UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
DIM0514 ARQUITETURA DE SOFTWARE

Janeto Erick da Costa Lima <janetoerick18@gmail.com>

1. Introdução

O presente relatório descreve o desenvolvimento de um projeto de gerenciamento de estacionamento, no qual se utiliza a ferramenta SysADL para a modelagem arquitetural do sistema. O SysADL é uma linguagem de modelagem que permite representar a arquitetura de sistemas de forma clara e precisa, facilitando o entendimento das interações entre os componentes do sistema.

O objetivo deste projeto é criar uma solução eficiente para a gestão de um estacionamento, com funcionalidades como controle de entrada e saída de veículos, monitoramento de ocupação das vagas e integração com sistemas de pagamento. Através da utilização do SysADL, busca-se obter uma visão abrangente da arquitetura do sistema, garantindo que os requisitos funcionais e não funcionais sejam atendidos.

Neste contexto, o relatório aborda o processo de modelagem arquitetural do sistema de estacionamento, detalhando as decisões de design, a estrutura dos componentes do sistema, bem como os principais desafios e soluções adotadas para a implementação da arquitetura.

2. Modelagem de Requisitos

Os requisitos foram elaborados a partir da análise detalhada das necessidades do estacionamento, incluindo a gestão de vagas, o controle de entrada e saída de veículos, a interação com os usuários e a integração com sistemas de pagamento. Na tabela 1, estão listados todos os requisitos, contendo seu código, nome, descrição e suas relações. Ao todo, são 17 requisitos funcionais e 1 requisito não funcional.

ID	Nome	Descrição	Relações
1	ControlarCancelaRF	O sistema deve ser capaz de controlar a cancela.	-

1.1	ControlarCancelaManualmenteRF	O sistema deve permitir que um usuário autorizado acione a cancela manualmente.	comp. 1
1.1.1	AbrirCancelaRF	O sistema deve abrir a cancela.	der. 1.1, der. 1.2.1, der. 1.2.7
1.1.2	FecharCancelaRF	O sistema deve fechar a cancela.	der. 1.1, der 1.1.3
1.1.3	VerificarPassagemVeiculoRF	O sistema deve ser capaz de verificar se o veículo passou pela cancela.	der. 1.1.1
1.2	ControlarCancelaAutomaticamenteRF	O sistema deve ser capaz de acionar a cancela.	comp. 1
1.2.1	FornecerTicketRF	O sistema deve fornecer ao usuário um ticket de acesso.	der 1.2
1.2.2	CapturarHorarioTicketRF	O sistema deve capturar o horário em que o ticket foi retirado.	der. 1.2.1
1.2.3	CalcularValorDoTicketRF	O sistema deve calcular o valor total do ticket com base na hora de entrada e saída.	der. 1.2.2
1.2.4	PagarTicketRF	O sistema deve permitir que o usuário pague o ticket.	der. 1.2.3
1.2.5	EscanearTicketRF	O sistema deve escanear o ticket.	der. 1.2
1.2.6	VerificarStatusDoTicketRF	O sistema deve ser capaz de verificar os status do ticket.	der. 1.2.5
1.2.7	ConfirmarPagamentoTicketRF	O sistema deve informar se o ticket está pago.	der. 1.2.6
2	GerenciarVagasRF	O sistema deve gerenciar a quantidade de vagas livres e	der. 2.2

		ocupadas.	
2.1	ExibirVagasRF	O sistema deve ser capaz de exibir a quantidade de vagas disponíveis.	der. 2
2.2	InformarPresencaDeVeiculoRF	O sistema deve informar a presença de veículo na vaga.	der. 2.3
2.3	MonitorarPresencaDeVeiculoRF	O sistema deve monitorar se há veículo em cada vaga.	-
3	DisponibilidadeRNF	O sistema deve operar 24 horas por dia, 7 dias por semana.	-

Tabela 1. Requisitos do sistema

Na tabela, vemos que ControlarCancelaRF divide-se em dois requisitos: ControlarCancelaManualmenteRF (*id.1.1*) e ControlarCancelaAutomaticamenteRF (*id.1.2*). Essa divisão do requisito foi feita para que o sistema tivesse a autonomia de controlar a abertura e fechamento das cancelas mediante situações pré estabelecidas. Além disso, deve permitir que um usuário autorizado consiga acionar manualmente a abertura e o fechamento, sendo esta alternativa importante para casos de indisponibilidade do sistema.

