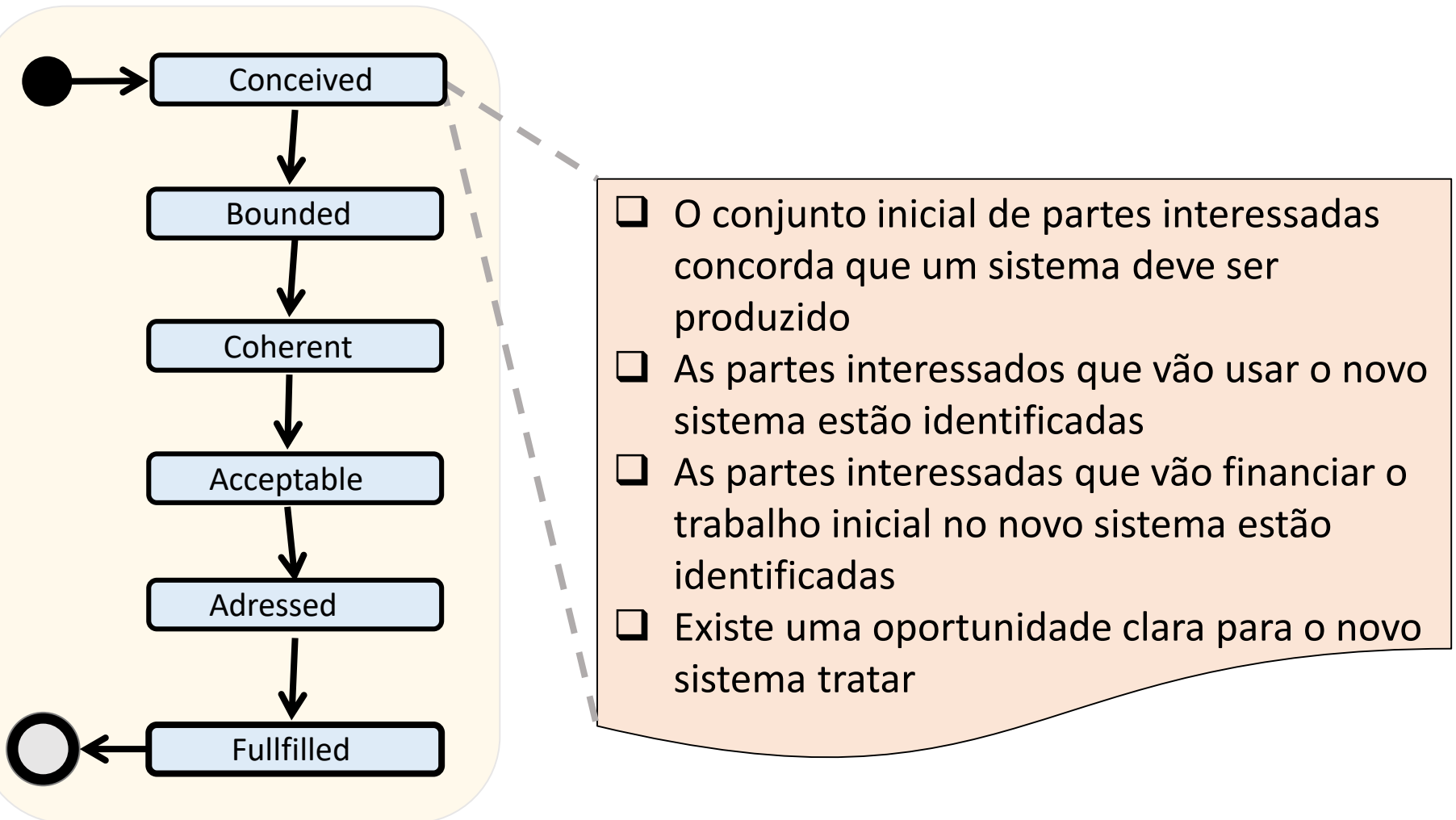
The requirements provide a coherent description of the essential characteristics of the new system.

