Desenvolvendo aplicações orientadas a objetos no E3 (Parte I)

Alexandre Balestrin Corrêa
abc@elipse.com.br

RT 003.03 - Criado: 06.06.2003 - Atualizado: 09.06.2003
Palavras-chave: E3, objetos, aplicação, projeto, desenvolvimento

Introdução

Com o surgimento do E3 e sua nova maneira de desenvolver aplicações, muitos projetos que antes se tornavam demorados por sua complexidade e extensão, agora podem ser desenvolvidos muito mais rapidamente. Utilizando os recursos de orientação a objetos e as novas bibliotecas disponíveis no E3, é possível criar projetos claros e objetivos e reduzir drasticamente o tempo de manutenção.

Neste artigo, veremos algumas técnicas para aproveitar o que há de melhor no novo paradigma de programação do E3.

1 Definindo a arquitetura do projeto

Primeiramente o desenvolvedor deve se perguntar: “Quais partes da aplicação que serão repetidas?”, “O que eu poderia tornar em objetos parametrizáveis?”.

As respostas mais comuns a estas perguntas são:

- Objetos de tela como bombas, válvulas, elevadores, silos etc;
- Telas, só mudando a sua parametrização (tags associados aos objetos);
- Objetos de dados, como alarmes, tags, históricos etc.

Mas mesmo criando estes objetos você pode estar se perguntando: “Quais são as vantagens deste trabalho?”. Vejamos:

- Tudo em uma só fonte (altera-se na definição e onde ele é utilizado é modificado automaticamente);
- Objetos formados por outros objetos;
- Criação de propriedades;

- Reutilização dos objetos em outros projetos.

Você irá notar que conforme for aumentando a sua experiência de utilização das bibliotecas do E3, cada vez mais o tempo de engenharia será voltado para o aprimoramentos dos objetos e não mais na aplicação em si, pois modificando a biblioteca automaticamente a aplicação é atualizada também.

Porém, antes de darmos ênfase para a criação de objetos, temos que revisar como são utilizadas as associações de propriedades.

2 Conceitos básicos para o desenvolvimento de objetos

Para o desenvolvimento de objetos, primeiramente temos que entender os conceitos básicos sobre associações, recursos de animação e do modo em que são utilizados dentro do E3. Para isto, utilizaremos problemas como exemplos de como fazer e de como não fazer.

Problema 1: Desejo fazer com que um texto apareça com as palavra "Ligado" quando o tag associado for 1 e "Desligado" quando ele for 0. Adicionalmente as palavras devem aparecer com as cores verde e vermelho, respectivamente.

Como **NÃO FAZER** isso: criar dois objetos texto, um deles com a cor do texto em verde e outro em vermelho, um sobre o outro, ambos tem uma associação da propriedade **Visible** com o mesmo tag, porém com condições inversas, como segue:

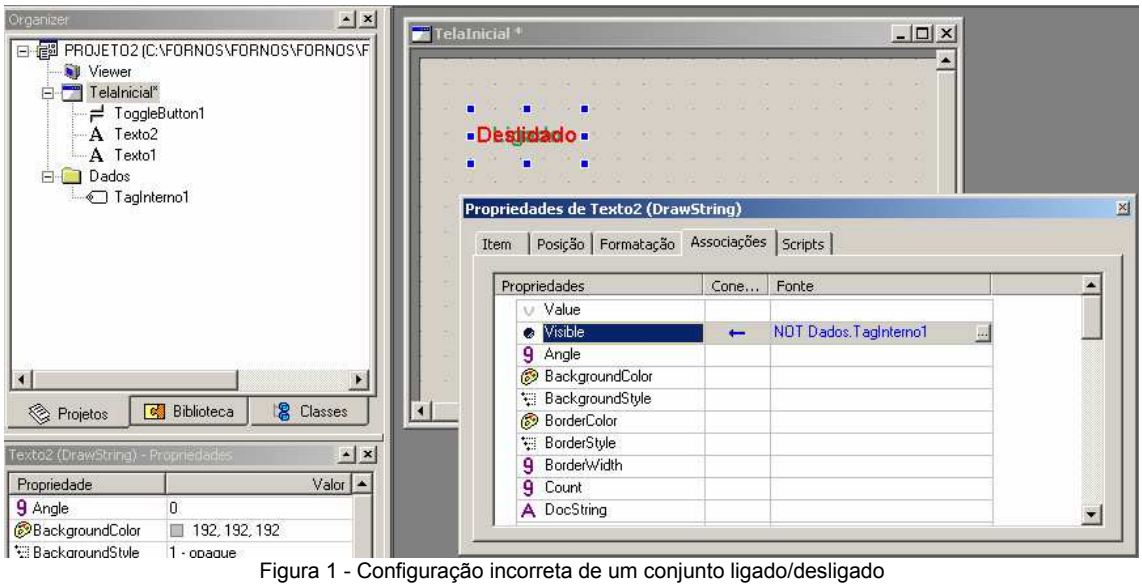

Figura 1 - Configuração incorreta de um conjunto ligado/desligado

Para um conjunto apenas de texto ligado/desligado isto pode parecer simples, mas imagine se tivermos algo do tipo:

Figura 2 - Configuração incorreta de vários conjuntos ligado/desligado

Com apenas 6 conjuntos já temos nada menos que 12 objetos e 12 associações. Veja que a edição do aplicativo começa a ficar complicada, pois temos objetos sobrepostos. Se por exemplo o nome do tag associado em um conjunto mudar, teremos que atualizá-lo nas suas duas associações.

Como **FAZER** isso: criar apenas um objeto texto, com duas associações:

- Associação digital da propriedade **Value** com o tag, sendo que esta setará o valor "Ligado" quando o tag estiver com o valor 1 e "Desligado" quando ele estiver com 0.
- Associação digital da propriedade **TextColor** com o tag, sendo que ela setará verde quando o tag estiver com o valor 1 e vermelho quando ele estiver com 0.

Vejamos a seguir como ficou a configuração da associação digital da propriedade **Value**:

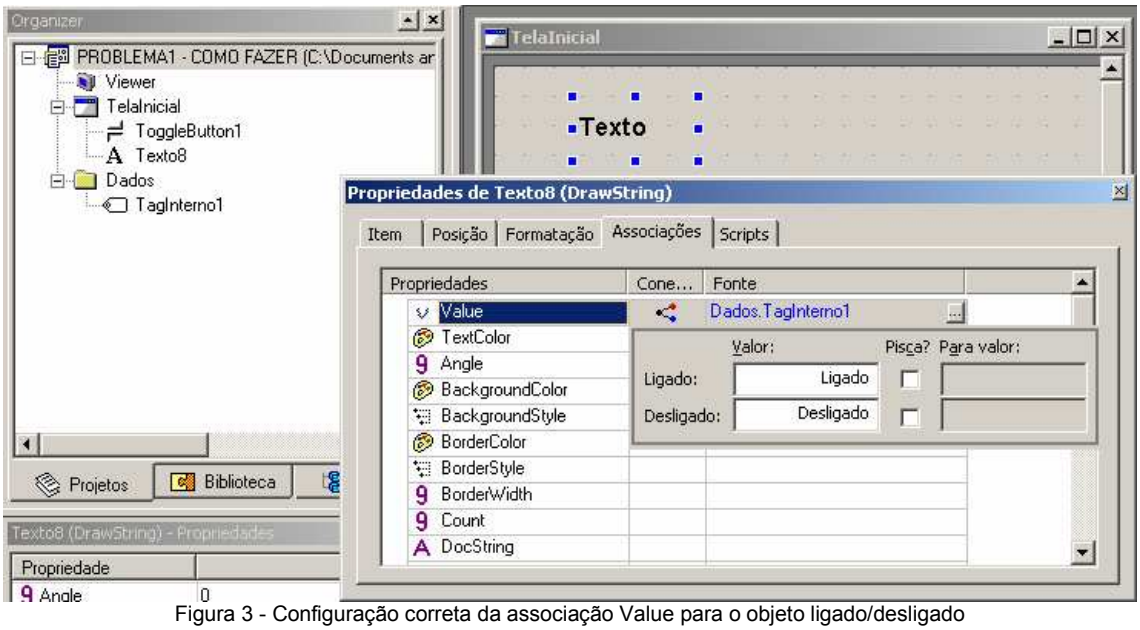

Figura 3 - Configuração correta da associação Value para o objeto ligado/desligado