Partindo de *ControlarCancelaManualmenteRF* (*id.1.1*), este se deriva em dois outros requisitos: *AbrirCancelaRF* (*id. 1.1.1*) e *FecharCancelaRF* (*id.1.1.2*), que diz respeito a abertura e fechamento da cancela, respectivamente. Para contribuir com a automação do sistema foi modelado o requisito VerificarPassagemVeiculoRF (*id.1.1.3*), no qual possibilita o fechamento da cancela, após sua abertura, quando um veículo atravessa-la.

Com relação à *ControlarCancelaAutomaticamenteRF* (*id.1.2*), é feita a derivação em outros dois requisitos que possibilitam o sistema a concretizar a abertura da cancela, são eles: *FornecerTicketRF* (*id.1.2.1*) e *EscanearTicketRF* (*id.1.2.5*). Como o controle do estacionamento será feito mediante ticket de acesso, o sistema deverá realizar o controle da cancela a partir da entrega e verificação dos tickets, sendo necessário a captura da hora de entrada do carro (*id.1.2.2*) para calcular o valor de estacionamento (*id.1.2.3*), possibilitando o pagamento do ticket pelo usuário (*id.1.2.4*).

3. Diagrama de Requisitos

O diagrama de requisitos teve dois núcleos de relacionamentos, um dos núcleos diz respeito ao controle das cancelas e do pagamento, e o outro tem relação ao gerenciamento das vagas. Primeiramente será exibido os núcleos separadamente, para melhor visualização, e no final será apresentado o diagrama de requisitos completo.