The requirements describe a system that is acceptable to the stakeholders.

Enough of the requirements have been addressed to satisfy the need for a new system in a way that is acceptable to the stakeholders.

The requirements have been addressed to fully satisfy the need for a new system.

Checklist para os estados de requisitos



Orientado a resultado

☐ **Requirements**

Conceived

Bounded

Coherent

Sufficient

Satisfactory

Fulfilled

- Good Requirements meets real needs
- Good Requirements has clear scope
- Good Requirements are coherent and well organized
- Good Requirements help drive development

☐ **Requirements**

Conceived

• Need for system agreed by initial stakeholders

• Users and customers identified

• Expected benefit of system agreed

1 / 6

Cada estado define uma check-list extensível

Estados das Competências

- Auxilia
- Aplica
- Domina
- Adapta
- Inova

Espaços de Atividade

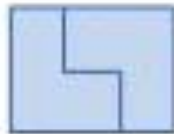
- Representam coisa a fazer
- Complementam os Alphas para fornecer um visão baseada em atividades do desenvolvimento de software



Evolução do Kernel

Essence Lite

Checklists



aligns a set of >



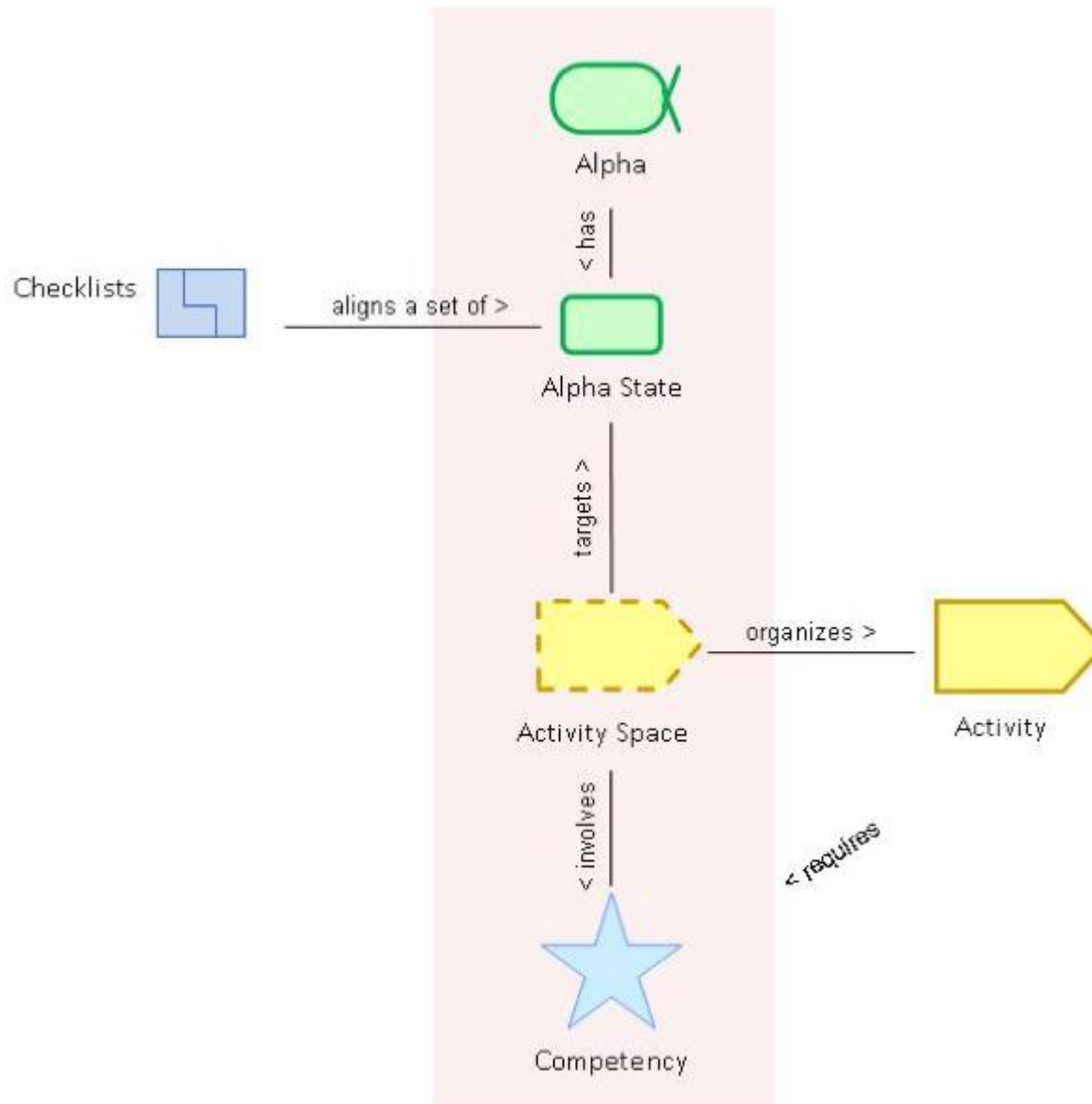
Alpha

< has

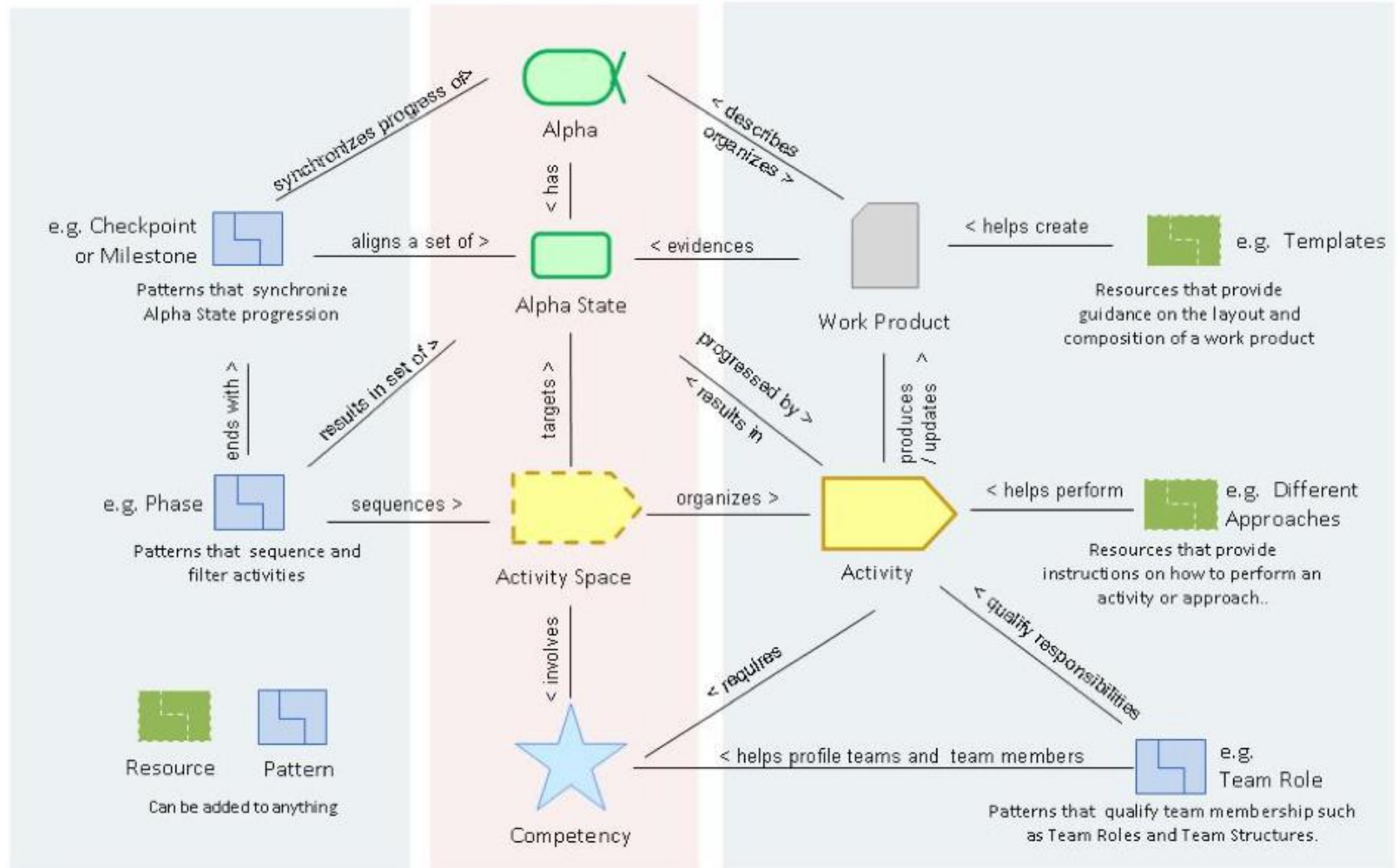


Alpha State

Essence Kernel



Essence Plus



Atividades: Cliente

- Explorar Possibilidades