E também da associação digital da propriedade **TextColor**:

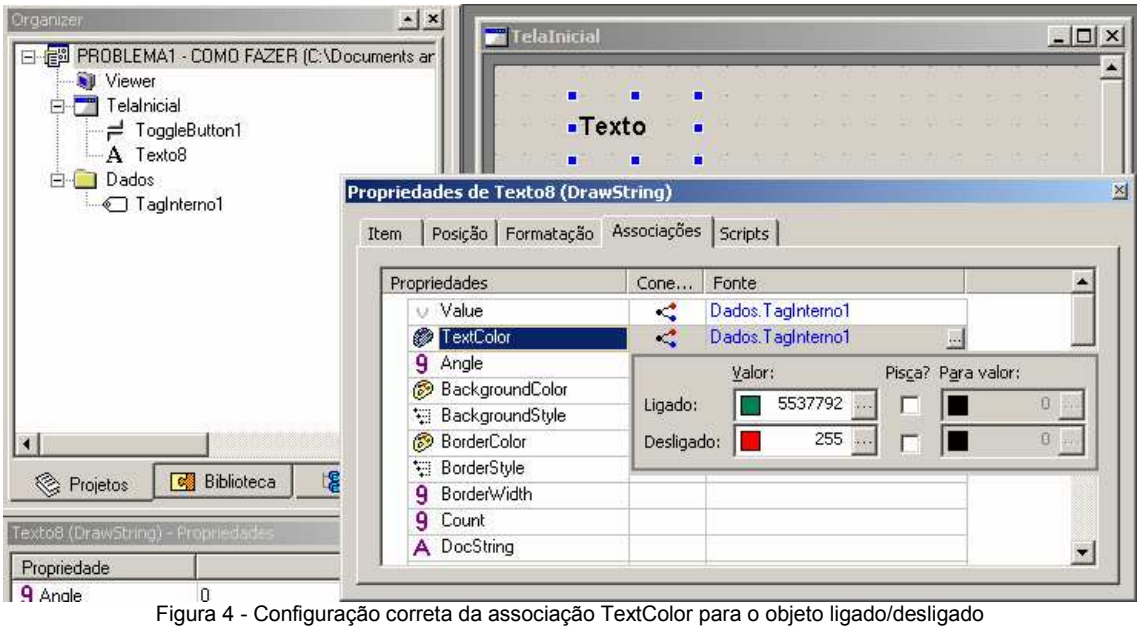

Figura 4 - Configuração correta da associação TextColor para o objeto ligado/desligado

Como no exemplo anterior, criaremos mais cinco cópias para visualizarmos como fica:

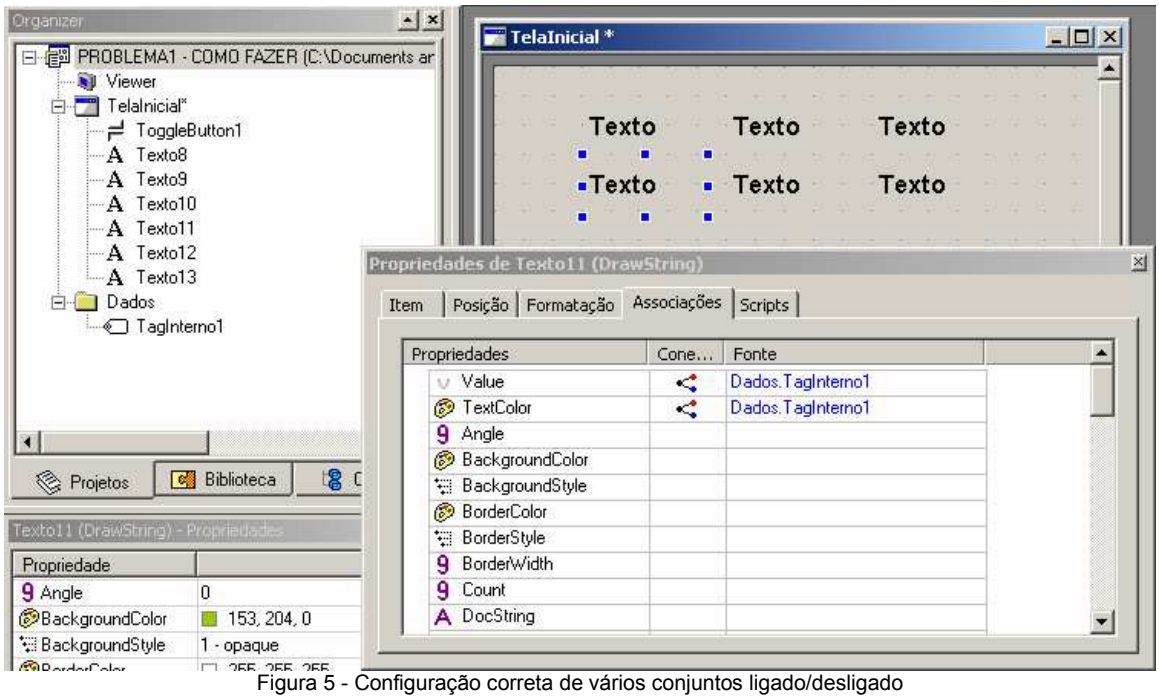

Figura 5 - Configuração correta de vários conjuntos ligado/desligado

Desta maneira, conseguimos ver melhor os benefícios da associação ter sido feita em apenas um objeto ao invés de dois:

- Agora temos apenas seis objetos na tela;
- A edição foi facilitada pois não existem mais objetos sobrepostos, a configuração da cor fica em um lugar apenas;
- Caso o nome do tag mude, só necessitamos mudar as duas associações em um só objeto.

Ainda assim, o ideal não seria que eu tivesse apenas que fazer modificações em apenas um lugar? Essa questão vai ser resolvida mais adiante com o uso de bibliotecas.

Problema 2: Como posso fazer para que um desenho de um caminhão percorra um caminho pré-estabelecido de acordo com a variação do valor de um tag? O tag vai assumir valores digitais de 0 até 3.

Como **NÃO FAZER** isso: criar quatro caminhões (objetos **DrawPicture**), e em cada um deles configurar uma conexão simples da propriedade **Visible** ao tag, só variando um pouco a condição verdadeira, como segue:

