REQUISITOS E SISTEMAS REATIVOS

[TE-265 ENGENHARIA DE SISTEMAS BASEADA EM MODELOS] [2023]

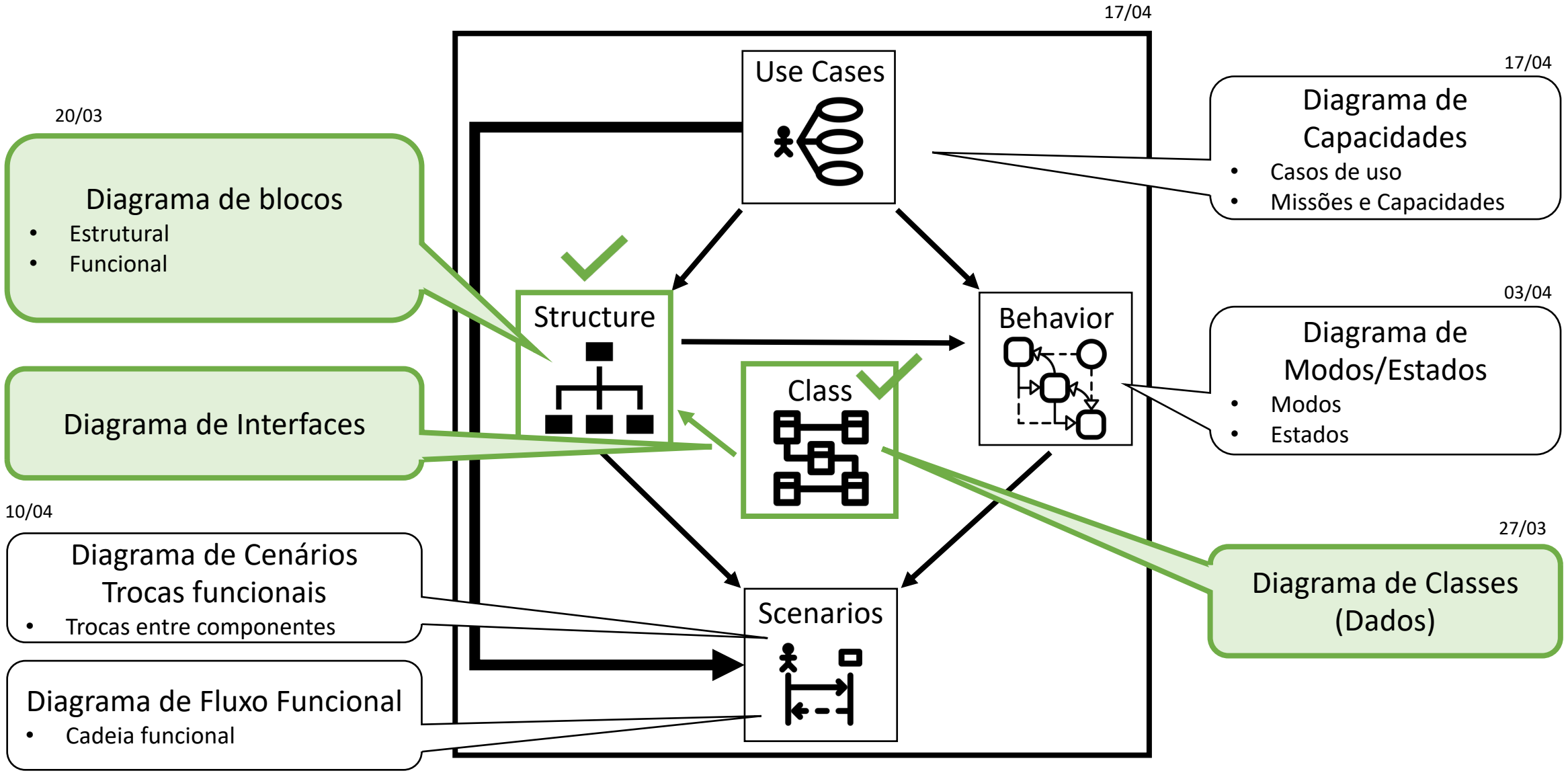
AULA	TEORIA	INDIVIDUAL	GRUPO
1 06-Mar	Introdução e Apresentação de Engenharia de Sistemas	Resumo dos princípios	Definição do grupo. Montagem de apresentação do tema.
2 13-Mar	Pitch: Temas Frameworks e Stakeholders	Trabalho sobre MBSE	Elicitar stakeholders
3 20-Mar	Arquitetura e Funções. Coesão e acoplamento.	Exercícios de análise estruturada	Mapa de interações
4 27-Mar	Ciclo de Vida e CONOPs		Ciclo de Vida e CONOPs
5 03-Apr	Pitch: Descrição livre da captura do problema Requisitos	Exercícios de correção de requisitos	Requisitos dos stakeholders
6 10-Apr	Modelagem Estrutural da Arquitetura	Exercícios de fixação	Relatório e Gravação de 5min com explicação
7 17-Apr	Modelagem de Comportamento da Arquitetura	Exercícios de fixação	
8 24-Apr	P1 - Questões conceituais e Mini-Case		

AULA	TEORIA	GRUPO
9 08-May	Introdução ao Arcadia e Análise do Contexto	Análise do Contexto e apontamento de necessidades
10 15-May	Pitch: Análise do Contexto - NOP Intervenção Sistêmica	Intervenção Sistêmica e requisitos da missão do sistema
11 22-May	Pitch: Análise do Sistema - ROP OPM e Exploração de Alternativas	Montagem de Alternativas
12 29-May	Pitch: Alternativas Arquitetura Conceitual e desdobramentos	Arquitetura Conceitual e requisitos do sistema
13 05-Jun	Pitch: Arquitetura Conceitual - RTLI Revisita de Requisitos e o processo de Verificação e Validação	Apontamento de etividades de verificação
14 12-Jun	Pitch: Propostas de Verificação Arquitetura Concreta e Carta Morfológica	Arquitetura Concreta com decisões tecnologicas
15 19-Jun	Pitch: Arquitetura Final - Especificações Revisão e desdobramentos para especialidades	Relatório e Gravação de 5min com explicação.
16 26-Jun	P2 - Apresentação final da relação entre as etapas	

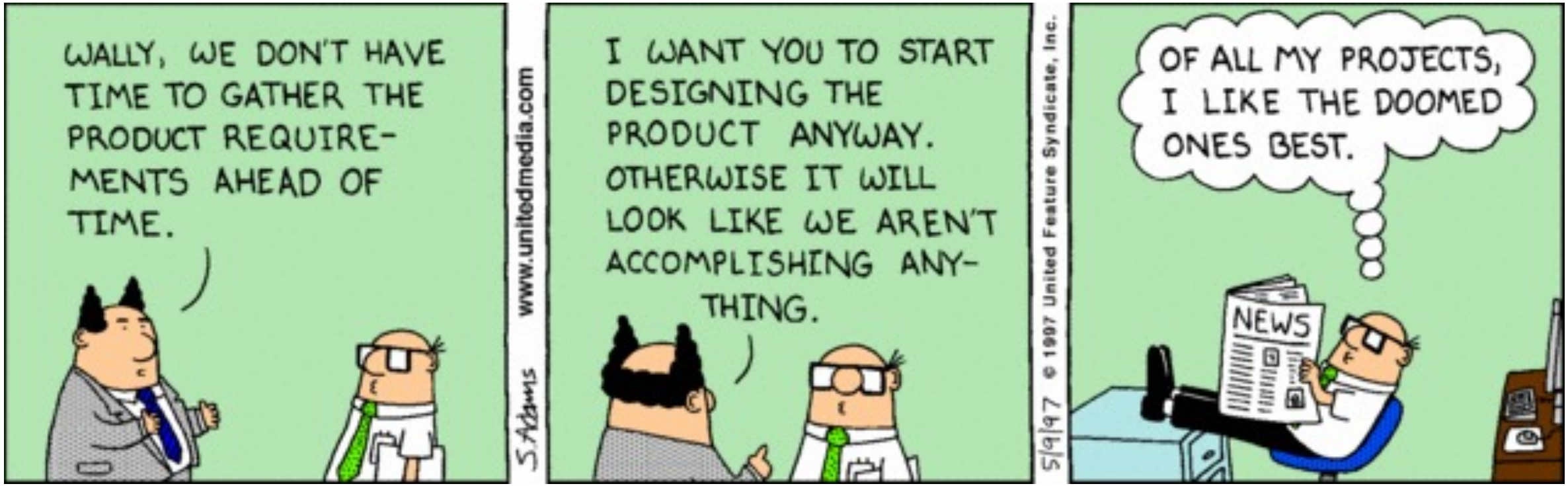
EXAME	Graduação (grupo): Desenvolvimento de um mini-case de um subsistema usando o Capella
03-Jul	Pós-graduação (grupo): Entrega de um artigo (Formato do SIGE) descrevendo seu case e atividades.
17-Jul	

AULA 05	Requisitos e Sistemas Reativos (Rev2)				
OBJETIVOS	05-01 - Introduzir o tema da Engenharia de Requisitos 05-02 - Introduzir o diagrama de máquina de estados				
DATA:	03-Apr				
	TÍTULO	#	TÓPICO	ATIVIDADE INDIVIDUAL	ATIVIDADE EM GRUPO
HORA 01	Pitchs				
		1			
		2			
		3			
		4			
		5			
HORA 02	Requisitos				
		1	Definição		
		2	Tipos de Requisitos: funcionais e não funcionais		
		3	Árvore de requisitos e rastreabilidade		
		4	Critério de Sucesso		
		5	Padrões de sintaxe	Exercícios de correção de requisitos	Escrever 5 requisitos funcionais e 2 não funcionais de usuário.
HORA 03	SysML - Máquina de Estados				
		1	Sistemas Reativos		
		2	Diagrama de máquina de estados	Exercícios de fixação	
		3			
		4			
		5			
			*** Próximas atividades**		

ROTEIRO DOS DIAGRAMAS



REVISÃO DE REQUISITOS


WALLY, WE DON'T HAVE TIME TO GATHER THE PRODUCT REQUIRE-MENTS AHEAD OF TIME.
I WANT YOU TO START DESIGNING THE PRODUCT ANYWAY. OTHERWISE IT WILL LOOK LIKE WE AREN'T ACCOMPLISHING ANY-THING.
OF ALL MY PROJECTS, I LIKE THE DOOMED ONES BEST.
NEWS
www.unitedmedia.com
S.Adams
5/9/97 © 1997 United Feature Syndicate, Inc.

DEFINIÇÕES DA IEEE STD 1220-1994.

- **Requisito.** Uma declaração que identifica uma capacidade, característica ou fator de qualidade que limita uma necessidade de produto ou processo para a qual uma solução será buscada.
  - **Requirement.** A statement identifying a capability, physical characteristic, or quality factor that bounds a product or process need for which a solution will be pursued.

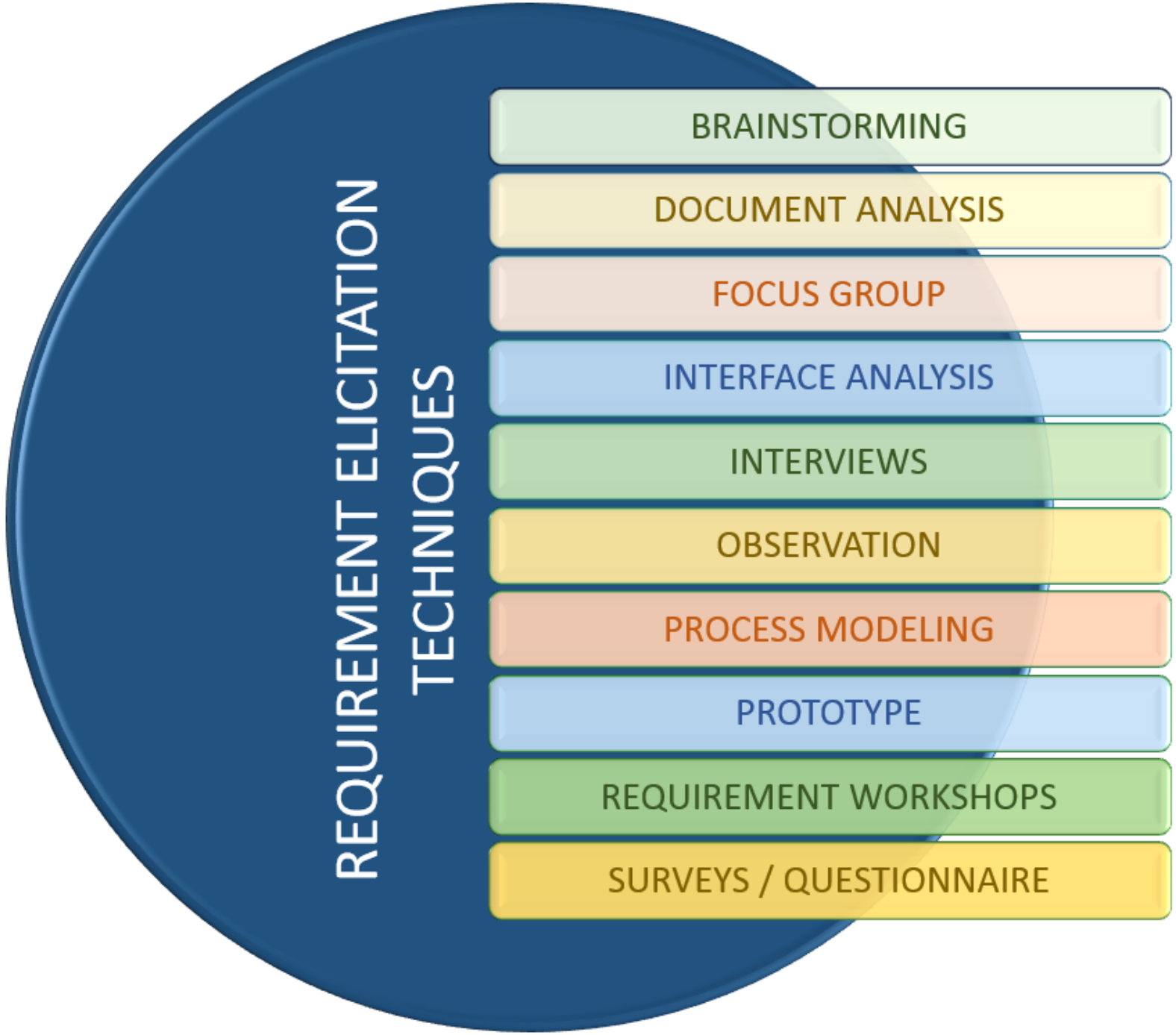
- **Restrição.** Uma limitação ou requisito mandatório que restringe a solução de projeto ou a implementação do processo de engenharia de sistemas, não é alterável pela atividade executora.
  - **Constraint.** A limitation or implied requirement that constrains the design solution or implementation of the systems engineering process, is not changeable by the performing activity, and is generally non allocable.