 - Inclui as oportunidades a serem abordadas e a identificação de partes interessadas
- Entender Necessidades das Partes Interessadas
 - EnGarantir a Satisfação das Partes Interessadas
- Usar o Sistema

Cenários de Uso

Compania X, Ana e Beto

- Company X executa vários projetos de desenvolvimento, grandes e pequenos
- Ana, a executiva, quer melhorar a capacidade de desenvolvimento
- Para isso, designa Beto
- Smith encontra com vários times, de diferentes backgrounds e experiências e os ajuda

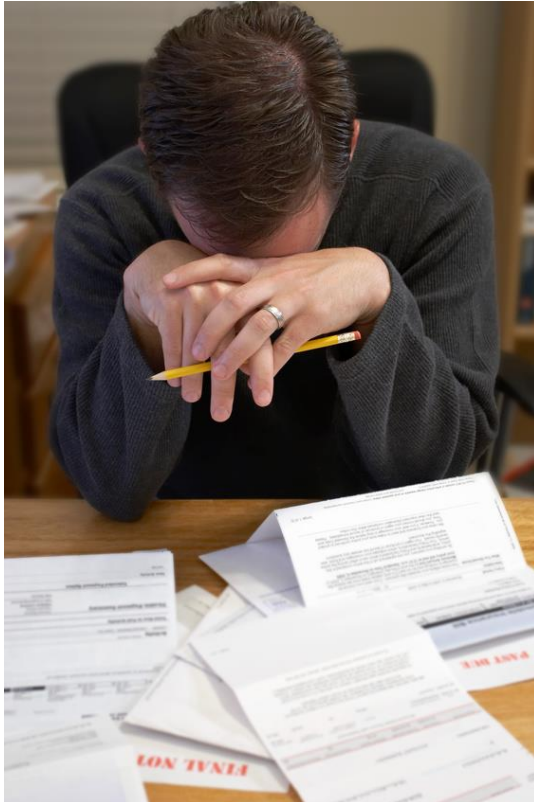


Cenário: Encontro com time pequeno

- Beto encontra um time pequeno A
- Precisa entender o status do desenvolvimento
- Como fazer isso?
- Procurar por o que?
- Como chegar a uma conclusão?

