Aplicação da IEC 61508 na Prototipação de Protocolos Seguros de Comunicação

William Vidal, Rodrigo Dobler, Sérgio Cechin, Taisy Weber e João Netto

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{wrcvidal, rjdobler, cechin, taisy, netto}@inf.ufrgs.br

Resumo. *Visando garantir os níveis de segurança funcional adequados para a correta troca de informações entre os dispositivos de controle e instrumentação em uma planta industrial, as normas IEC 61508 e IEC 61784-3 impõem o uso de protocolos de comunicação seguros. A codificação desse tipo de protocolo deve ser realizada caso a caso e certificada para cada novo equipamento de automação. Neste contexto, este artigo apresenta um protótipo do protocolo PROFIsafe. A construção do protótipo partiu da especificação do protocolo e seguiu as recomendações da norma IEC 61508 para o desenvolvimento de software embarcado de segurança. O conjunto de regras e padrões de codificação desenvolvidos e a experiência na validação do protótipo serão aplicados no desenvolvimento de equipamentos para sistemas instrumentados de segurança.*

Abstract*. To ensure adequate integrity safety levels for the proper exchange of information between control and instrumentation devices in an industrial plant, IEC 61784-3 and IEC61508 standards impose the use of safe communication protocols. For each new equipment such protocols must be specially implemented and certified. In this context, this paper presents a prototype of PROFIsafe protocol. The prototyping started from the protocol specification and followed the recommendations of IEC 61508 for the development of safe embedded software. The set of coding rules we developed and our experience in the validation of the prototype will be incorporated in the development of equipment for safety instrumented systems.*

1. Introdução

Vivemos uma era de grandes desenvolvimentos tecnológicos, na qual se buscam novas tecnologias que possibilitem maior desempenho e eficiência nos processos industriais. Entretanto esse desenvolvimento não pode ser realizado sem a preocupação com a redução de riscos. Defeitos podem conduzir a acidentes graves que provocam perdas financeiras, afetem negativamente o meio ambiente ou causam danos à vida humana.

Sistemas computacionais são amplamente empregados para controle de processos críticos, para coleta e processamento de dados, para simulações e previsões, para gerência de instalações e para uma série de outras funções em ambientes industriais e empresariais. Sistemas computacionais são usados também para aumentar a segurança de instalações industriais executando as funções necessárias para prevenir ou mitigar acidentes nessas instalações (Leveson 1995). A correta operação do sistema em resposta a suas entradas, quando executando funções de segurança, está relacionada à segurança

funcional (Dunn 2003). Exemplos de funções de segurança são: controle de parada emergencial em indústrias químicas; bloqueio preventivo em maquinários pesados; sinais luminosos de alerta; alarmes de incêndio e detecção de gases tóxicos.

Um atributo importante para avaliar a segurança funcional (*functional safety*) é a sua integridade de segurança (Bell 2011) - *safety integrity*. Ela é definida como a probabilidade de um sistema realizar com sucesso a função de segurança a ele atribuída. No lugar de probabilidade de sucesso, o usual é mencionar a probabilidade de ocorrência de defeitos ao realizar a função de segurança. Dependendo do tipo de demanda da função de segurança, são usadas duas medidas para determinação do nível de integridade de segurança: probabilidade de defeito perigoso ao demandar a função (PFD), ou probabilidade de defeito perigoso por hora (PFH).

Para permitir a descentralização de funções nas plantas industriais (Thomesse 2005), sistemas de automação industrial, incluindo tanto controle como instrumentação de segurança, são baseados em barramentos de comunicação (Neumann 2007). A interação entre os elementos de uma rede de automação deve ser realizada de forma rápida, confiável e robusta. Protocolos de comunicação segura, tais como o FF SIS, openSafety, PROFIsafe e o Safety-over-EtherCAT, foram propostos para atingir esse objetivo.

Especificações dos protocolos estão disponíveis para que fornecedores possam desenvolver estes protocolos para as plataformas de hardware e software de seus produtos (Malik and Mühlfeld 2003). Não existem, entretanto, códigos prontos e pré-certificados. Para cada novo equipamento de um sistema de automação ou instrumentação, o protocolo escolhido deve ser codificado seguindo a especificação e obedecendo estritamente as recomendações da norma IEC 61508 (Bell 2006). Não basta provar a conformidade com a especificação. Todas as regras aplicáveis para um determinado nível de integridade de segurança devem ter sido seguidas para pleitear a certificação (Bilich and Hu 2009).