- **Especificação.** Um documento que descreve completamente um elemento físico ou suas interfaces em termos de requisitos (funcionais, desempenho, restrições e características físicas) e as condições e procedimentos de qualificação para cada requisito.
  - **Specification.** A document that fully describes a physical element or its interfaces in terms of requirements (functional, performance, constraints and physical characteristics) and the qualification conditions and procedures for each requirement.

IMPORTÂNCIA DE TER BONS REQUISITOS

- Os requisitos informam **o que o sistema precisa fazer** (requisitos funcionais).
- **Quão bem** o sistema precisa fazer isso (requisitos de desempenho)
- **Em que ambiente** o sistema tem de funcionar (requisitos ambientais).
- O que o sistema deve fazer **para se encaixar** em outros sistemas (requisitos de interface).
- O que os subsistemas/montagens/componentes de para fazer com que tudo funcione (alocação de requi**nível inferior devem fazer**sitos/recursos).
- O que você precisa fazer **antes de operar** (atividades de verificação).
- E, basicamente, quando você terminar (os requisitos são atendidos).
-

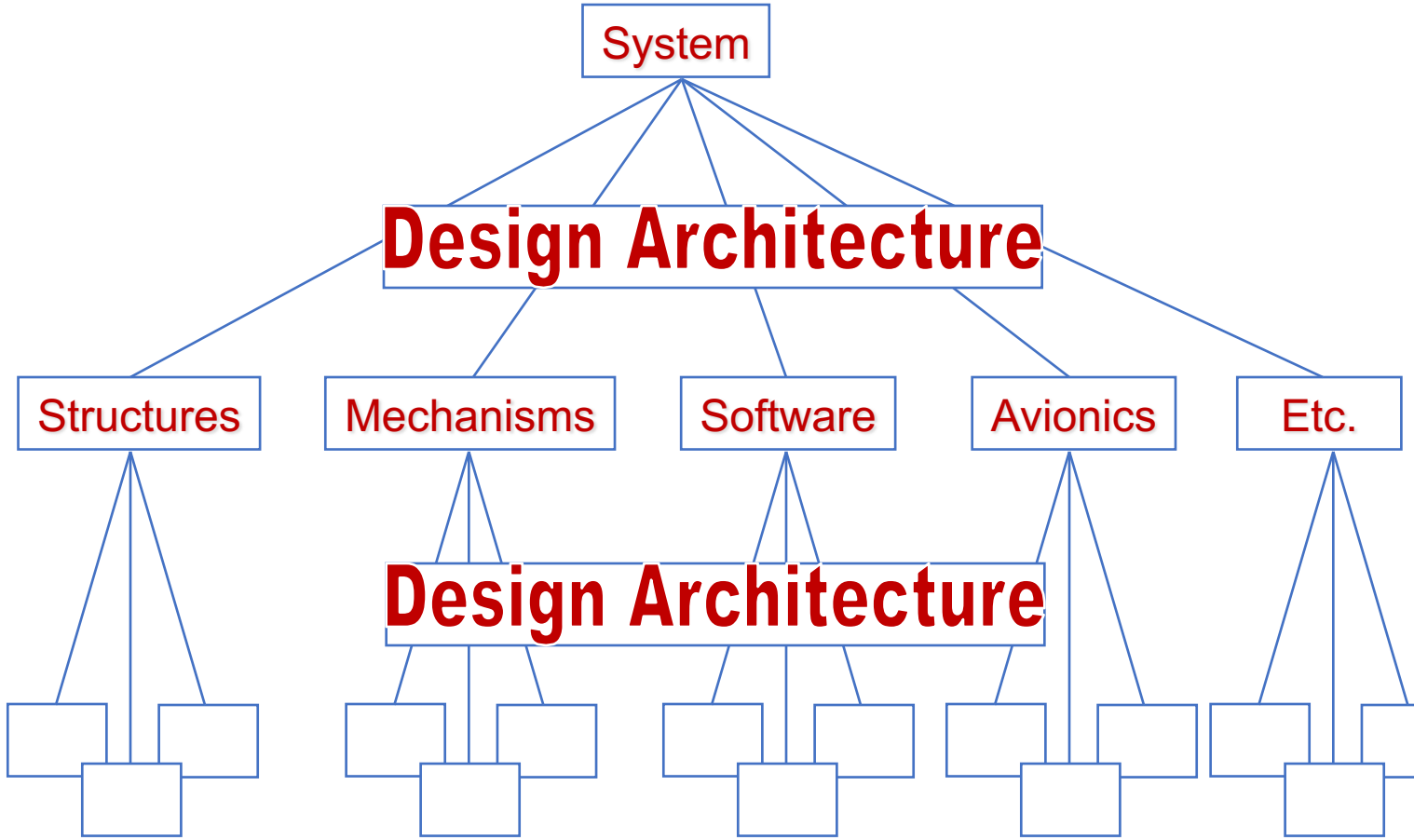

I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?
© Scott Adams, Inc./Dist. by UFS, Inc.


REQUIREMENT ELICITATION TECHNIQUES
BRAINSTORMING
DOCUMENT ANALYSIS
FOCUS GROUP
INTERFACE ANALYSIS
INTERVIEWS
OBSERVATION
PROCESS MODELING
PROTOTYPE
REQUIREMENT WORKSHOPS
SURVEYS / QUESTIONNAIRE

DECOMPOSIÇÃO E DERIVAÇÃO DOS REQUISITOS

- **Os requisitos são desenvolvidos em níveis diferentes de hierarquia**, você começa com os requisitos alocados a você (com seus requisitos pai), mas não é aí que você para!
  - O mesmo processo, identificação de stakeholders, ConOps, etc., devem/podem ser aplicado em **todos os níveis** da hierarquia de requisitos.
  - O tamanho e a profundidade dessas atividades podem diminuir, e alguns produtos podem ser combinados ou informais, mas você deve passar pelo mesmo processo de pensamento para garantir que o conjunto de requisitos de cada nível esteja **consistente, completo e correto**.

EXEMPLO DE DECOMPOSIÇÃO



O que o sistema deve fazer

O que os subsistemas devem fazer para o o Sistema

O que os componentes devem fazer para o subsistema

RASTREABILIDADE

- Rastreabilidade é a relação de requisitos para cima e para baixo na hierarquia de requisitos.
  - Normalmente, isso é **pai para filho (e o inverso)**, mas também pode ser ponto a ponto.
  - Ajuda a confirmar a **alocação completa e precisa** dos requisitos
  - **Todos os requisitos pais têm filhos?**
    - **Todos os requisitos filhos têm pais (ou são órfãos)?**
  - Ajuda a identificar o "**padrão ouro**", verificando a adição de requisitos desnecessários.
  - Vital para avaliar o **impacto das mudanças** nos requisitos.
  - *Obviamente, uma Ferramenta de Gerenciamento de Requisitos (banco de dados) é muito útil.*
    - *Mas também usamos tabelas de rastreabilidade em documentos do Word/Excel.*

INFORMAÇÕES DE FUNDAMENTAÇÃO(ARRAZOADO)

- **Razões**: **Muitas vezes, a razão para o requisito não é óbvia**, e pode ser perdido se não for registrado quando o requisito está sendo documentado. A razão pode apontar para uma restrição ou conceito de operações. Se houver um requisito de origem claro ou um estudo comercial que explique o motivo, ele deve ser referenciado.
- **Premissas**: Se um requisito foi escrito assumindo a conclusão de um programa de desenvolvimento de tecnologia ou uma missão de tecnologia bem-sucedida, a premissa deve ser documentada.
- **Relações de documentos**: As relações com as operações esperadas do produto (por exemplo, expectativas sobre como as partes interessadas usarão um produto) devem ser documentadas. Isso pode ser feito com um link para o ConOps.
- **Restrições de projeto**: As restrições impostas pelas decisões tomadas à medida que o projeto evolui devem ser documentadas. Se o requisito indicar um método de implementação, a fundamentação deve indicar a razão da tomada a decisão de limitar a solução a este único método de implementação.

EXEMPLO DE MODELO DE ESTRUTURA

- Os requisitos são mais comumente expressos como declarações de linguagem natural, embora as linguagens de requisitos matemáticos gráficos e formais também sejam usadas.
- Usando linguagem natural, as métricas de qualidade de requisitos podem ser desenvolvidas por meio da análise de cada declaração de requisito nos elementos de um modelo estrutural de um bom requisito.

EXEMPLO DE UMA ESTRUTURA

- **Actor**. Este é o sujeito da sentença - a coisa que está sendo especificada. Exemplos (bons e ruins!) são: “the system”, “the interface”, “the function”, ...
- **Conditions of Action**. Isso define as condições durante as quais a ação deve ocorrer, por exemplo. “in Replay Mode”, e/ou as condições iniciais: “upon receipt of a message”, “power having been applied”, ...
- **Action**. Este é um verbo - a ação a ser tomada pelo ator (sujeito). Exemplos são: “shall calculate”, “shall display”, “shall fly”, “shall be displayed”.....
- **Constraints of Action**. Estes qualificam a ação, por exemplo. “at a resolution of 400 x 1000 pixels”, “within limits imposed by vehicle speed”, ..., e incluir desempenho
- **Object of Action**. Este é um substantivo, e é a coisa sobre a qual se age ao tomar a ação. São exemplos: “the message”, “the input signal”, ...
- **Refinement/Source of Object**. Estes qualificam o objeto, por exemplo (refinamento): “of flash priority", por exemplo (fonte): “from DISCON”.
- **Refinement/Destination of Action**. Estes qualificam ainda mais a ação e podem ser adicionais às Restrições de Ação. “in accordance with IEEE 802.11g”, “to DISCON”.

EXEMPLO UTILIZANDO O TEMPLATE

Element	Text
Actor:	The system,
Condition:	upon receipt of a message,
Action:	shall switch
Object of Action:	that message
Constraints of Action:	within 10 milliseconds of receipt
Refinement of Object:	for messages in ACP128 format having a valid routing indicator
Source of Object:	from the message input port,
Destination of Object:	to a message output port,
(Further) Refinement of Action:	corresponding to the routing indicator in the message.