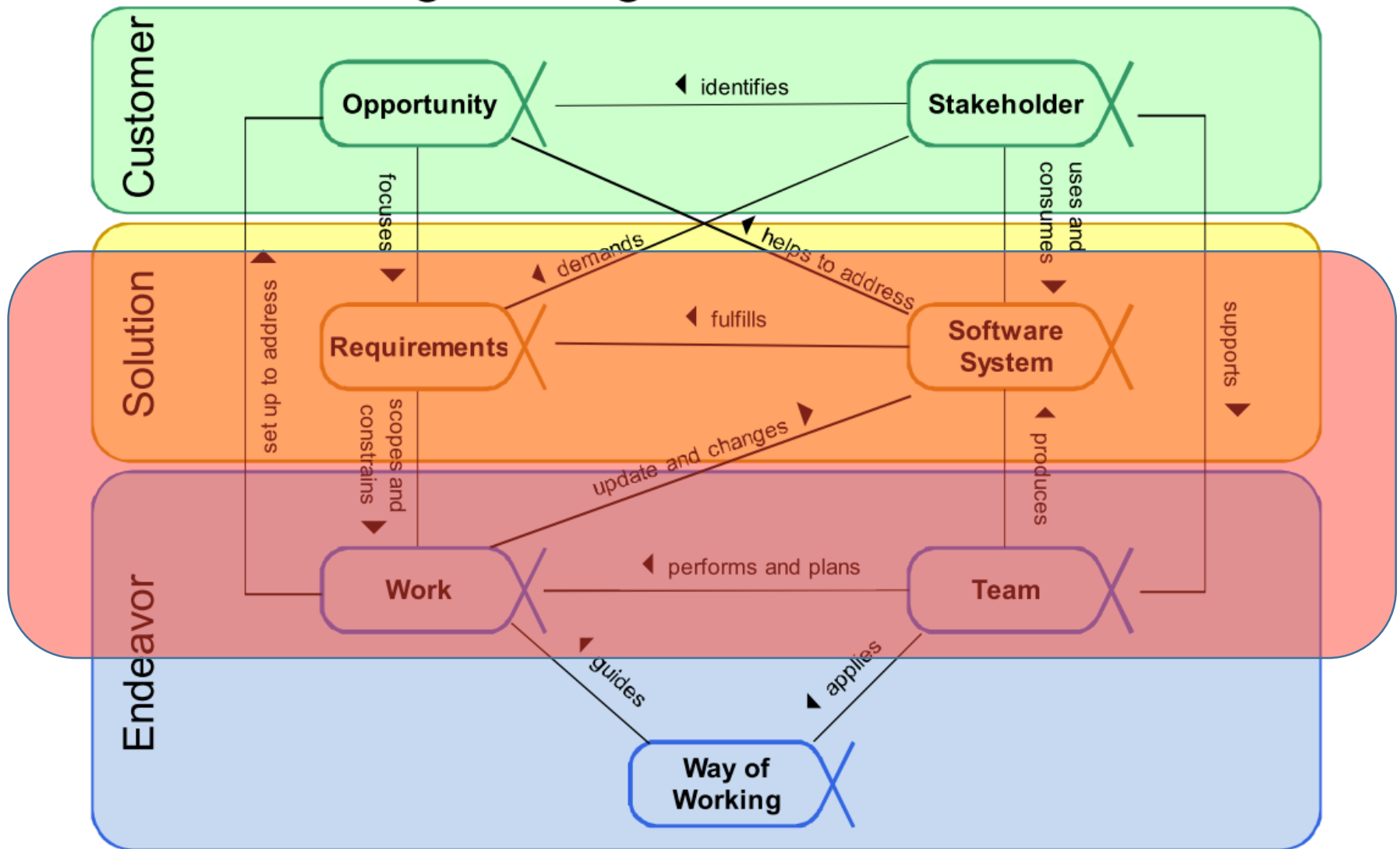


Collecting the evidence



- O time dá documentos para Beto
 - Casos de Negócio
 - Modelo de Caso de Uso
 - Visão do Sistema
- Beto faz algumas perguntas
- O que perguntar?

O Kernel - Alphas



Qual a situação atual?

