Engenharia de Software

Caso Prático *Relógio-Agenda*

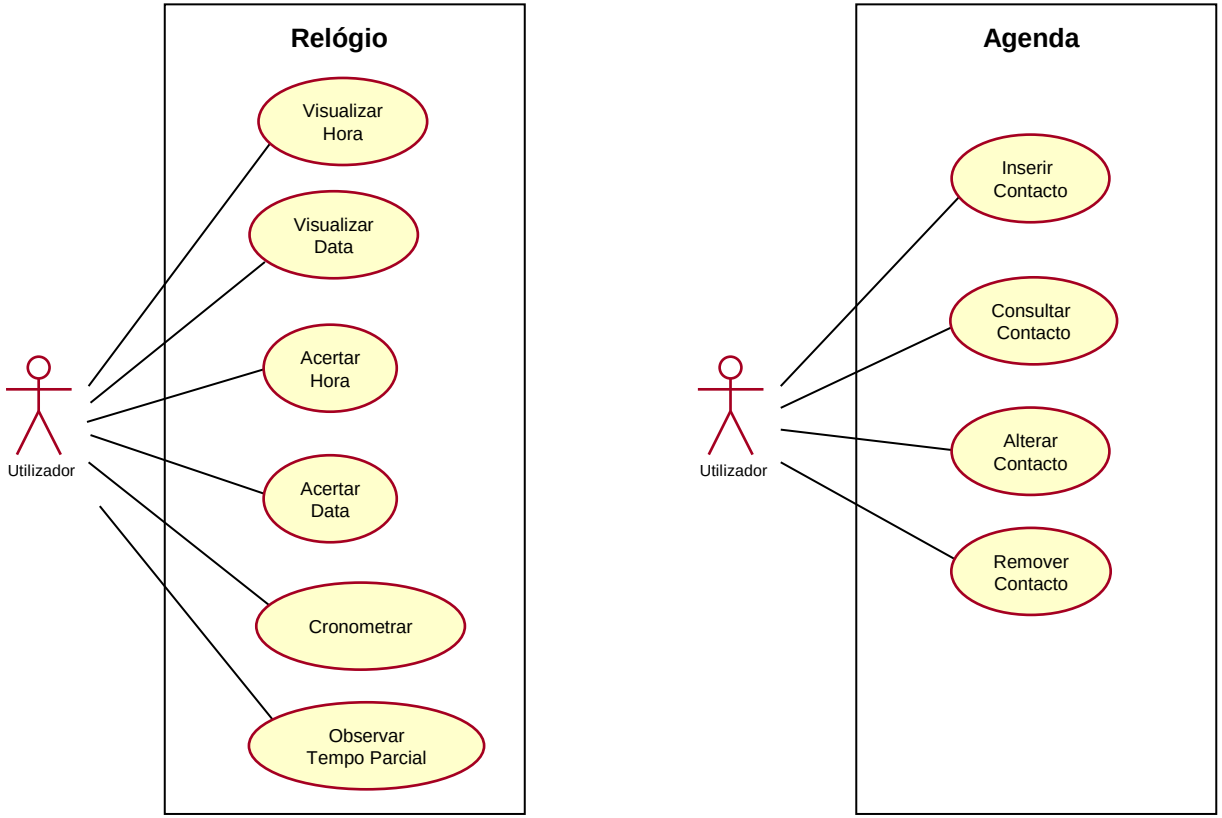
Luís Morgado

Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

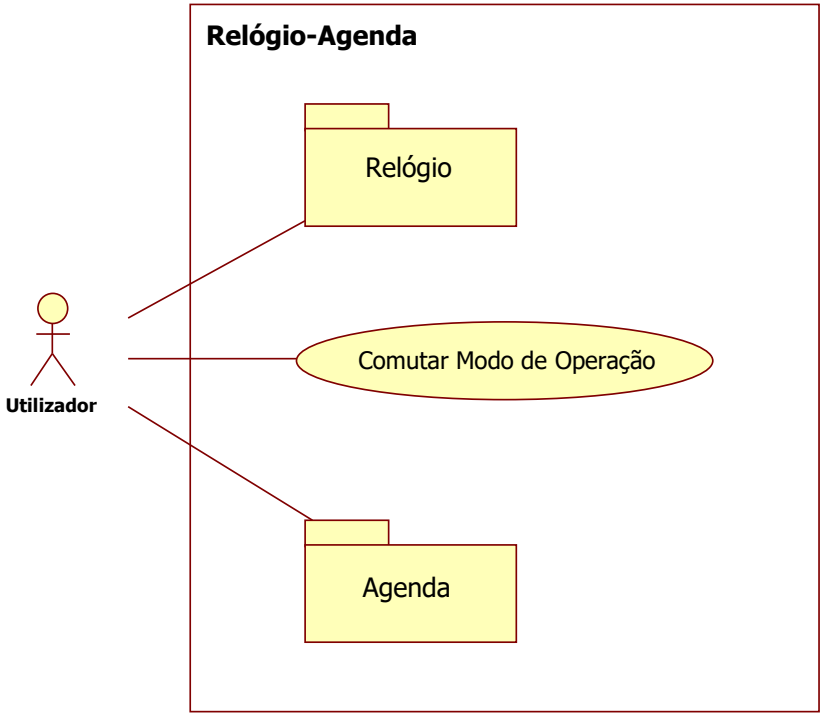
Iteração Inicial

- Objectivo:
  - Estudo do problema e de uma possível solução
- Actividades:
  - Elaboração do documento de visão
  - Identificação dos principais casos de utilização
  - Elaboração de um esboço de arquitectura geral

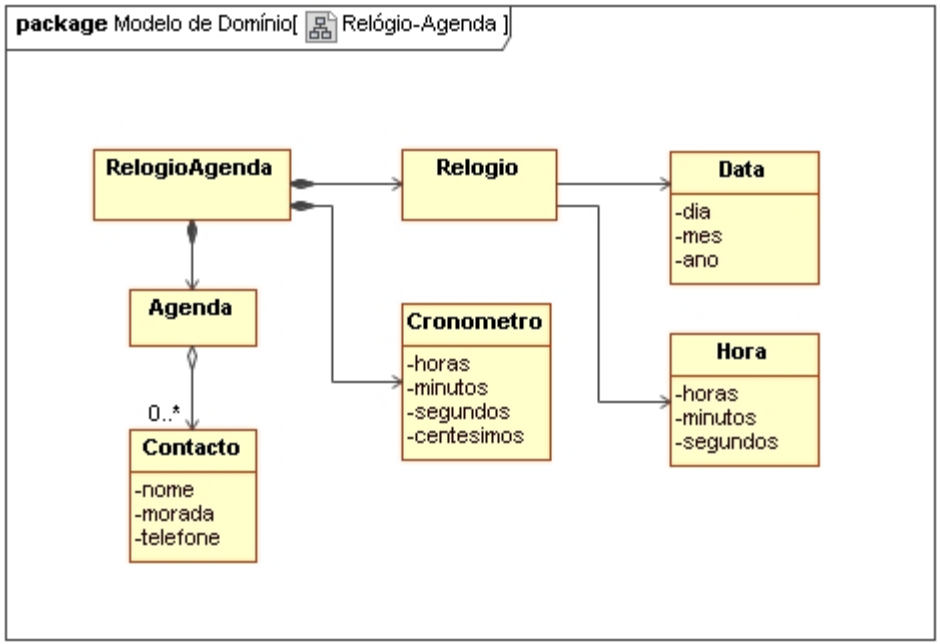
Modelo de Casos de Utilização



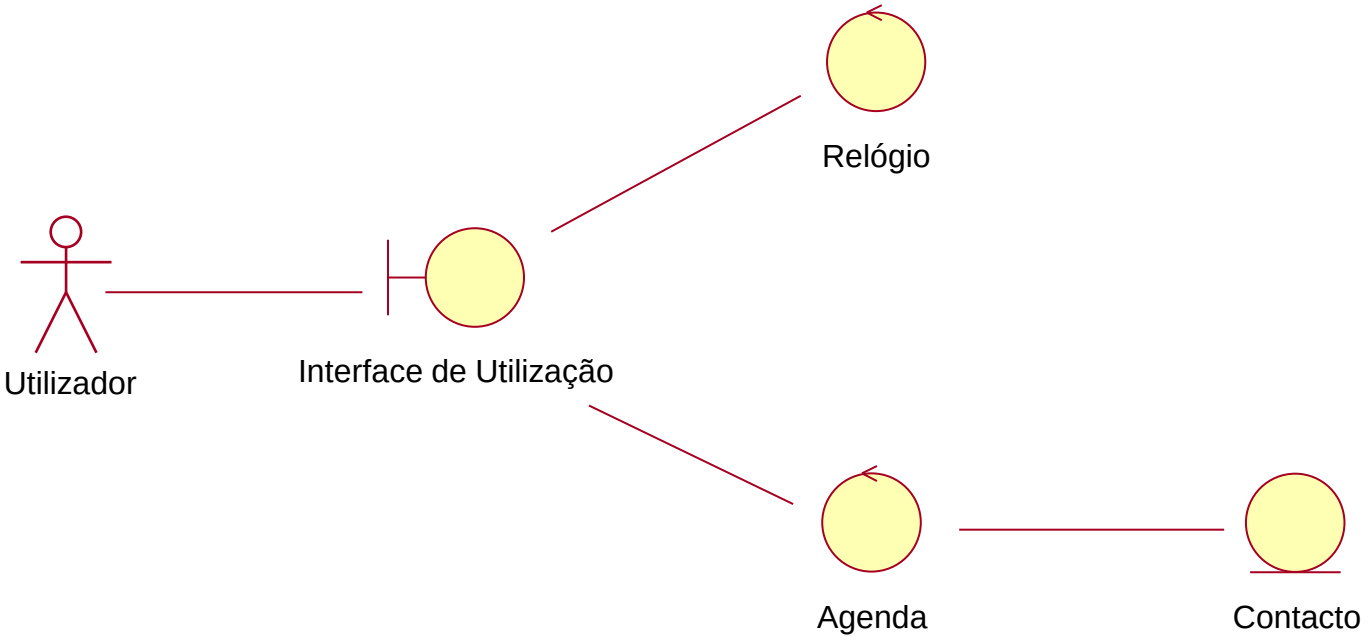
Modelo de Casos de Utilização



Modelo de Domínio



Modelo de Análise

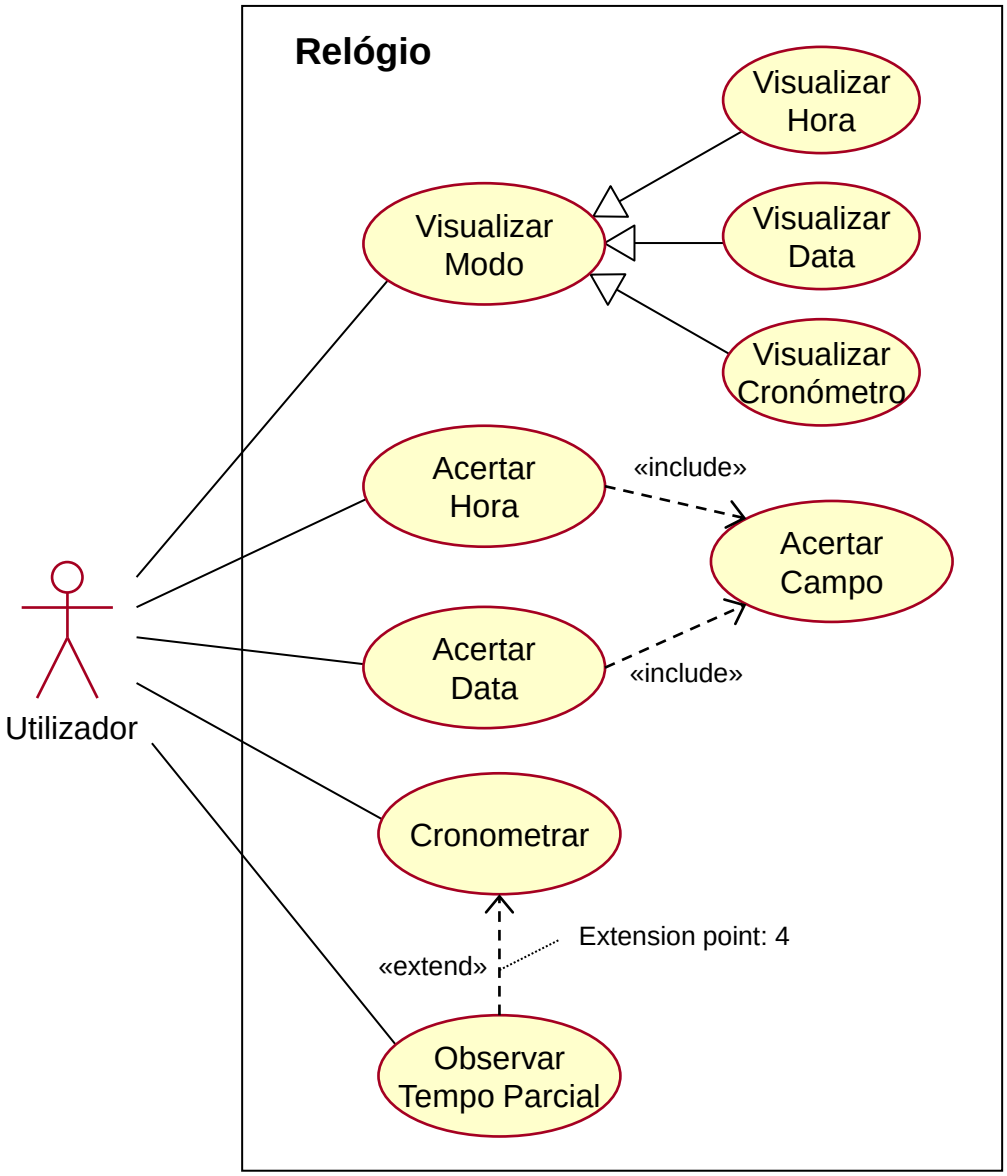


Proporciona uma visão geral da organização do sistema

Iteração de Análise de Requisitos 1

- Objectivo:
  - Compreensão e descrição da solução a realizar
- Actividades:
  - Detalhe e descrição do modelo de casos de utilização e especificação suplementar
  - Projecto de subsistemas
  - Elaboração de protótipo demonstrador

Modelo de Casos de Utilização



Modelo de Casos de Utilização

Descrição de Caso de Utilização: *Simulador Relógio-Agenda*

Designação: **Visualizar Modo**

Resumo: Este caso de utilização permite ao utilizador visualizar a hora, a data ou o cronómetro.

Actores: Utilizador

Pré-condições:

Cenário principal:

1. O caso de utilização inicia-se quando o utilizador pretende visualizar a hora, a data ou o cronómetro, estando o relógio no modo respectivo.
2. O utilizador observa a informação pretendida e o caso de utilização termina.

Cenário alternativo 1:

1. No passo 1 do cenário principal o sistema está em modo relógio mas não está no modo pretendido.
2. O utilizador pressiona o botão MODE as vezes necessárias para colocar o sistema no modo pretendido (**ver requisito R7**).
3. O utilizador observa a informação e o caso de utilização termina.

Cenário alternativo 2:

1. No passo 1 do cenário principal, o sistema está em modo agenda.
2. O utilizador pressiona o botão MEMO.
3. O sistema retorna ao modo relógio (**ver requisito R8.2**).
4. Aplica-se o cenário anterior adequado.

Modelo de Casos de Utilização

Descrição de Caso de Utilização: *Simulador Relógio-Agenda*

Designação: **Visualizar Hora**
Resumo: Este caso de utilização concretiza *Visualizar Modo* para **Modo = Hora**.

Descrição de Caso de Utilização: *Simulador Relógio-Agenda*

Designação: **Visualizar Data**
Resumo: Este caso de utilização concretiza *Visualizar Modo* para M**odo = Data**.

Descrição de Caso de Utilização: *Simulador Relógio-Agenda*

Designação: **Visualizar Cronómetro**
Resumo: Este caso de utilização concretiza *Visualizar Modo* para M**odo = Cronómetro**.

Modelo de Casos de Utilização

Descrição de Caso de Utilização: *Simulador Relógio-Agenda*

Designação: **Acertar Hora**

Resumo: Este caso de utilização permite ao utilizador acertar a hora do relógio.

Actores: Utilizador

Pré-condições: O sistema encontra-se em modo de visualização de hora.

Cenário principal:

1. O caso de utilização inicia-se quando o utilizador pretende acertar a hora estando o relógio em modo hora.
2. O utilizador pressiona o botão FUNCTION.
3. O campo SEGUNDOS do relógio fica a piscar.
4. **Incluir Acertar Campo.**
5. O utilizador pressiona o botão MODE.
6. O campo SEGUNDOS deixa de piscar e passa a piscar o campo MINUTOS.
7. **Incluir Acertar Campo.**
8. O utilizador pressiona o botão MODE.
9. O campo MINUTOS deixa de piscar e passa a piscar o campo HORAS.
10. **Incluir Acertar Campo.**
11. O utilizador pressiona o botão FUNCTION.
12. O campo HORAS do relógio deixa de piscar e o caso de utilização termina.

Modelo de Casos de Utilização

Descrição de Caso de Utilização: *Simulador Relógio-Agenda*

Designação:	**Acertar Campo**
Resumo:	Este caso de utilização permite ao utilizador acertar um campo de hora ou data.
Actores:	Utilizador
Pré-condições:	O sistema encontra-se em modo de visualização de hora ou de data com um campo seleccionado (a piscar).

Cenário principal:

1. O caso de utilização inicia-se quando o utilizador pressiona o botão ADV.
2. Enquanto o campo não apresentar o valor pretendido pelo utilizador:
   a) O sistema incrementa o campo seleccionado (**ver requisitos R2 e R5**).
3. O caso de utilização termina.

Modelo de Casos de Utilização

Descrição de Caso de Utilização: *Simulador Relógio-Agenda*

Designação: **Utilizar Cronómetro**
Resumo: Este caso de utilização permite ao utilizador cronometrar uma actividade.
Actores: Utilizador
Pré-condições: O sistema está em modo de cronómetro com o cronómetro parado.

Cenário principal:

1. O caso de utilização inicia-se quando o utilizador pressiona o botão ADV.
2. O cronómetro inicia a contagem **(ver requisito R1**).
3. O utilizador pressiona o botão FUNCTION.
4. O cronómetro mantém o tempo parcial.
5. Internamente o cronómetro continua a contar mas não é feita a actualização dos campos.
6. Após algum tempo o utilizador pressiona novamente o botão FUNCTION.
7. O cronómetro retoma a actualização dos campos de acordo com a contagem interna.
8. Após algum tempo o utilizador pressiona novamente o botão ADV.
9. O cronómetro pára.
10. O utilizador pressiona o botão FUNCTION (função *Reset*).
11. O sistema coloca todos os campos do cronómetro a zero.
12. O caso de utilização termina.