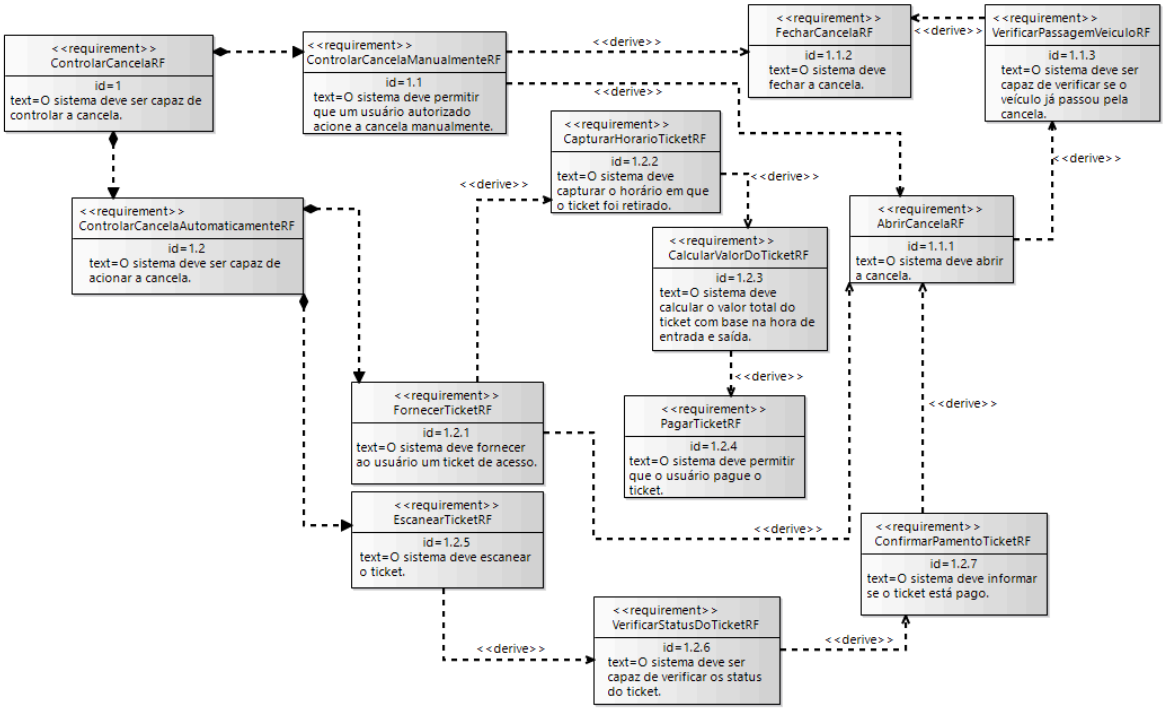

Figura 1. Requisitos relacionados com acionar cancela

A figura 1 mostra os relacionamentos de composição entre o requisito *1* e suas composições *1.1* e *1.2*, com o complemento da composição do requisito *1.2* nos requisitos *1.2.1* e *1.2.5*. Ainda, os relacionamentos de derivação dos demais requisitos no núcleo.

Com relação a figura 2, é mostrado os requisitos referentes a gerência do controle de vagas dentro do estacionamento. Isto é feito com o requisito *2*, os requisitos que o derivam, *2.1*, e de quem ele deriva (*2.2* e *2.3*).