O objetivo do trabalho foi desenvolver o protótipo de um protocolo de segurança seguindo as normas cabíveis e estabelecer um conjunto de regras e restrições de codificação que auxilie na codificação de protocolos de segurança para diferentes tipos de plataformas de hardware e software. Como as regras são extremamente restritivas, elas formam uma lista de proibições e alternativas. O protótipo desenvolvido servirá posteriormente para aplicar as estratégias de validação recomendadas pela IEC 61508 e para treinar a equipe de desenvolvedores na empresa parceira quanto à aplicabilidade e cobertura de tais estratégias.

Entre os protocolos de comunicação segura, este trabalho focou o PROFIsafe, pois esse era o interesse da empresa parceira. O protocolo pode ser implementado a partir da sua especificação ou usando o pacote comercial PROFIsafe StarterKit. Em ambos os casos, o código deve ser adaptado para a plataforma final. Neste trabalho a implementação será efetuada a partir da especificação do protocolo.

O artigo apresenta um breve resumo da norma IEC 61508 e o protocolo PROFIsafe, as principais decisões de projeto sobre protótipo desenvolvido e os resultados alcançados.

2. Normas de segurança funcional

Nas últimas décadas, uma grande quantidade de normas de segurança foi proposta por organizações internacionais para vários setores de atividades (Esposito, Cotroneo, and Silva 2011). Algumas normas apresentam uma natureza mais genérica, como a IEC 61508 (Fowler and Bennett 2000). Outras são direcionadas para áreas específicas, como aviação, energia nuclear, máquinas, processos, indústria automobilística. Várias dessas normas específicas são derivadas da IEC 61508, incluindo a 61784-3, que padroniza perfis de comunicação segura.

2.1. IEC 61508

A IEC 61508 (IEC 1999) trata da segurança funcional de sistemas elétricos, eletrônicos e eletrônicos programáveis e serve como base para outras normas relacionadas com segurança (Bell 2006). O principal objetivo da norma é orientar o trabalho de equipes técnicas no desenvolvimento de equipamentos computacionais de segurança, visando alcançar níveis compatíveis com os exigidos por agências reguladoras em vários domínios de aplicação, incluindo os setores de óleo e gás, geração e distribuição de energia, máquinas e equipamentos, e transportes, entre outros (Gall and Wen 2010).

A norma cobre todo o projeto e desenvolvimento de hardware e software necessários para executar a função de segurança. São considerados o sensor, o controle lógico e o atuador final, incluindo os sistemas de comunicação. Adicionalmente, são consideradas quaisquer ações que possam ser realizadas por operadores humanos (Schonbeck, Rausand, and Rouvroye 2010).

Para classificar os sistemas, a norma especifica quatro níveis de desempenho para uma função de segurança, chamados de níveis de integridade de segurança, SIL (*Safety Integrity Level*). O nível mais baixo corresponde a SIL 1, enquanto o mais alto corresponde ao SIL 4. Os requisitos para um determinado SIL são mais rigorosos na medida em que se aumenta o SIL. Quanto maior o SIL, maior deve ser a garantia de uma menor probabilidade de ocorrência de defeitos perigosos.

Normas para segurança funcional têm em comum a obrigatoriedade ou a recomendação do uso de técnicas de prevenção e de tolerância a falhas para a redução de risco, além do projeto criterioso e documentado, tanto do hardware como de software, desde as primeiras fases do ciclo de desenvolvimento do sistema. As normas permitem certificação de sistemas construídos segundo suas recomendações e que tenham sido devidamente verificados e validados.

Normas de segurança tratam tanto do hardware como do software. Muitos acidentes devidos a falhas de software têm sido reportados na literatura em várias áreas críticas (Grottke and Trivedi 2007), (Zhivich and Cunningham 2009), (Hardy 2013), (Alemzadeh et al. 2013) onde os prejuízos à qualidade de vida são diretos e muitas vezes irremediáveis. Aplicar criteriosamente normas de segurança não fornece, entretanto, garantia absoluta de evitar acidentes provocados por falhas de software e hardware, mas ajuda a reduzir consideravelmente os riscos.