Especificação Suplementar

Especificação suplementar: *Simulador Relógio-Agenda*

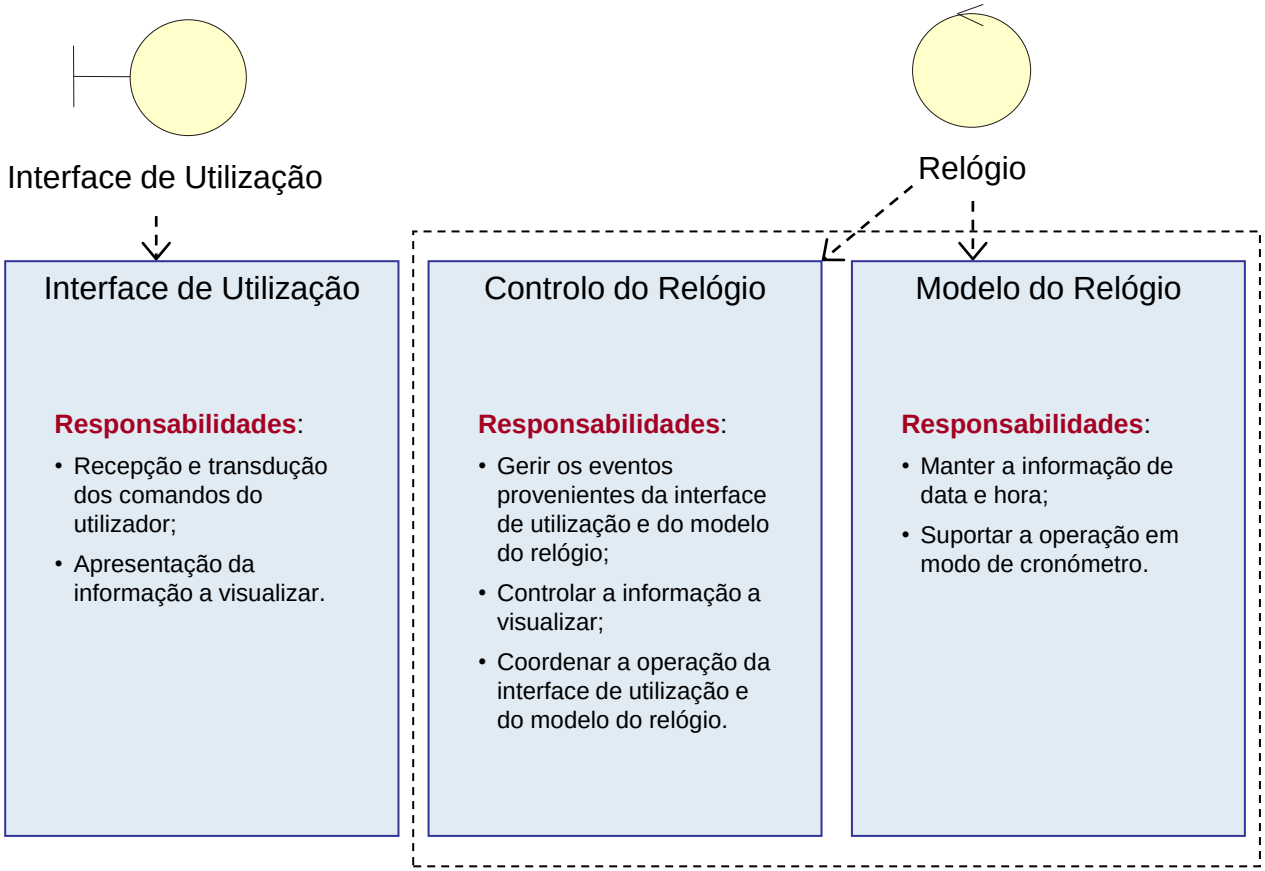
Ref.	Descrição	Categoria
R1	O cronómetro deve ter uma resolução de 10 [ms].	Obrigatório
R2	Os campos do relógio (hora e data) admitem as seguintes gamas de valores:	Obrigatório
R2.1	Hora: 0-23	Obrigatório
R2.2	Minutos: 0-59	Obrigatório
R2.3	Segundos: 0-59	Obrigatório
R2.4	Dia: 1-31	Obrigatório
R2.5	Mês: 1-12	Obrigatório
R2.6	Ano: 0-99	Obrigatório
R3	Os campos do cronómetro admitem as seguintes gamas de valores:	Obrigatório
R3.1	Hora: 0-99	Obrigatório
R3.2	Minutos: 0-59	Obrigatório
R3.3	Segundos: 0-59	Obrigatório
R3.4	Centésimos de segundo: 0-99	Obrigatório

Especificação Suplementar

Especificação suplementar: *Simulador Relógio-Agenda*

Ref.	Descrição	Categoria
R4	Todos os campos do relógio nos três modos (hora, data e cronómetro), devem ser apresentados com dois dígitos.	Obrigatório
R5	O incremento dos campos é feito dentro dos limites respectivos, com retorno ao valor inicial após o valor máximo.	Obrigatório
R6	No incremento da data deve ser tido em conta o facto do ano poder ser bissexto.	Opcional
R7	A comutação entre os modos do relógio ocorre de acordo com a seguinte sequência cíclica: Hora ⇨ Data ⇨ Cronómetro	Obrigatório
R8	Na comutação de modo devem verificar-se os seguintes requisitos:	Obrigatório
R8.1	Deve ser possível mudar de modo estando o cronómetro em operação, devendo essa operação ser mantida mesmo quando o cronómetro não está visível.	Obrigatório
R8.2	Nos modos hora, data e agenda a mudança de modo resulta no reinício do modo actual.	Obrigatório

Projecto de Subsistemas



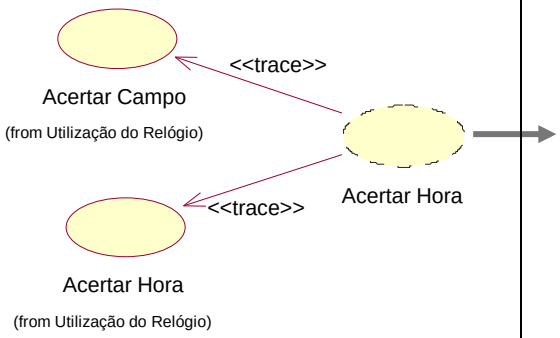
Padrão *Layers*

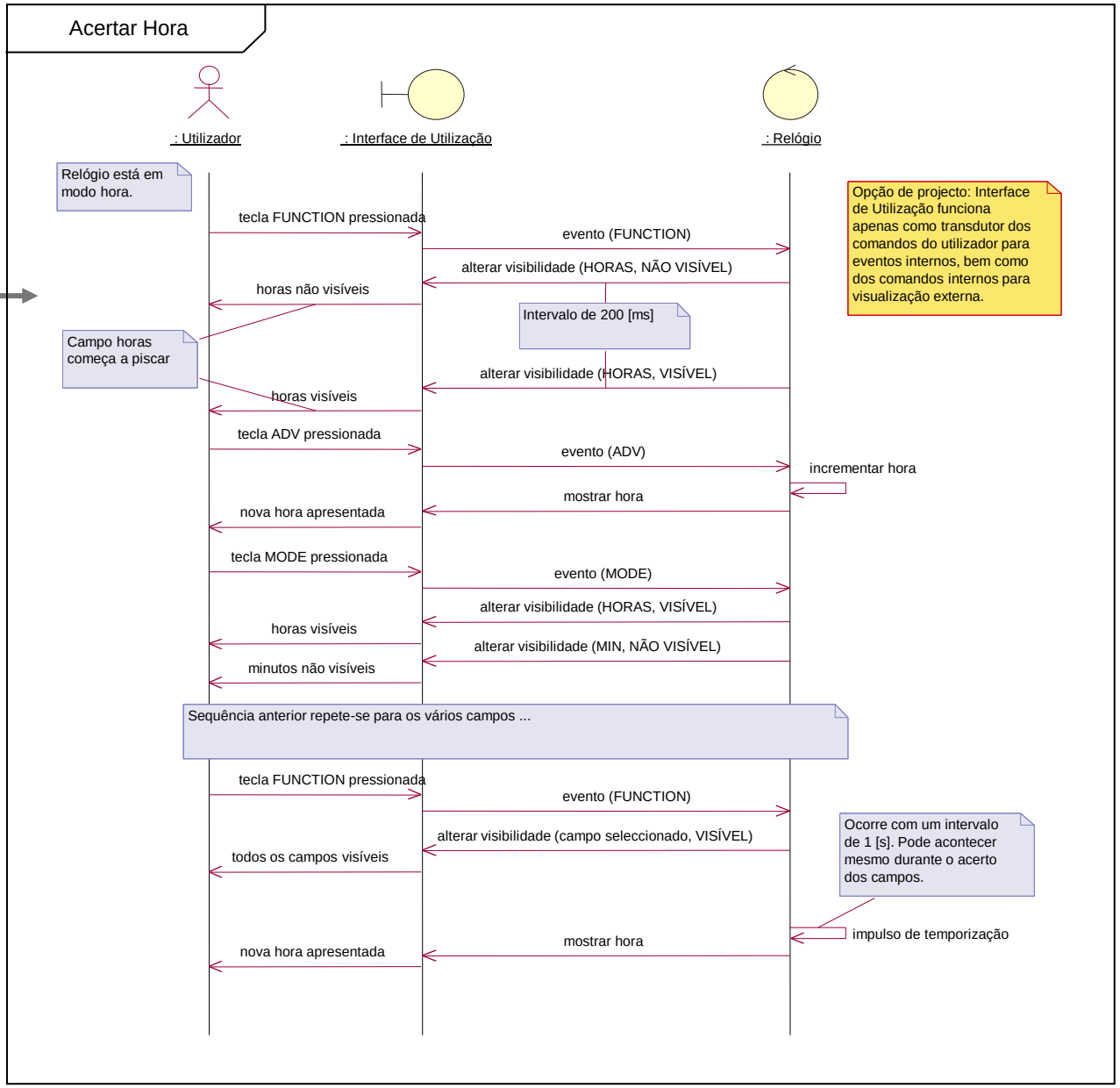
Modelo Computacional Genérico

Iteração de Transição Análise-Projecto

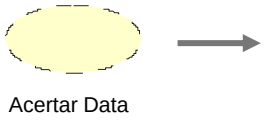
- Objectivo:
  - Compreensão e descrição de como construir a solução a realizar
- Actividades:
  - Realização de casos de utilização seleccionados
  - Projecto de subsistemas (detalhe)
  - Projecto de mecanismos gerais
  - Protótipo de teste de unidades e subsistemas críticos