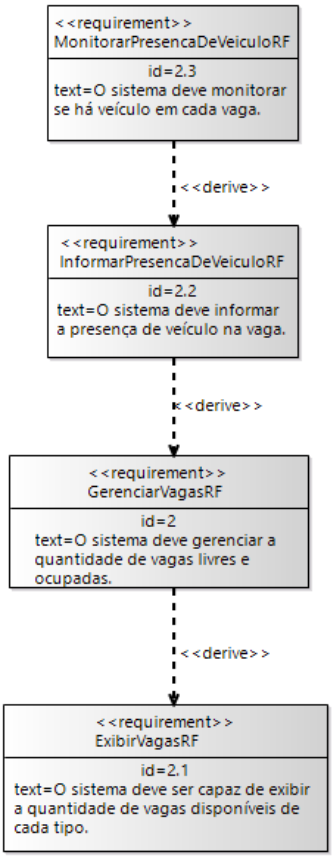

Figura 2. Requisitos relacionados com gerenciar vagas

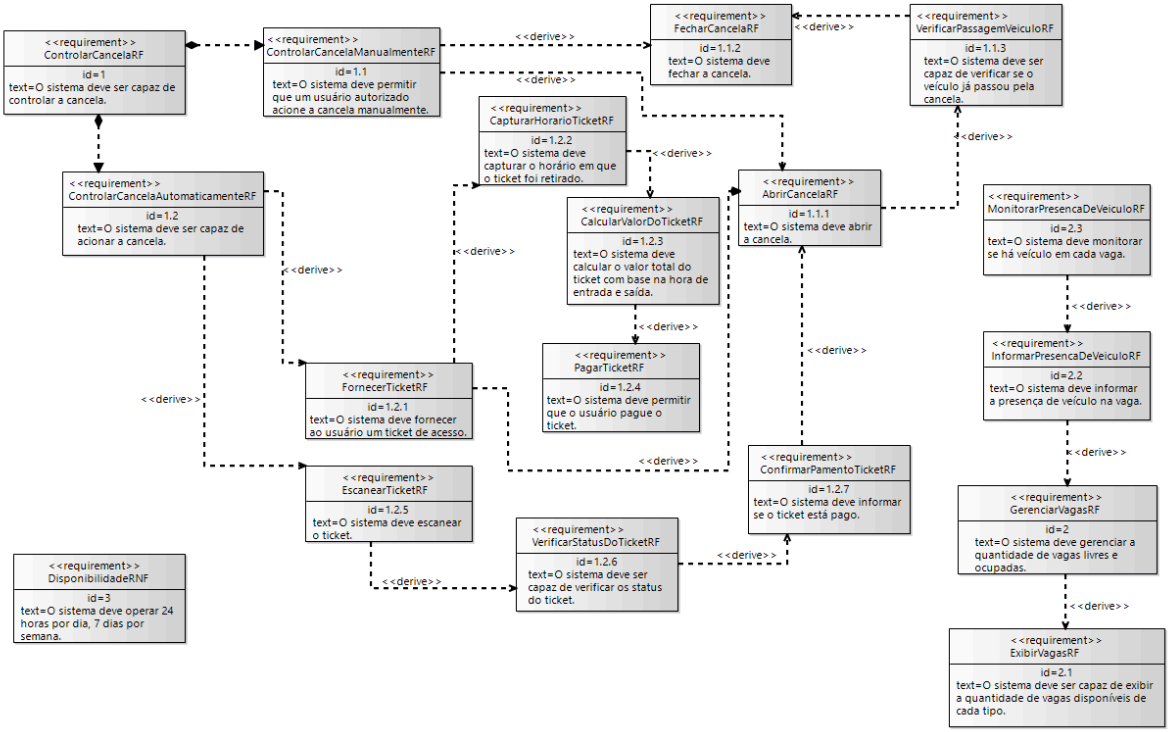

Figura 3. Diagrama de Requisitos

Além dos já citados, na figura 3, há também a presença do requisito não funcional *DisponibilidadeRNF (id.3)*.

4. Visão Estrutural

Para a modelagem da visão estrutural do sistema de estacionamento, foram separados quatro processos de construção para separar os principais elementos da arquitetura, ou seja, os tipos de valor, portas, conectores e componentes, os quais são essenciais para o desenvolvimento da arquitetura de software no SysADL.

Para cada etapa de criação dos elementos foi criado um Diagrama de Definição de Bloco (BDD), para definir os respectivos elementos ligados a eles. Esses diagramas serão apresentados nas subseções seguintes.

4.1. Diagrama de Valores

Para a construção da arquitetura utilizando o SysADL, os valores são os elementos atômicos necessários para iniciar a modelagem estrutural do sistema. Esse diagrama faz referência a todos os valores que serão utilizados. A representação dos valores na linguagem são feitos através dos elementos gráficos: tipos de valor (Value Type), tipo de dado (dataType) e enumeração (enumeration). Além disso, pode ter um acréscimo de dimensões (dimensions) e unidades (unit), no qual, não é utilizado no sistema proposto. O diagrama de valores do sistema é mostrado na figura 4.