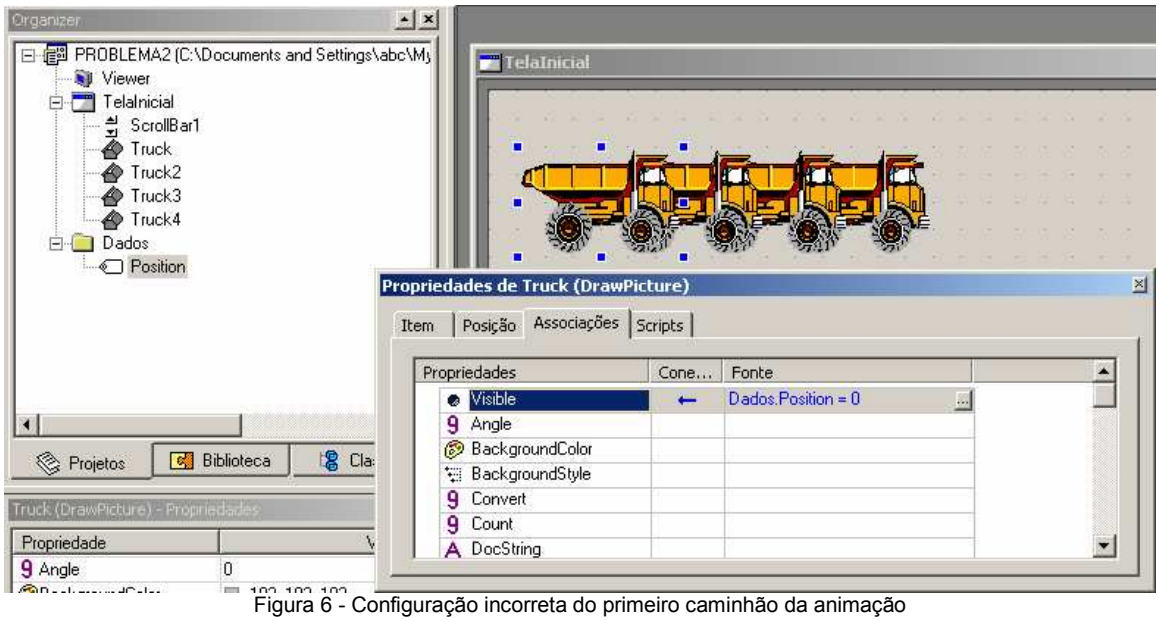

Figura 6 - Configuração incorreta do primeiro caminhão da animação

Figura 7 - Configuração incorreta do último caminhão da animação

Esta abordagem funciona perfeitamente, porém não é otimizada, pois:

- Cria quatro conexões com a fonte (caso a fonte mude, teremos que mudar as quatro conexões);

- Caso o valor do tag possa no futuro ter sua precisão aumentada, isto é, tenha uma maior número de combinações, teremos que colocar mais caminhões;
- Caso tenhamos que mudar a posição inicial do objeto, teremos que mover os outros objetos também.

Poderíamos ainda tentar fazer uma implementação um pouco mais otimizada, como deixar somente um único objeto e criar uma conexão analógica da propriedade **X** do caminhão ao tag, como segue:

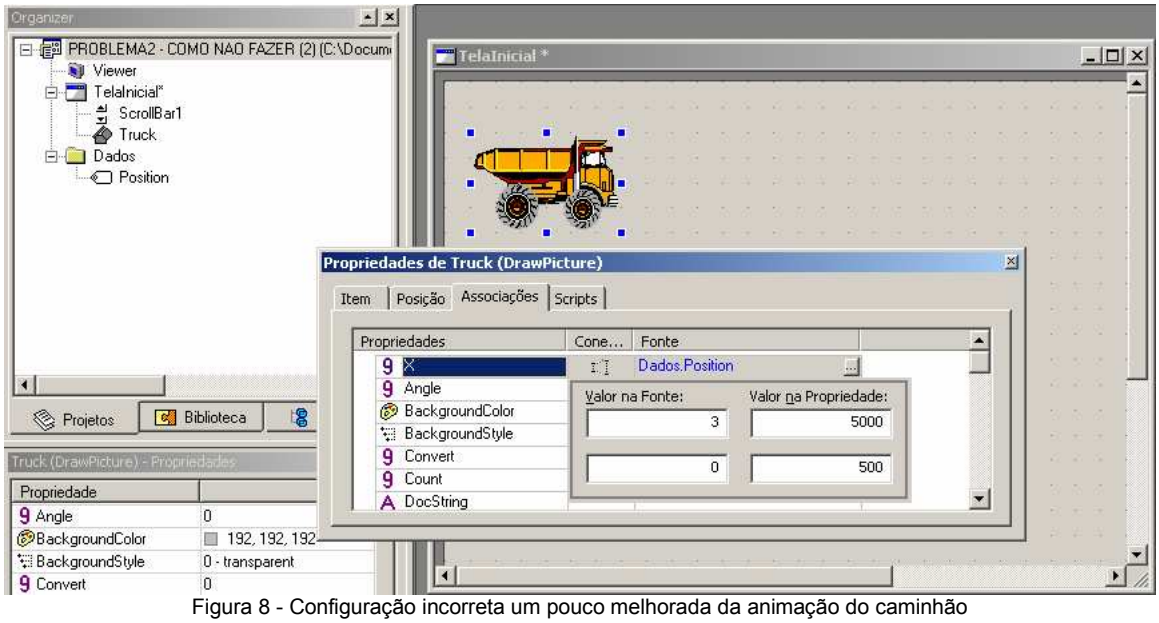

Figura 8 - Configuração incorreta um pouco melhorada da animação do caminhão

Então como configurado acima, o E3 fará um conversão do valor do tag para o valor da propriedade, sendo assim quando o tag variar de 0 a 3, automaticamente valores entre 500 e 5000 serão atribuídos para a propriedade **X** do caminhão, fazendo o movimento.

Mas ainda assim temos um problema nesta solução: como saber especificar com exatidão onde terminará a animação? A resposta vem a seguir.

Como **FAZER** isso: selecionamos a imagem e clicamos no botão "Animar com Translação". O E3Studio criará uma imagem "fantasma" do desenho indicando sua posição final. Cabe salientar que o objeto "caminhão" no Organizer fica dentro de um novo objeto chamado **DynamicMove**. Utilizamos o quadrado verde que fica no centro do objeto para posicioná-lo onde desejado.

Configure as propriedade **RangeMax** e **RangeMin** do **DynamicMove** para "3" e "0" respectivamente e crie uma conexão simples da propriedade **Value** para o tag como mostrado a seguir:

Figura 9 - Configuração correta da animação do caminhão

Isto fará com que automaticamente o objeto se desloque de um lugar para outro, trazendo as seguintes vantagens:

- A imagem "fantasma" indica a posição final do caminhão de uma maneira intuitiva e clara;
- Se for necessário mover a posicão inicial do objeto, a posição final é deslocada automaticamente;
- É possível testar a movimentação alterando-se o valor da propriedade **Value** do **DynamicMove** durante a edição no E3Studio.

Problema 3: Como fazer uma animação do estado de uma bomba através da mudança da sua cor? Por exemplo, digamos que tenhamos três estados: "ligada", "desligada" e "em manutenção", com as cores verde, vermelha e cinza respectivamente.

Como **NÃO FAZER** isso: criando três bitmaps com as três cores e fazendo uma conexão simples da propriedade **Visible** com o tag (parecido com o feito no problema 1), cada bitmap tem uma cor configurada como cor transparente, como segue:

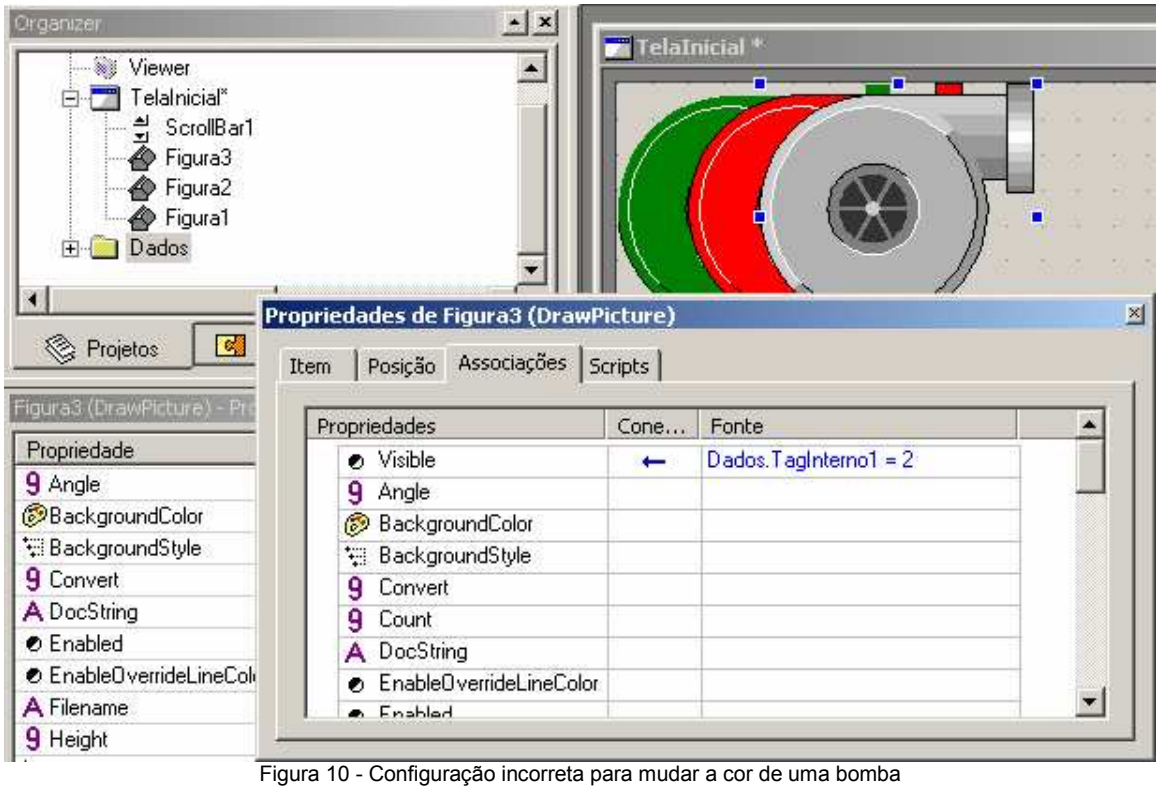

Figura 10 - Configuração incorreta para mudar a cor de uma bomba

Neste exemplo colocamos as bombas lado a lado apenas por clareza, pois eles estariam que estar sobrepostos. Esta abordagem contém os seguintes problemas:

- Temos três bitmaps e se tivermos que modificar seu desenho, teremos que efetuar as modificações nas três versões, por exemplo: se quisermos mudar de verde para azul o estado de ligado, teremos que modificar o bitmap fonte;
- Temos três conexões para o mesmo tag, se o nome do tag for modificado, teremos que alterá-lo nas três conexões;
- A edição também fica complicada pois temos três objetos sobrepostos.

Como **FAZER** isso: no E3 deve-se sempre dar prioridade a arquivos com o formato WMF (Windows MetaFile), pois é possível modificar suas características de desenho como cor e tipo do preenchimento **sem alterar o arquivo fonte**. Então para resolver este problema colocar o WMF na tela e fazer uma conexão por tabela da propriedade **OverrideFillColor** com o tag, como segue:

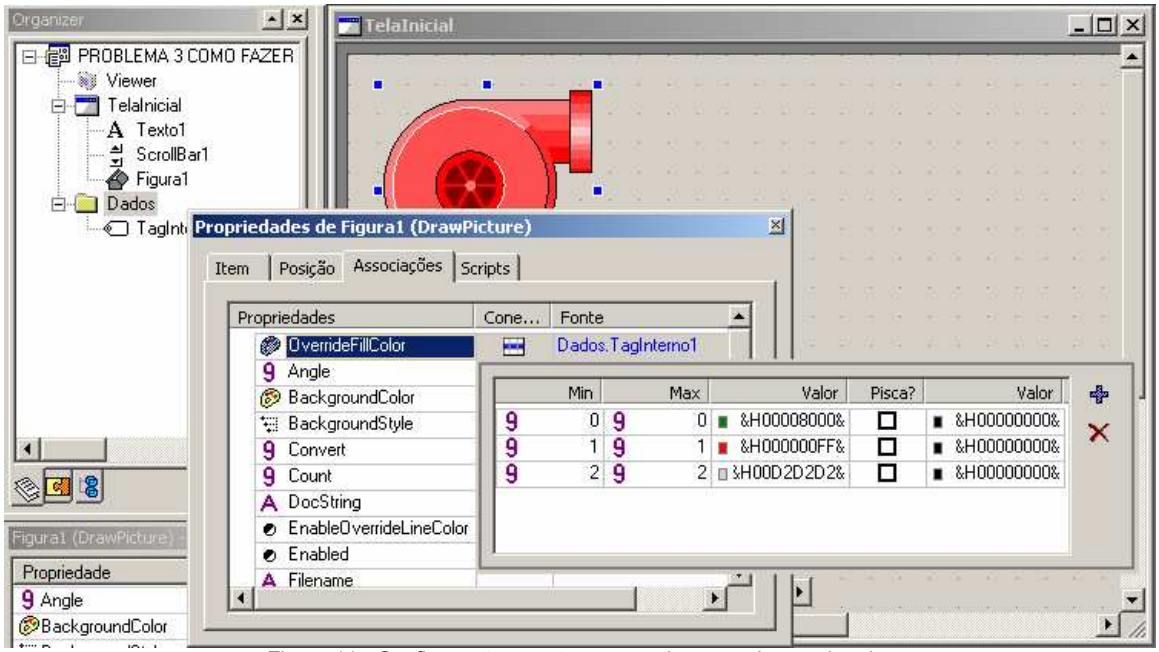

Figura 11 - Configuração correta para mudar a cor de uma bomba

Além dos benefícios deste tipo de associação com os já explicados no problema 1, a utilização de arquivos do formato WMF tem as seguintes vantagens em relação aos bitmaps:

- É possível redefinir o tipo de preenchimento do WMF através da propriedade **OverrideFillMode** para “Wireframe”, “SolidFill” ou “ByBrightness”;
- É possível redefinir a cor do preenchimento do WMF;
- É possível redefinir a cor da linha do WMF;
- É vetorial, portanto não perde resolução mesmo quando for efetuado um zoom.

Como dito anteriormente, todas essas modificações no desenho não modificam o arquivo original, apenas a maneira que o desenho é visualizado.

Você pode estar se perguntando: mas o que acontece se eu necessitar ter vários destes objetos configurados na minha aplicação e necessitar modificá-los depois de algum tempo as suas definições? Terão que serem modificados um a um? Todas essas dúvidas serão esclarecidas na **Parte II** deste documento.

Desenvolvendo aplicações orientadas a objetos no E3 (Parte II)

Alexandre Balestrin Corrêa
abc@elipse.com.br

RT035.06 - Criado: 24/03/2006 – Modificado: 06/04/2006
Palavras-chave: E3, objetos, aplicação, projeto, desenvolvimento

Resumo

Neste artigo discutiremos a maneira como são criados e usados objetos de biblioteca XControl e XObject no E3, com base em uma aplicação simples de dois tanques. Serão também abordados assuntos relativos à criação de propriedades e associações entre objetos e propriedades, assim como encapsulamento e reaproveitamento de objetos.

1 Introdução

Vimos na parte I deste artigo a maneira correta de criar associações para animação dos controles de tela. Nesta parte II, discutiremos a melhor estratégia para o desenvolvimento de uma aplicação para uma planta industrial imaginária.

2 Descrição da planta

A planta deste exemplo consiste em dois tanques, cada um deles com uma válvula que permite a entrada de líquido, e uma bomba de saída. Ao se abrir a válvula através de um clique do mouse, o tanque começa a encher. O tanque esvazia ao clicar-se sobre a bomba de saída.

Cada válvula fica verde quando aberta e apresenta a legenda “Opened”. Caso contrário, ela fica vermelha e apresenta a legenda “Closed”. Já o objeto bomba quando ligado fica verde e com a legenda “On”. Caso contrário, ele fica vermelho e com a legenda “Off”.

A seguir, vemos o sinótico da planta.