METADADOS

TABLE 4.2-2 Requirements Metadata

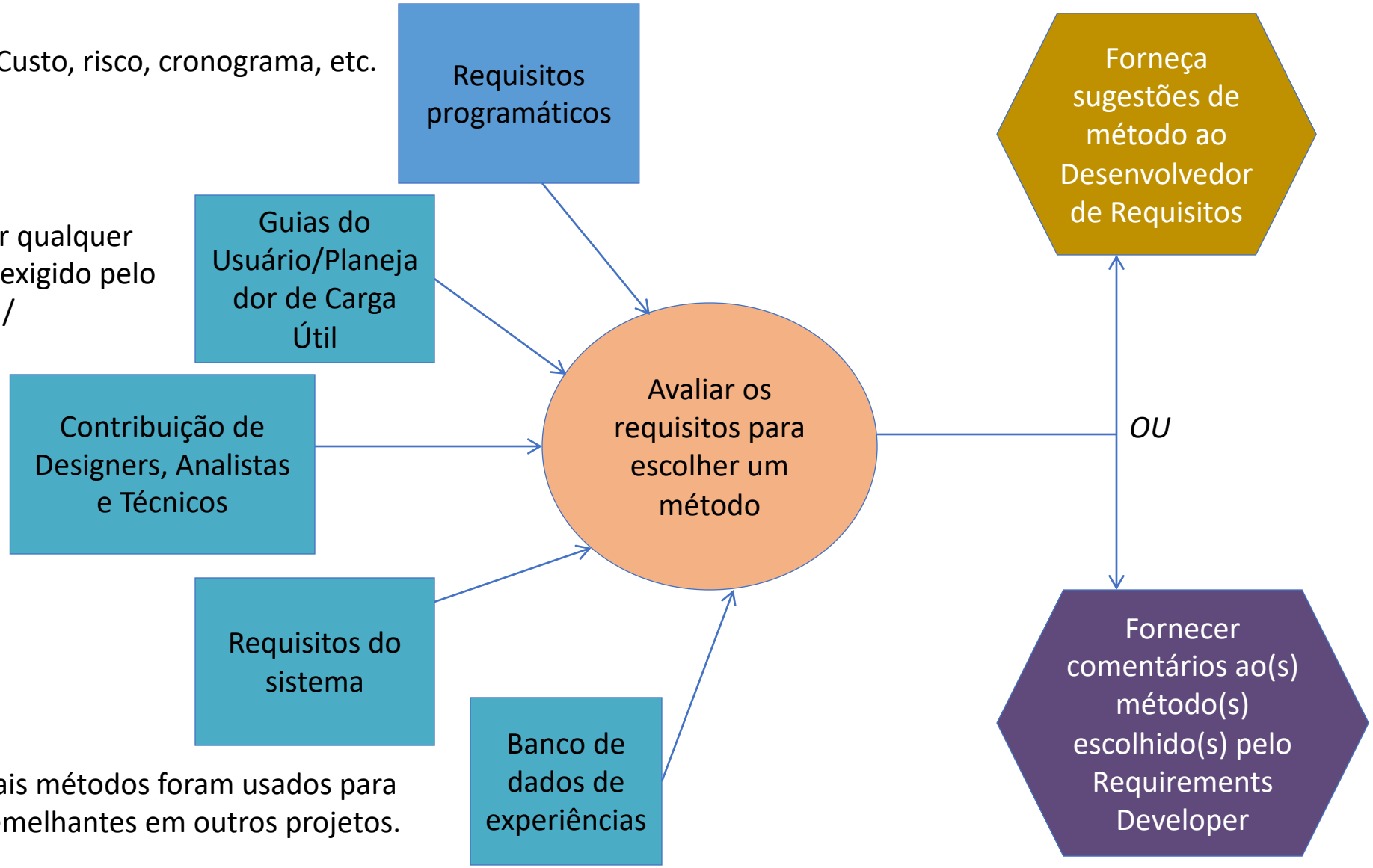
Item	Function
Requirement ID	Provides a unique numbering system for sorting and tracking.
Rationale	Provides additional information to help clarify the intent of the requirements at the time they were written. (See “Rationale” box below on what should be captured.)
Traced from	Captures the bidirectional traceability between parent requirements and lower level (derived) requirements and the relationships between requirements.
Owner	Person or group responsible for writing, managing, and/or approving changes to this requirement.
Verification method	Captures the method of verification (test, inspection, analysis, demonstration) and should be determined as the requirements are developed.
Verification lead	Person or group assigned responsibility for verifying the requirement.
Verification level	Specifies the level in the hierarchy at which the requirements will be verified (e.g., system, subsystem, element).

ESCOLHENDO UM MÉTODO V&V

- Custo, risco, cronograma, etc.

- Revise-os para procurar qualquer método de verificação exigido pelo veículo de lançamento / transportador

- Pesquise quais métodos foram usados para requisitos semelhantes em outros projetos.



ESCOLHENDO UM MÉTODO

Teste é usado quando:

*Técnicas analíticas não produzem resultados adequados

* Existem modos de falha que comprometem a segurança, os sistemas espaciais ou os objetivos da missão

*Componentes associados a interfaces críticas do sistema

Demonstração é usada quando:

*A verificação das funções projetadas pode ser realizada através da observação

*Os resultados tendem a ser "aprovação/reprovação", "sim/não"

*Natureza mais subjetiva, como fatores humanos ou capacidade de manutenção

*Usa sistemas de voo reais ou representativos

A inspeção é usada quando:

*Desenhos, documentos ou dados podem ser verificados visualmente para verificar se as características físicas foram projetadas no produto

*Normalmente usado para características de projeto, métodos de construção, mão de obra, dimensões e registros

A análise é usada quando:

*Técnicas analíticas precisas são possíveis

*O teste não é uma opção econômica

*A verificação por inspeção não é adequada

*Técnicas e modelos analíticos foram validados

Design Requirement

3.2.2.15.34 Recovery Force Communications
The product shall provide a communications system capable of communicating with the recovery forces pre- and post- landing

SE – Translates Operational Objectives into Design Requirements
Design – Provides assessment of requirements implementation
Test – Provides assessment of requirements verifiability

Design Verification

Verification Objective	Pass / Fail (Success Criteria)
Perform Integrated System Test of the communications system capability to provide a voice communications and beacon with recovery forces pre and post landing within an integrated hardware / software environment	**Testing will show that the communications system can transmit and receive audio at frequencies and ranges (power) represented by standard ground recovery force communications devices as defined in TBD**
Perform a demonstration of the communications systems capability to provide voice and beacon communications with recovery forces pre and post landing while within a representative environment and using a production equipment configuration	**Demonstration will show the ability for the communications systems to verbally communicate with the on-board communication production configuration equipment. The demonstration will also show beacon tracking within communication ranges established by TBD.**

SE – Provides compliance of the design requirement
Test / Implementation Group – Ensures Verification Implementation Feasibility
Advises alternatives to support programmatics
Assesses completeness
Provides verifiability assessment

Traceability

Verification Cross Reference Matrix

Paragraph #	N/A	I	A	M/S	D	T
3.2.2.15.34						VR-5T
3.2.2.15.34					VR-5D	

SE – Verification Allocation and Traceability Assurance

- **VR-5T: Prove that the product's communications system is capable of communicating with the ground command team by performing an integrated system test within an integrated hardware/software environment. Testing will show that the product can transmit and receive to standard ground recovery forces audio at frequencies represented by communications devices defined in (TBD).**

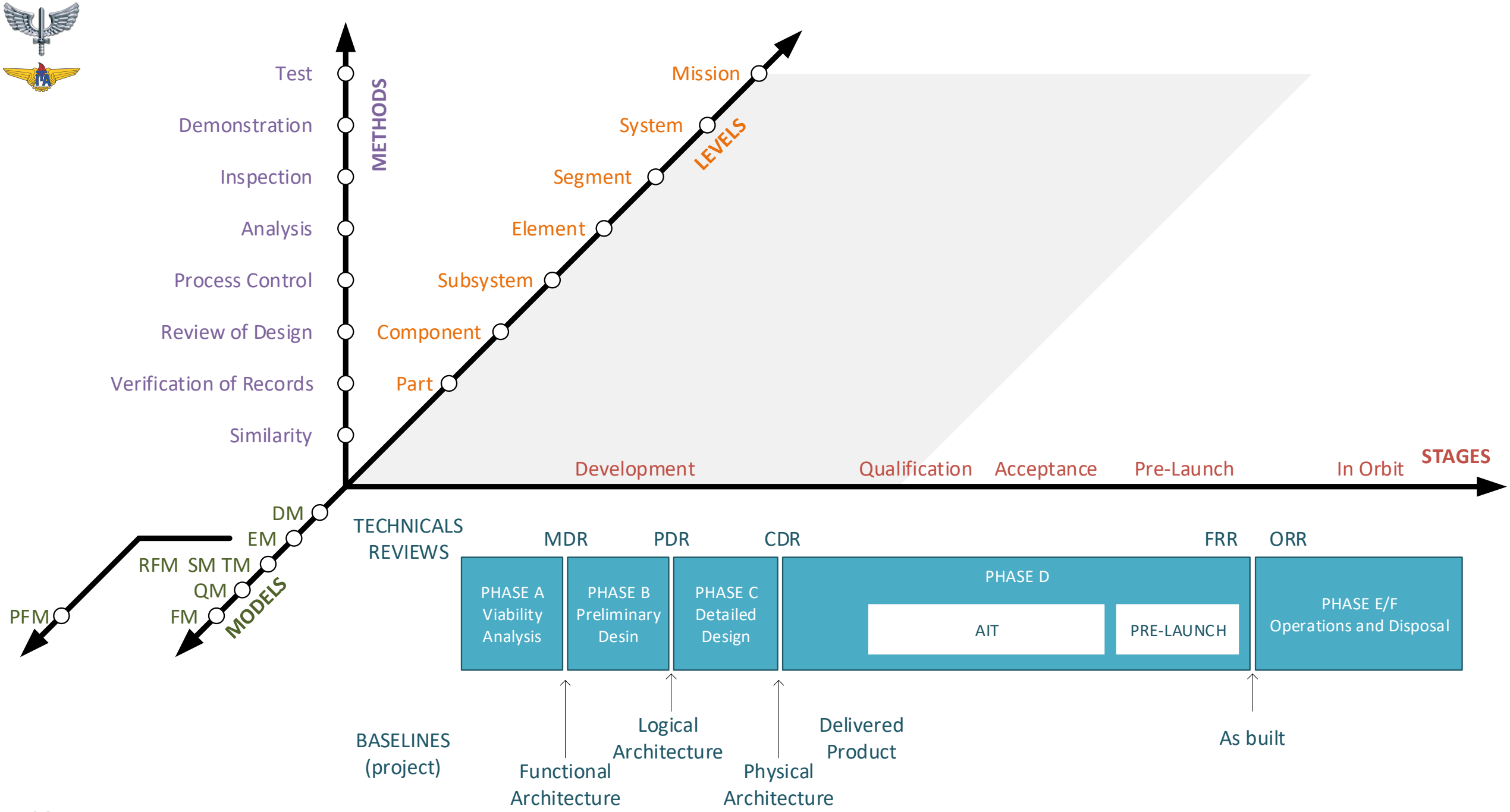
Verification Objective

Verification Method

Environment

Note – there are no Special Conditions

Success Criteria


METHODS
Test
Demonstration
Inspection
Analysis
Process Control
Review of Design
Verification of Records
Similarity
LEVELS
Mission
System
Segment
Element
Subsystem
Component
Part
STAGES
Development
Qualification
Acceptance
Pre-Launch
In Orbit
MODELS
DM
EM
RFM SM TM
QM
FM
PFM
TECHNICALS REVIEWS
MDR
PDR
CDR
FRR
ORR
PHASE A Viability Analysis
PHASE B Preliminary Desin
PHASE C Detailed Design
PHASE D
AIT
PRE-LAUNCH
PHASE E/F Operations and Disposal
BASELINES (project)
Functional Architecture
Logical Architecture
Physical Architecture
Delivered Product
As built

Appendix C: How to Write a Good Requirement—Checklist

C.1 Use of Correct Terms

- ☐ Shall = requirement
- ☐ Will = facts or declaration of purpose
- ☐ Should = goal

C.2 Editorial Checklist

Personnel Requirement

- ☐ The requirement is in the form "responsible party shall perform such and such." In other words, use the active, rather than the passive voice. A requirement should state who shall (do, perform, provide, weigh, or other verb) followed by a description of what should be performed.

Product Requirement

- ☐ The requirement is in the form "product ABC shall XYZ." A requirement should state "The product shall" (do, perform, provide, weigh, or other verb) followed by a description of what should be done.
- ☐ The requirement uses consistent terminology to refer to the product and its lower-level entities.
- ☐ Complete with tolerances for qualitative/performance values (e.g., less than, greater than or equal to, plus or minus, 3 sigma root sum squares).
- ☐ Is the requirement free of implementation? (Requirements should state WHAT is needed, NOT HOW to provide it; i.e., state the problem not the solution. Ask, "Why do you need the requirement?" The answer may point to the real requirement.)
- ☐ Free of descriptions of operations? (Is this a need the product should satisfy or an activity involving the product? Sentences like "The operator shall…" are almost always operational statements not requirements.)

Example Product Requirements

- ☐ The system shall operate at a power level of…
- ☐ The software shall acquire data from the…
- ☐ The structure shall withstand loads of…
- ☐ The hardware shall have a mass of…

C.3 General Goodness Checklist

- ☐ The requirement is grammatically correct.
- ☐ The requirement is free of typos, misspellings, and punctuation errors.
- ☐ The requirement complies with the project's template and style rules.
- ☐ The requirement is stated positively (as opposed to negatively, i.e., "shall not").
- ☐ The use of "To Be Determined" (TBD) values should be minimized. It is better to use a best

https://ntrs.nasa.gov/citations/20170001761

SISTEMAS REATIVOS E MÁQUINAS DE ESTADO

DEFINIÇÃO DE SISTEMAS REATIVOS

- Um sistema reativo é um sistema que, quando ligado, é capaz de criar os efeitos desejados em seu ambiente, **habilitando, aplicando ou impedindo** eventos no ambiente.
- Propriedades:
  - Interação contínua (sem terminação)
  - O sistema responderá a estímulos externos, e
  - a resposta depende do seu estado atual

SISTEMAS TRANSFORMACIONAIS

- Sistemas reativos contrastam com sistemas transformadores, **que existem para transformar uma entrada em uma saída**.