Requirements

Software System

Work

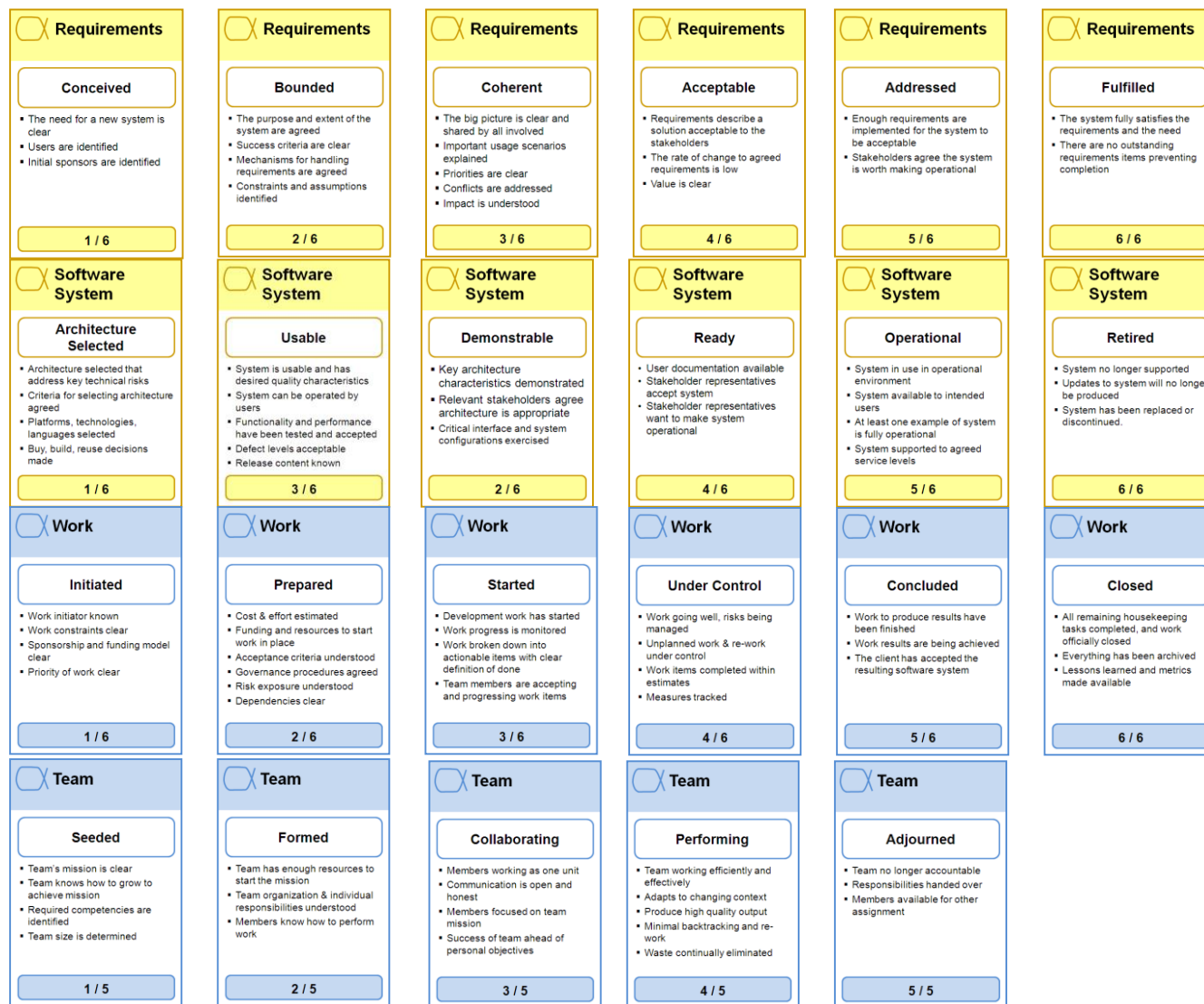
Team

Requirements <ul style="list-style-type: none"> Conceived The need for a new system is clear Users are identified Initial sponsors are identified 1 / 6	Requirements <ul style="list-style-type: none"> Bounded The purpose and extent of the system are agreed Success criteria are clear Mechanisms for handling requirements are agreed Priorities are clear Conflicts are addressed Impact is understood 2 / 6	Requirements <ul style="list-style-type: none"> Coherent The big picture is clear and shared by all involved Important usage scenarios explained Priorities are clear Conflicts are addressed Impact is understood 3 / 6	Requirements <ul style="list-style-type: none"> Acceptable Requirements describe a solution acceptable to the stakeholders The rate of change to agreed requirements is low Value is clear 4 / 6	Requirements <ul style="list-style-type: none"> Addressed Enough requirements are implemented for the system to be acceptable Stakeholders agree the system is worth making operational 5 / 6	Requirements <ul style="list-style-type: none"> Fulfilled The system fully satisfies the requirements and the need There are no outstanding requirements items preventing completion 6 / 6
Software System <ul style="list-style-type: none"> Architecture Selected Architecture selected that address key technical risks Criteria for selecting architecture agreed Platforms, technologies, languages selected Buy, build, reuse decisions made 1 / 6	Software System <ul style="list-style-type: none"> Usable System is usable and has desired quality characteristics System can be operated by users Functionality and performance have been tested and accepted Defect levels acceptable Release content known 3 / 6	Software System <ul style="list-style-type: none"> Demonstrable Key architecture characteristics demonstrated Relevant stakeholders agree architecture is appropriate Critical interface and system configurations exercised 2 / 6	Software System <ul style="list-style-type: none"> Ready User documentation available Stakeholder representatives accept system Stakeholder representatives want to make system operational 4 / 6	Software System <ul style="list-style-type: none"> Operational System in use in operational environment System available to intended users At least one example of system is fully operational System supported to agreed service levels 5 / 6	Software System <ul style="list-style-type: none"> Retired System no longer supported Updates to system will no longer be produced System has been replaced or discontinued. 6 / 6