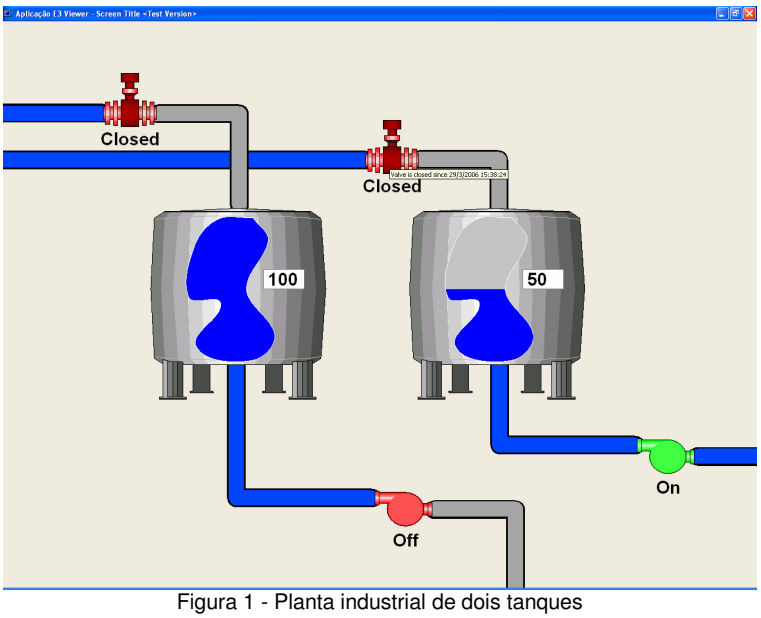

Figura 1 - Planta industrial de dois tanques

3 Desenvolvendo a biblioteca de objetos

O primeiro passo para o desenvolvimento de uma aplicação no E3 é determinar quais funcionalidades do processo serão repetidas e podem ser transformadas em objetos do E3.

Nesta aplicação vemos claramente os objetos de tela que são repetidos: tanques, bombas, válvulas e tubos. Após a identificação dos objetos, é necessário definir cada uma de suas funcionalidades e propriedades, além é claro, do seu desenho.

Também é necessário que seja executada no servidor a lógica de funcionamento dos tanques. Esta lógica está definida no XObject chamado "TankLogic".

Vejamos a seguir como cada um destes objetos será construído.

Criando o XObject TankLogic

Inicialmente criaremos um projeto chamado **Plant** com uma biblioteca chamada **PlantLib**. Na biblioteca PlantLib ficarão as definições dos XObjects e XControls. No projeto Plant ficarão os objetos executados no servidor e as telas do Viewer.

O objeto TankLogic (Figura 2) terá três propriedades para controle do processo de um tanque: *ValveOpened* informa se a válvula de entrada do tanque está aberta ou fechada; *Meter* informa o nível de líquido do tanque; e *PumpActivated* reporta se a bomba de saída do tanque está ligada ou desligada.

Uma recomendação interessante é o preenchimento da coluna "Texto de Ajuda" de cada propriedade adicionada a um ElipseX (Figura 2). O texto de cada propriedade é mostrado no AppBrowser, auxiliando futuramente na manutenção da aplicação.

Para fins didáticos, não vamos entrar em detalhes de implementação deste objeto; é necessário saber apenas que quando a propriedade *ValveOpened* estiver em True o tanque irá começar a encher até que a propriedade *Meter* chegue ao limite de 100 unidades. Ao ser ligada a bomba de saída através da propriedade *PumpActivated* o tanque é esvaziado. Com a bomba ligada e a válvula aberta a propriedade *Meter* continua com valor constante.

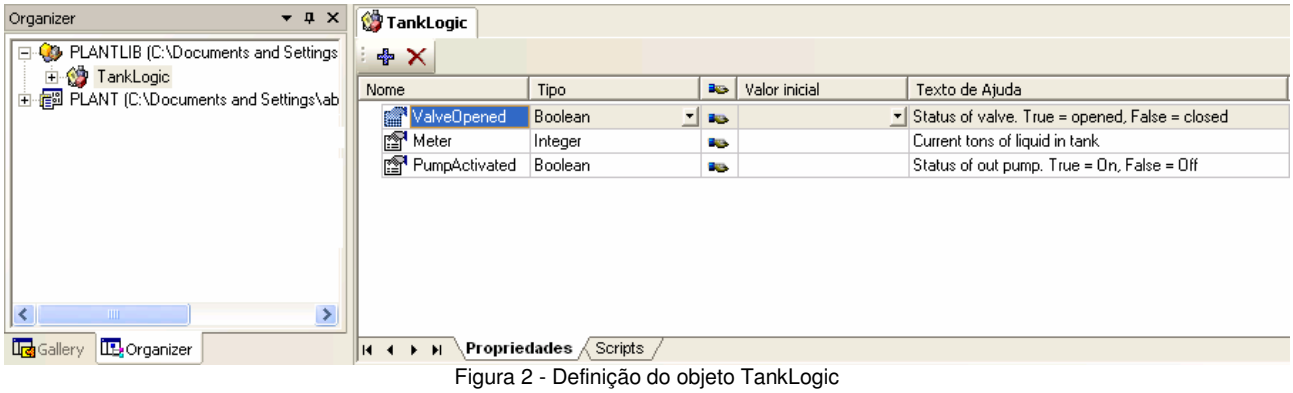

Figura 2 - Definição do objeto TankLogic

Criando o XControl Tank

O objeto Tank é composto do desenho do reservatório mais dois mostradores da quantidade de líquido, um que mostra uma barra vertical analógica e outro com o valor em toneladas presente no tanque (Figura 3).

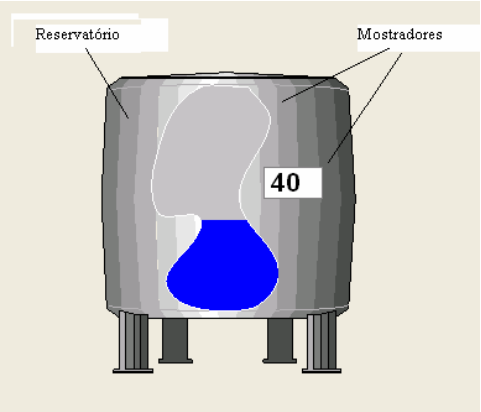

Figura 3 - Objeto Tank

Com o objetivo de demonstrar a flexibilidade das bibliotecas ElipseX, vamos dividir o objeto Tank em duas partes: uma apenas com o desenho do reservatório e outra com o desenho dos mostradores. Com isso, podemos ver que é possível um objeto ElipseX ser construído usando outros objetos previamente criados.

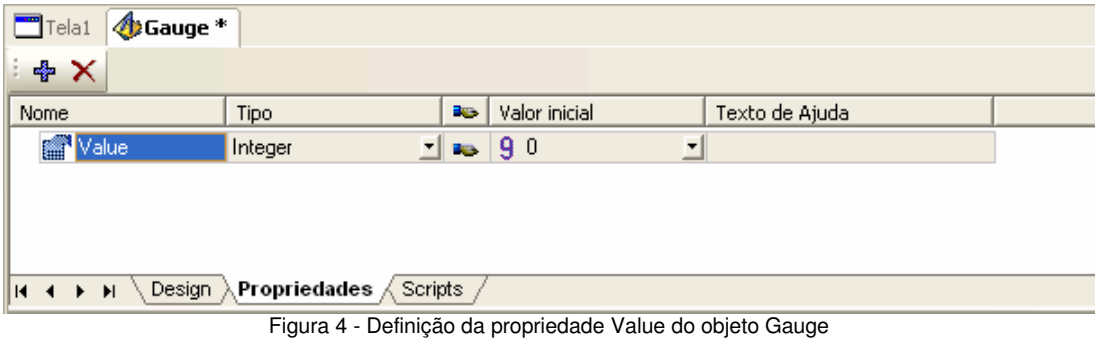

Figura 4 - Definição da propriedade Value do objeto Gauge

O objeto Gauge conterá os mostradores e tem uma propriedade chamada *Value* (ver Figura 4), que será usada para animar os controles internos do objeto. O desenho do objeto Gauge está na Figura 5.