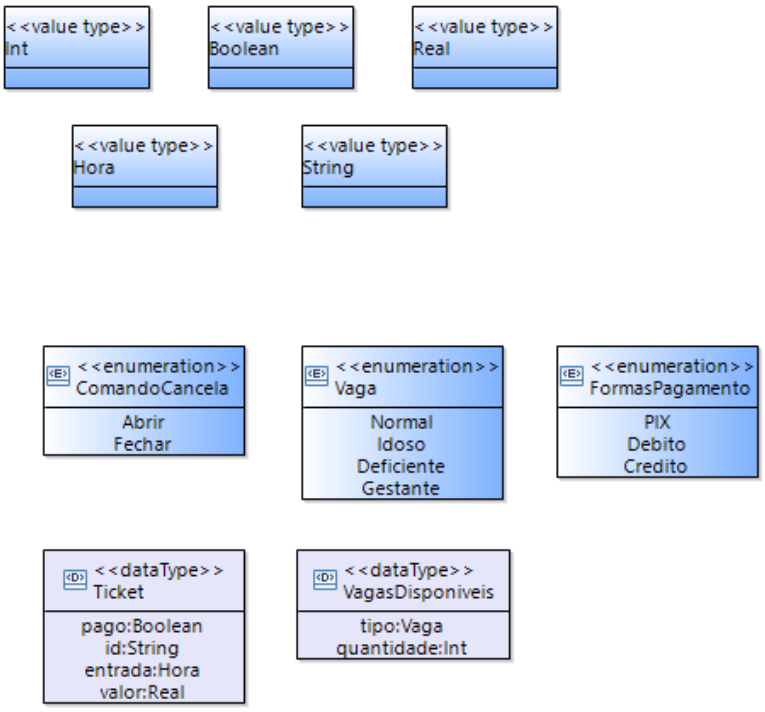

Figura 4. BDD de valores

Foram utilizados dois dataType, *Ticket* e *VagasDisponiveis*. Cada um é usado em núcleos diferentes, citado no diagrama de requisitos. *O Ticket* é utilizado no núcleo de controle das cancelas e *VagasDisponiveis* no núcleo de gerência de vagas. Com isso foram criadas três enumerations, *ComandoCancela*, *Vaga* e *FormasPagamento*, que tem como foco os tipos de comando que a cancela deverá receber (abrir e fechar), os tipos de vagas possíveis de ter no estacionamento (normal, idoso, deficiente e gestante) e as formas de pagamento aceitáveis pelo sistema, respectivamente. Além disso, há a presença de cinco valueType: *Int, Boolean, Real, String* e *Hora*.

4.2. Diagrama de Portas

As portas são elementos acoplados nos componentes, no qual retratam a entrada e saída de dados. Para cada tipo de dado manipulado no sistema foram criadas suas portas de entrada e saída. O diagrama apresentando as portas é mostrado na figura 5.

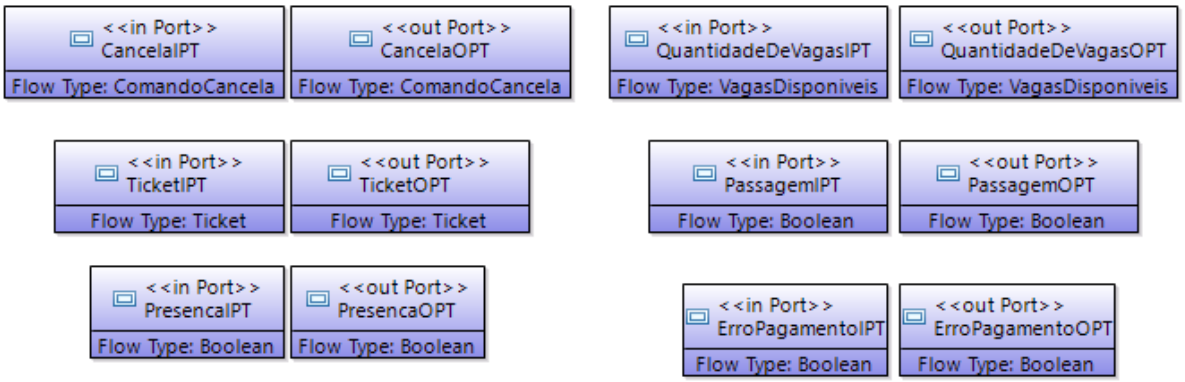

Figura 5. BDD de portas

4.3. Diagrama de Conectores

Os conectores são a representação da manipulação dos dados pelas portas citadas anteriormente, neles são retratados quais valores são passados em cada tipo de porta. No sistema proposto é feito o uso de seis conectores simples, a apresentação deles está presente no diagrama da figura 6.