2.2. IEC 61784-3 e o conceito de Black Channel

Quando a função de segurança é realizada por um sistema de automação distribuído, com o uso da comunicação de dados, deve-se estimar a taxa de erros residuais deste

processo de comunicação e considerá-la na determinação das taxas de defeito daquela função de segurança como um todo. Para isso, a norma especifica que devem ser considerados erros como repetições, perda de mensagens, inserção de mensagens, mensagens entregues fora de ordem, corrupção, atraso e mascaramento, que correspondem a erros de endereçamento.

Alternativamente, pode-se utilizar a abordagem do canal escuro de comunicação (*Black Channel*). Este canal implementa um protocolo de comunicação de mais baixo nível, que não precisa seguir as recomendações da norma, e que suporta um protocolo de mais alto nível, este sim responsável pela segurança da comunicação. A principal vantagem do uso do *Black Channel* é que partes do canal de comunicação não precisam ser projetadas e validadas de acordo com a IEC 61508. Entretanto, conforme a IEC 61784-3, que trata dos perfis de comunicação de segurança funcional, mecanismos necessários para assegurar baixa taxa de defeitos devem ser implementados nos subsistemas relacionados com a segurança e que tenham interface com este canal de comunicação.

O conceito de *Black Channel* foi incorporado na versão de 2010 da IEC 61508, mas já aparecia em outras normas específicas de comunicação e publicações anteriores (Neumann 2007). O conceito é extremamente útil para o desenvolvedor de sistemas de segurança. É possível, desta forma, usar uma pilha convencional de protocolos de comunicação e se concentrar apenas na codificação do topo da pilha. Protocolos de comunicação seguros como EtherCAT, PROFIsafe, INTERBUS, OpenSafety e FOUNDATION Fieldbus se apoiam no conceito de *Black Channel* (Neumann 2007). Uma dada codificação desses protocolos é passível de ser certificada segundo a IEC 61508 até o nível SIL 3.

Perfis de protocolos seguros foram normalizados pela IEC 61784-3. A IEC 61784-3 baseia-se nas definições de segurança da IEC 61508. Ela estabelece que o *Black Channel* não deve consumir mais do que 1% do PFD ou do PFH, correspondente ao SIL projetado para a função de segurança.

3. Protocolo PROFIsafe

PROFIsafe é um protocolo de comunicação criado pela Profibus e Profinet Internacional. Sua especificação segue a norma IEC 61508. O nível de segurança que pode ser alcançado através PROFIsafe foi avaliado pela Agência de Inspeção Técnica alemã TÜV SÜD, que certificou a especificação do protocolo para aplicações até SIL 3. Deve ser observado que só a especificação está certificada. Cada nova codificação para uma dada plataforma deve passar por todo o processo de certificação para esta plataforma específica. Também não basta apenas usar um protocolo seguro para contar com um sistema de segurança, todos os componentes de hardware e software do sistema devem ser também certificados.

Com o protocolo PROFIsafe, as redes PROFIBUS e PROFINET, da Profibus e Profinet International, passaram a possibilitar que os seus equipamentos e sistemas pudessem ser usados na implementação de funções de segurança.

Muitos fabricantes de componentes seguros como, por exemplo, Siemens, WAGO, BECKHOFF, participaram da criação de padrões, o que permitiu o desenvolvimento de um portfólio extenso e completo de produtos (Malik and Mühlfeld 2003). Estes produtos são usados para realizar funções de segurança em processos

industriais, no setor de transportes, em equipamentos de guindastes, teleféricos, na indústria automotiva, entre outros. Como exemplo dos equipamentos desenvolvidos com PROFIsafe, pode-se citar: portas remotas de entrada e saída, sensores ópticos, sistemas de controle de segurança, gateways seguros, sensores de segurança, dispositivos com funções de segurança integrada, válvulas e bloqueios de válvulas.