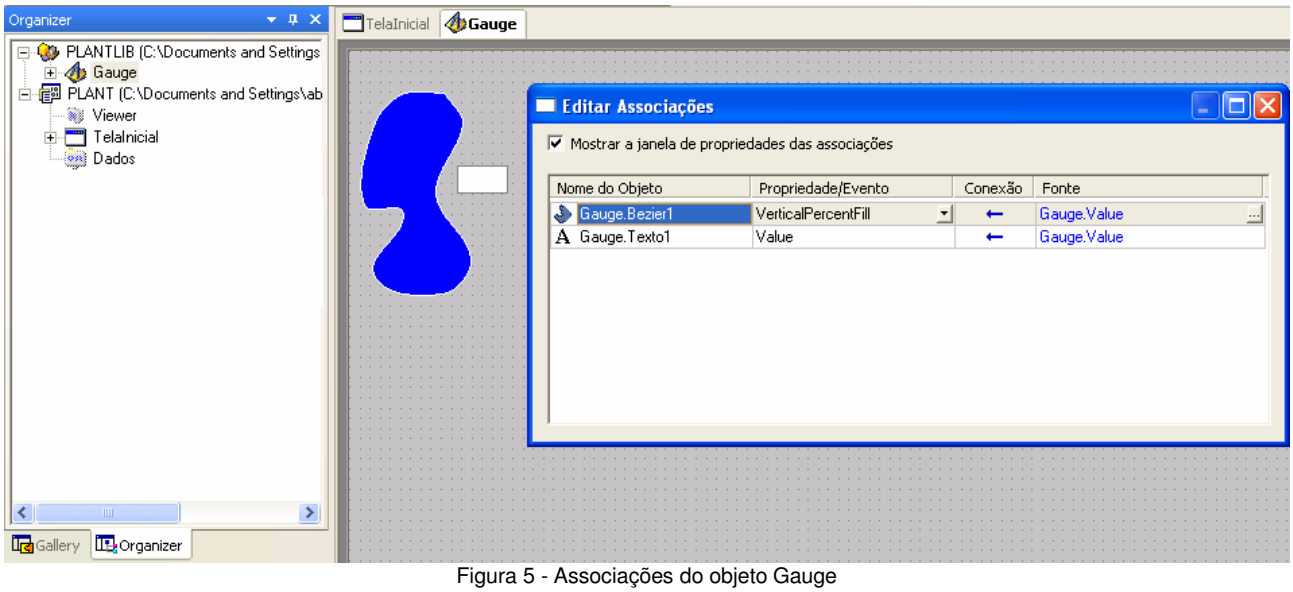

Figura 5 - Associações do objeto Gauge

Observe que os dois objetos que formam o Gauge estão associados à propriedade *Value* do Gauge. Isto é, ao modificar-se a propriedade *Value* do Gauge automaticamente este valor será enviado para os dois objetos associados a esta propriedade.

Depois disso partimos para o objeto Tank em si, que é composto de um desenho de um reservatório mais o objeto Gauge (Figura 6).

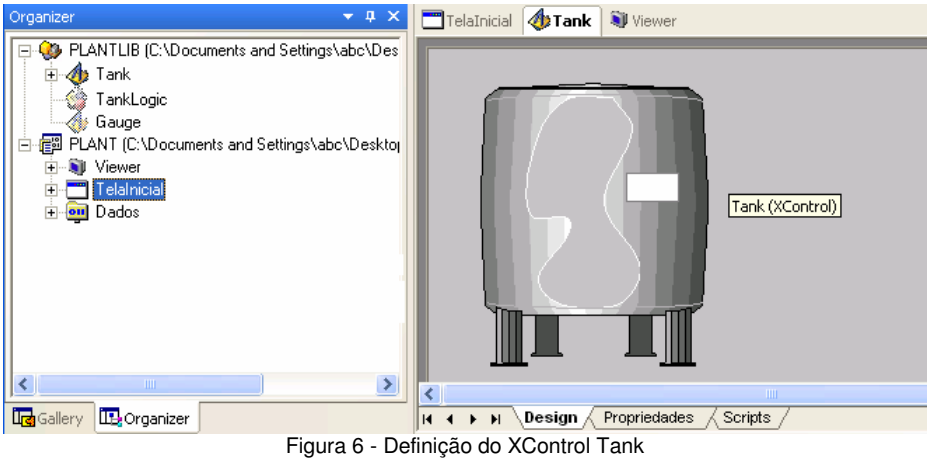

Figura 6 - Definição do XControl Tank

Agora temos que pensar nas propriedades deste objeto. Temos duas abordagens: a primeira seria adicionar uma propriedade do tipo Integer (Inteiro) e associá-la à propriedade *Value* do objeto Gauge1. A segunda opção é criar uma propriedade do tipo TankLogic e associar o *Value* do objeto Gauge1 à propriedade *Meter* do TankLogic.

Para este exemplo, escolheremos a segunda opção (Figura 7).

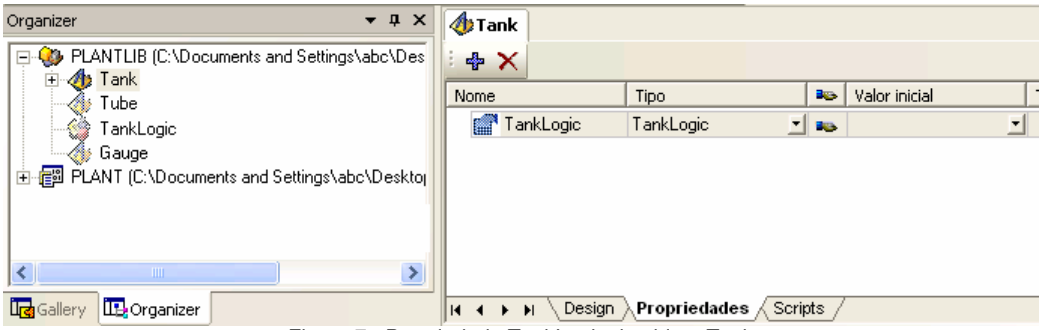

Figura 7 - Propriedade TankLogic do objeto Tank

Agora basta associarmos a propriedade *Value* do objeto Gauge1 à propriedade *Meter* do TankLogic do objeto Tank (Figura 8). Esta é a maneira com que o objeto Gauge1 irá acessar as propriedades do objeto externo associado à propriedade *TankLogic*, que é um objeto do tipo TankLogic.

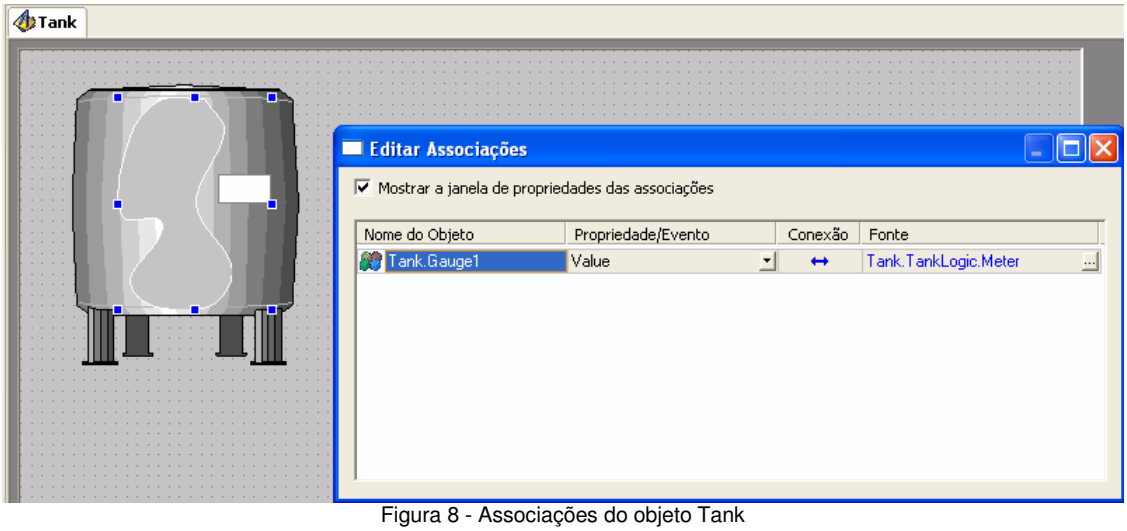

Figura 8 - Associações do objeto Tank

Criando o XControl Valve

O objeto Valve é composto por um desenho de uma válvula que fica vermelha quando fechada e verde quando aberta. Ao clicar no objeto, ele inverte o estado atual da propriedade *ValveOpened* do objeto TankLogic associado, fazendo com que a válvula seja aberta ou fechada.