Realização de Casos de Utilização

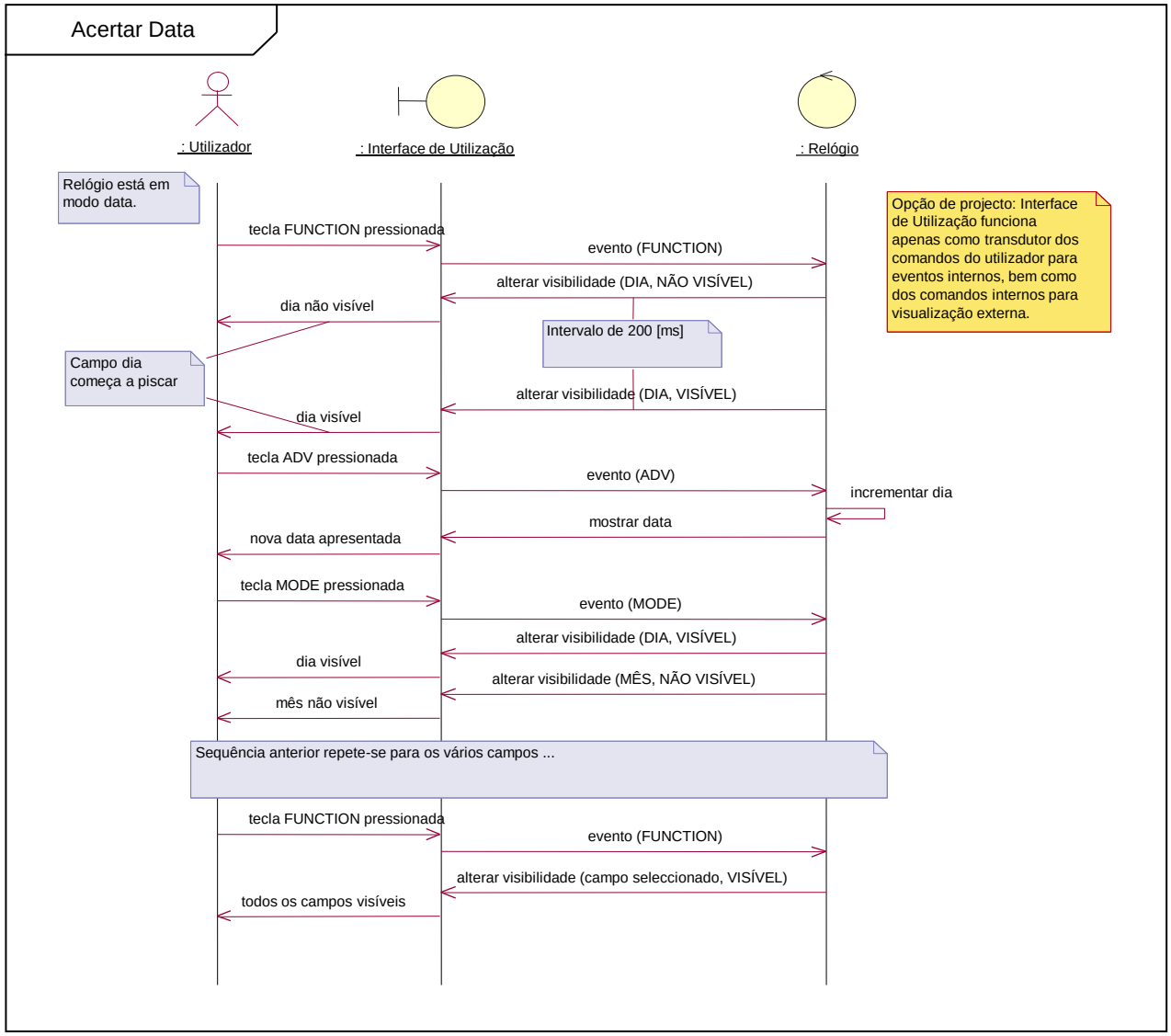




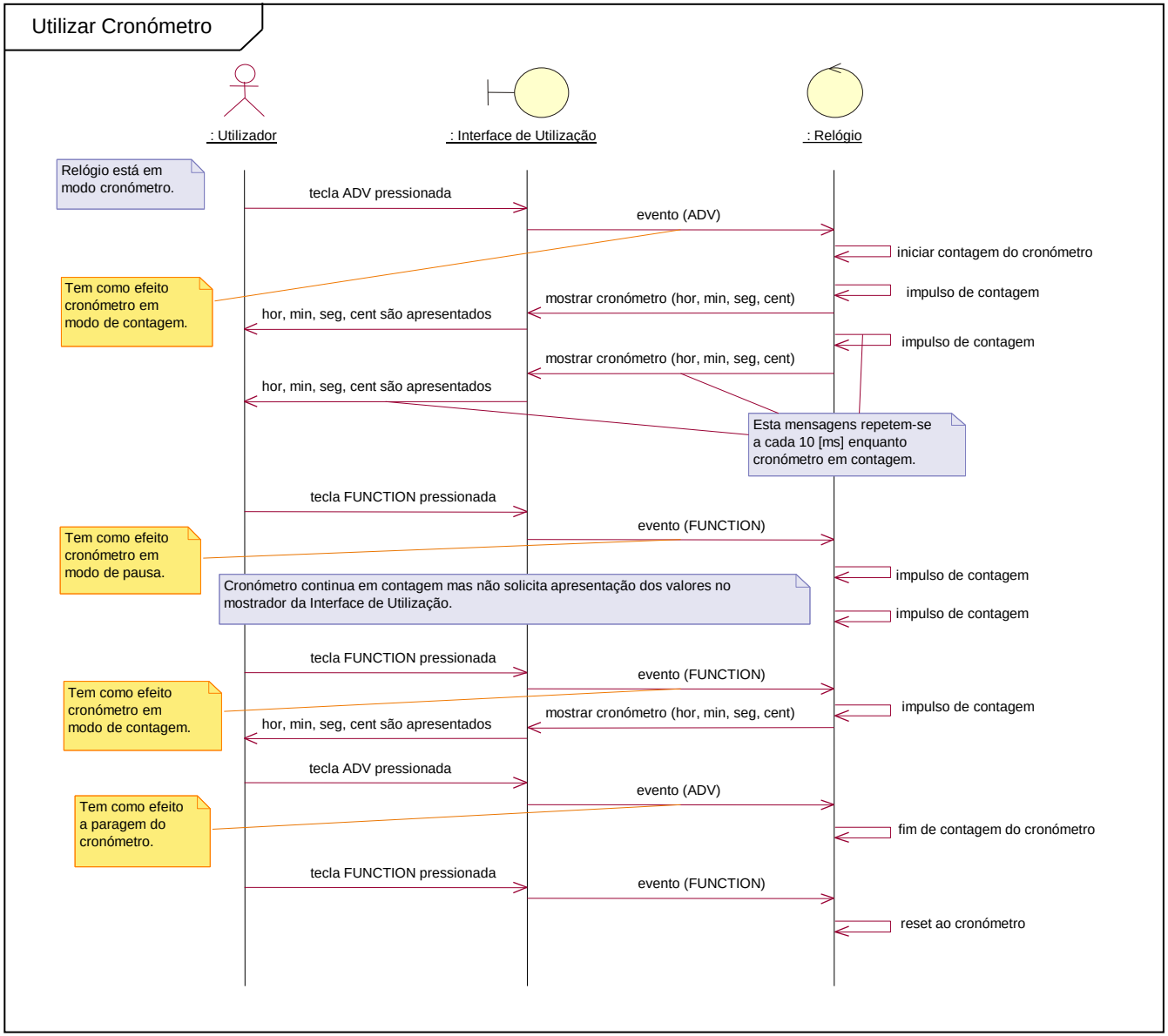
Realização de casos de Utilização



Acertar Data



Realização de casos de Utilização

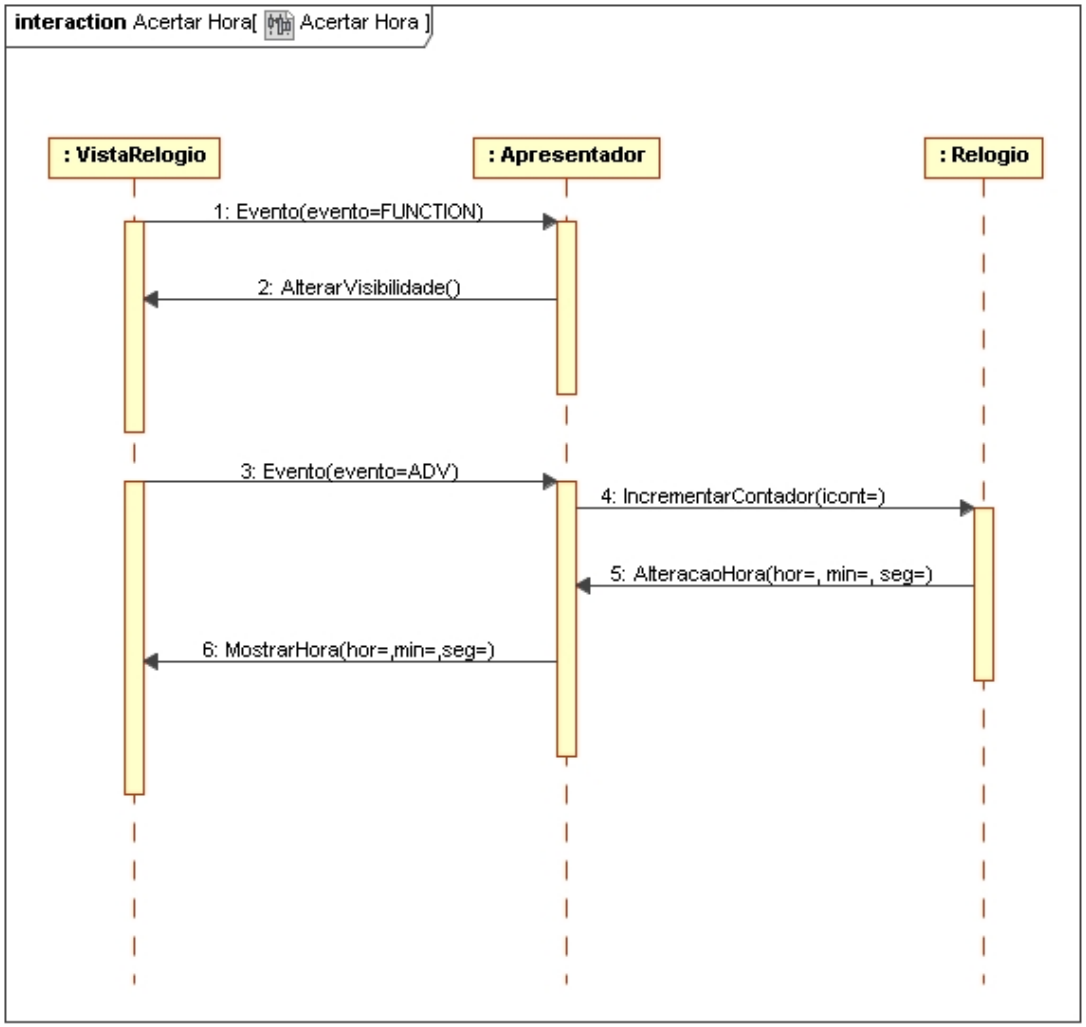


Realização de casos de Utilização

Operação como relógio

Padrão *MVP*

Acertar Hora

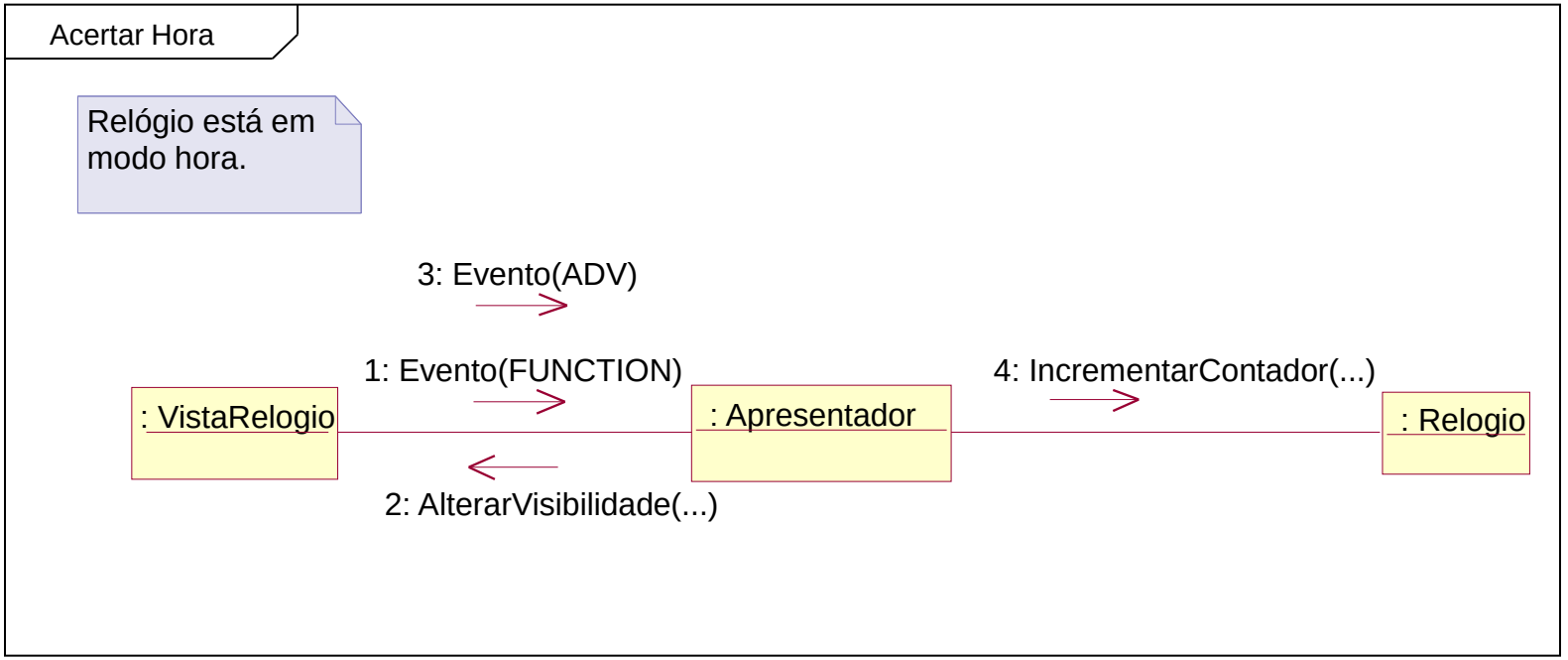


Realização de casos de Utilização

Operação como relógio

Padrão *MVP*

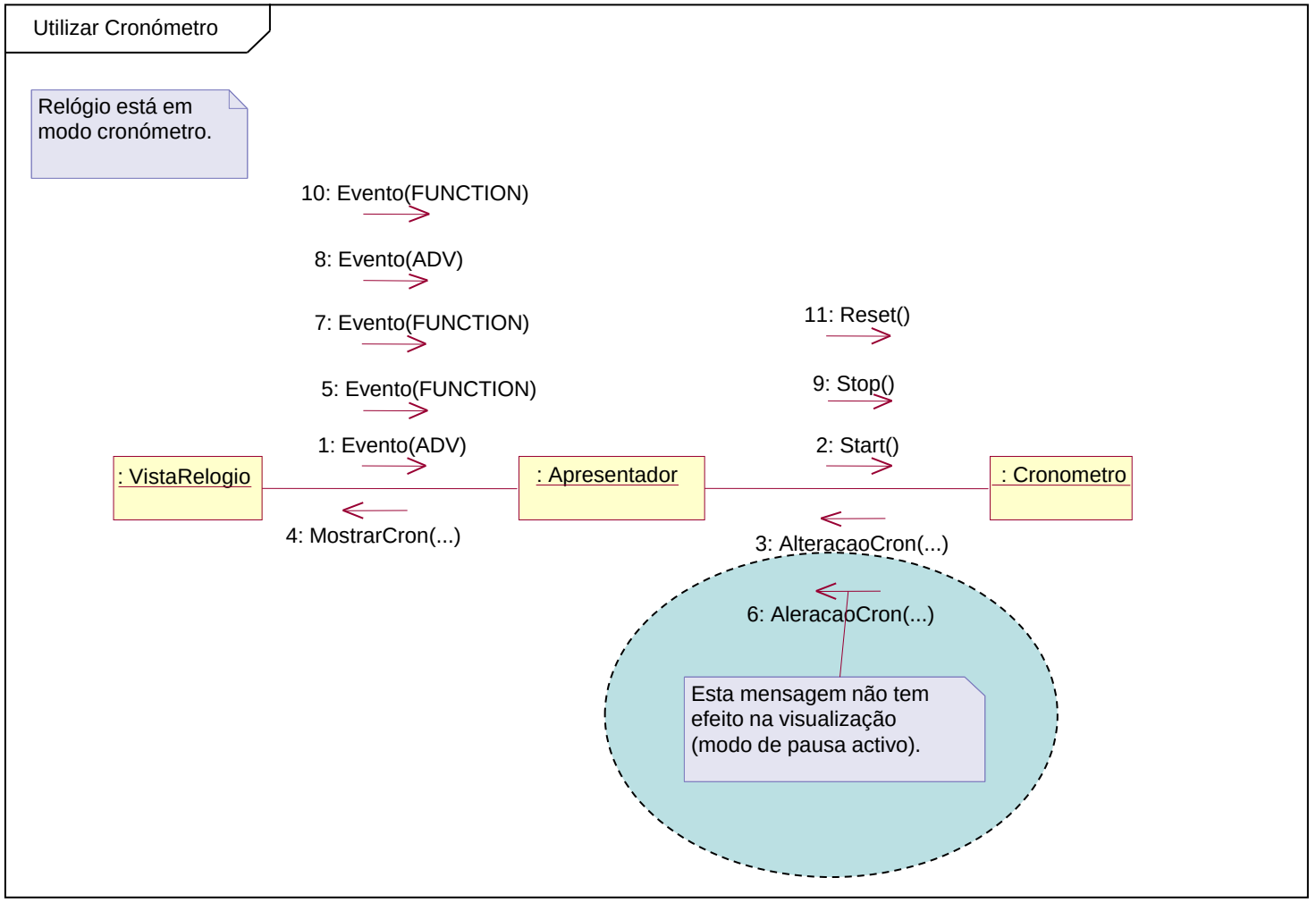
Acertar Hora



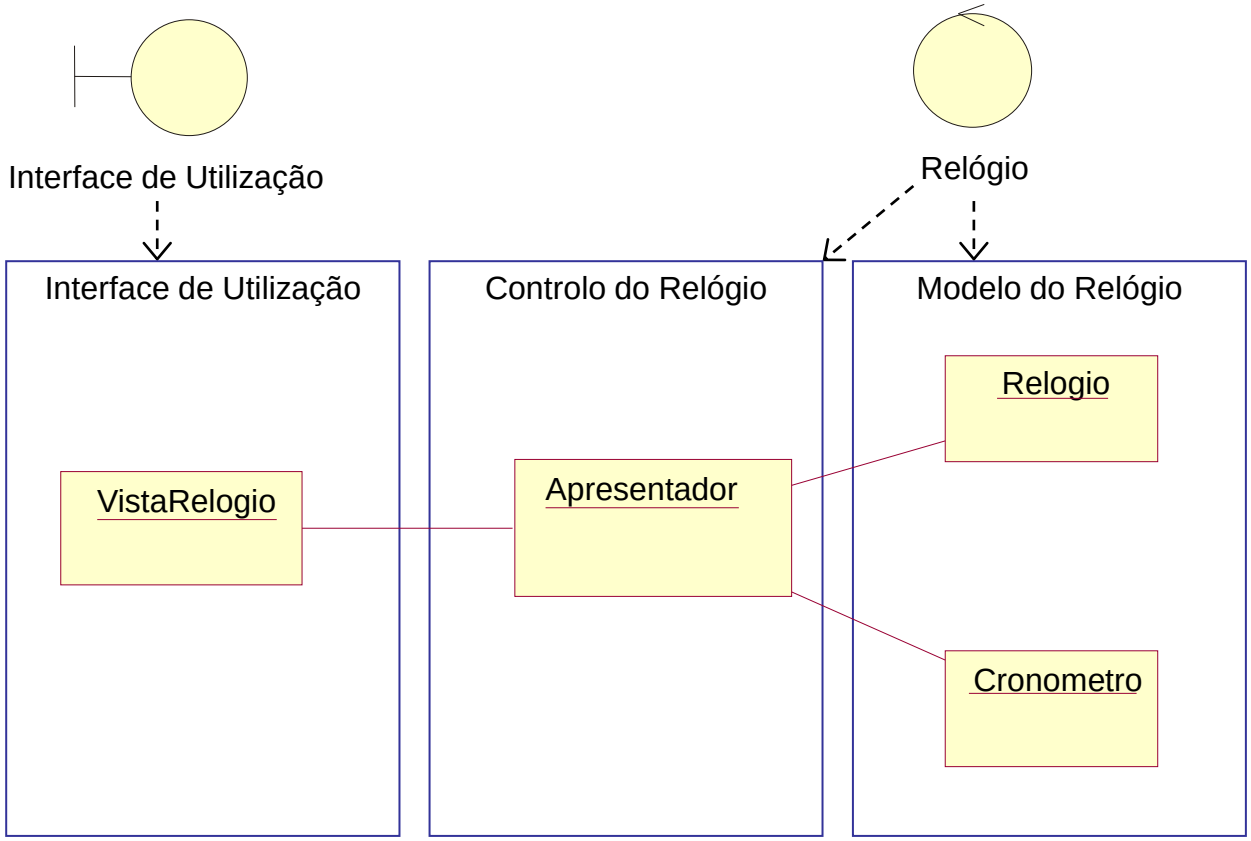
Realização de casos de Utilização

Operação como cronómetro

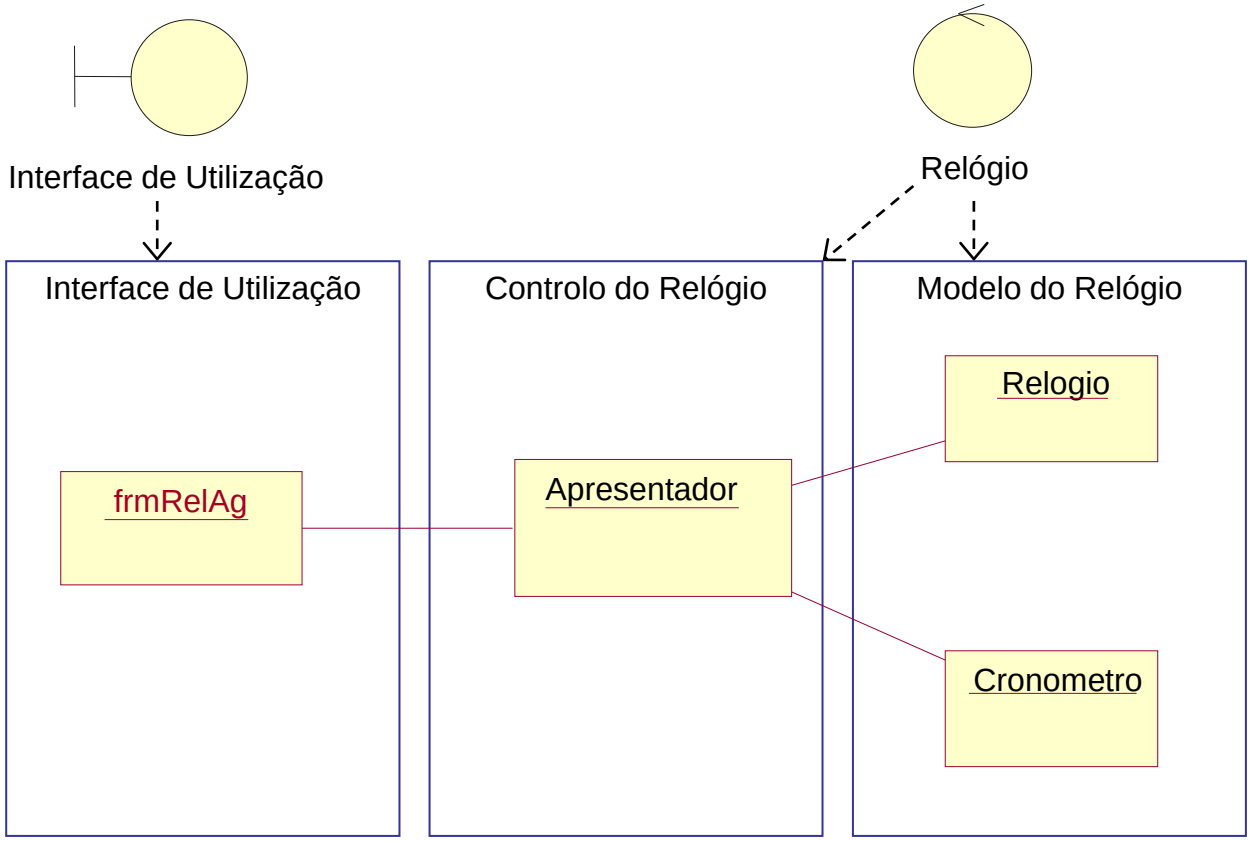
Utilizar Cronómetro



Projecto de Subsistemas

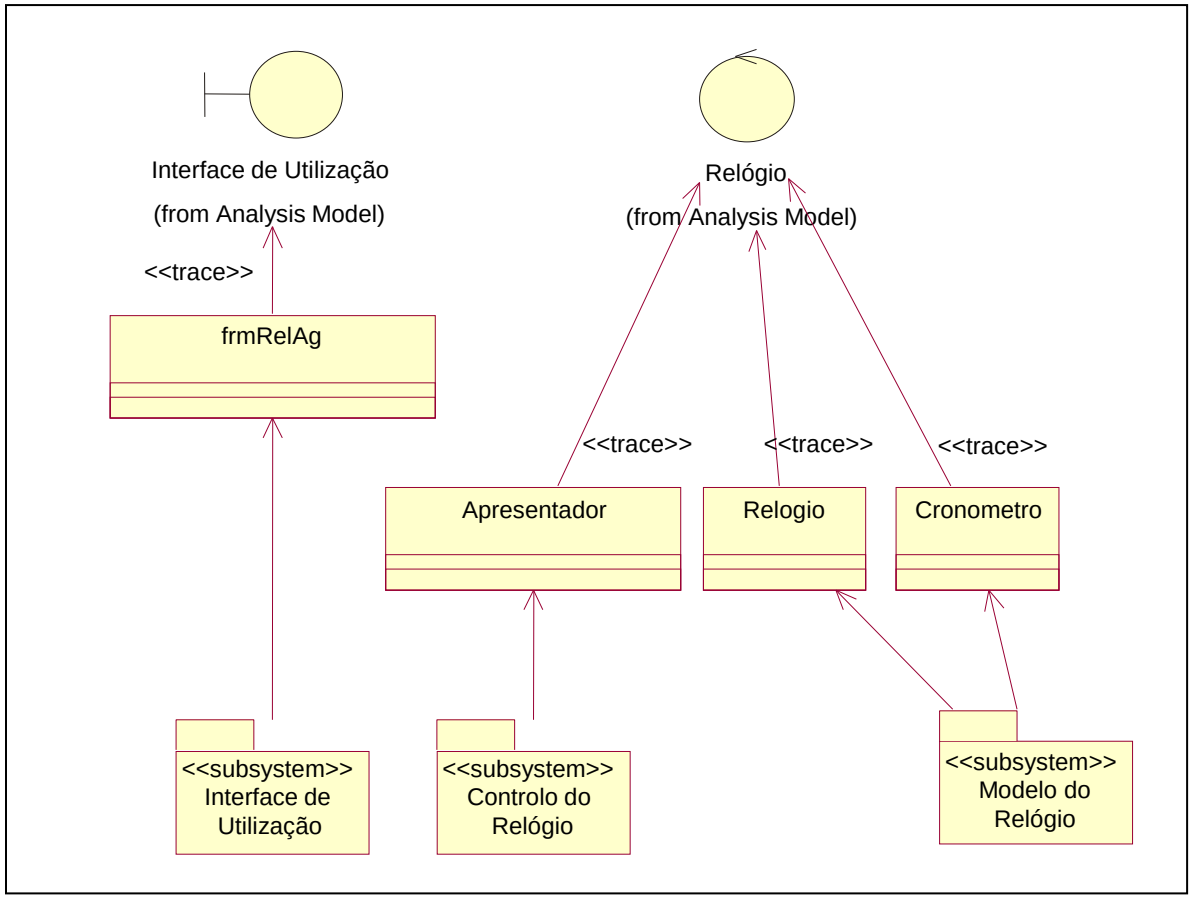


Projecto de Subsistemas



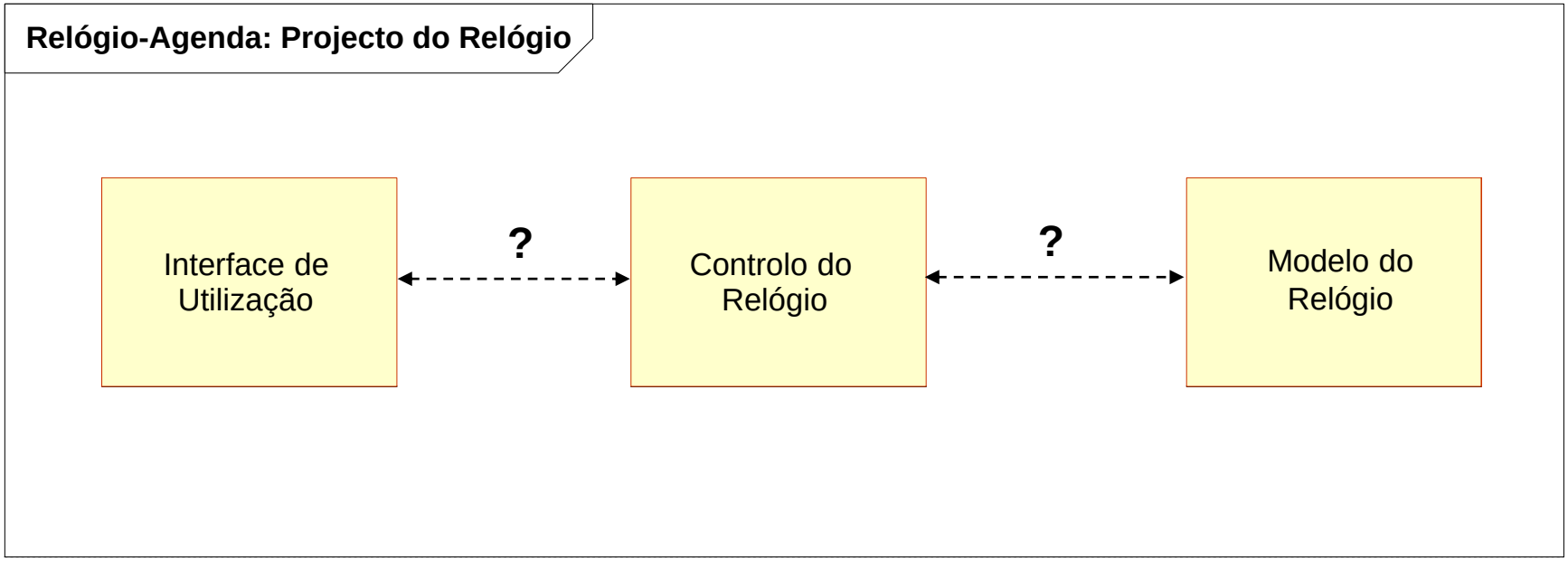
Adequação ao ambiente de desenvolvimento

Projecto de Subsistemas



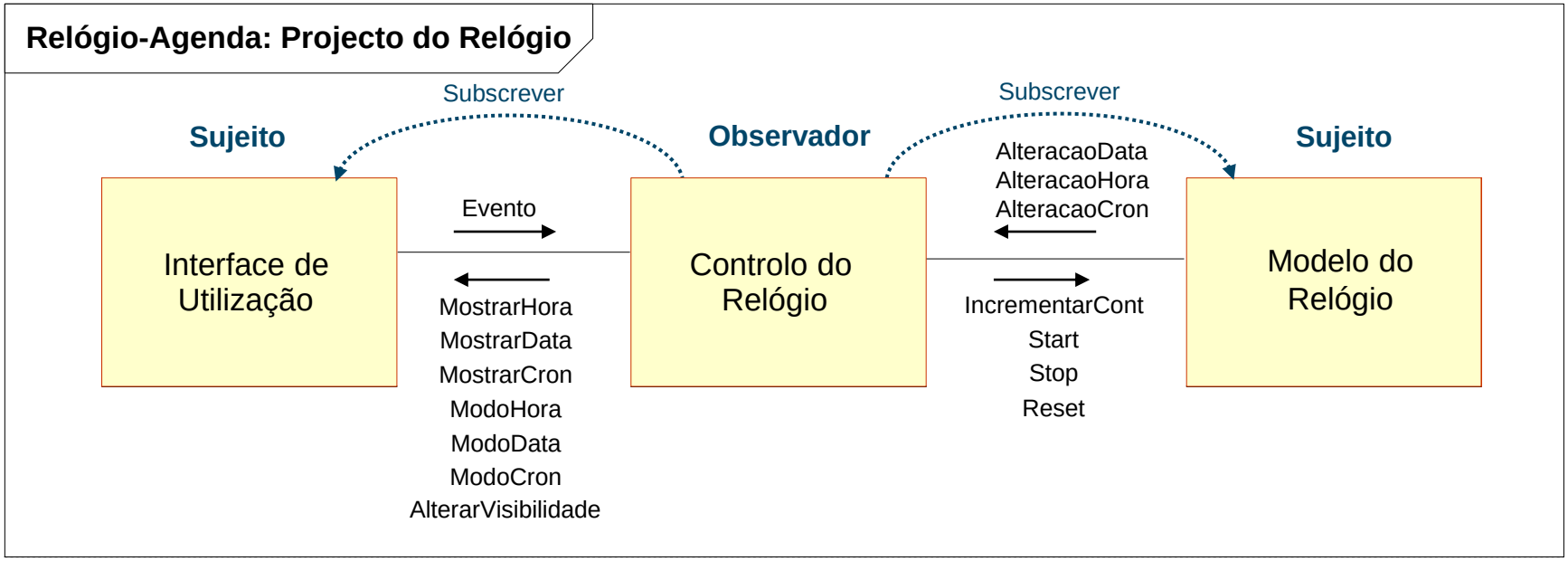
Mapeamento Análise - Projecto

Projecto de Subsistemas



Interacção entre subsistemas
Acoplamento entre subsistemas independente da implementação