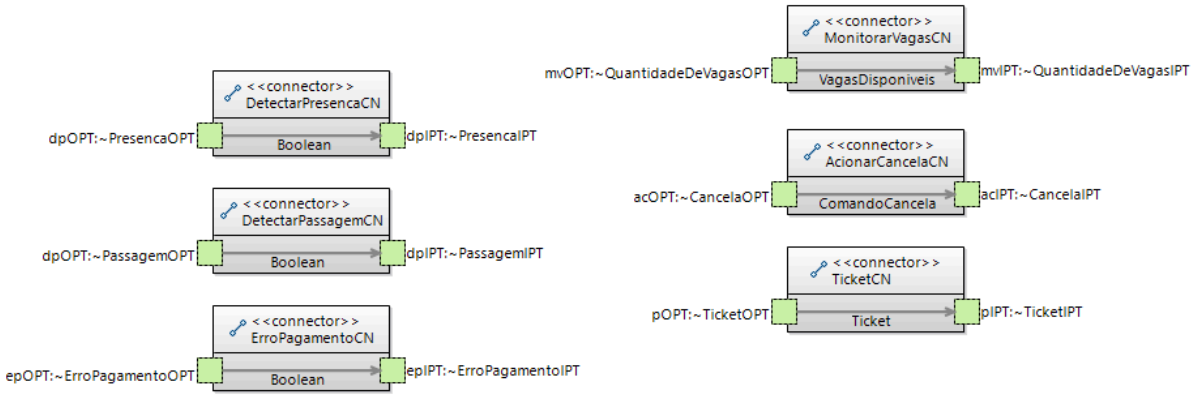

Figura 6. BDD de conectores

4.4. Diagrama de Componentes

Para a construção deste diagrama foi pensado em utilizar a estratégia centralizada, para ter o controle das informações concentradas no componente *SistemaDeControleCP*, com algumas iterações feitas fora do seu interior. Ao todo foram criados dez componentes, que estão apresentados no diagrama da figura 7 a seguir.

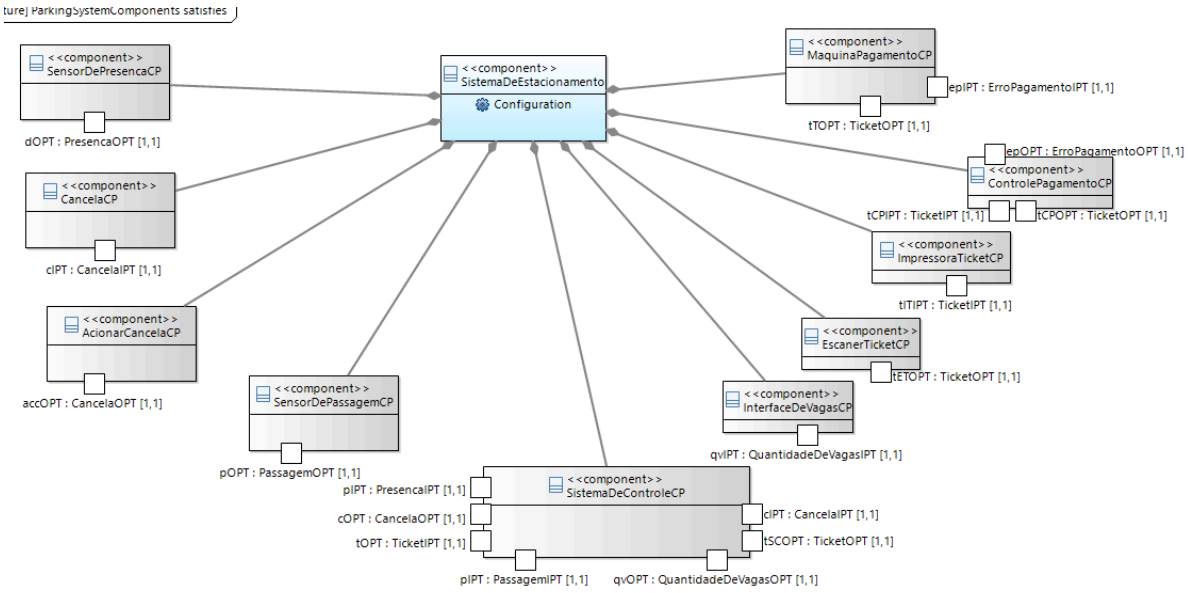

Figura 7. BDD de Componentes

Para o controle das cancelas foram modelados dois componentes, *CancelaCP* e *AcionarCancelaCP*, no qual tem como propósito o controle automático e manual da cancela, respectivamente. Além disso, há os sensores para capturar as informações da presença dos carros nas vagas (*SensorDePresencaCP*) e

passagem dos veículos pelas cancelas (*SensorDePassagemCP*), no qual este último é vital para a automatização do fechamento da cancela após a passagem dos automóveis.