3.1. Comunicação no PROFIsafe

O protocolo utiliza uma abordagem de canal simples e deve ser capaz de permitir o processamento nos equipamentos de controle, assim como a comunicação, de informações seguras e não seguras. Para isso, é necessário garantir que esses dois tipos de processamento ocorram de forma logicamente isolada. Essa abordagem permite a operação de sistemas que apresentam um único barramento físico de comunicação, o que é vantajoso do ponto de vista econômico.

O protocolo deve ser implementado sobre um *Black Channel*, o que o torna independente dos canais físicos de transmissão, sejam eles fios de cobre, fibras ópticas, *wireless* ou *backplanes*. As taxas de transmissão e os mecanismos de detecção de erros do *Black Channel* não têm qualquer interferência sobre o protocolo seguro. Entretanto o *Black Channel* deve garantir uma taxa máxima de defeitos. Se não operar abaixo desta taxa, não é possível garantir a segurança da camada construída sobre o canal inseguro.

O *Black Channel* torna desnecessária a avaliação de segurança dos elementos que o compõem: *backplanes* individuais, caminhos de transmissão e redes PROFIBUS e PROFINET. Toda a segurança do caminho, desde a geração do sinal seguro em um dispositivo remoto de entradas até o seu destino onde será processado, é obtida através dos mecanismos de proteção do protocolo seguro codificado sobre o Black Channel.

Na nomenclatura do PROFIsafe, existem três elementos: O F-Module, o F-Host (ou mestre) e o F-Device (ou escravo). Um módulo é representado por: entradas ou saídas digitais e analógicas, módulos de força, módulos de partida de motores e conversores de frequência, entre outros. Um mestre é um controlador. Um exemplo de dispositivo é um *scanner* a laser. Os parâmetros individuais de segurança variam de acordo com as diferentes tecnologias de segurança dos dispositivos e devem ser codificados e, principalmente, protegidos de operações que levem a defeitos no sistema.

3.2. Modelo de falhas

Erros na comunicação são ocasionados por falhas de hardware, interferência eletro-magnética e outros tipos de interferência ambiental. O modelo de falhas abrange erros de transmissão, repetições, perda de mensagens, inserção, sequenciamento, corrupção, atraso e mascaramento.

Erros de transmissão ocorrem quando causas externas ocasionam dados corrompidos. Corrupção é semelhante a erros de transmissão, mas tem como causa elementos do próprio sistema. Repetição ocorre quando mensagens velhas são repetidas. Um dispositivo de barramento com defeito pode apagar ou inserir uma mensagem ou modificar a ordem relativa das mensagens. Atrasos geralmente ocorrem quando os enlaces de comunicação estão saturados e as mensagens gastam mais tempo para circular do que o previsto. Mascaramento mistura mensagens relevantes e sem relevância para a segurança.

3.3. Mecanismos de detecção de erros

Para detectar e corrigir erros de comunicação, PROFIsafe foi especificado com os seguintes mecanismos: número sequencial de mensagens, controle de temporização, identificador único e CRC. Todos os mecanismos são tradicionais e aplicados largamente em inúmeros protocolos de comunicação. A diferença é a combinação dos diversos mecanismos.

O número de sequência permite identificar se as mensagens estão na ordem correta. O número de sequência detecta repetição não intencional, perda de mensagens, inserção de mensagens, sequência incorreta de mensagens e erros de repetição de mensagens. No protocolo, o número de sequência possui apenas um bit e tem que aparecer alternado entre duas mensagens. *Watchdog* detecta atrasos no caminho de comunicação. No protocolo, o *watchdog* detecta também falhas de perda de mensagens, mascaramento e inserções em mensagens.

O identificador único de conexão detecta mensagens mal encaminhadas entre transmissor e receptor. Esse identificador é conhecido unicamente pelos dois dispositivos envolvidos e é estabelecido a cada conexão. O identificador permite detectar endereçamento incorreto entre segmentos da rede, além de inserções e mascaramento em mensagens.

Finalmente, o CRC detecta bits de dados corrompidos nas mensagens. Adicionalmente, esta técnica permite detectar se algum módulo seguro realizou trocas de informações entre mensagens de segurança, e auxilia na detecção de mascaramento de mensagens.