- Propriedades:
  - Apenas para transformar a entrada na saída.
  - Após a transformação, ele termina
  - Não tem estados que sejam relevantes para entidades externas

EXEMPLOS DE SISTEMAS REATIVOS

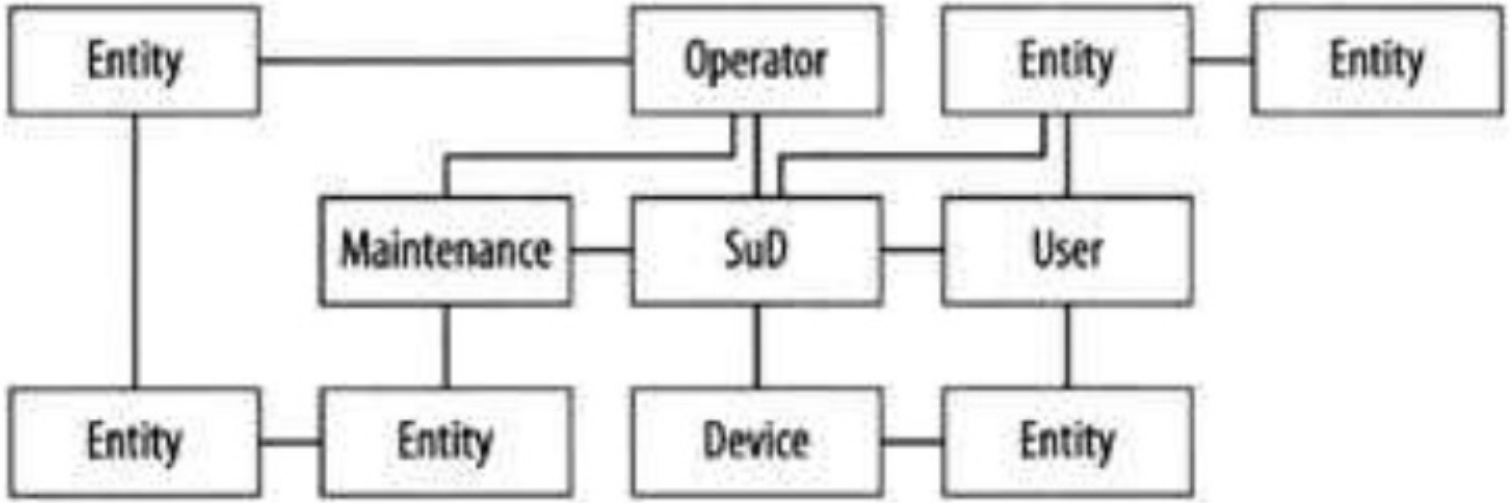
- **SISTEMAS EM TEMPO REAL** – a resposta depende do tempo. Ex.: software de controle, controle de elevador, caixas eletrônicos
- **SISTEMAS CRÍTICOS** – comportamentos errados podem levar à perda de vidas. Ex.: sistemas biofísicos, computadores de bordo (automóvel/aeronave/nave espacial/navio)
- **SISTEMAS EMBARCADOS** – restrições de implementação. Ex.: móvel, software de firmware integrado, IoT.

- Sistemas que gerenciam infraestruturas críticas: gerenciamento de tráfego aéreo, trem, reatores nucleares, etc.

O AMBIENTE

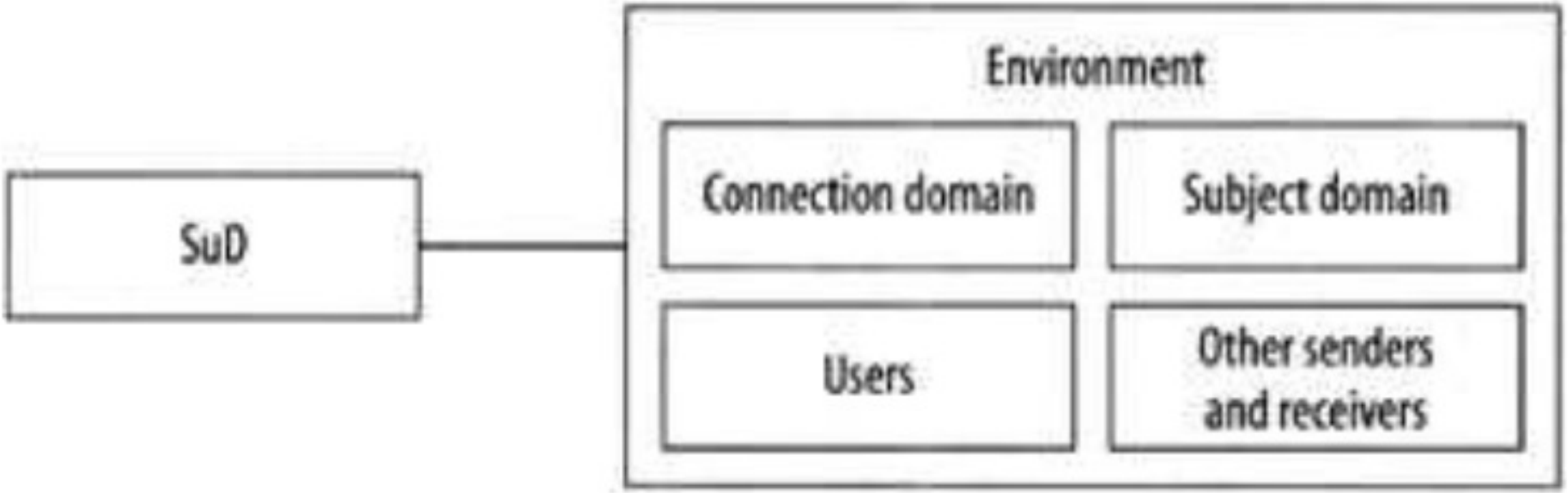
AMBIENTE

- Precisamos identificar a parte do mundo que é relevante como ambiente do sistema **– Identificar as interfaces do SuD**
  - **As interfaces do sistema podem mudar de acordo com o objetivo ou a fase de teste.**



AMBIENTE

- O ambiente de um sistema contém vários assuntos e domínios conectados.



DOMÍNIO DO ASSUNTO

- DOMÍNIO – domínio do conhecimento que define as dimensões do ambiente (domínio de controle de atitude: roda de reação, magnetômetro, sensor solar, etc.)
  - Entidades de domínio: roda de reação, magnetômetro, etc.
  - Eventos: aplicação de torque, leitura de medição

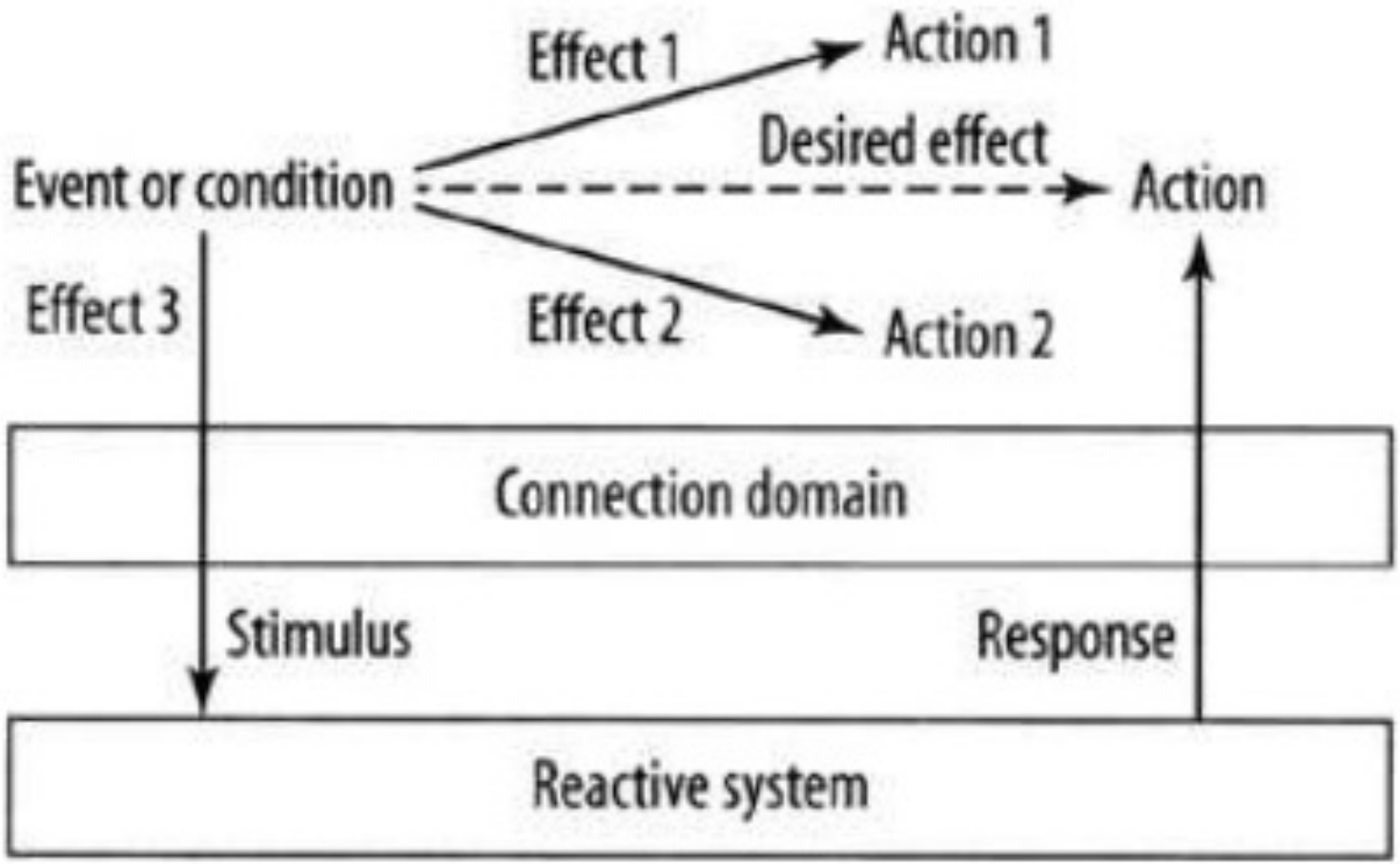
DOMÍNIO DE CONEXÃO

- Canais de comunicação que permitem a troca de mensagens entre o SuD e o Ambiente.
- Precisamos modelar o canal de comunicação porque isso pode levar a atrasos, dados corrompidos, etc. – **muitos erros acontecem nas interfaces.**

COMPORTAMENTO ESTÍMULO-RESPOSTA

CADEIAS DE CAUSA E EFEITO

- A função de um sistema reativo é **responder à ocorrência de eventos ou condições no ambiente**, causando mudanças no ambiente.



EVENTOS, CONDIÇÕES E AÇÕES

- **EVENTO** – algo que acontece no mundo.
  - **EVENTOS EXTERNOS**: mudança discreta na condição do ambiente
  - **EVENTOS TEMPORAIS**: passagem de um tempo significativo ao qual se espera que o sistema responda
- **CONDIÇÃO** – estado do mundo que persiste por algum período de tempo diferente de zero.
- **AÇÕES** – eventos do ponto de vista do iniciador

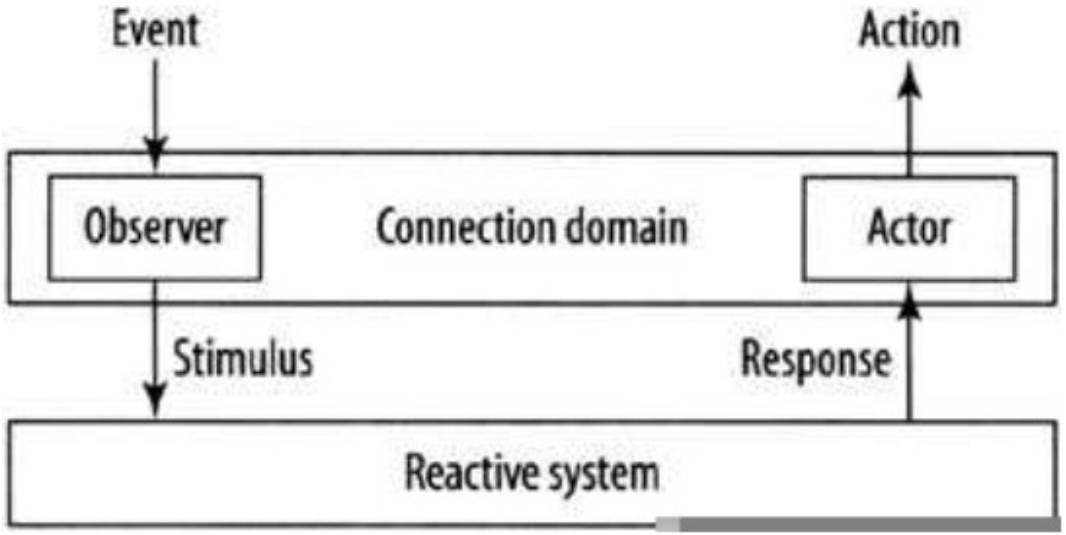
ESTÍMULOS

- O estímulo de um sistema é um **evento na interface do sistema causado pelo ambiente**.
- Eventos externos ocorrem em algum lugar do ambiente; os **estímulos ocorrem na interface** do sistema

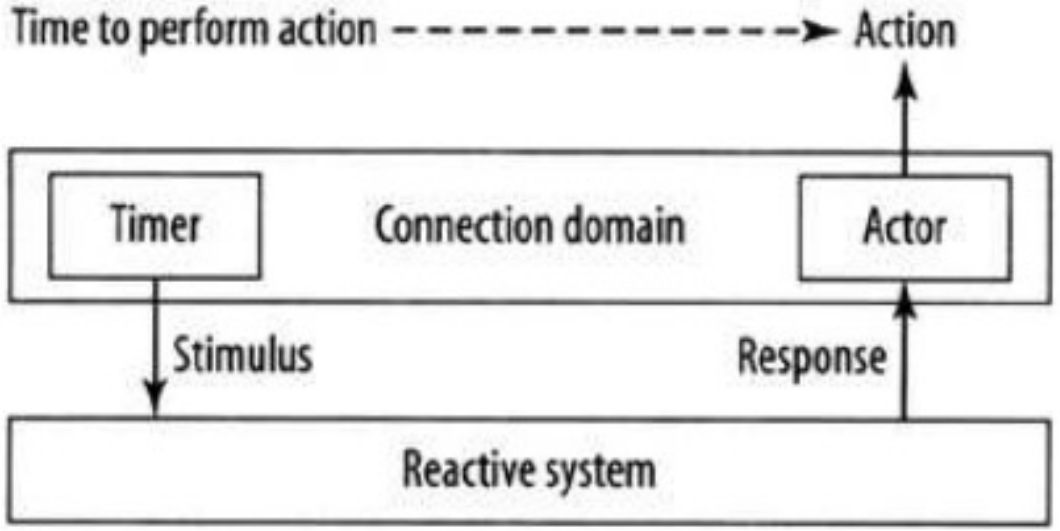
RESPOSTA

- O sistema reativo **responde** a um evento externo ou temporal atualizando seu estado **ou produzindo uma saída**.
- Uma resposta de saída de um sistema é um **evento na interface** do sistema, causado pelo sistema.
- O efeito desejado do estímulo pode consistir em uma ou mais ações desejadas, a serem causadas por várias respostas.