O primeiro passo para a construção deste objeto é a criação de uma propriedade chamada *TankLogic* do tipo TankLogic, fazendo com que este objeto possa ser associado a um objeto TankLogic do servidor.

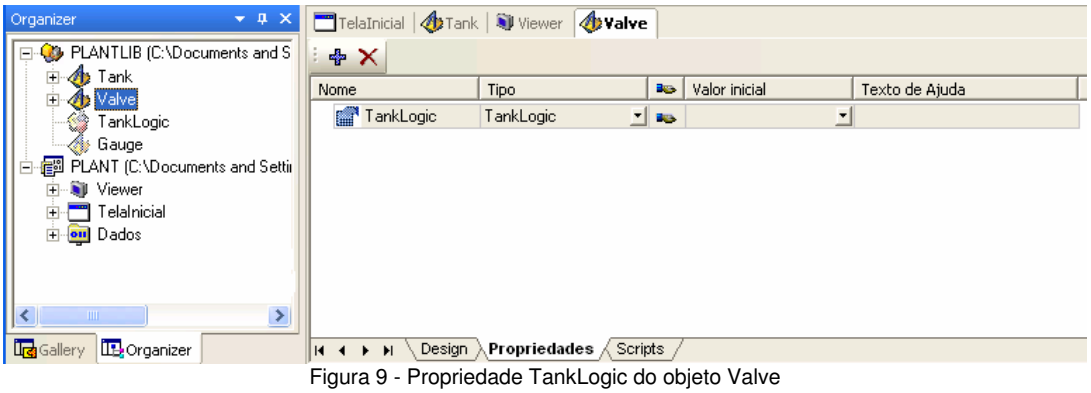

Figura 9 - Propriedade TankLogic do objeto Valve

Após, temos o desenho do objeto Valve e suas associações (Figura 10), onde podemos observar que tanto o objeto Texto1 quanto a Figura1 foram associados à propriedade *ValveOpened* do TankLogic da Valve.

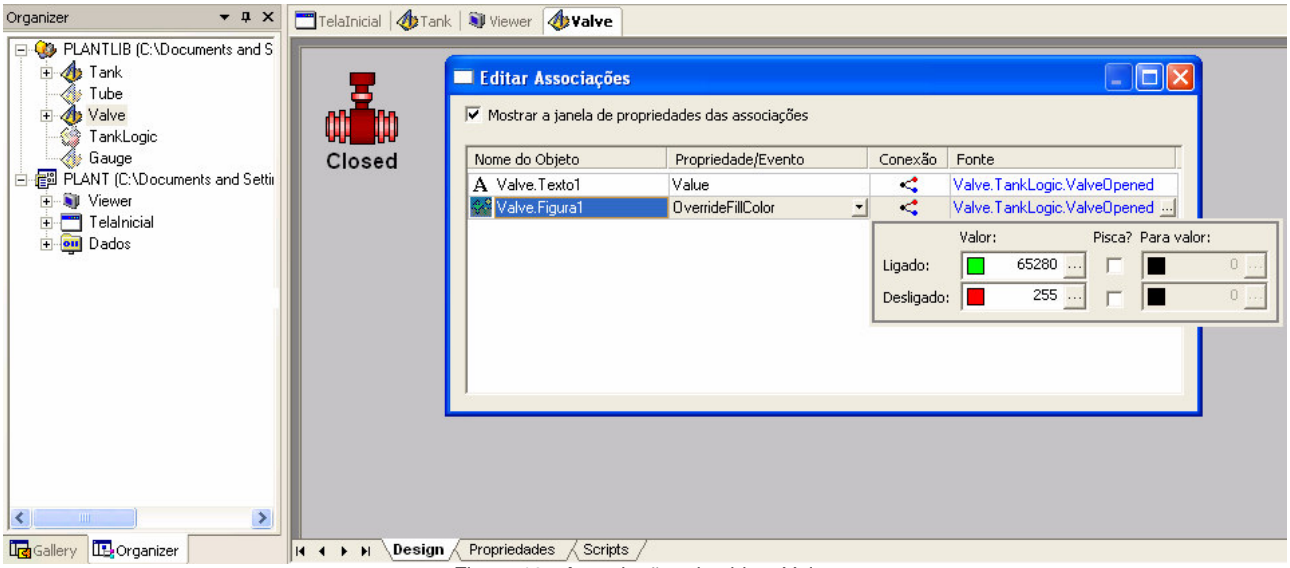

Figura 10 - Associações do objeto Valve

Agora precisamos de um script que faça com que a propriedade *ValveOpened* do TankLogic seja invertida ao clicar na válvula. Isto é feito através de um script *Click()* do objeto Figura1 (como vemos na Figura 11).

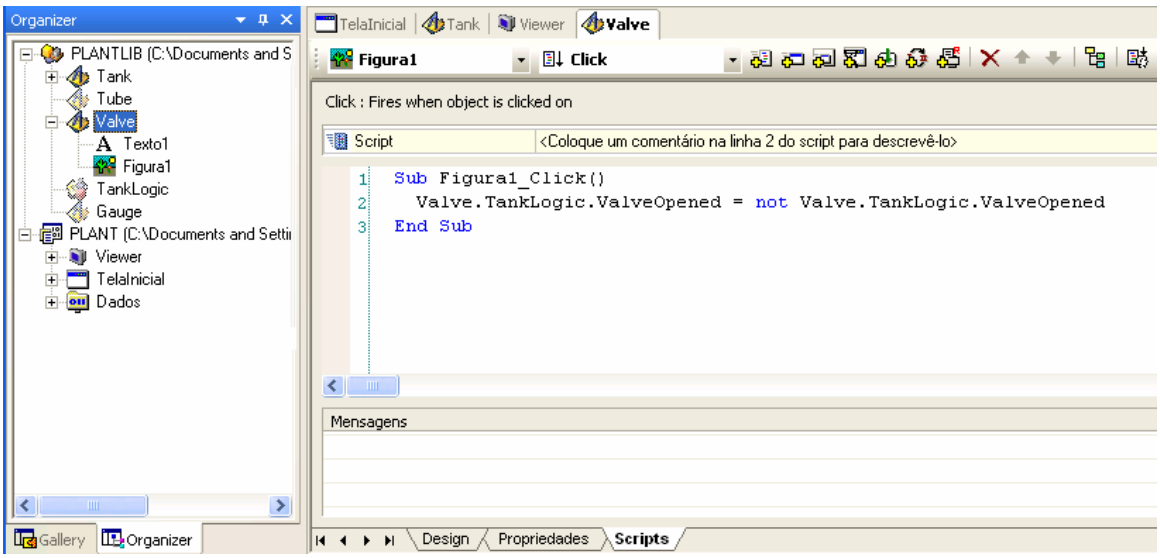

Figura 11 - Script Click do objeto Figura1 do Valve

Criando o XControl Pump

O objeto Pump é praticamente igual ao objeto Valve, com duas diferenças: seu desenho é uma bomba e ele inverte a propriedade *PumpActivated* do objeto TankLogic (Figura 12).