Ademais, foram pensados mais quatro componentes, um responsável para a amostragem de vagas disponíveis (*InterfaceDeVagasCP*) e quatro responsáveis pela manipulação dos tickets, *MaquinaPagamentoCP*, *ImpressoraTicketCP, EscanerTicketCP* e *ControlePagamentoCP,* sendo este último responsável pelo gerencia dos pagamentos de forma mais específica.

A modelagem do IBD referente ao SistemaDeEstacionamento é mostrada na figura 8 a seguir.

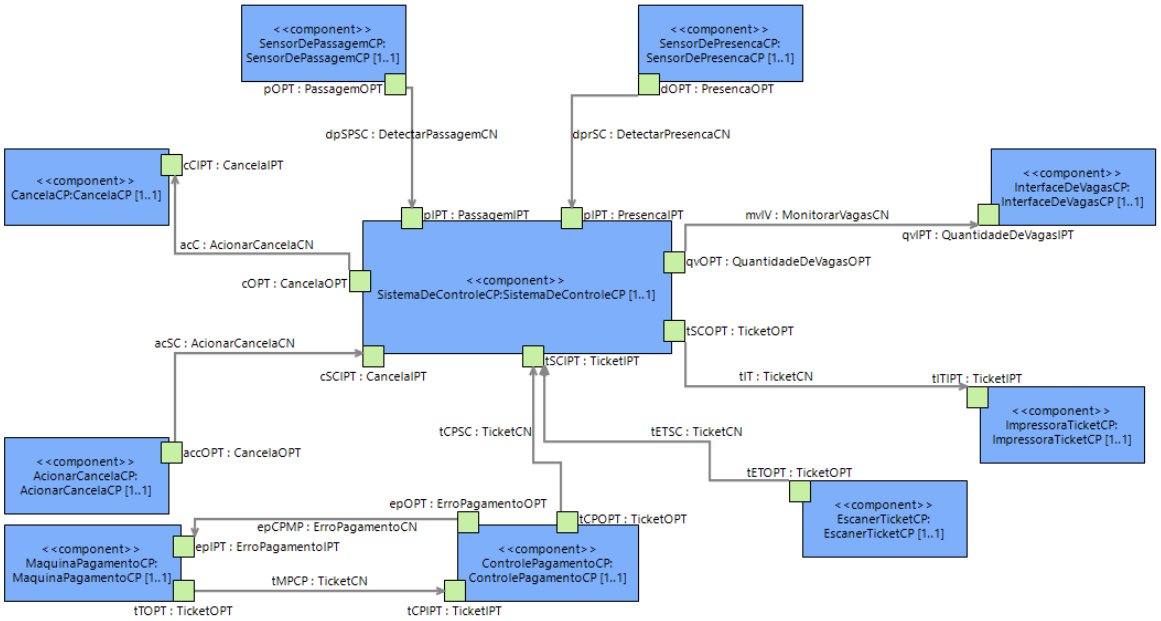

Figura 8. IBD do SistemaDeEstacionamento

Para a arquitetura do sistema foi projetado o diagrama estrela com o foco concentrado em *SistemaDeControleCR,* que tem como função gerenciar os dados e distribuí-los aos demais elementos do sistema para gerar resultados.

Falando mais em relação a *ControlePagamentoCP*, ele terá a funcionalidade de, a partir do ticket mostrado pelo usuário na *MaquinaPagamentoCP*, fazer o cálculo do valor total a ser pago e processar o pagamento de acordo com a forma de pagamento desejado (PIX, Débito ou Crédito), no final dessa iteração, se tudo ocorrer bem, mandar o valor do ticket atualizado para o sistema de controle com o status de pago igual a verdadeiro e confirmando o procedimento concluído para *MaquinaPagamentoCP*, permitindo assim, caso o ticket em questão seja lido pelo *EscanerTicketCP*, aciona a cancela. Caso contrário em que o pagamento não seja efetuado como programado, será enviada apenas a mensagem de erro igual a ‘True’ para a *MaquinaPagamentoCP.*