RESUMO



Eventos-Estímulos

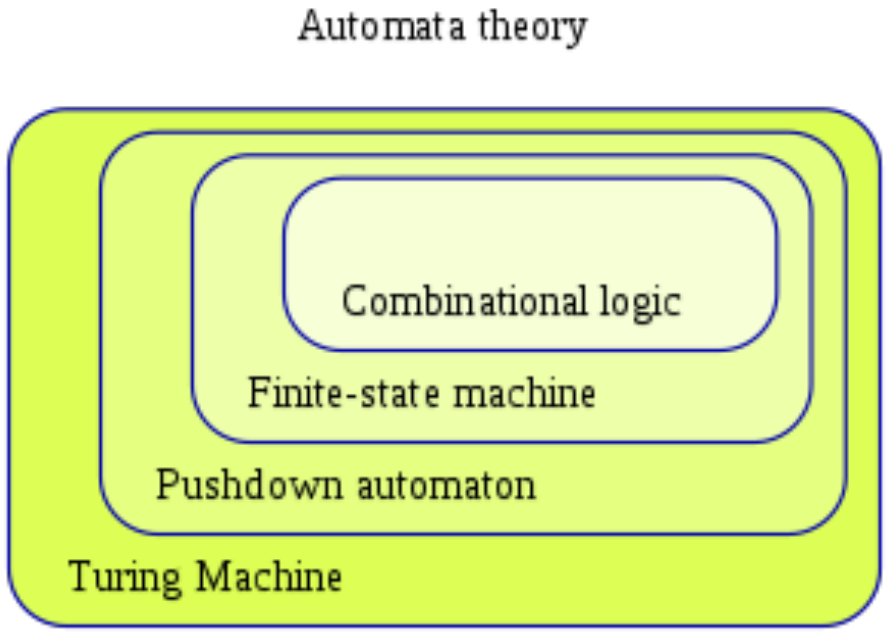


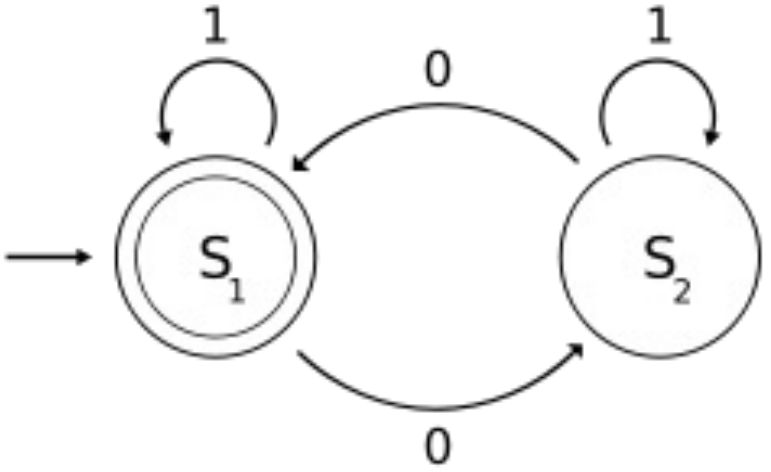
Eventos Temporais

DESCREVENDO COMPORTAMENTO

TEORIA DOS AUTÔMATOS

- Um autômato finito determinístico é representado formalmente por uma quintupla <Q,Σ,δ,q_0,F>, onde:
- Q é um conjunto finito de estados.
- Σ é um conjunto finito de símbolos, chamado de alfabeto do autômato.
- δ é a função de transição, ou seja, δ: Q × Σ → Q.
- q_0 é o estado inicial, ou seja, o estado do autômato antes de qualquer entrada ter sido processada, onde $q_0 \in Q$.
- F é um conjunto de estados de Q (ou seja, $F \subseteq Q$) chamados estados de aceitação.
-

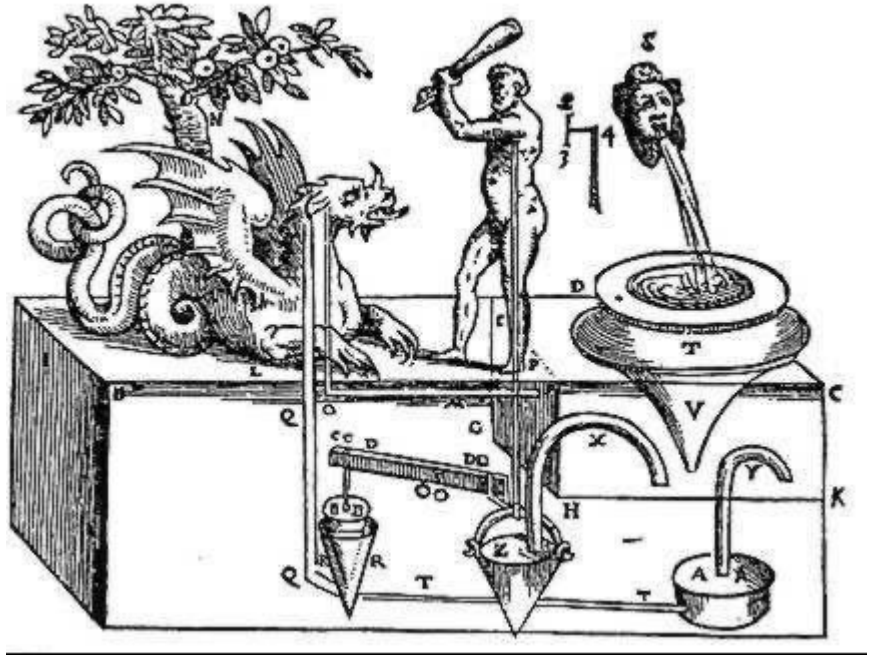




https://en.wikipedia.org/wiki/Automata_theory

AUTÔMATO

- A própria palavra autômato, intimamente relacionada à palavra "automação", denota processos automáticos que realizam a produção de processos específicos. Simplificando, a teoria dos autômatos lida com a lógica da computação em relação a máquinas simples, referidas como autômatos.
- Autômatos são modelos abstratos de máquinas que executam cálculos em uma entrada movendo-se através de uma série de estados ou configurações. Em cada estado da computação, uma função de transição determina a próxima configuração com base em uma porção finita da configuração atual.

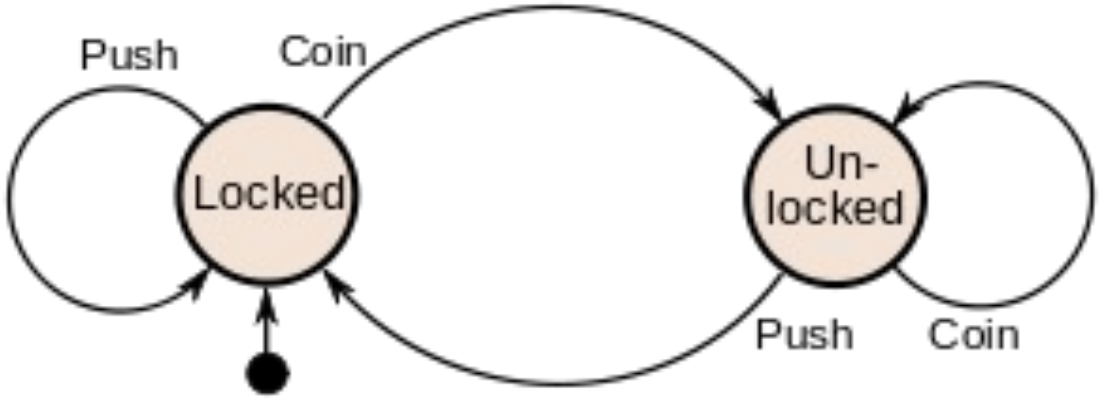


FERRAMENTA: MÁQUINAS DE ESTADO

- Uma máquina de estados finitos (FSM) ou autômato de estados finitos (FSA, plural: autômatos), autômato finito, ou simplesmente uma máquina de estados, é um modelo matemático de computação.
- **É uma máquina abstrata que pode estar em exatamente um de um número finito de estados a qualquer momento.**
- O FSM pode **mudar de um estado para outro** em resposta a algumas entradas externas; a mudança de um estado para outro é chamada de **transição**.
- Uma FSM é definida por uma lista de seus estados, seu estado inicial e as condições para cada transição.
-

https://en.wikipedia.org/wiki/Finite-state_machine

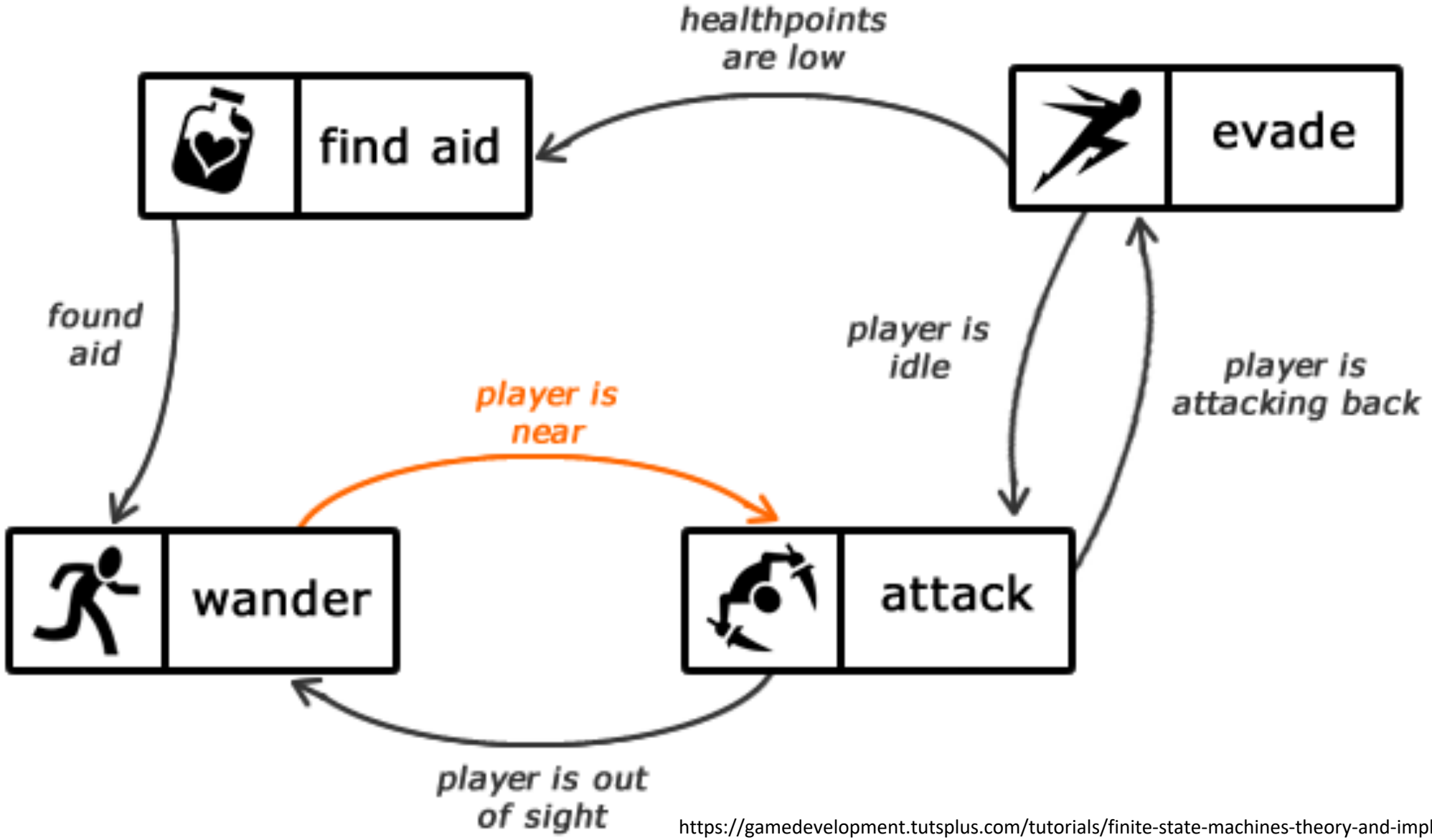
EXEMPLO



Current State	Input	Next State	Output
Locked	coin	Unlocked	Unlocks the turnstile so that the customer can push through.
	push	Locked	None
Unlocked	coin	Unlocked	None
	push	Locked	When the customer has pushed through, locks the turnstile.