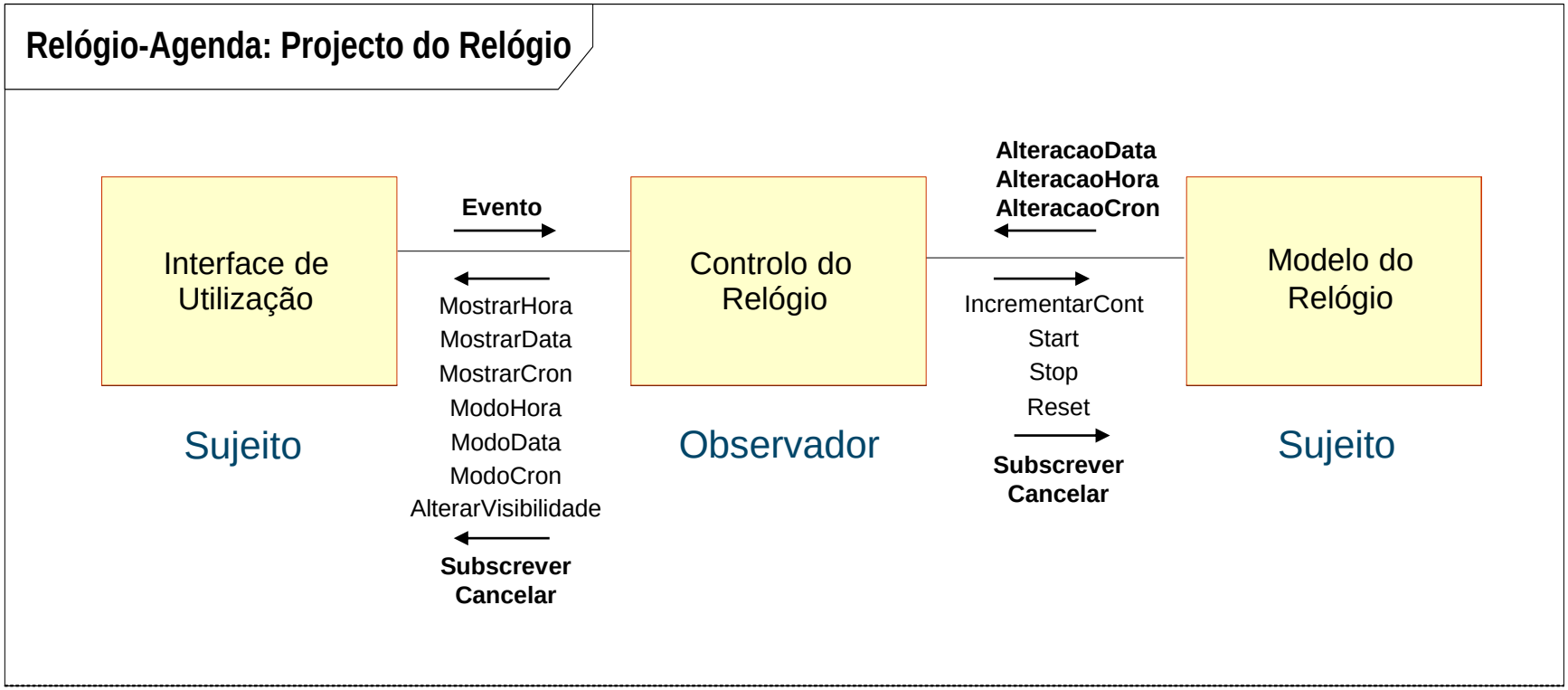
Projecto de Subsistemas



A partir de diagramas de sequência e de comunicação

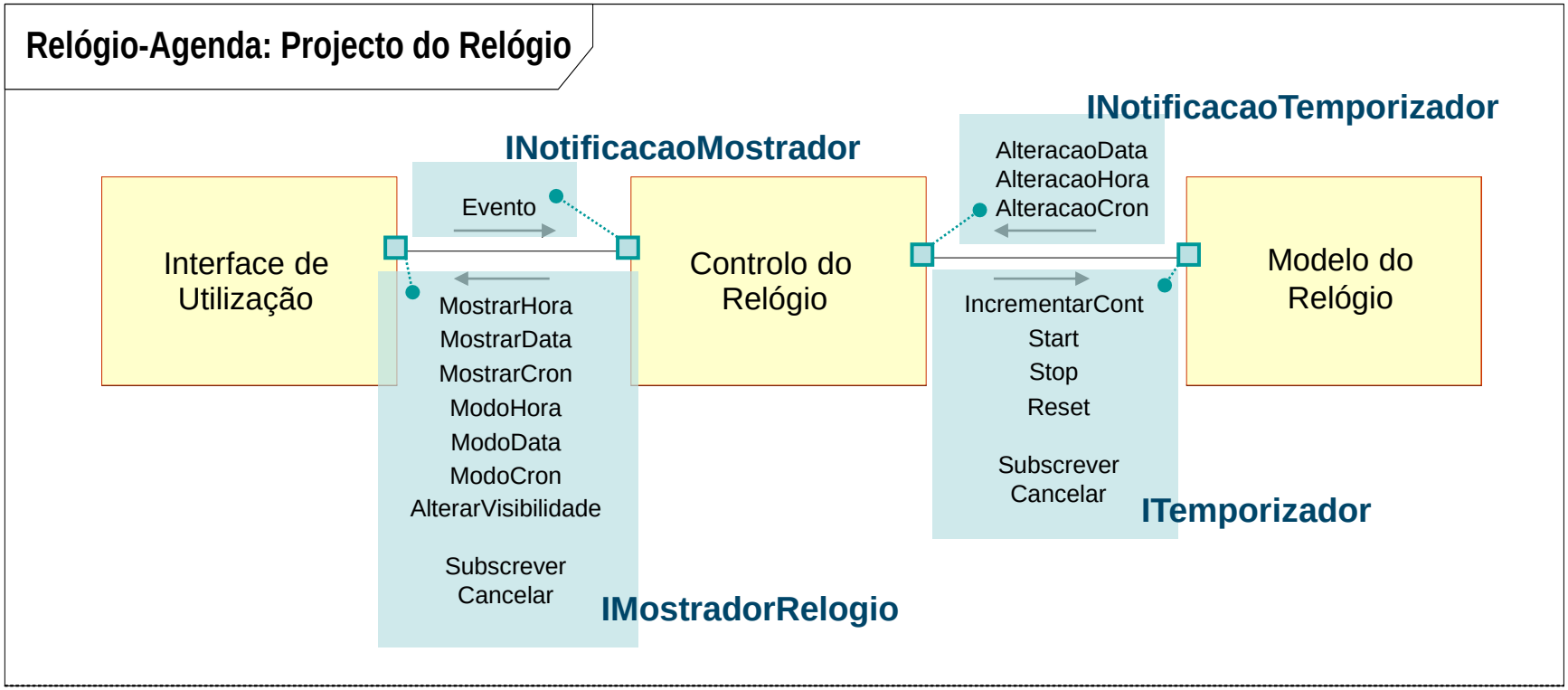
Padrão *Observer*

Projecto de Subsistemas



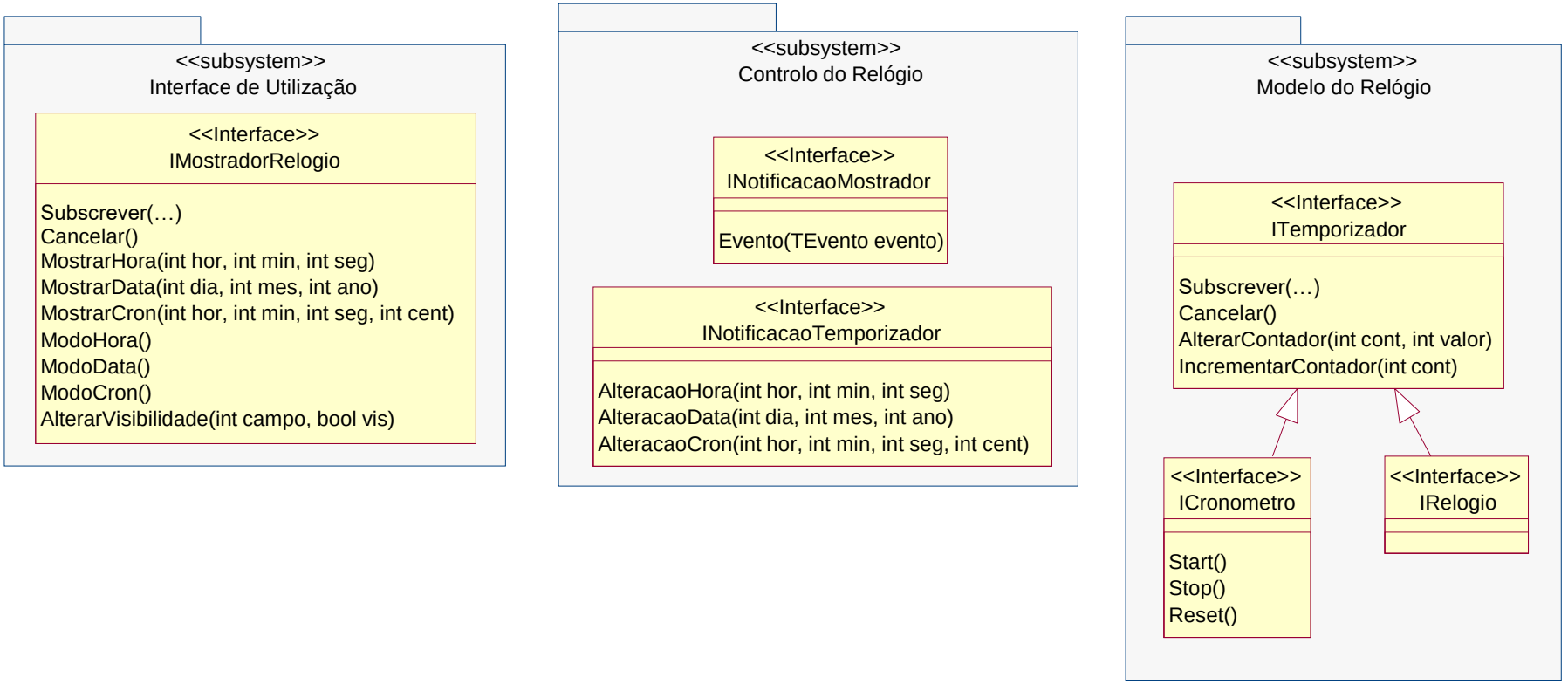
Padrão *Observer*

Projecto de Subsistemas



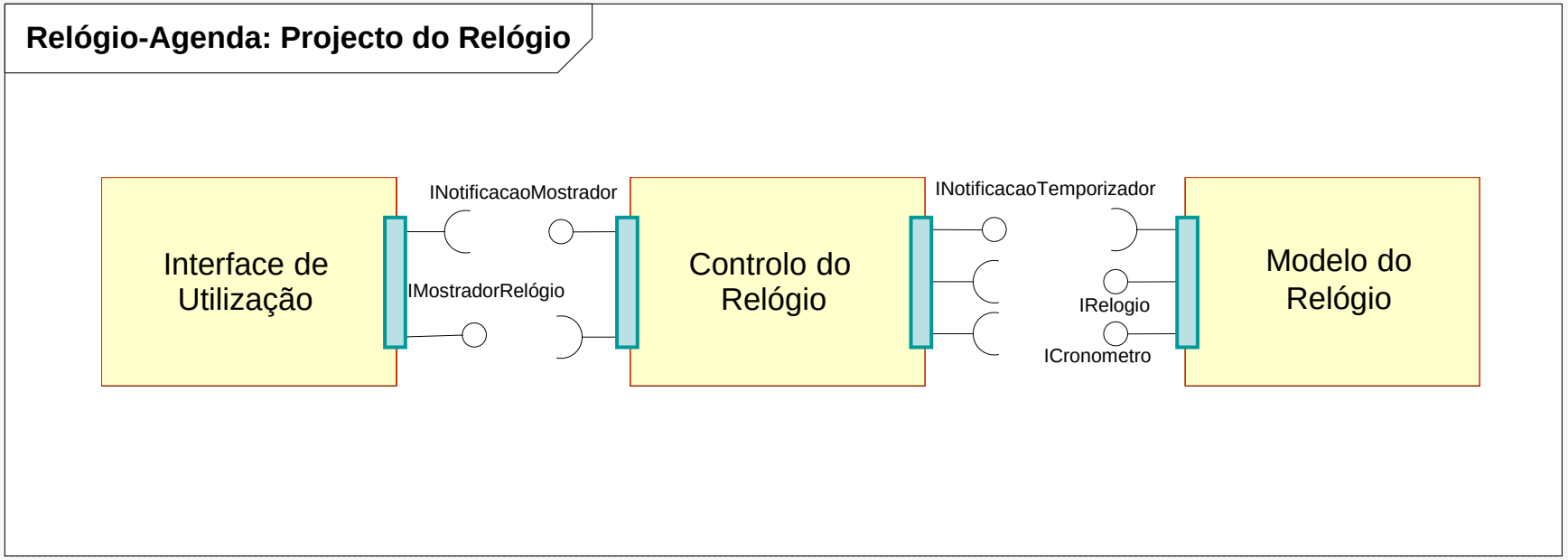
Definição de Interfaces

Projecto de Subsistemas



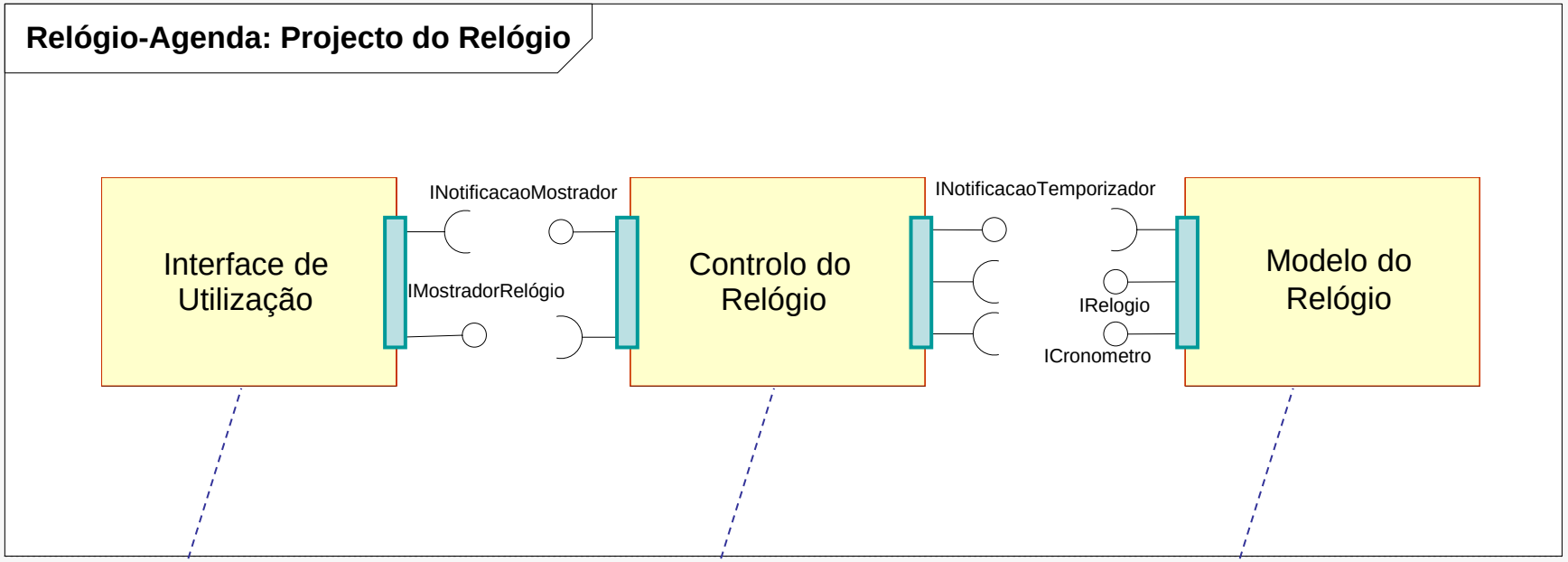
Refinamento de interfaces de interacção entre subsistemas

Projecto de Subsistemas



Definição de interfaces de interacção entre subsistemas
Modularização, encapsulamento

Projecto de Subsistemas



Gestão da interacção com o utilizador (apenas em termos de transdução)

Gestão e coordenação da interacção entre o modelo interno e a interface de utilização

Gestão e evolução dos contadores de data, hora e cronómetro

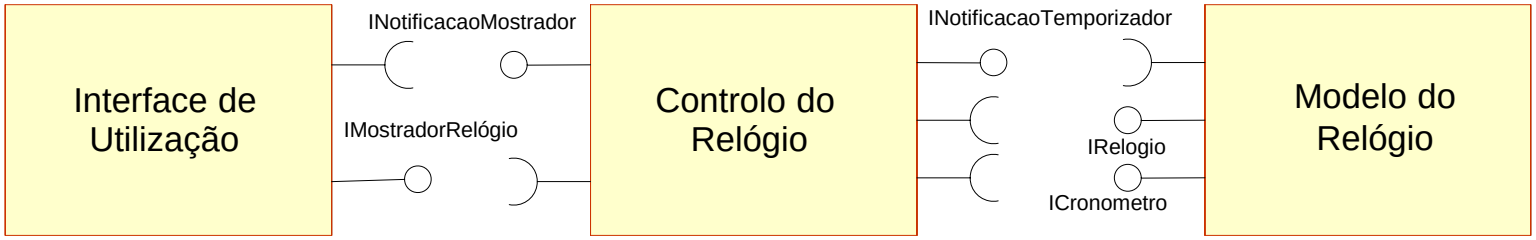
Atribuição de responsabilidades
(adequação a partir das responsabilidades identificadas na iteração anterior)

Iteração de Projecto 1

- Objectivo:
  - Definir os mecanismos que suportam a operação dos subsistemas com maior risco associado
- Actividades:
  - Realização de casos de utilização seleccionados
  - Projecto de mecanismos (detalhe)
  - Gestão de configurações
  - Protótipo de teste de unidades e subsistemas críticos

Projecto de Mecanismos

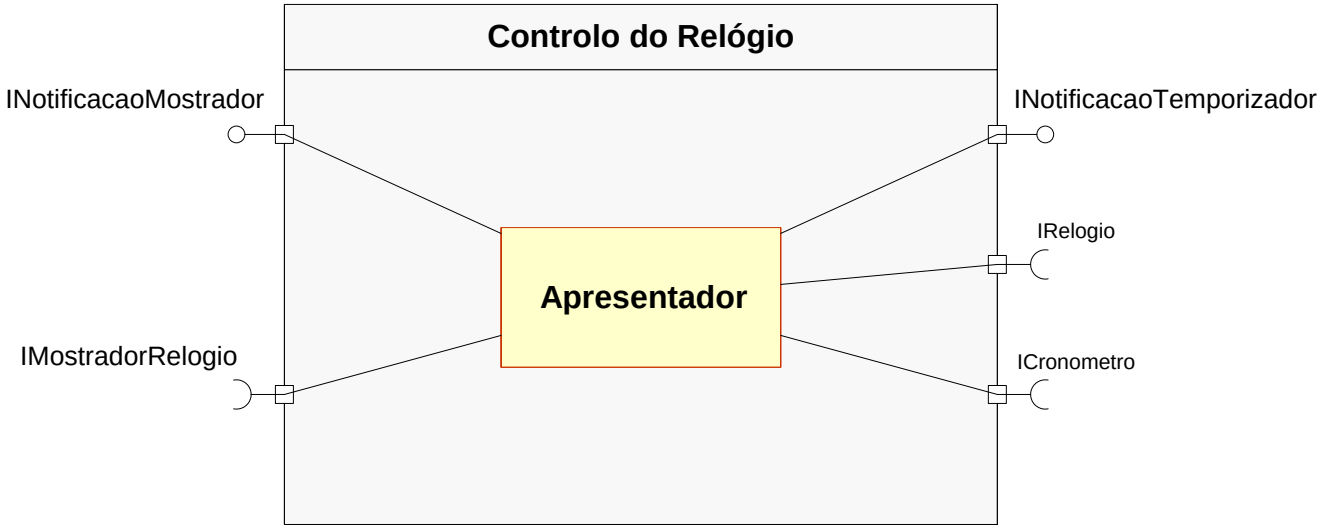
Focagem da iteração
Redução de risco



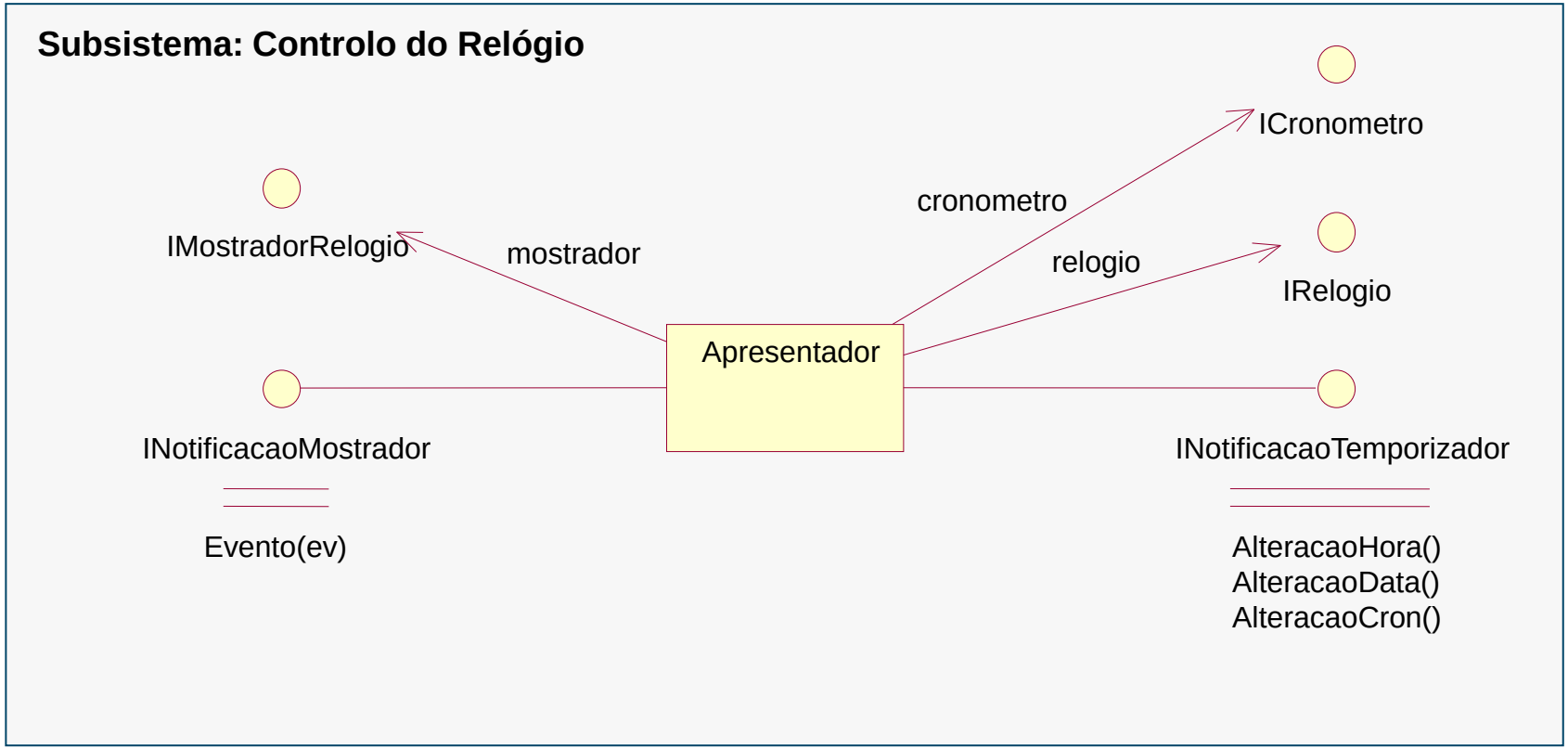
Risco de desenvolvimento ?	Risco de desenvolvimento ?	Risco de desenvolvimento ?
Baixo	**Alto**	**Baixo**