No PROFIsafe, o CRC é calculado em múltiplos níveis. Primeiramente, ao longo da parametrização inicial do protocolo, é calculado o CRC1, composto de 2 octetos. O CRC, que é então encapsulado no pacote, é chamado de CRC2. Ele é formado pelos campos do pacote denominados dados de saída, F-Parametros, Status/Control Byte e Número Consecutivo. Para trazer mais segurança à técnica, o CRC é recalculado quando o pacote chega ao seu destino e comparado byte a byte com o campo CRC2 presente no pacote. Isso é possível, pois todos os bytes necessários para recalcular o CRC se encontram no próprio pacote ou já foram estabelecidos na fase inicial de parametrização do PROFIsafe.

4. Construção do protótipo

A empresa parceira no projeto pretende desenvolver uma linha de produtos de segurança certificados em conformidade com a IEC 61508. Para atingir tal objetivo é preciso adquirir experiência com a aplicação da norma, que é extensa, abrangendo mais do que 800 páginas de recomendações, sugestões e restrições para todas as fases do ciclo de vida de um produto de hardware e software. A certificação leva em conta não apenas o produto, mas toda a cultura de desenvolvimento e produção da empresa, portanto, o período de adaptação é longo.

O trabalho de prototipação irá fornecer subsídios para a empresa definir métodos e técnicas apropriados à produção de sistemas de segurança visando certificação de conformidade com a IEC61508. Assim, definiu-se que o protótipo do subsistema de comunicação seria um meio adequado para a aplicação das técnicas recomendadas de desenvolvimento e validação e permitiria adquirir experiência suficiente para estabe-

lecer padrões de codificação a serem seguidos pela empresa.

4.1. Sistemas de segurança versus sistemas computacionais convencionais

Há algumas diferenças entre o processo de desenvolvimento de software e hardware em projetos tradicionais e em projetos com requisitos de segurança. Nesses últimos, a criatividade do desenvolvedor é restrita a aplicar técnicas que já se mostraram adequadas na prática (Wassyng et al. 2011). Novidades acadêmicas e inovações não são bem acolhidas pelas normas de segurança funcional.

O principal desafio é aplicar técnicas de tolerância a falhas e engenharia de software no desenvolvimento de dispositivos de segurança levando criteriosamente em conta requisitos de segurança funcional e seguindo as normas apropriadas para o domínio da aplicação e sem explosão dos custos de desenvolvimento.

4.2. Modelo de comunicação

O sistema executa uma aplicação com dados recebidos pelos sensores e devolve os resultados para os atuadores. As portas de entrada (F-Devices) que coletam dados dos sensores se comunicam com o controlador (F-Host) através de um *Black Channel*. O controlador se comunica com as portas de saída (F-Devices) que fornecem dados aos atuadores através de outro *Black Channel*. O protocolo de comunicação é o PROFIsafe.

O modelo de comunicação do PROFIsafe é do tipo um para um (mestre/escravo), tendo o mestre como o F-Host e o escravo como o F-Device. O F-Device só envia mensagens em resposta às mensagens de solicitação de pedidos. Se o F-Device for de entrada, a resposta contém os dados de entrada. Se o F-Device for de saída, o dispositivo recebe os dados e confirma ao host o recebimento da mensagem. Como o dispositivo jamais gera uma mensagem espontaneamente e apenas responde a solicitações, no protótipo do PROFIsafe foi usada uma arquitetura sem threads, onde a aplicação determina o passo de execução de todos os mecanismos implementados na parte segura da arquitetura.

A discussão sobre a implementação nos próximos itens foca a codificação do protocolo como executada nos F-Devices. O F-Host tem uma implementação semelhante, mas como ele atua como mestre, seu código é mais complexo.

4.3. Arquitetura do sistema

O objetivo do protocolo é garantir a comunicação segura. Dessa forma, a arquitetura proposta inicialmente busca assegurar a separação entre a parte segura e a parte não segura.