UM EXEMPLO DE JOGO

FSM representando o cérebro de um inimigo.



https://gamedevelopment.tutsplus.com/tutorials/finite-state-machines-theory-and-implementation--gamedev-11867

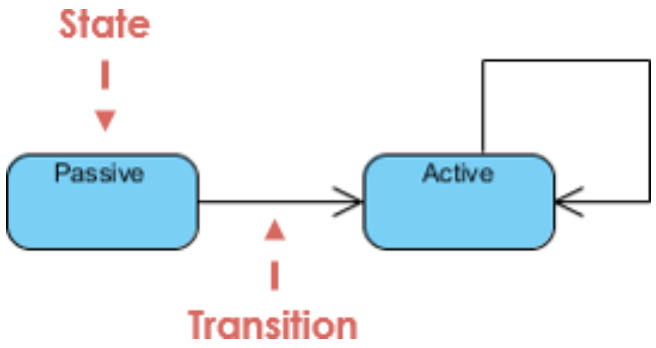
O DIAGRAMA DA MÁQUINA DE ESTADO

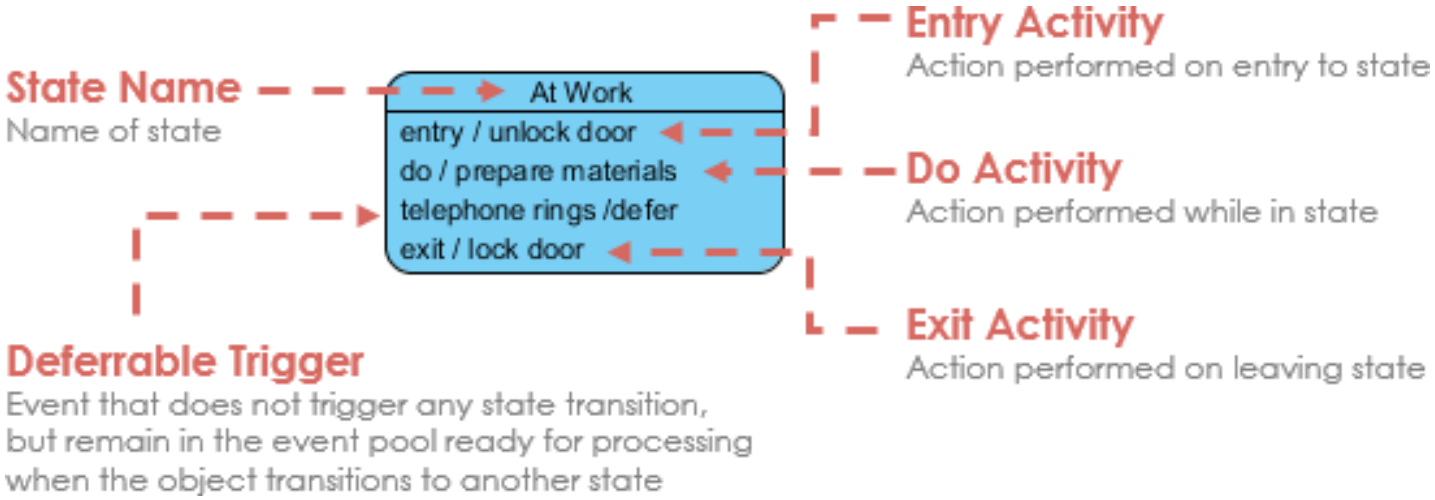
https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

VISÃO GERAL

- O comportamento de uma entidade não é apenas uma consequência direta de suas entradas, mas também depende de seu estado anterior. A **história passada de uma entidade pode ser melhor modelada** por um diagrama de máquina de estados finitos ou tradicionalmente chamado de autômatos.
- Os Diagramas de Máquina de Estado (ou às vezes chamados de diagrama de estado, máquina de estado ou gráfico de estado) mostram os diferentes estados de uma entidade.
- Os diagramas de máquina de estado também **podem mostrar como uma entidade responde a vários eventos** mudando de um estado para outro.

ESTADO

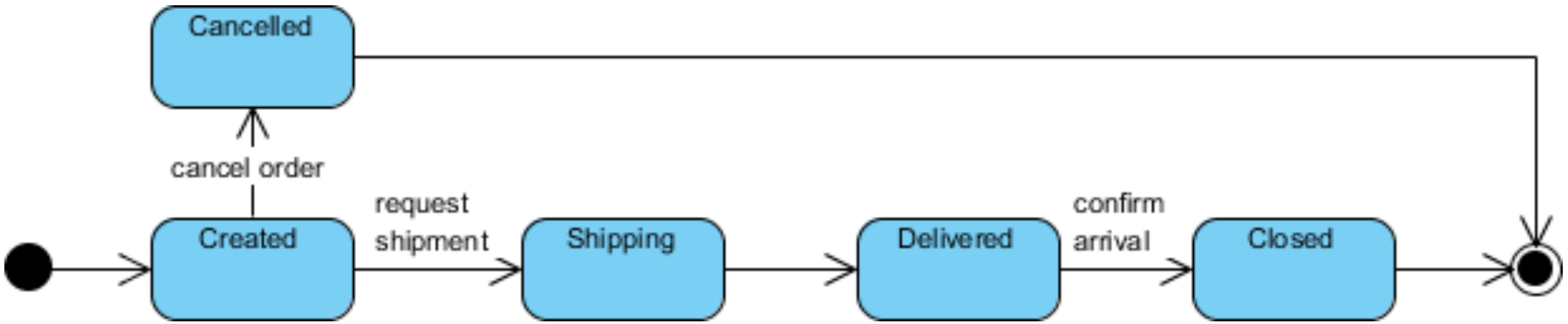




- Um estado é uma restrição ou uma situação no ciclo de vida de um objeto, na qual uma restrição se mantém, o objeto executa uma atividade ou aguarda um evento.
- Existem várias características dos estados em geral, independentemente de seus tipos:
  - Estado representa as condições dos objetos em determinados pontos no tempo.
  - Um estado é frequentemente associado a uma abstração de valores de atributo de uma entidade que satisfaz alguma(s) condição(ões).
  - Uma entidade muda seu estado não apenas como consequência direta da entrada atual, mas também depende de algum histórico passado de suas entradas.

ESTADOS INICIAIS E FINAIS

- O estado inicial de um diagrama de máquina de estados, conhecido como **pseudo-estado inicial**, é indicado com um círculo sólido. **Uma transição deste estado mostrará o primeiro estado real**
- O **estado final** de um diagrama de máquina de estado é mostrado como círculos concêntricos. Uma máquina de estado de **loop aberto representa um objeto que pode terminar antes que o sistema termine**, enquanto um diagrama de máquina de estado **de loop fechado não tem um estado final**; se for o caso, então o objeto vive até que todo o sistema termine.



TRANSIÇÃO

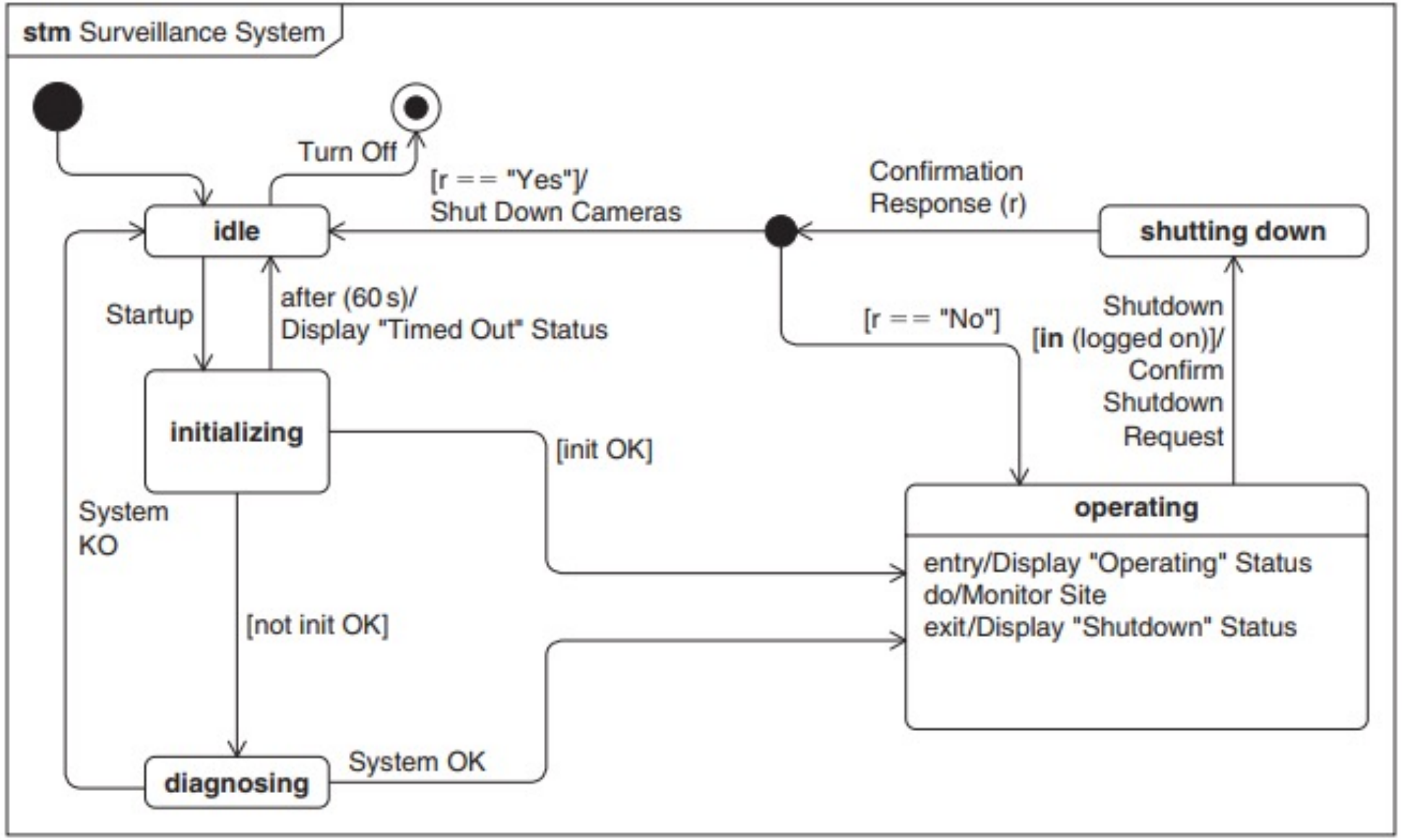
- **As linhas de transição retratam o movimento de um estado para outro**. Cada linha de transição é rotulada com o **evento que causa a transição**.
  - Compreender as transições de estado faz parte da análise e do projeto do Sistema
- Várias transições ocorrem quando eventos diferentes resultam no encerramento de um estado ou quando há condições de proteção (guarda) nas transições
- **Uma transição sem um evento e uma ação é conhecida como transições automáticas**

O DIAGRAMA DA MÁQUINA DE ESTADO DO SYSML

VISÃO GERAL

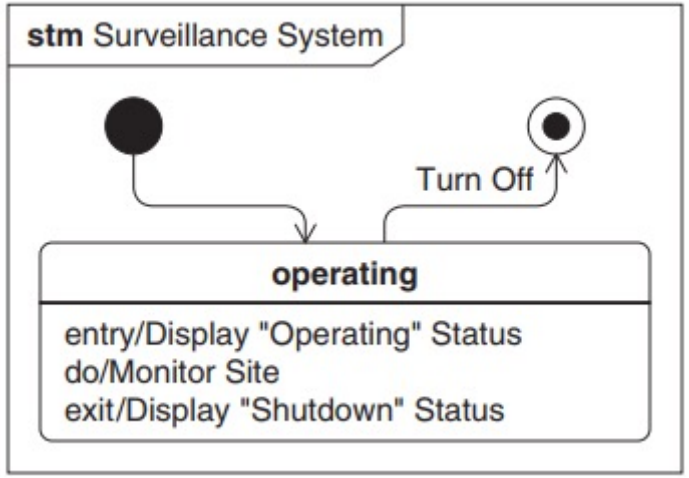
- As máquinas de estado são normalmente usadas para descrever o comportamento dependente do estado de um element da arquitetura ao longo de seu ciclo de vida, que é definido em termos de seus estados e as transições entre eles..
- A máquina de estado define como o comportamento do elemento muda à medida que ele transita entre diferentes estados e enquanto está em diferentes estados.
- As máquinas de estado podem ser usadas para descrever uma ampla gama de comportamentos relacionados à estados, desde um simples interruptor de lâmpada até os modos complexos de uma aeronave avançada.

EXEMPLO DE MÁQUINA DE ESTADO



ESPECIFICANDO ESTADOS

- **REGIÃO**: descrever o comportamento relacionado ao estado da máquina de estado.
- **ESTADO**: representa alguma condição significativa na vida de um elemento, normalmente porque representa alguma mudança na forma como responde aos eventos e quais comportamentos ele executa.
  - **Cada estado pode conter comportamentos de entrada e saída** que são executados sempre que o estado é inserido ou encerrado, respectivamente. Além disso, o estado pode **conter um comportamento que é executado uma vez que o comportamento de entrada tenha sido concluído**



TRANSIÇÕES DE ESTADO

- Uma transição especifica quando uma mudança de estado ocorre dentro de uma máquina de estado.
  - Trigger (Gatilho)
  - Guard (Guarda)
  - Effect (Efeito)

GATILHO

- Um gatilho identifica os possíveis estímulos que causam a ocorrência de uma transição. O SysML tem quatro tipos principais de eventos de disparo.
  - Um **evento de sinal** indica que uma nova mensagem assíncrona correspondente a um sinal chegou. Um evento de sinal pode ser acompanhado por vários argumentos que podem ser usados no efeito de transição.
  - Um **evento de tempo** indica que um determinado intervalo de tempo passou desde que o estado atual foi inserido (relativo) ou que um determinado instante no tempo foi atingido (absoluto).
  - Um **evento de alteração (mudança)** indica que alguma condição foi satisfeita (normalmente que algum conjunto específico de valores de atributo mantém).
  - Um **evento de chamada** indica que uma operação no bloco proprietário da máquina de estado foi solicitada. Um evento de chamada também pode ser acompanhado por vários argumentos.