Projecto de Mecanismos

Delimitação de âmbito de intervenção
Modularização e encapsulamento

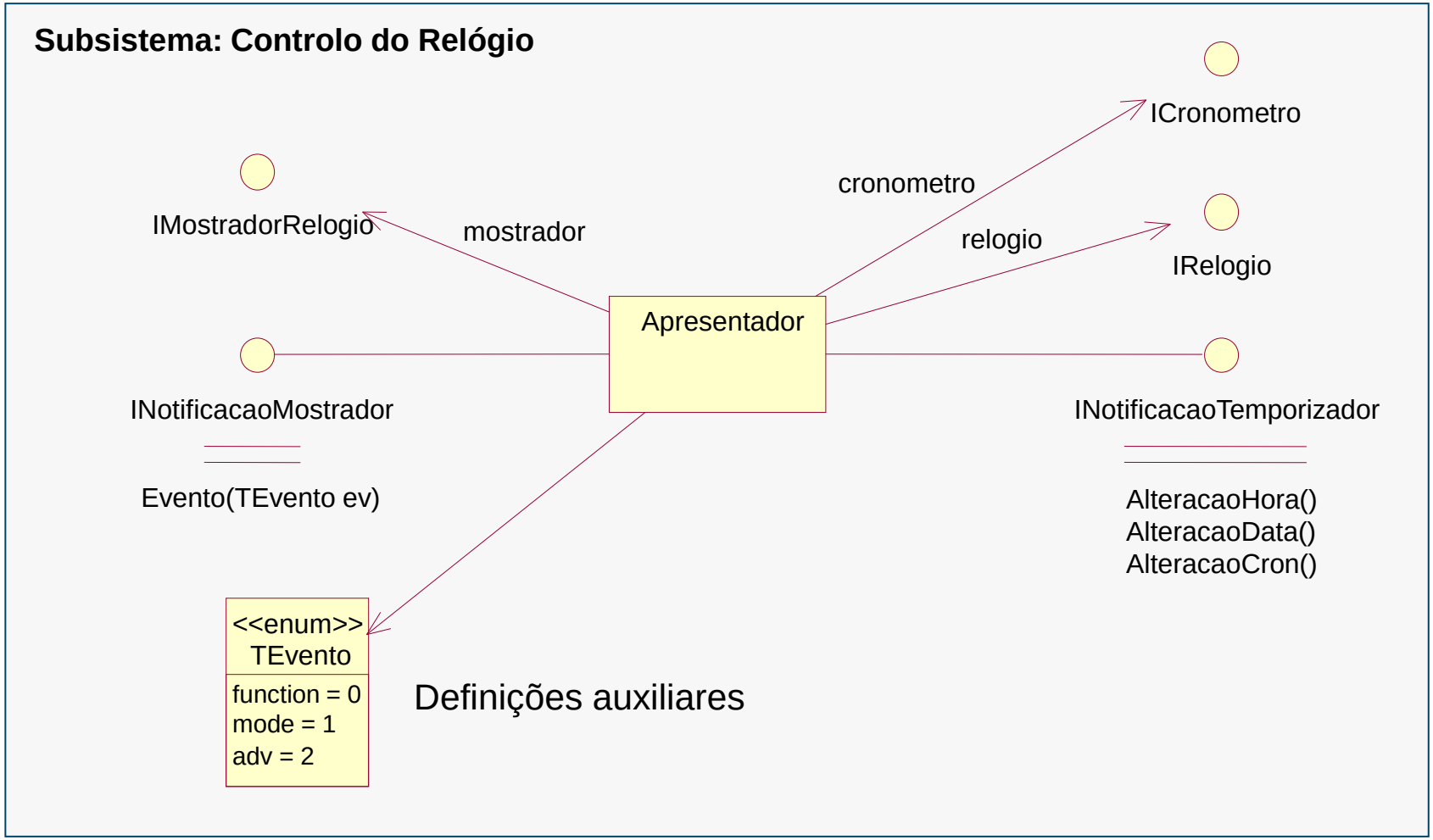


Projecto de Mecanismos

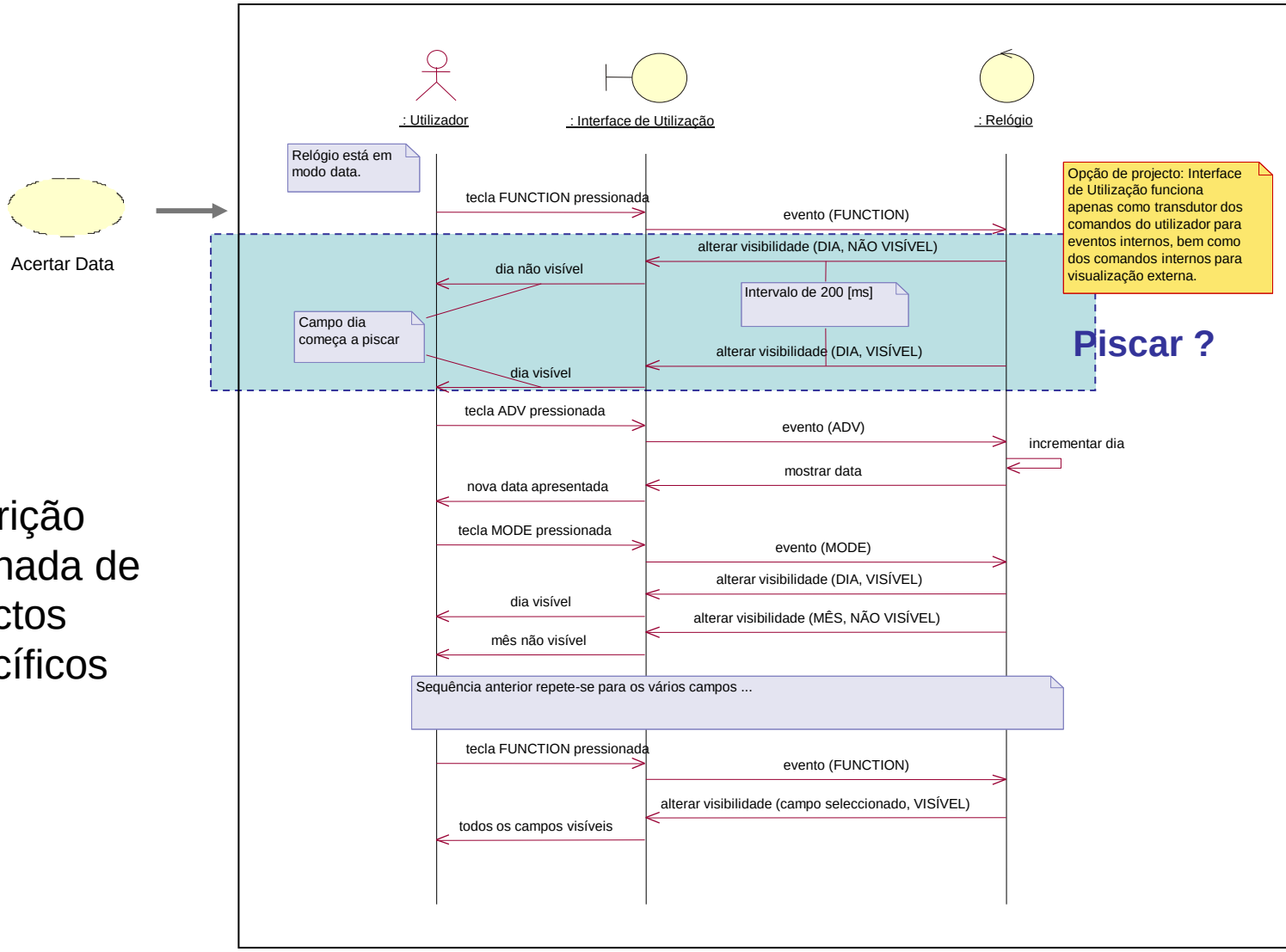


Definição de comportamento suportado

Projecto de Mecanismos

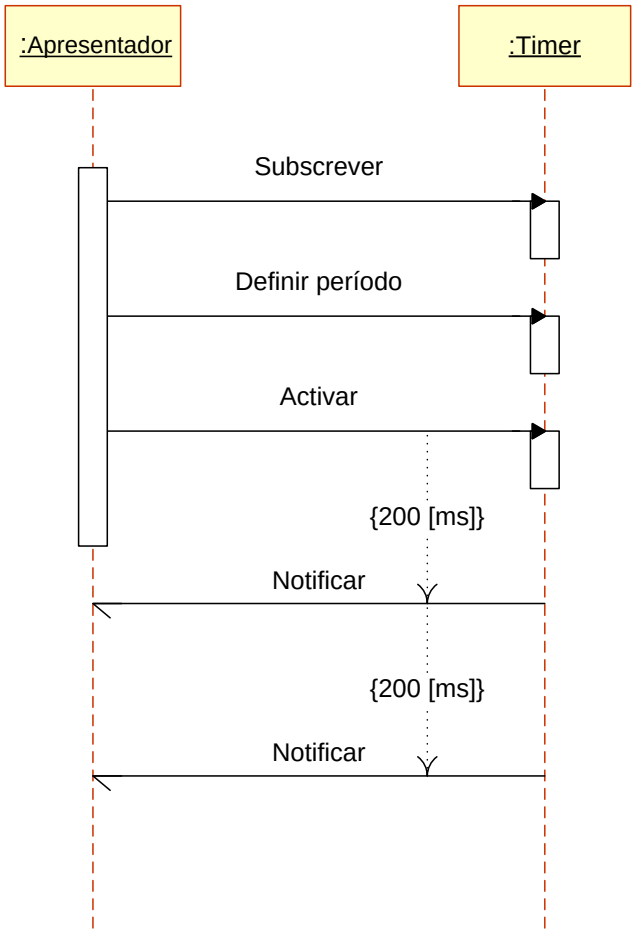


Projecto de Mecanismos

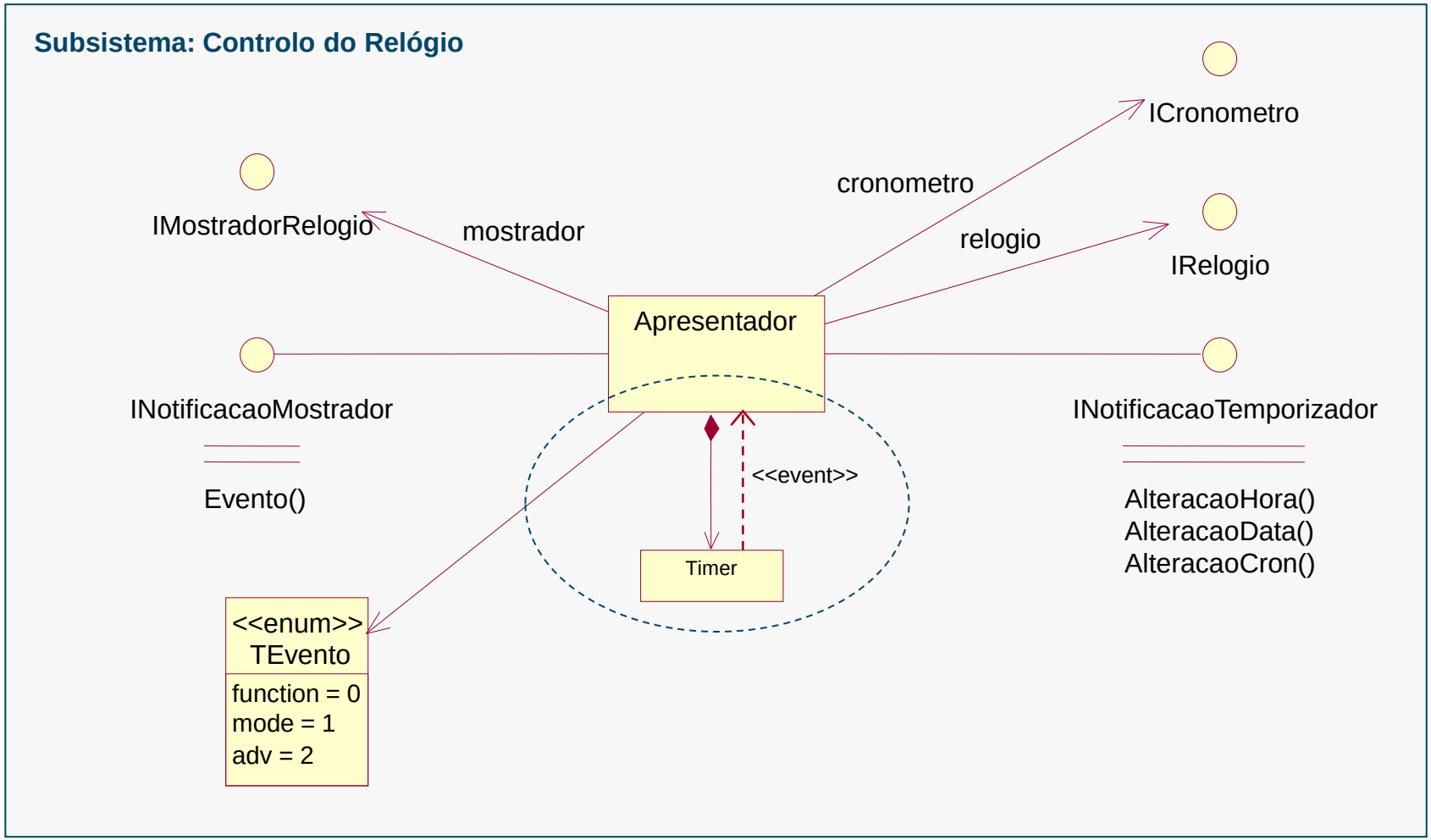


Descrição detalhada de aspectos específicos

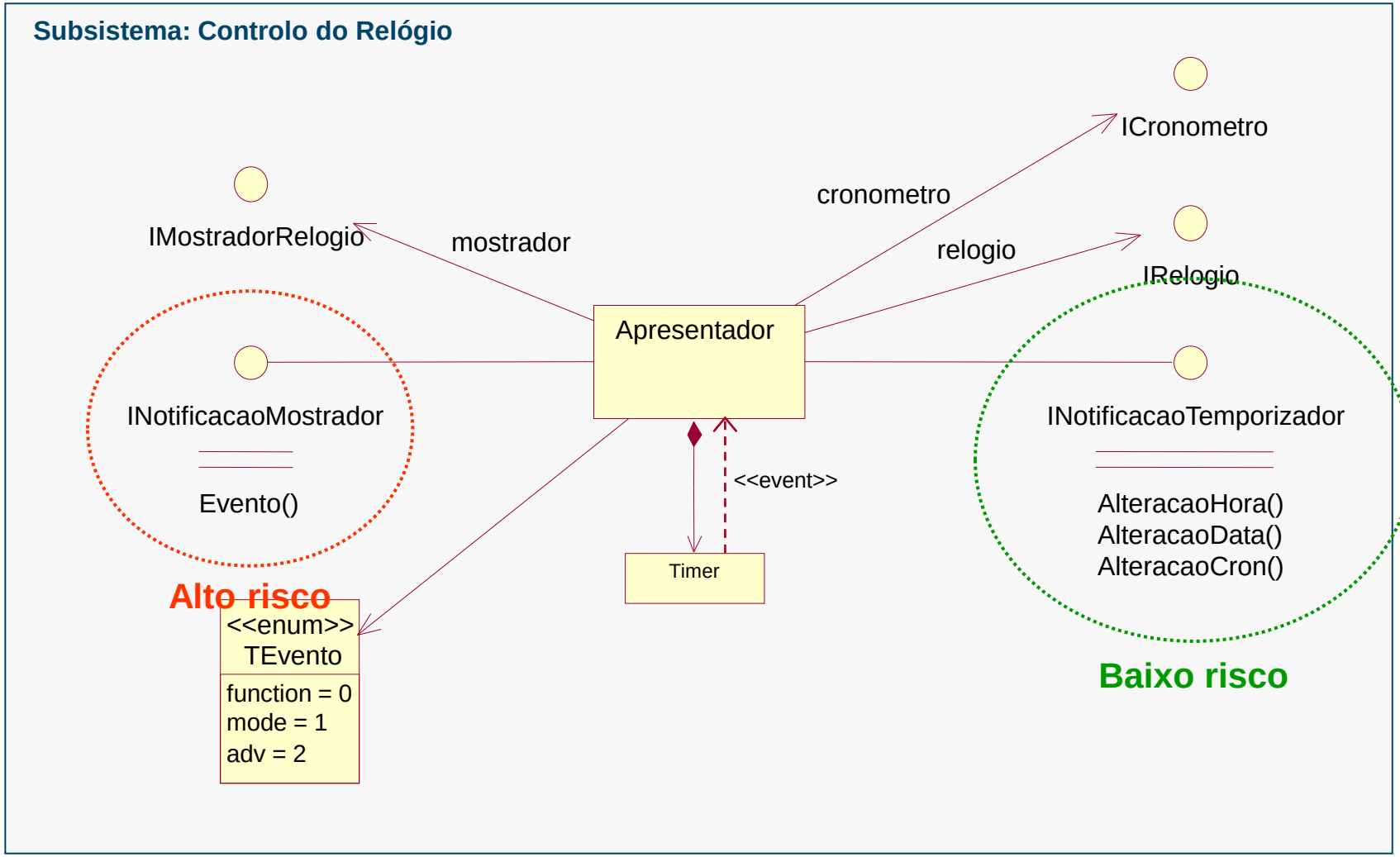
Projecto de Mecanismos



Projecto de Mecanismos

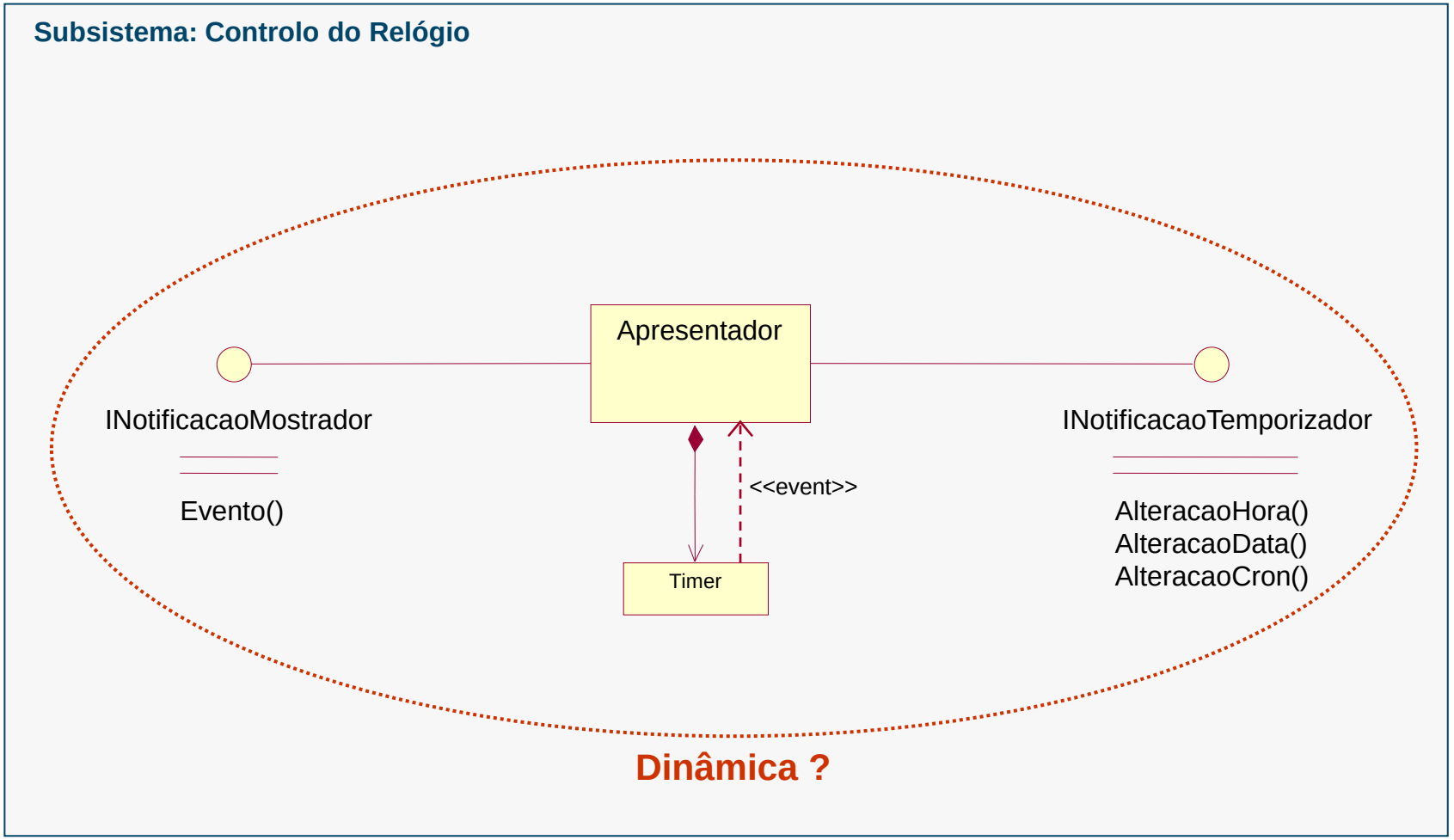


Projecto de Mecanismos



Definição de comportamento

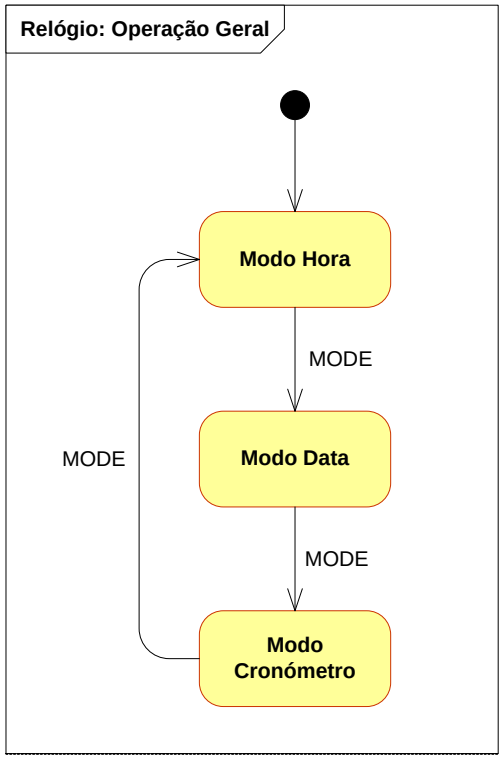
Projecto de Mecanismos



Projecto de Mecanismos

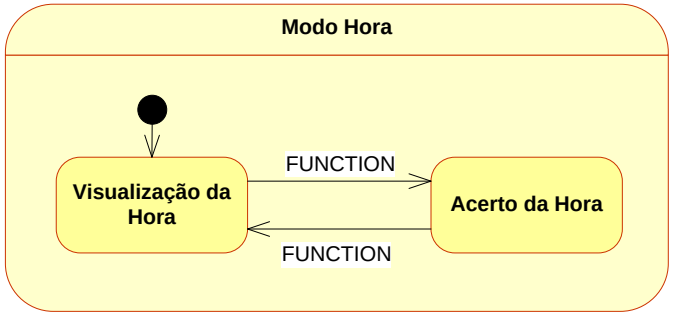
R7	A comutação entre os modos do relógio ocorre de acordo com a seguinte sequência cíclica: Hora ⇨ Data ⇨ Cronómetro	Obrigatório