A codificação do protocolo segue o conceito de *Black Channel*. Os mecanismos padrões de comunicação são componentes da parte não segura do sistema. O código da aplicação e uma porção do código do protocolo de comunicação ficam na parte segura. A codificação deve obedecer à norma: entre outras limitações pode-se citar a impossibilidade de uso de ponteiros (o que levou ao uso das chamadas de sinalização e envio de dados byte-a-byte) e de threads na parte segura (essa limitação pode ser removida, caso exista um sistema de gerência de threads que seja certificado).

Com isso, foi proposta uma arquitetura com duas partes e três camadas (Figura 1): a parte segura, onde está a aplicação; o código do protocolo seguro; e a parte não segura,

representando o início do *Black Channel*, onde está a biblioteca do canal de comunicação. Além disso, cada camada comunica-se com as camadas superiores ou inferiores através de um mecanismo de buffers de mensagens.

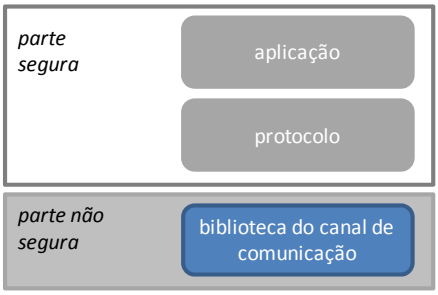

Figura 1 – Arquitetura de 3 camadas

4.4. Interação entre as partes seguras e não seguras

A troca de informação entre as partes segura e não segura não está claramente definida nas normas, e precisou ser desenvolvida. Nossa primeira abordagem está detalhada na Figura 2, onde os buffers de entrada e saída das mensagens estão nas camadas não segura e segura, respectivamente.

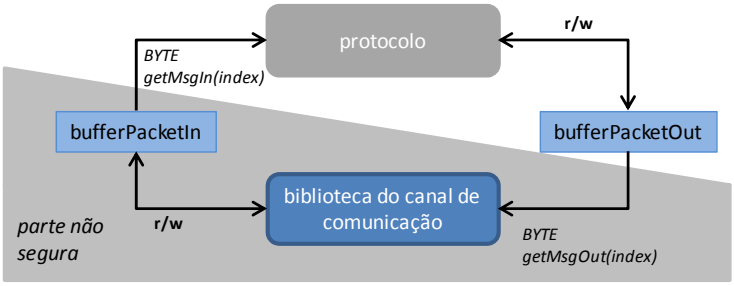

Figura 2 – arquitetura dos buffers de entrada e saída

Essa abordagem deixa o código com alto acoplamento. Além disso, ele fica pouco coeso e não padronizado. A escrita e leitura diretamente nos buffers requerem o posicionamento do código em uma das camadas (segura ou não segura) e o acesso aos buffers que estão na outra camada. Esta abordagem não oferece o isolamento de código requerido pelas normas, consequentemente, ela teve que ser aprimorada para garantir esse isolamento.

A solução foi restringir a parte não segura no que diz respeito ao acesso aos buffers internos da parte segura. Isso foi realizado por funções específicas de controle de acesso, usadas apenas para leitura.

Informações entre camadas de software do F-Device são trocadas através de chamadas de função. Conforme pode ser visto na Figura 3, são duas funções de "descida" e duas de "subida". Além disso, cada par possui uma função para a passagem de dados e outra para sinalização (que não têm parâmetros). O principal motivo para o desenvolvimento desta solução, além dos apurados anteriormente, foi a restrição do uso de ponteiros pela norma IEC 61508.

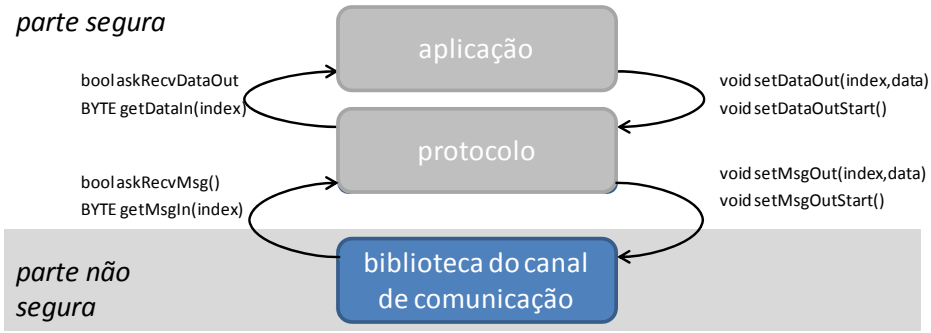

Figura 3 - arquitetura do protótipo

4.5. Recepção e envio de pacotes

O pacote PROFIsafe é formado por dados (até no máximo 123 bytes), CRC (3 ou 4 bytes) e *Status/Control Byte* (1 byte). Neste protótipo, optou-se por *payloads* de 12 bytes (correspondendo ao modo de operação rápido previsto na especificação), formando um pacote de 16 bytes.