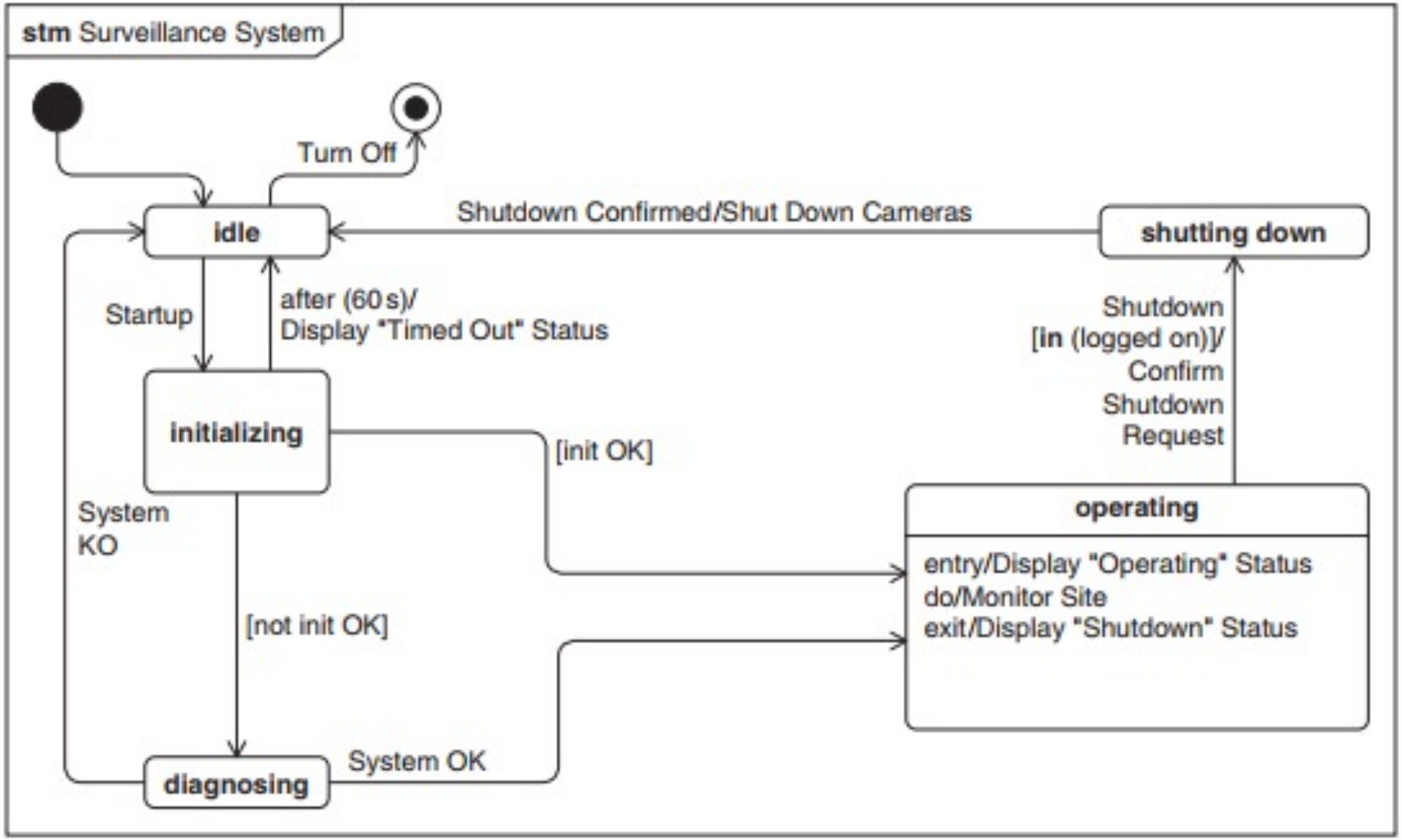
GUARDA

- A guarda de transição contém uma expressão que deve ser avaliada como verdadeira para que o efeito de uma transição ocorra ou a propria transição. A guarda é especificada usando uma restrição, que inclui uma expressão textual para representar a condição de guarda.
- Quando um evento satisfaz um gatilho, a proteção na transição é avaliada.
  - Se for avaliado como verdadeiro, a transição será acionada;
  - se for avaliado como falso, o evento será consumido sem efeito.

EFEITO

- O efeito é um **comportamento executado durante a transição** de um estado para outro.
- O efeito de transição pode ser um comportamento arbitrariamente complexo que pode incluir ações de sinal de envio ou chamadas de operação usadas para interagir com outros elementos.
  - Se a transição for acionada, primeiro o comportamento de saída do estado atual (origem) será executado, o efeito de transição será executado e, finalmente, o comportamento de entrada do estado de destino será executado.

TRANSIÇÕES


stm Surveillance System
Turn Off
idle
Shutdown Confirmed/Shut Down Cameras
shutting down
Startup
after (60 s)/
Display "Timed Out" Status
Shutdown
[in (logged on)]/
Confirm
Shutdown
Request
initializing
[init OK]
System
KO
operating
entry/Display "Operating" Status
do/Monitor Site
exit/Display "Shutdown" Status
[not init OK]
System OK
diagnosing

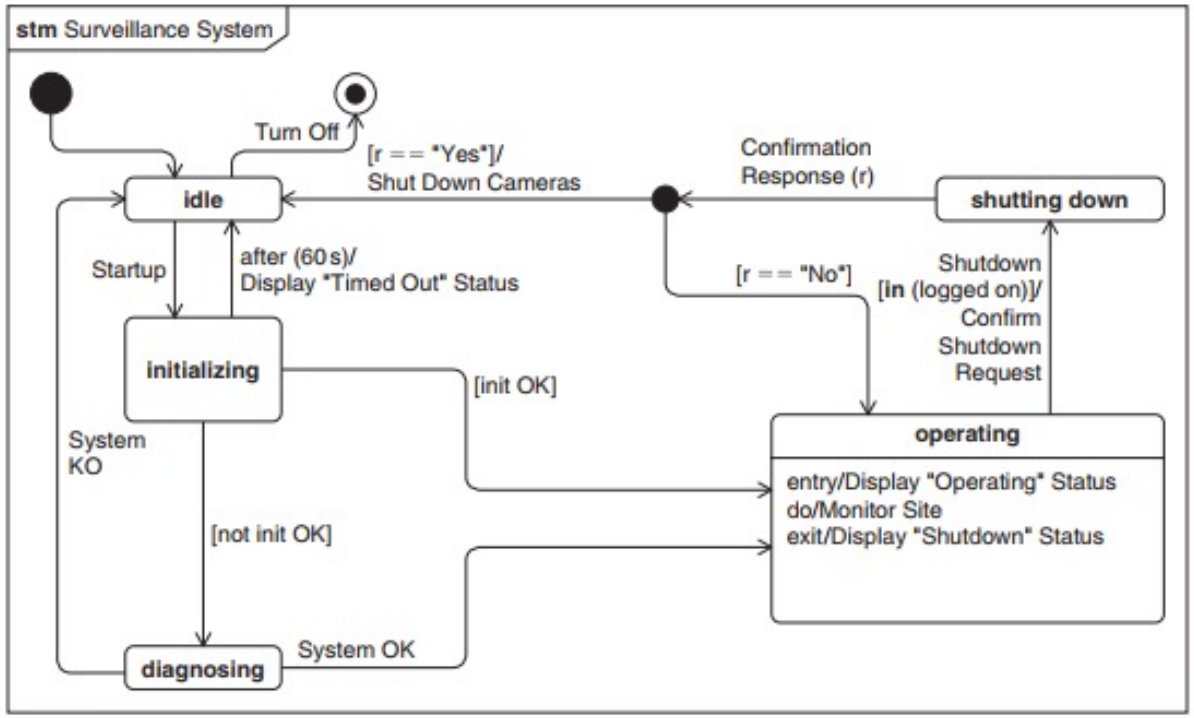
PSEUDOESTADOS: INICIALIZAÇÃO E CONCLUSÃO

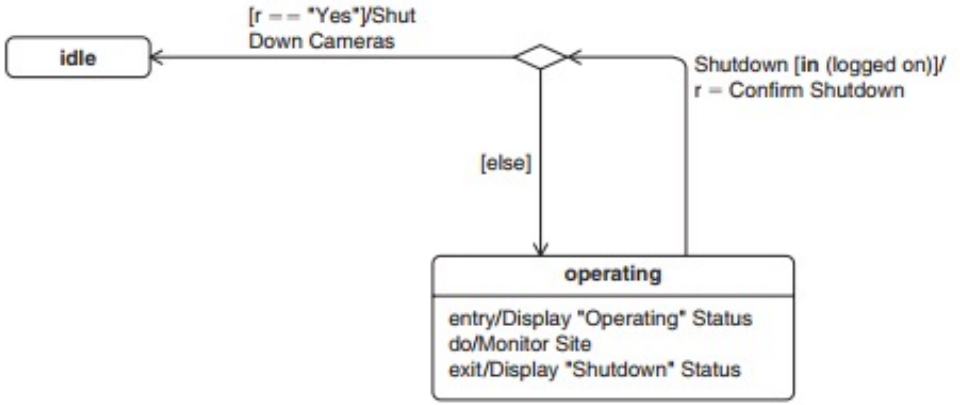
- A inicialização e a conclusão de uma região são descritas usando um pseudoestado inicial e um estado final, respectivamente.
- Um **pseudoestado inicial** é usado para determinar o estado inicial de uma região.
- Quando o estado ativo de uma região é o estado final, a região foi concluída e não ocorrem mais transições dentro dela. Assim, um **estado final não pode ter transições de saída**.
- O pseudoestado **"terminate"** é sempre associado ao estado de uma máquina de estado inteira.
  - Se um pseudoestado de terminação for atingido, o comportamento da máquina de estado será encerrado. Um pseudoestado de terminação tem o mesmo efeito que atingir os estados finais de todas as regiões da máquina de estado.

TRANSIÇÕES DE ROTEAMENTO USANDO PSEUDOESTADOS

- Há uma variedade de situações em que uma simples transição direta entre dois estados não é suficiente para expressar a semântica necessária..
  - Um **pseudoestado de junção** é usado para construir um caminho de transição composto entre estados. A transição composta **permite que mais de um caminho de transição alternativo entre estados seja especificado**, embora apenas um caminho possa ser tomado em resposta a qualquer evento único..
  - O **pseudoestado de escolha** também tem múltiplas transições de entrada e transições de saída e, como o pseudoestado de junção, é parte de uma transição composta entre estados. O comportamento do pseudoestado de escolha é distinto do de um pseudoestado de junção na medida em que **os guardas em suas transições de saída não são avaliados até que o pseudoestado de escolha tenha sido atingido**.

ROUTINGS

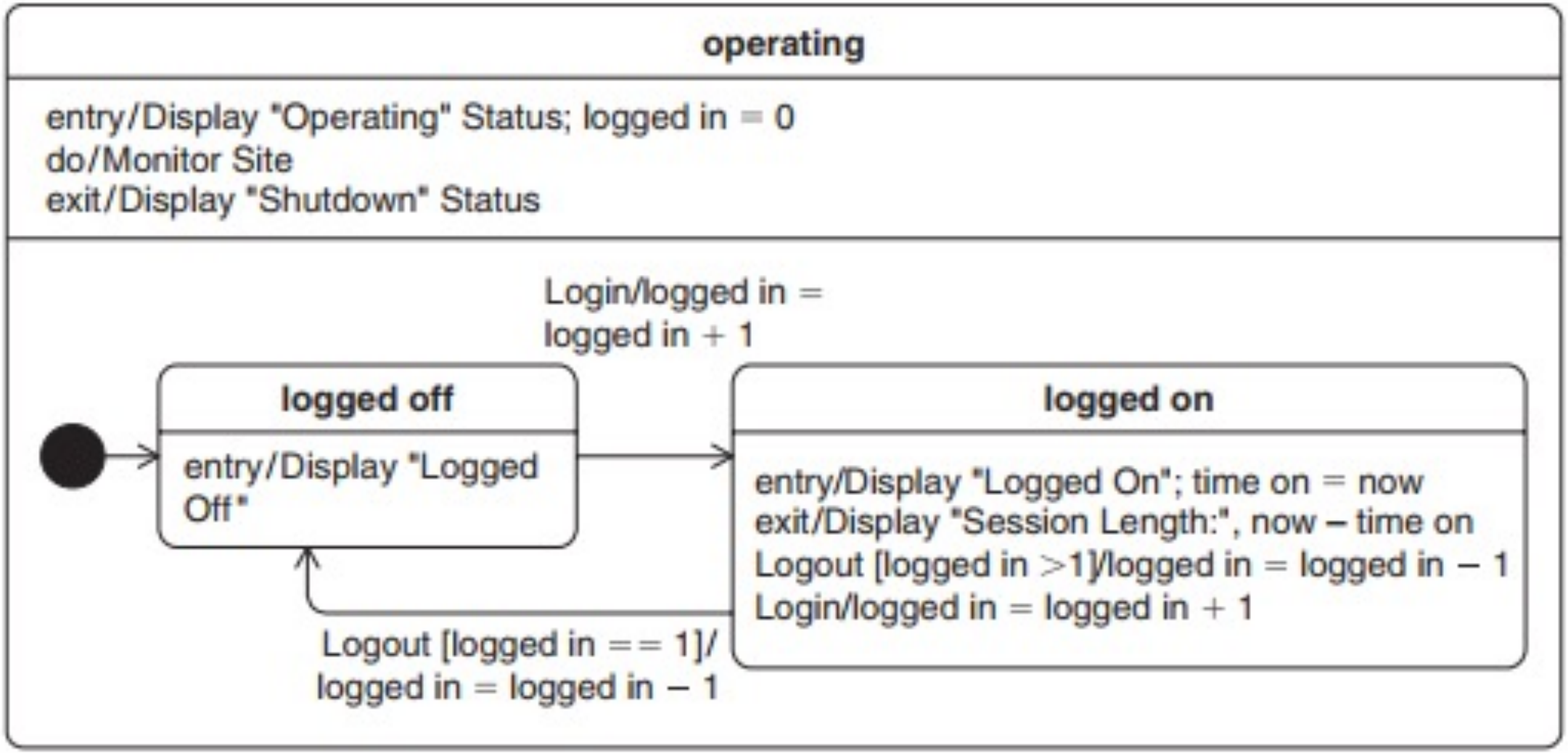

stm Surveillance System
Turn Off
[r == "Yes"]/
Shut Down Cameras
idle
Confirmation
Response (r)
shutting down
Startup
after (60 s)/
Display "Timed Out" Status
[r == "No"]
Shutdown
[in (logged on)]/
Confirm
Shutdown
Request
initializing
[init OK]
System
KO
operating
entry/Display "Operating" Status
do/Monitor Site
exit/Display "Shutdown" Status
[not init OK]
diagnosing
System OK


idle
[r == "Yes"]/Shut
Down Cameras
Shutdown [in (logged on)]/
r = Confirm Shutdown
[else]
operating
entry/Display "Operating" Status
do/Monitor Site
exit/Display "Shutdown" Status

ESTADOS COMPOSTOS – REGIÃO ÚNICA

- **Indiscutivelmente, a situação mais comum é um estado composto que tem uma única região.**
  - Uma região normalmente conterá um pseudoestado inicial e um estado final, um conjunto de pseudoestados e um conjunto de subestados, que podem ser estados compostos..
  - Se a região tiver um estado final, um evento de conclusão será gerado quando esse estado for atingido.
- Um estado composto pode ter muitas regiões, que podem conter subestados.