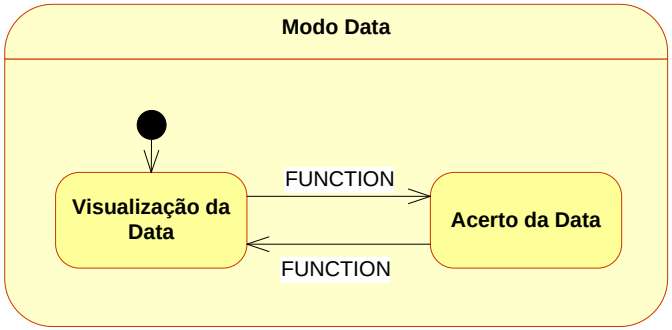
A comutação entre modos ocorre através do botão MODE
(Caso de Utilização **Visualizar Modo)**



Projecto de Mecanismos

Casos de Utilização **Visualizar Modo**, **Acertar Hora** e **Acertar Data**



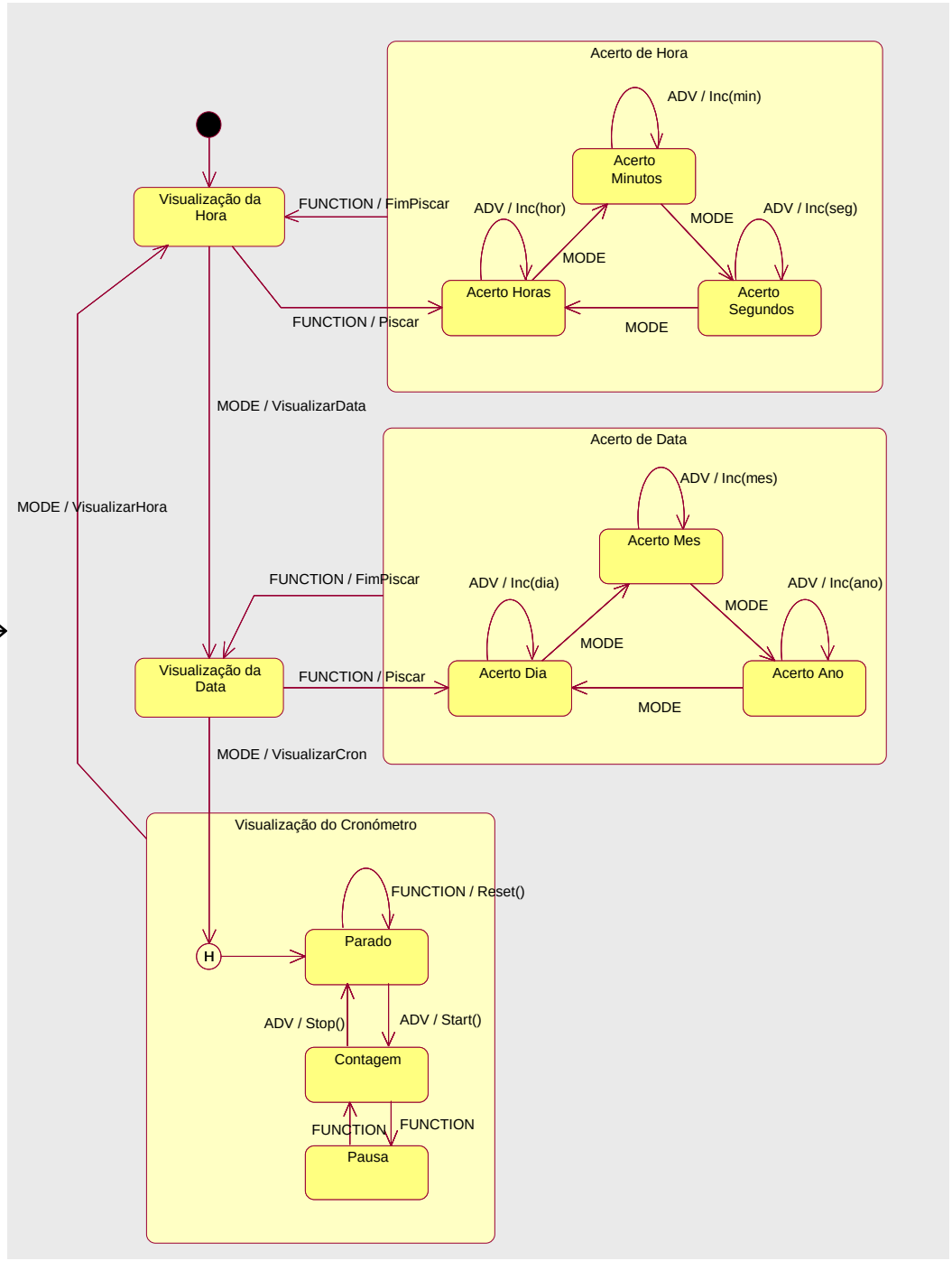


Projecto de Mecanismos

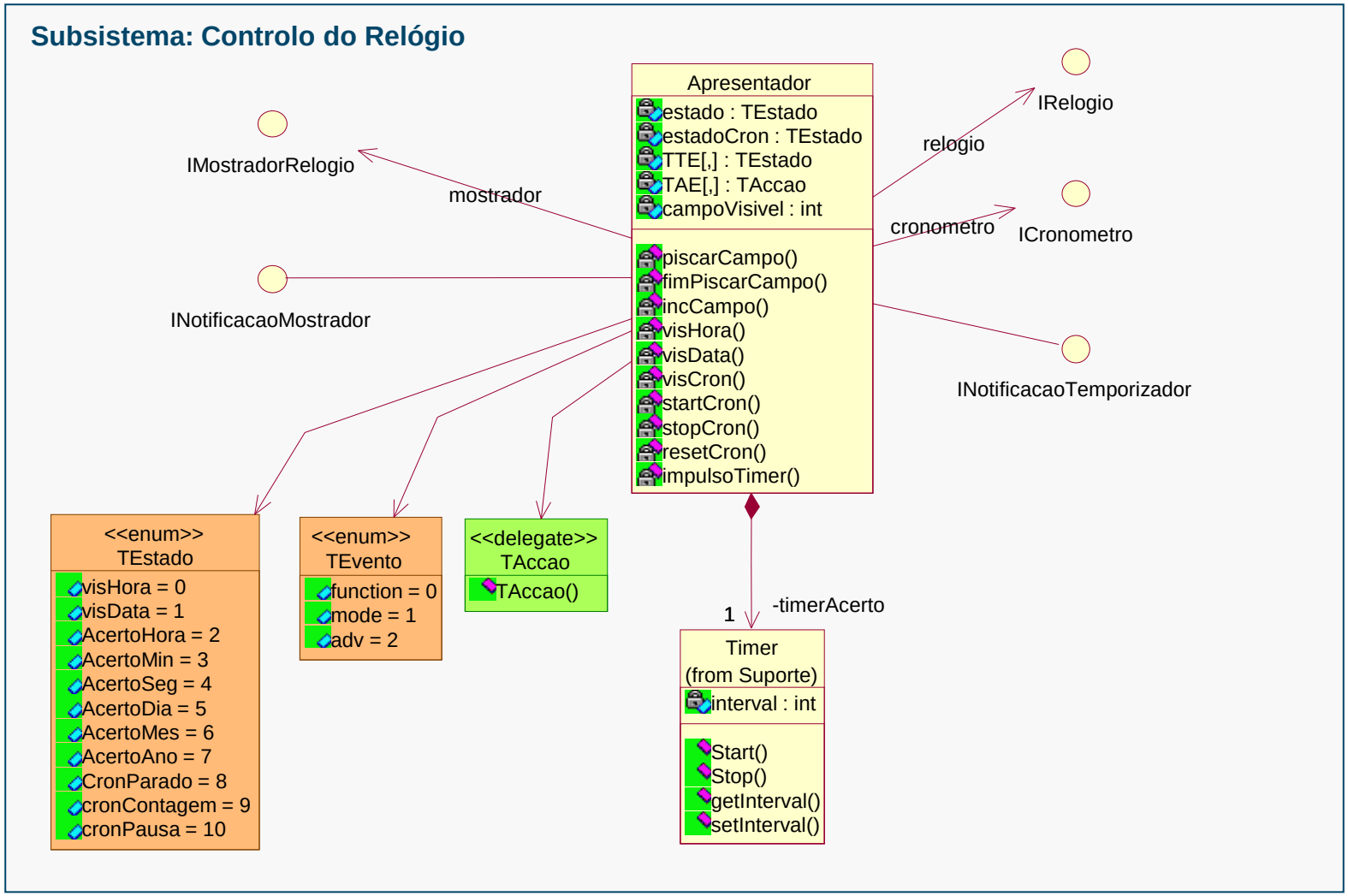
Detalhe da Dinâmica

Subsistema: Controlo do Relógio

Apresentador

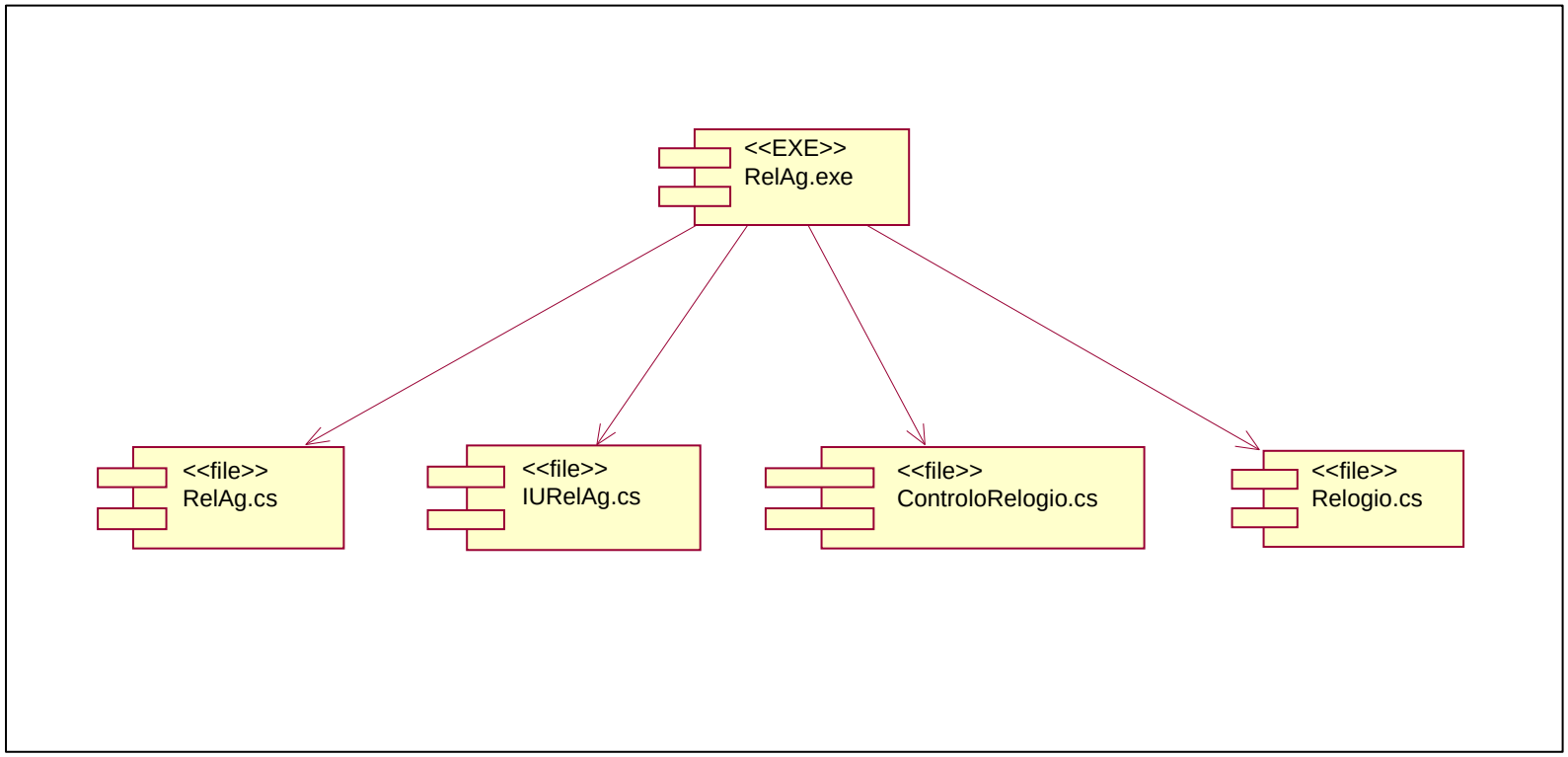


Projecto de Mecanismos



Detalhe de mecanismos

Gestão de Configurações



Modelo de implementação

Protótipo Demonstrador

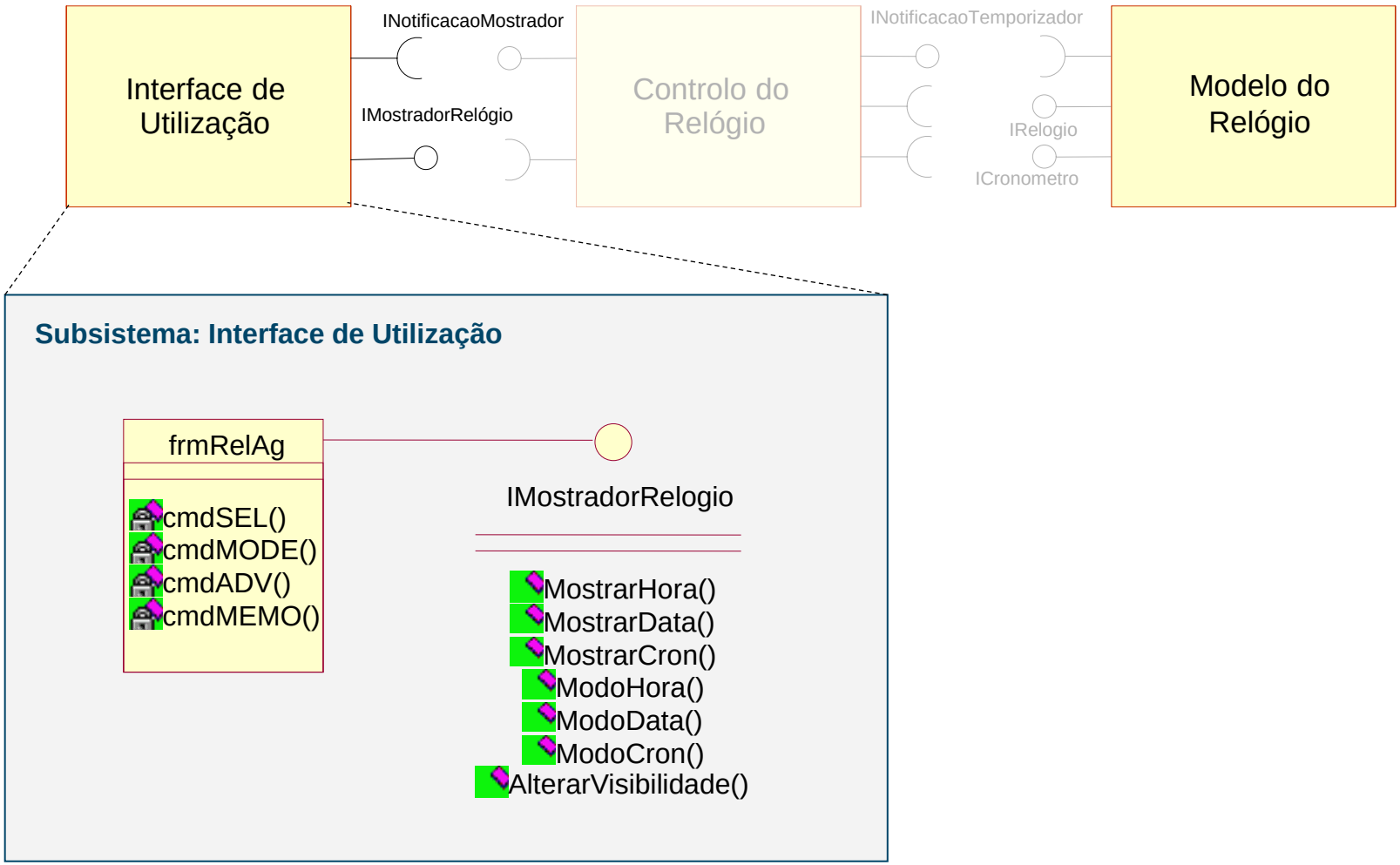
Modo relógio

Modo agenda

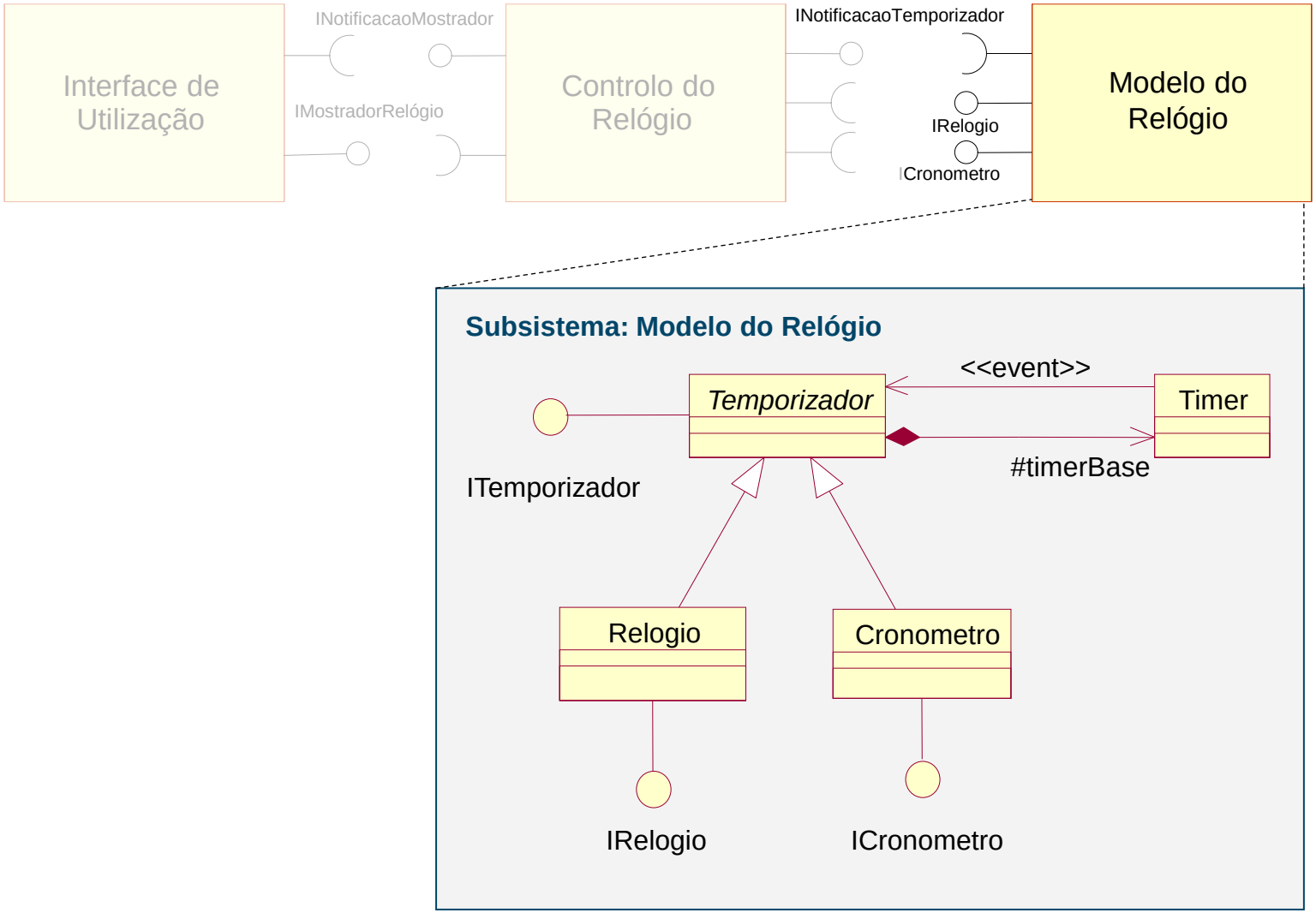
Iteração de Projecto 2

- Objectivo:
  - Detalhar os mecanismos que suportam a operação dos vários subsistemas
  - Garantir a integração correcta dos subsistemas
- Actividades:
  - Projecto de mecanismos
  - Projecto detalhado
  - Testes de integração
  - Protótipo funcional preliminar