Work <ul style="list-style-type: none"> Initiated Work initiator known Work constraints clear Sponsorship and funding model clear Priority of work clear 1 / 6	Work <ul style="list-style-type: none"> Prepared Cost & effort estimated Funding and resources to start work in place Acceptance criteria understood Governance procedures agreed Risk exposure understood Dependencies clear 2 / 6	Work <ul style="list-style-type: none"> Started Development work has started Work progress is monitored Work broken down into actionable items with clear definition of done Team members are accepting and progressing work items 3 / 6	Work <ul style="list-style-type: none"> Under Control Work going well, risks being managed Unplanned work & re-work under control Work items completed within estimates Measures tracked 4 / 6	Work <ul style="list-style-type: none"> Concluded Work to produce results have been finished Work results are being achieved The client has accepted the resulting software system 5 / 6	Work <ul style="list-style-type: none"> Closed All remaining housekeeping tasks completed, and work officially closed Everything has been archived Lessons learned and metrics made available 6 / 6
Team <ul style="list-style-type: none"> Seeded Team's mission is clear Team knows how to grow to achieve mission Required competencies are identified Team size is determined 1 / 5	Team <ul style="list-style-type: none"> Formed Team has enough resources to start the mission Team organization & individual responsibilities understood Members know how to perform work 2 / 5	Team <ul style="list-style-type: none"> Collaborating Members working as one unit Communication is open and honest Members focused on team mission Success of team ahead of personal objectives 3 / 5	Team <ul style="list-style-type: none"> Performing Team working efficiently and effectively Adapts to changing context Produce high quality output Minimal backtracking and re-work Waste continually eliminated 4 / 5	Team <ul style="list-style-type: none"> Adjourned Team no longer accountable Responsibilities handed over Members available for other assignment 5 / 5	

Plano: Determine Estado Atual



Estado Atual!



Cliente

- Todo projeto de desenvolvimento de software tem ao menos um cliente para o software que ele produz
- Qualquer processo adotado precisa integrar a perspectiva do cliente no projeto para prevenir que uma solução não apropriada seja produzida
- Lembrem do CHAOS Report:

FIM