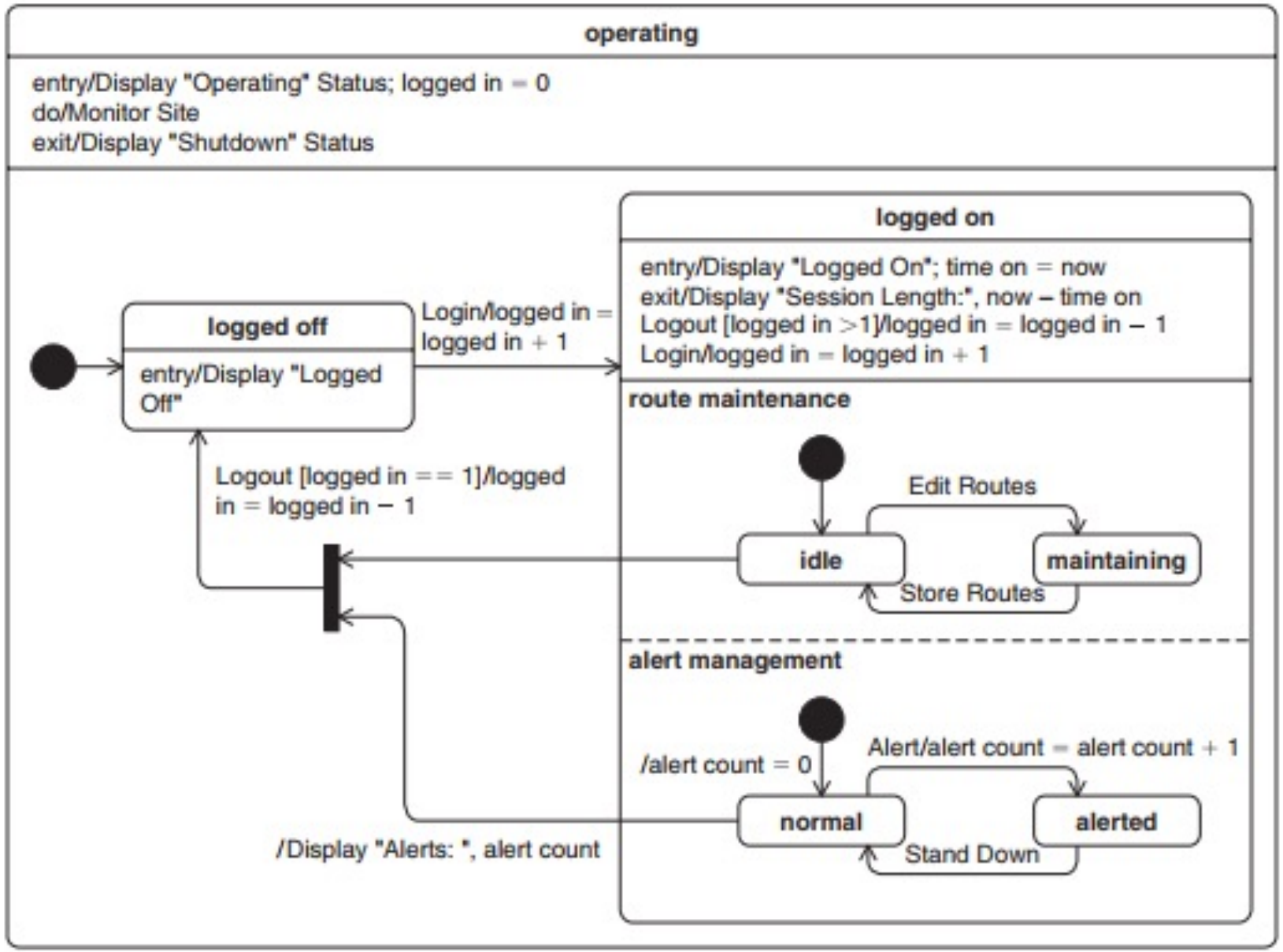
ESTADOS COMPOSTOS – REGIÃO ÚNICA (DOIS SUBESTADOS)



ESTADOS COMPOSTOS - SIMULTANEIDADE

- Quando um **estado composto ortogonal** está ativo, cada região tem **seu próprio estado ativo** que é independente das outras, e **qualquer evento de entrada é analisado independentemente** dentro de cada região.
  - Uma transição que termina no estado composto **acionará transições do pseudoestado inicial** de cada região, portanto, deve haver um pseudoestado inicial em cada região para que tal transição seja válida.
- Da mesma forma, um **evento de conclusão para o estado composto ocorrerá quando todas as regiões estiverem em seu estado final**.

ESTADOS COMPOSTOS - SIMULTANEIDADE



PSEUDOESTADO DA HISTÓRIA

- Em alguns cenários de design, é desejável manipular um evento de exceção interrompendo o comportamento da região atual, respondendo ao evento e, em seguida, retornando ao estado em que a região estava no momento da interrupção.
  - Um **pseudoestado de história profunda** registra os estados de todas as regiões na hierarquia de estado abaixo e incluindo a região que possui o pseudoestado de história profunda.
  - Um **pseudoestado de histórico** raso registra apenas o estado de nível superior da região que o possui.

CONSIDERAÇÕES FINAIS

Table A.21 State Machine Diagram State Nodes

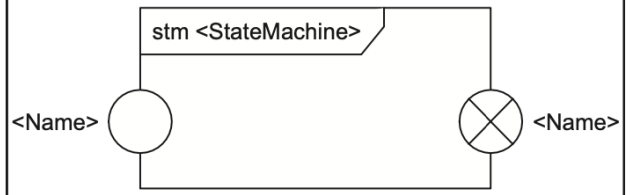
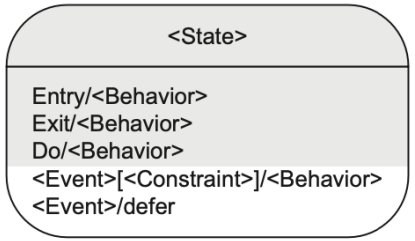
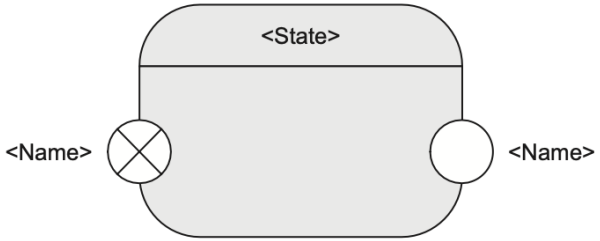
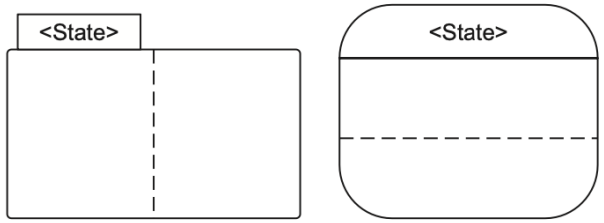
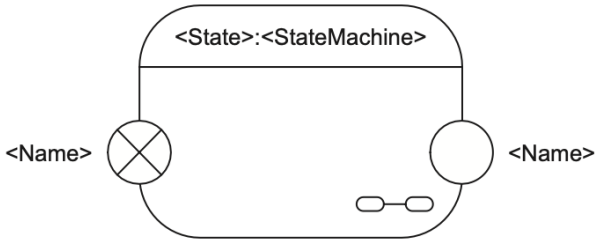
Diagram Element	Notation	Description	Section
State Machine with Entry- and Exit-Point Pseudostate Nodes		A state machine may have entry- and exit-point pseudostates, which are similar to junctions. On state machines, entry-point pseudostates can only have outgoing transitions and exit-point pseudostates can only have incoming transitions.	11.6.5
Atomic State Node		A state represents some significant condition in the life of a block. Each state may have entry and exit behaviors, and a do behavior. An atomic state node may also show transitions that are local to the state and events that are deferred while the state machine is in this state.	11.3
Composite State with Entry- and Exit-Point Pseudostate Nodes		A composite state is a state with nested regions; the most common case is a single region. A composite state may have entry- and exit-point pseudostates that act like junction pseudostates. Entry points have incoming transitions from outside the state and exit points have the opposite.	11.6.1
Composite State Node with Multiple Regions		A composite state may have many regions, which may each contain substates. These regions are orthogonal to each other and so a composite state with more than one region is sometimes called an orthogonal composite state.	11.6.2
Sub–State Machine Node with Connection Points		A state machine may be reused using a kind of state called a submachine state. A transition ending on a submachine state will start its referenced state machine. Transitions may also be connected to connection points on the boundary of the state. The symbol in the lower right refers to a lower level statemachine diagram. This can also be represented by a rake symbol.	11.6.5

Table A.22 State Machine Diagram Pseudostate and Transition Nodes

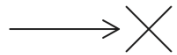
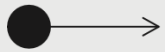
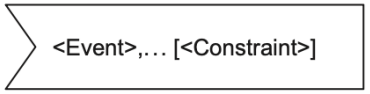
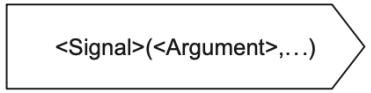
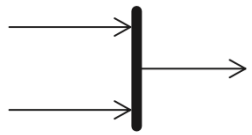
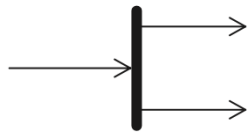
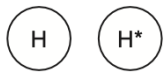
Diagram Element	Notation	Description	Section
Terminate Pseudostate Node		If a terminate pseudostate is reached, then the behavior of the state machine terminates.	11.3
Initial Pseudostate Node		An initial pseudostate specifies the initial state of a region.	11.3
Final State Node		The final state indicates that a region has completed execution.	11.3
Choice Pseudostate Node		The outgoing transitions of a choice pseudostate are evaluated once it has been reached.	11.4.2
Junction Pseudostate Node		A junction pseudostate is used to construct a compound transition path between states.	11.4.2
Trigger Node		This node represents all the transition's triggers, with the descriptions of the triggering events and the transition guard inside the symbol.	11.4.3
Action Node		<EffectExpression> describes the effect of the transition, either the name of a behavior or the body of an opaque behavior.	11.4.3
Send Signal Node		This node represents a send signal action. The signal's name, together with any arguments that are being sent, are shown within the symbol.	11.4.3
Join Pseudostate Node		A join pseudostate has a single outgoing transition and many incoming transitions. When all of the incoming transitions can be taken, and the join's outgoing transition is valid, then all the transitions happen.	11.6.2
Fork Pseudostate Node		A fork pseudostate has a single incoming transition and many outgoing transitions. When an incoming transition is taken to the fork pseudostate, all of the outgoing transitions are taken.	11.6.2
History Pseudostate Node		A history pseudostate represents the last state of its owning region, and a transition ending on a history pseudostate has the effect of returning the region to the state it was last in.	11.6.4

Table A.23 State Machine Diagram Paths

Diagram Element	Notation	Description	Section
Time Event Transition Path	after <TimeExpression>[<Constraint>]/<Behavior> → at <TimeExpression>[<Constraint>]/<Behavior> →	Time events indicate either that a given time interval has passed since the current state was entered (after), or that a given instant of time has been reached (at). The transition can also include a guard and effect.	11.4.1
Signal Event Transition Path	<Signal>(<Attribute>, …)[<Constraint>]/<Behavior> →	Signal events indicate that a new asynchronous message has arrived. A signal event may be accompanied by a number of arguments, which may be assigned to attributes. The transition can also include a guard and effect.	11.4.1
Call Event Transition Path	<Operation>(<Attribute>, …)[<Constraint>]/<Behavior> →	Call events indicate that an operation on the state machine's owning block has been requested. A call event may also be accompanied by a number of arguments, which may be assigned to attributes. The transition can also include a guard and effect.	11.5
Change Event Transition Path	when <Expression>[<Constraint>]/<Behavior> →	Change events indicate that some condition has been satisfied (normally that some specific set of attribute values hold). The transition can also include a guard and behavior/effect.	11.7

ROTEIRO DOS DIAGRAMAS