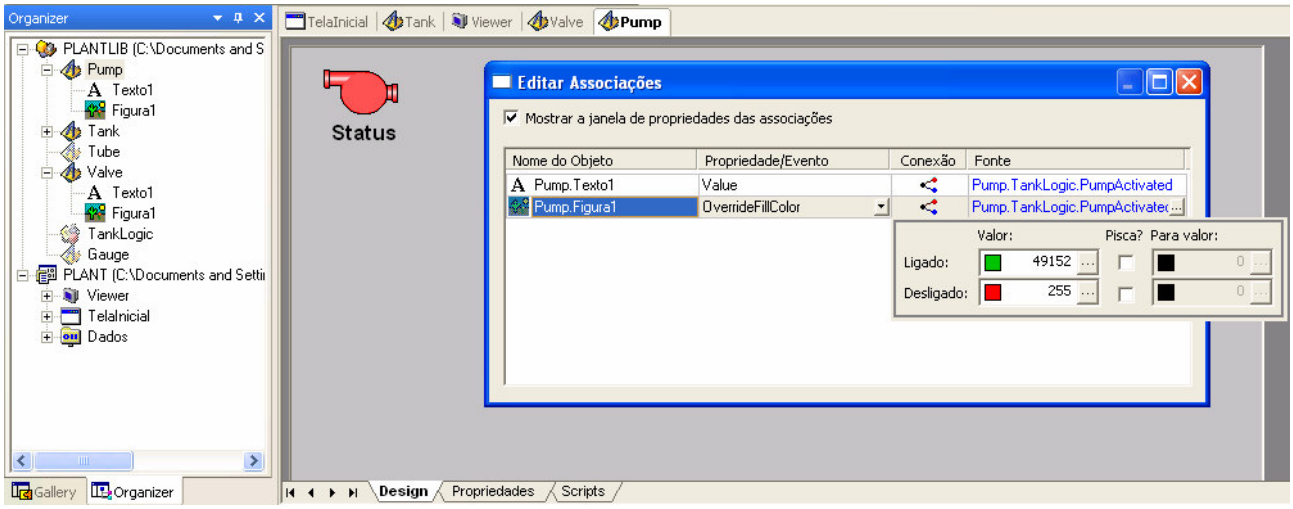

Figura 12 - Associações do objeto Pump

4 Construindo a aplicação

Uma vez pronta a biblioteca, é necessário usar os objetos criados e inseri-los em um projeto. Para isto, basta criar os objetos no projeto Plant conforme desejado. Como a aplicação tem dois tanques, precisamos de dois simuladores do processo de tanques, portanto é necessário criar dois objetos TankLogic na pasta Dados do servidor (Figura 13).

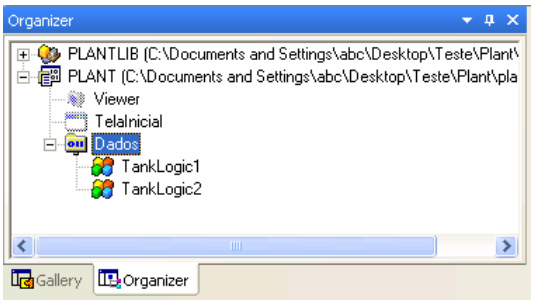

Figura 13 - Objetos TankLogic instanciados no servidor

Estes dois objetos, TankLogic1 e TankLogic2, serão depois associados aos objetos de tela, que farão a interface com o usuário para a abertura/fechamento da válvula e ligamento/desligamento da bomba.

Na TelaInicial incluiremos os objetos de tela que criamos na biblioteca, como os tanques, bombas e válvulas, além dos tubos (ver Figura 14). Cada objeto destes vai ser associado com seu respectivo TankLogic.

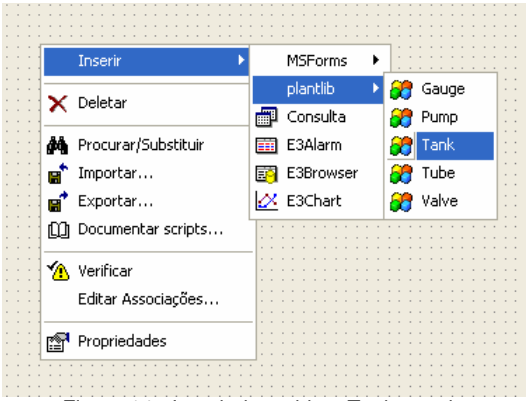

Figura 14 - Inserindo o objeto Tank na tela

Como última etapa do desenvolvimento deste aplicativo deve-se associar os objetos de tela aos seus respectivos objetos TankLogic (Figura 15).

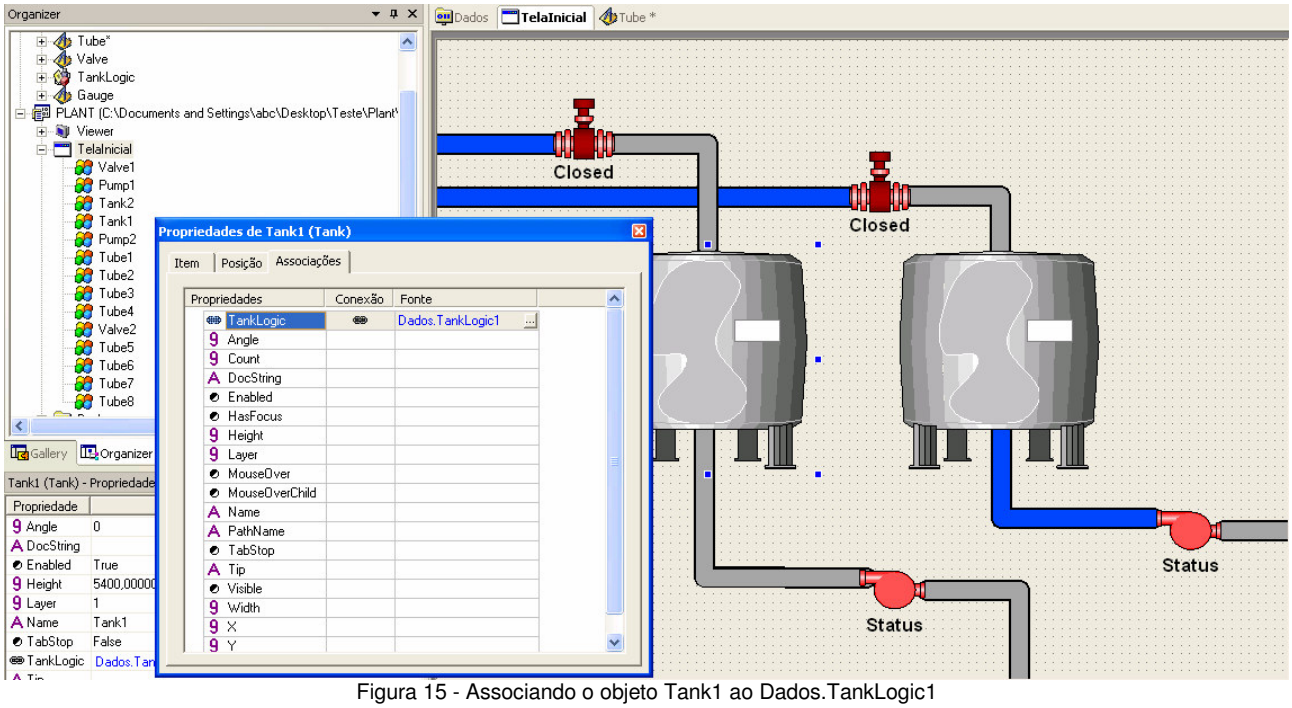

Figura 15 - Associando o objeto Tank1 ao Dados.TankLogic1

Vemos assim como é fácil criar e manter aplicações que tenham usado os recursos das bibliotecas do E3. Depois de criada a biblioteca, basta criar os objetos e associá-los, tornando o trabalho mais simples.

É importante salientar que na biblioteca ficam apenas as definições dos objetos, sendo necessário inserí-los no projeto e configurar suas propriedades para que tenham o funcionamento desejado.

5 Conclusão

Vimos nesta aplicação como é simples e rápida a criação de objetos funcionais com o uso das bibliotecas do E3. Com elas, quebra-se o paradigma de termos que criar toda a aplicação do início, pois é possível reaproveitar objetos que tenham sido criados em outras aplicações.

6 Referências bibliográficas

Corrêa, Alexandre B. **Desenvolvendo aplicações orientadas a objetos no E3 (Parte I).**
http://www.elipse.com.br/download.asp?sSalvarComo=rt003&sArquivo=download/managerFiles/tb_artigo_arquivo/arquivo_4.pdf

Maciel, Paulo H. **Desenvolvendo bibliotecas no E3**.
http://www.elipse.com.br/download.asp?sSalvarComo=&sArquivo=download/managerFiles/tb_artigo_arquivo/arquivo_6.pdf