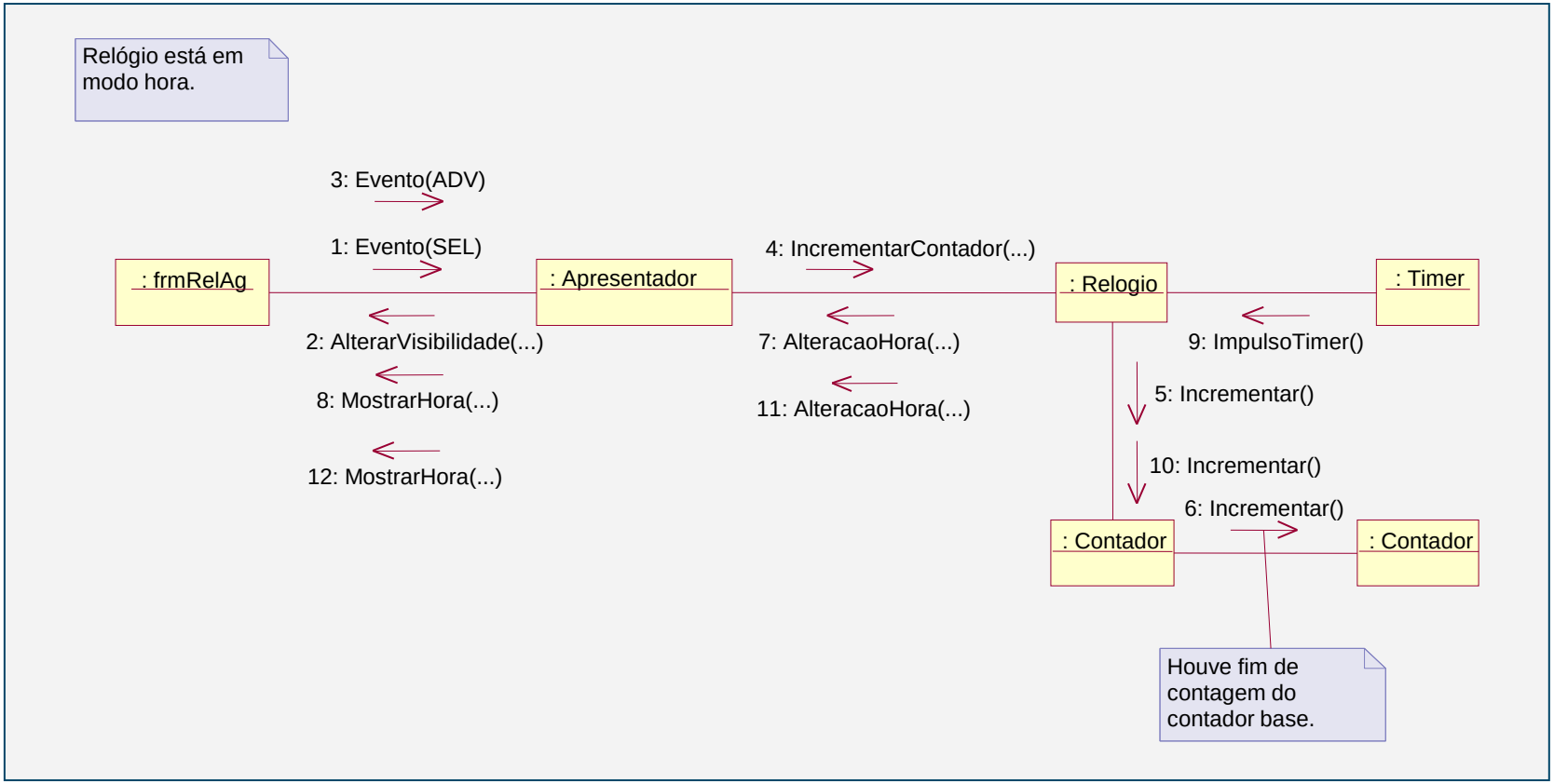
Projecto de Mecanismos



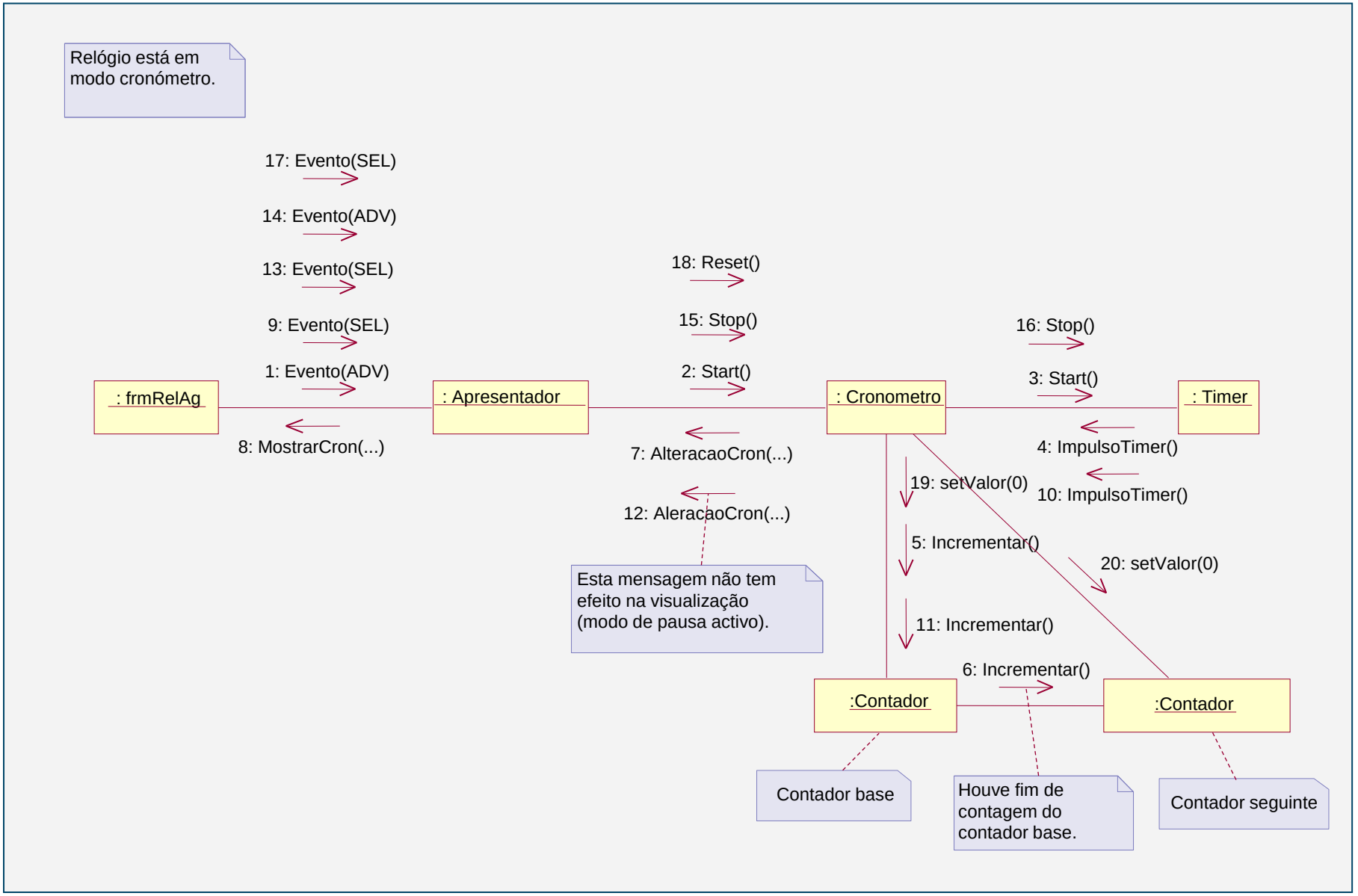
Projecto de Mecanismos



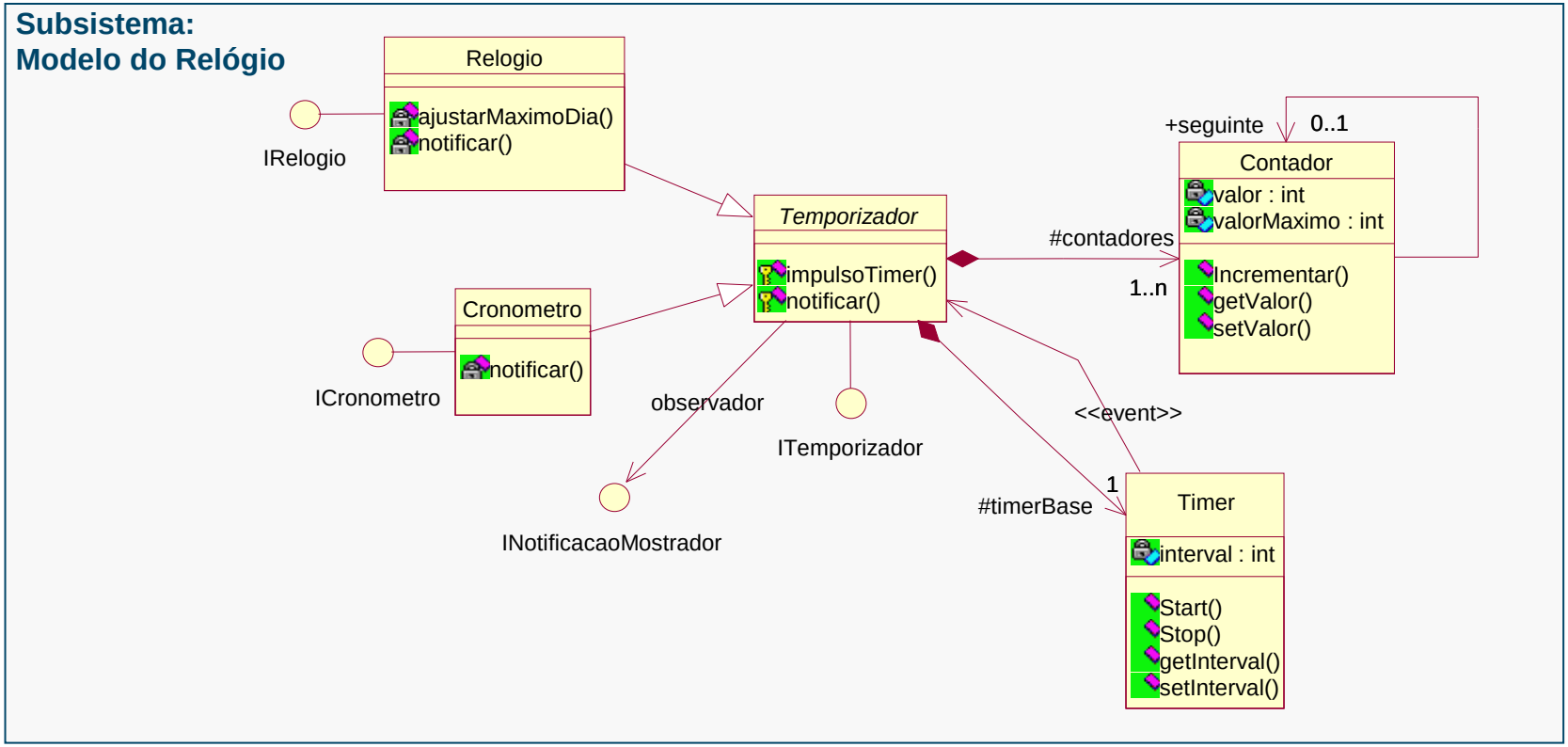
Projecto Detalhado



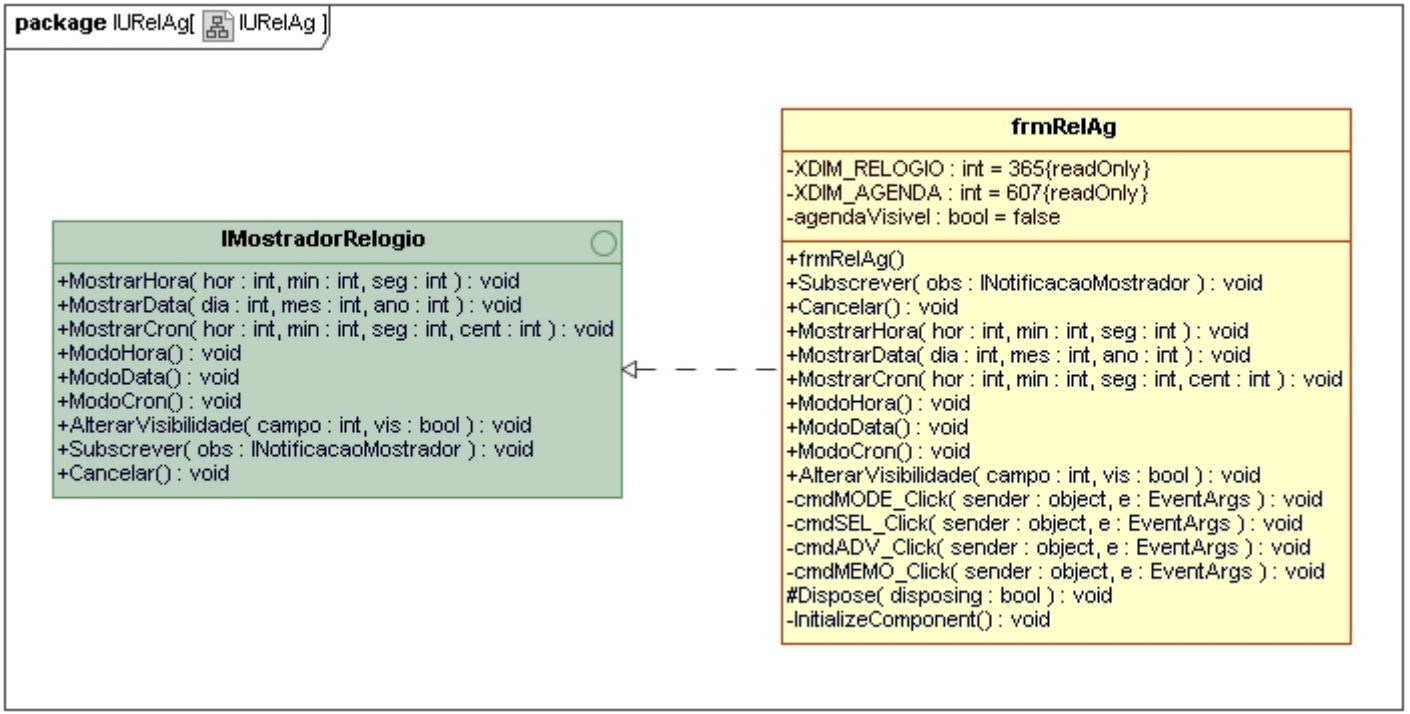
Projecto de Detalhado



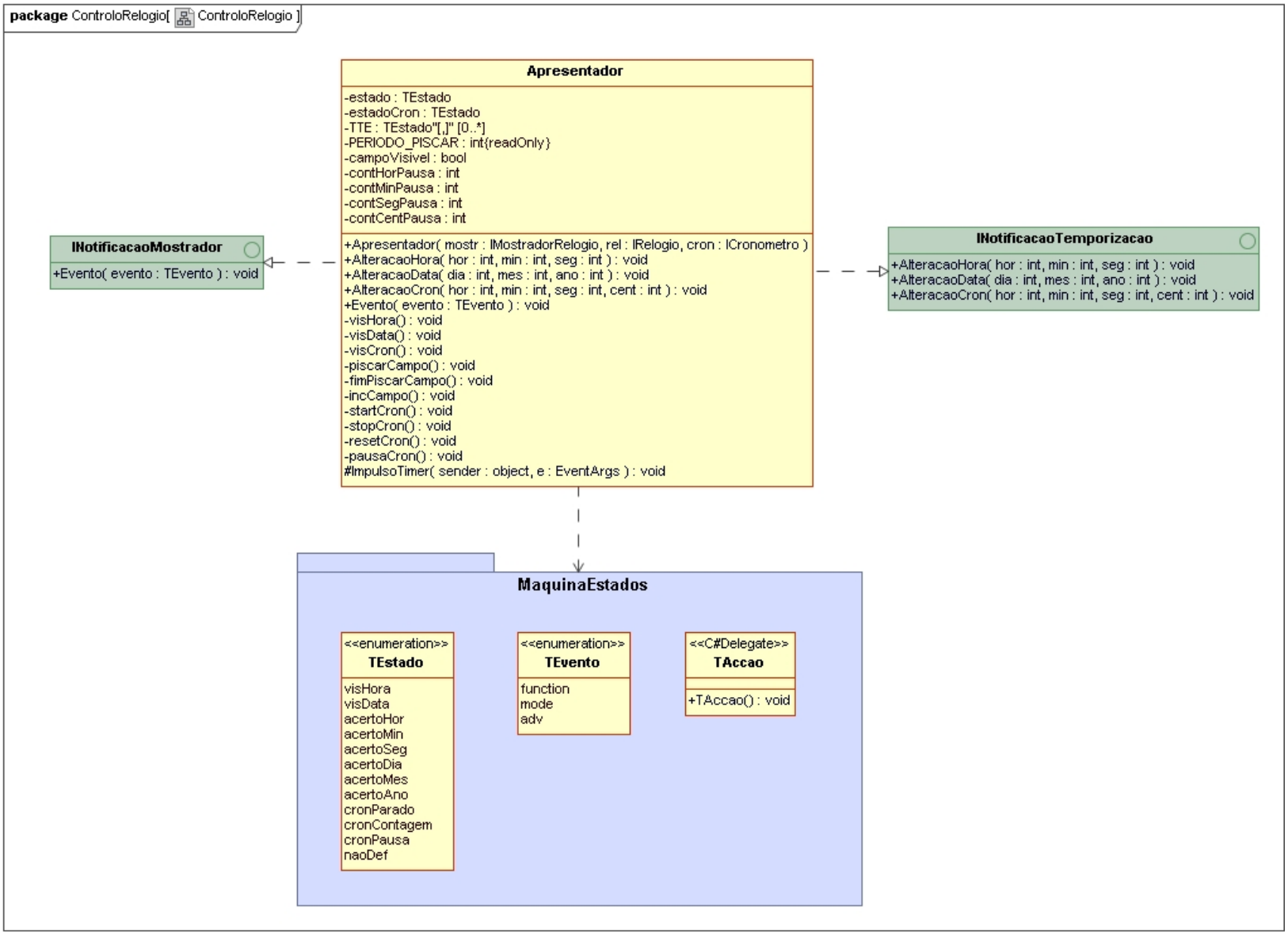
Arquitectura Detalhada



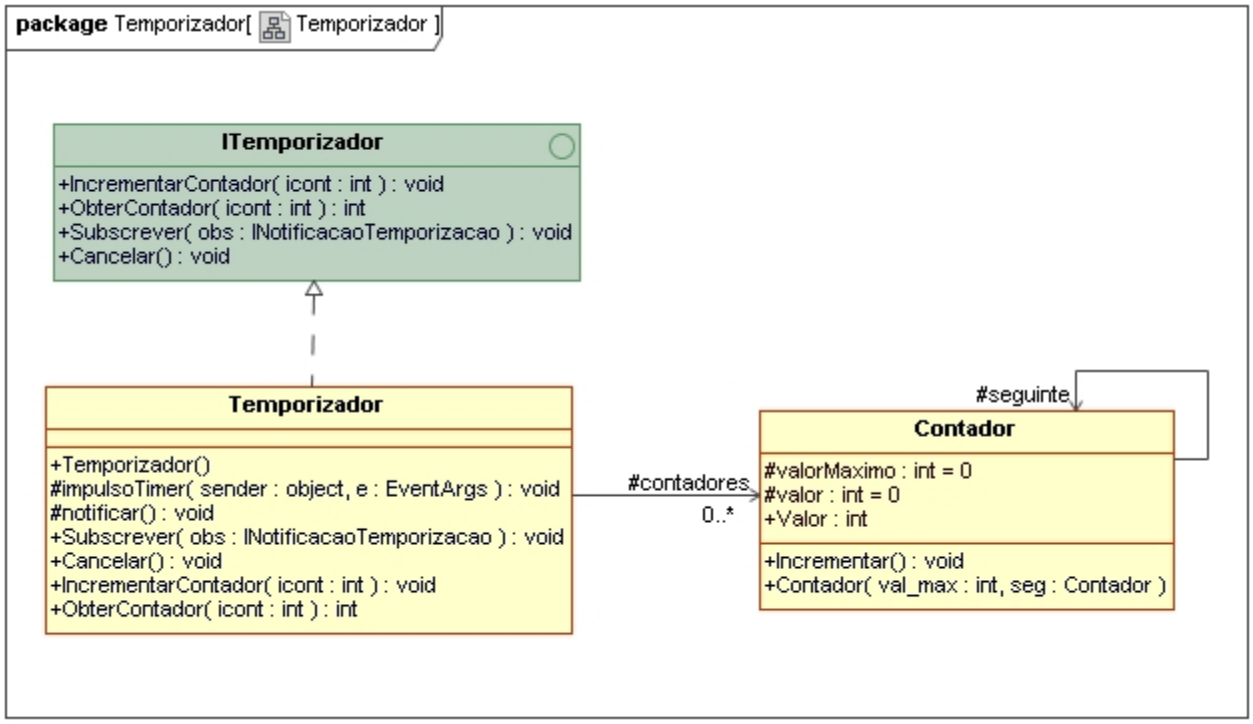
Arquitectura Detalhada



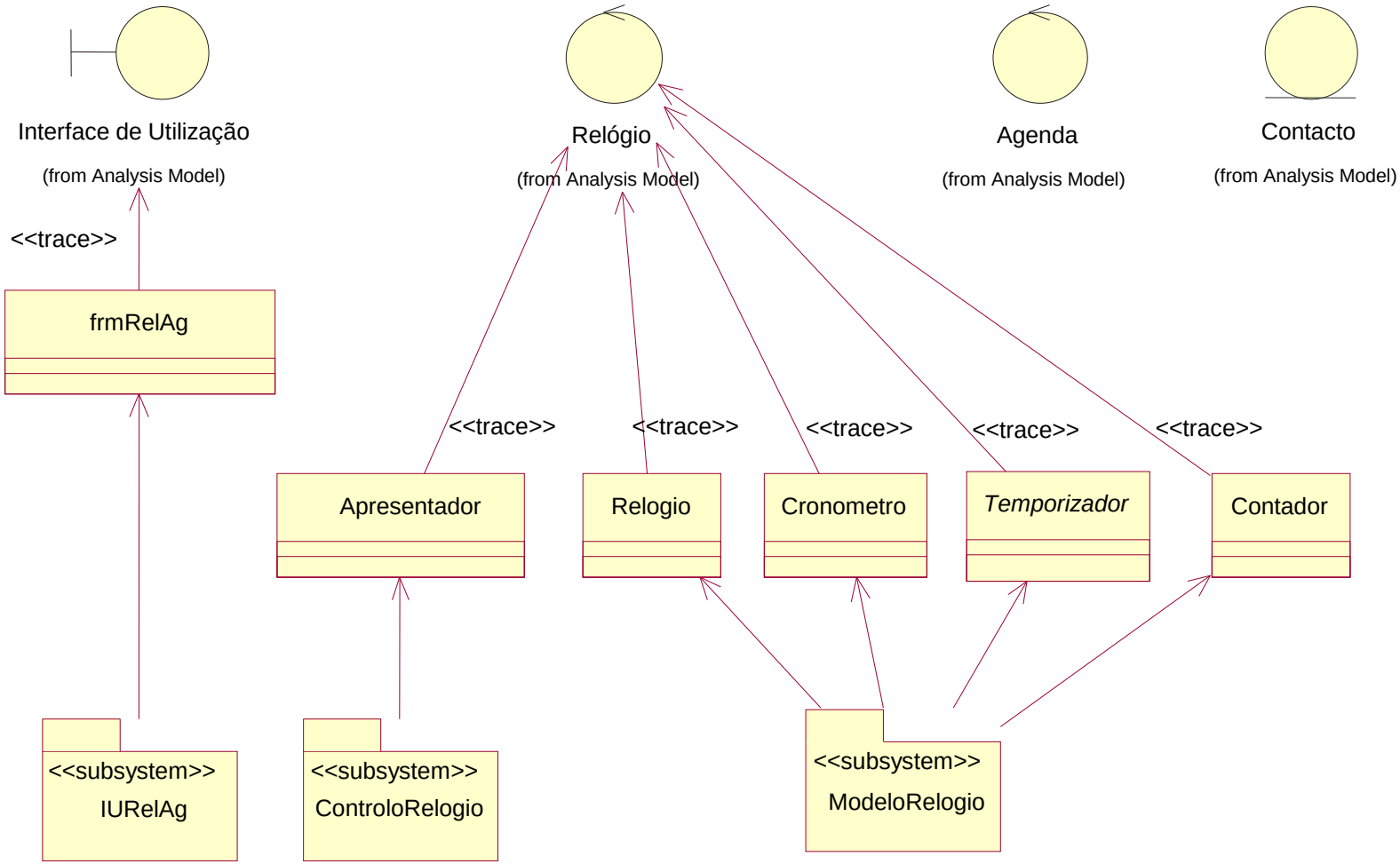
Arquitectura Detalhada



Arquitectura Detalhada

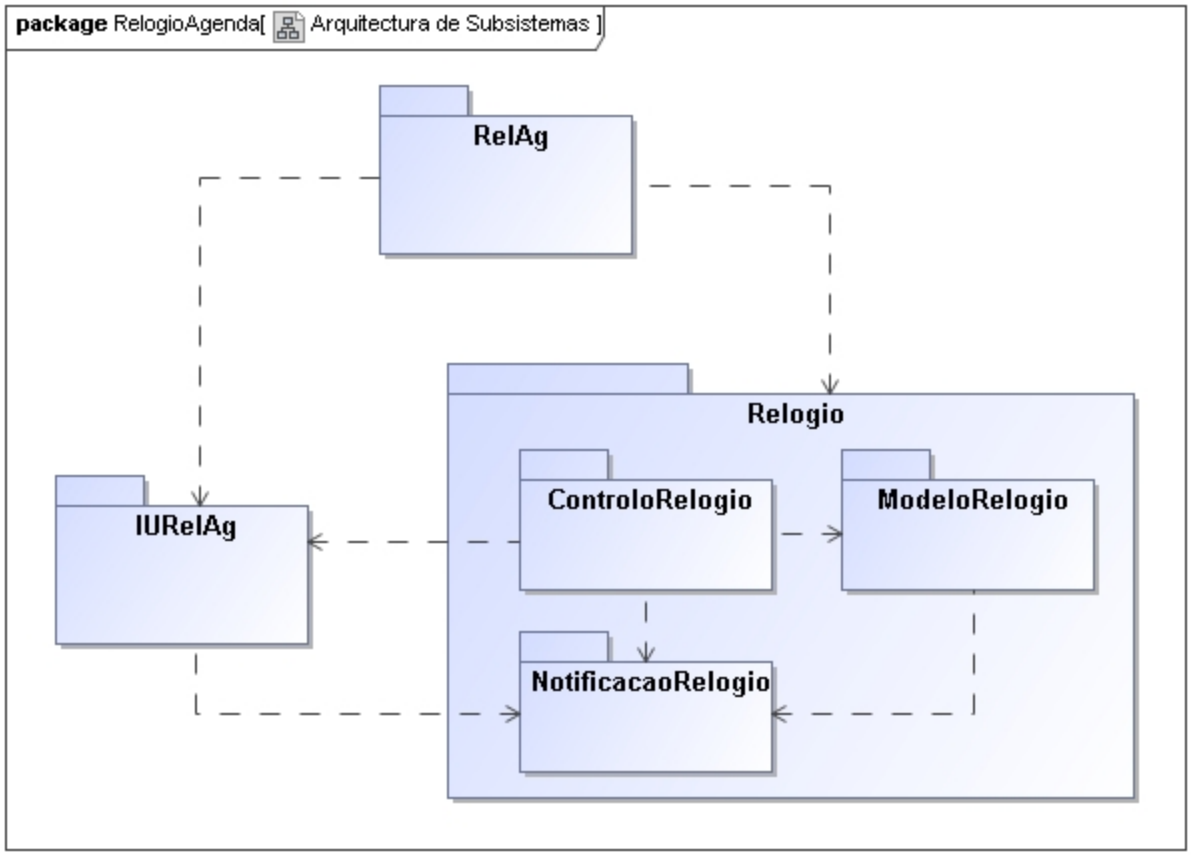


Projecto de Subsistemas

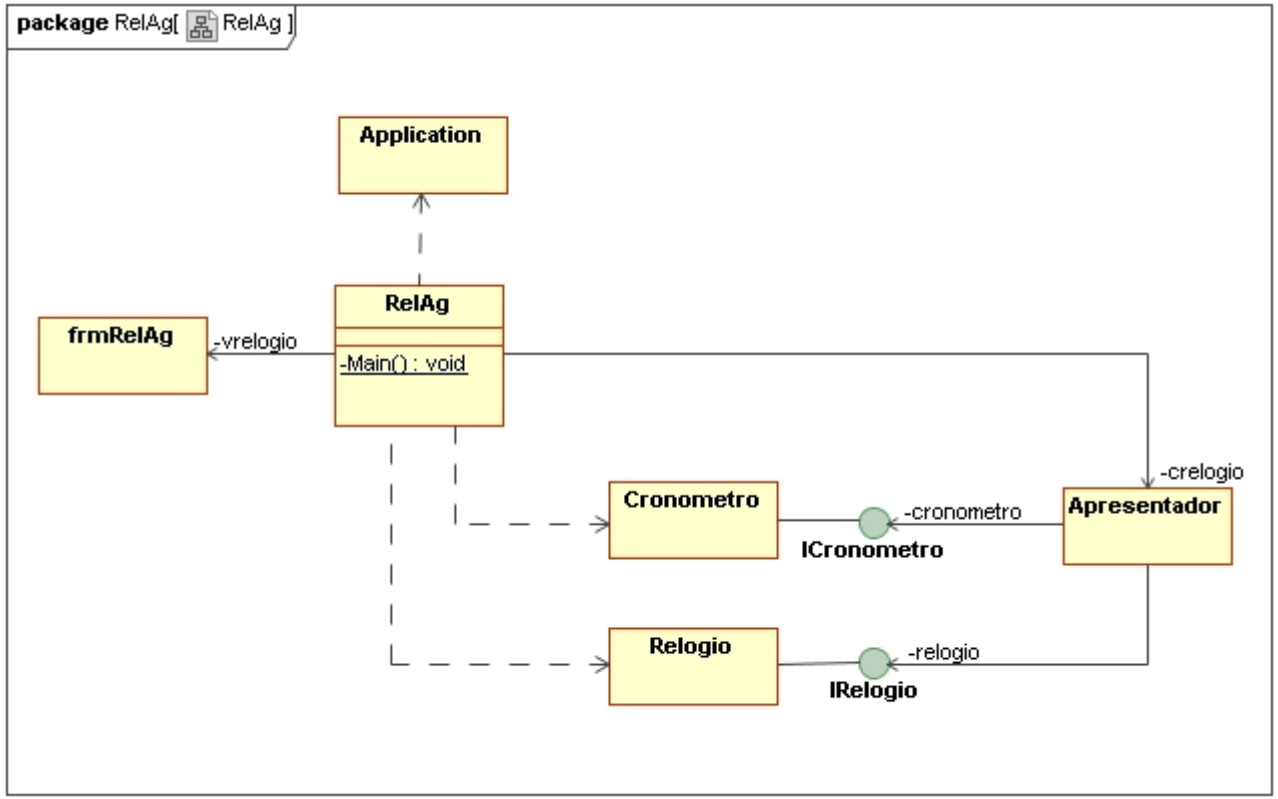


Mapeamento Análise – Projecto (refinamento)

Arquitectura de Subsistemas

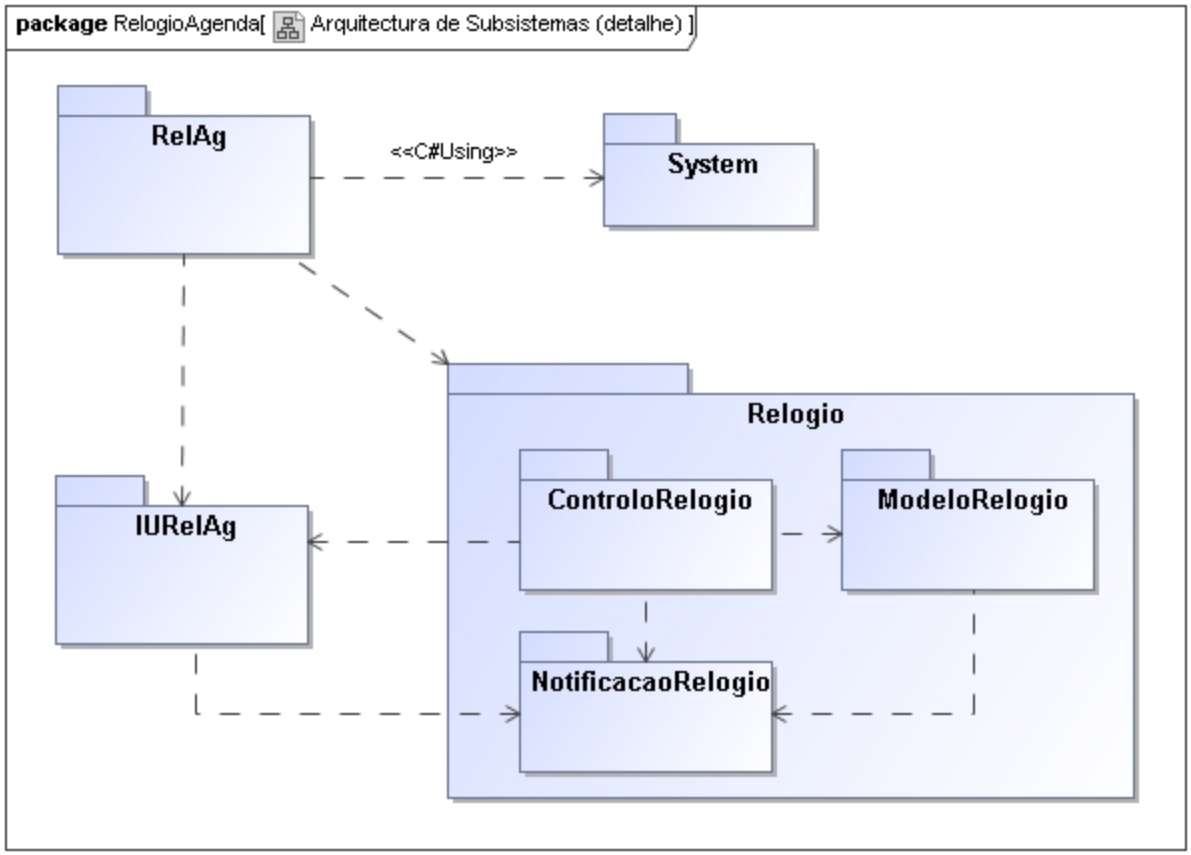


Detalhe de Subsistemas

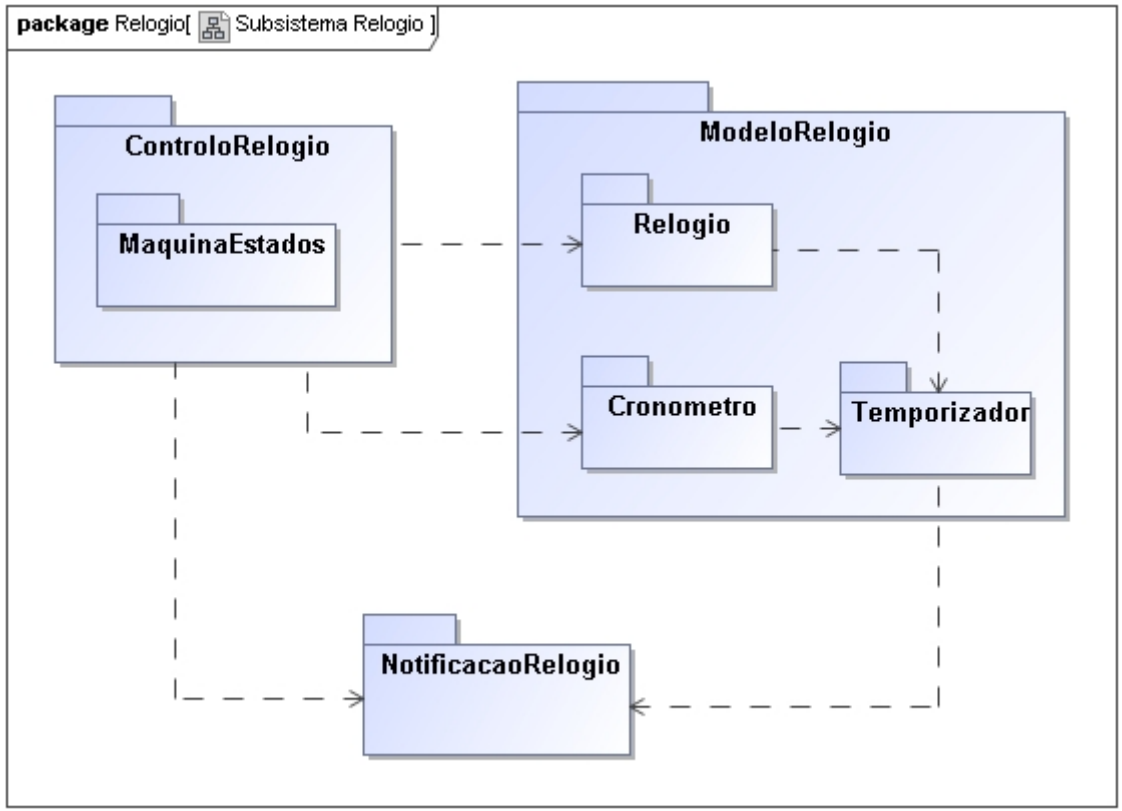


Configuração da aplicação com injecção de dependências

Arquitectura de Subsistemas



Arquitectura de Subsistemas



Iteração de Construção 1

- Objectivo:
  - Implementar os vários subsistemas
  - Obter uma versão funcional do sistema
- Actividades:
  - Projecto detalhado
  - Programação
  - Testes de integração e testes de sistema
  - Sistema funcional (versão preliminar)

Implementação

Código estrutural
Compilação sem erros

Código comportamental
Realização e teste
de comportamento

```
/// <summary>
/// Temporizador: representa um encadeamento de contadores
/// </summary>
class CTemporizador
{
    protected INotificacaoTemporizacao observador;
    protected CContador[] contadores;

    protected Timer timer;

    public CTemporizador()
    {
    }

    protected void impulsoTimer(object sender, EventArgs e)
    {
    }

    protected virtual void notificar()
    {
    }

    /// <summary>
    /// Implementação da interface ITemporizador
    /// </summary>

    public void Subscrever(INotificacaoTemporizacao obs)
    {
    }

    public void Cancelar()
    {
    }

    public void IncrementarContador(int icont)
    {
    }

    public int ObterContador(int icont)
    {
    }
}
```

Implementação de Mecanismos

```
/// <summary>
/// Apresentador do relógio:
/// realiza a gestão e coordenação dos serviços de relógio e cronómetro
/// </summary>
partial class Apresentador : INotificacaoMostrador, INotificacaoTemporizacao
{
    // Estado (estado inicial: visualizar hora com cronómetro parado)
    private static TEstado estado = TEstado.visHora;
    private static TEstado estadoCron = TEstado.cronParado;

    // Tabela de Transição de Estados:
    // definição da dinâmica do controlador
    private TEstado[,] TTE = new TEstado[11, 3]
        {   // FUNCTION              MODE                 ADV
            {TEstado.acertoHor,     TEstado.visData,     TEstado.naoDef},
            {TEstado.acertoDia,     TEstado.cronParado,  TEstado.naoDef},
            {TEstado.visHora,       TEstado.acertoMin,   TEstado.acertoHor},
            {TEstado.visHora,       TEstado.acertoSeg,   TEstado.acertoMin},
            {TEstado.visHora,       TEstado.acertoHor,   TEstado.acertoSeg},
            {TEstado.visData,       TEstado.acertoMes,   TEstado.acertoDia},
            {TEstado.visData,       TEstado.acertoAno,   TEstado.acertoMes},
            {TEstado.visData,       TEstado.acertoDia,   TEstado.acertoAno},
            {TEstado.cronParado,    TEstado.visHora,     TEstado.cronContagem},
            {TEstado.cronPausa,     TEstado.visHora,     TEstado.cronParado},
            {TEstado.cronContagem,  TEstado.visHora,     TEstado.naoDef}
        };
```

Implementação de Mecanismos

```
/// <summary>
/// Processar eventos do mostrador do relógio:
/// activação de comportamento e transição de estado de acordo com as tabelas
/// de activação de comportamento (TAE) e de transição de estado (TTE)
/// <summary>
public void Evento(TEvento evento)
{
    // Activação de comportamento
    TAccao accao = TAE[(int)estado, (int)evento];
    if (accao != null) accao();

    // Transição de estado
    TEstado novoEstado = TTE[(int)estado, (int)evento];

    // Manter histórico do estado do cronometro
    if (novoEstado == TEstado.visHora && evento == TEvento.mode) estadoCron = estado;
    if (novoEstado == TEstado.cronParado && evento == TEvento.mode) novoEstado = estadoCron;

    // Actualizar estado
    if (novoEstado != TEstado.naoDef) estado = novoEstado;
}
```

Implementação de Mecanismos

```
/// <summary>
/// Acções de estado/evento
/// <summary>

private static void visHora()
{
    mostrador.ModoHora();
    mostrador.MostrarHora(relogio.ObterContador(2),
                          relogio.ObterContador(1),
                          relogio.ObterContador(0));
}

private static void visData()
{
    mostrador.ModoData();
    mostrador.MostrarData(relogio.ObterContador(3),
                          relogio.ObterContador(4),
                          relogio.ObterContador(5));
}

private static void visCron()
{
    mostrador.ModoCron();
    mostrador.MostrarCron(cronometro.ObterContador(3),
                          cronometro.ObterContador(2),
                          cronometro.ObterContador(1),
                          cronometro.ObterContador(0));
}

private static void piscarCampo()
{
    timerAcerto.Start();
}
```

Implementação de Mecanismos

```
private static void fimPiscarCampo()
{
    timerAcerto.Stop();

    mostrador.AlterarVisibilidade(0, true);
    mostrador.AlterarVisibilidade(1, true);
    mostrador.AlterarVisibilidade(2, true);
}

private static void incCampo()
{
    switch (estado)
    {
        case TEstado.acertoHor: relogio.IncrementarContador(2); break;
        case TEstado.acertoMin: relogio.IncrementarContador(1); break;
        case TEstado.acertoSeg: relogio.IncrementarContador(0); break;
        case TEstado.acertoDia: relogio.IncrementarContador(3); break;
        case TEstado.acertoMes: relogio.IncrementarContador(4); break;
        case TEstado.acertoAno: relogio.IncrementarContador(5); break;
    }
}

private static void startCron()
{
    cronometro.Start();
}

private static void stopCron()
{
    cronometro.Stop();
}
```

Sistema Versão 1

Modo relógio

Modo agenda

Testes de aceitação