Todo o envio de pacote pelo F-Device é precedido de uma recepção de pacote. Para envio de pacotes, a aplicação monta a sua mensagem através de setDataOut() que repassa os dados para a camada abaixo (camada de protocolo). Esses dados são salvos, byte-a-byte, em uma área de dados chamada de bufferDataAplicationSend. Em seguida, a aplicação sinaliza que terminou o envio dos dados através de setDataOutStart(). Com isso, o protocolo é notificado que os dados de entrada estão prontos para o seu uso exclusivo.

Prontamente, a camada de protocolo começa o encapsulamento do pacote. Os dados são transferidos para o módulo que calcula o CRC. Após, é criado o byte de controle/status que é encapsulado em bufferToSend para formar o pacote PROFIsafe. Após o encapsulamento, o pacote está pronto em bufferToSend que agora é transferido, byte-a-byte, para a camada inferior (denominada biblioteca) através da chamada setMsgOut(). A sinalização setMsgStart() avisa para a camada de biblioteca que os dados foram transferidos.

Nota-se que para a camada de protocolo, a sinalização que a mensagem foi enviada é a chamada *setMsgStart().* A camada de biblioteca funciona como uma caixa preta; não interessa de que forma a mensagem irá chegar ao seu destino.

4.6. Sistema de interrupção

O código do protocolo é embarcado em um módulo de hardware com poucos recursos computacionais. No protótipo foi necessário incorporar um sistema de interrupção, uma vez que não há um sistema operacional disponível no dispositivo que possa tratar as interrupções do hardware.

Não existe restrição ao uso de um sistema operacional embarcado. Entretanto essa opção implicaria em contar com um sistema operacional certificado, o que representaria um custo muito elevado para equipamentos de segurança simples como módulos de entrada e saída. Por essa razão, faz parte da arquitetura do protótipo um sistema de interrupção. Sua primeira função é a detecção de *timeout*, onde um contador

é decrementado (*watchdog*) e a sinalização ocorre quando o contador zera. A segunda função está relacionada à recepção do pacote. Foi preciso criar um semáforo binário para garantir a exclusividade de acesso ao sinal que informa o recebimento de uma nova mensagem. Este sinal é atualizado pelo sistema de interrupções. O sistema de interrupção opera como uma *thread*, responsável por realizar tais verificações. Esse mecanismo encontra-se na parte não segura da arquitetura, ou seja, está no *Black Channel*.

4.7. Testes com o protótipo

O código do protótipo foi validado por testes sob falhas. Foi usada como carga de trabalho a troca de mensagens entre os dispositivos mestre e escravo, onde foram injetados e detectados atrasos e corrupção de mensagens.

Nos testes realizados, em condições normais de operação, a troca de mensagens entre os dispositivos mestre e escravo ocorreu de maneira correta. Nos testes com injeção de atraso, quando foi inserido um atraso maior do que o valor máximo tolerado pelo protocolo para o recebimento de mensagens, este foi detectado pelo mecanismo de timeout. A corrupção de mensagens foi provocada pela alteração de um byte no pacote, a qual foi detectada no F-Device por meio de CRC.

Os testes realizados são ainda incompletos, não abrangendo todos os tipos de falhas necessários para ativar todos os mecanismos de detecção de erros do protocolo implementado, que estão previstas no modelo de falhas. Os próximos testes de injeção de falhas irão considerar o modelo de falhas completo.

5. Resultados

O processo de desenvolvimento teve que considerar os aspectos relativos à especificação do PROFIsafe e às recomendações da IEC 61508.

Notamos que a especificação do PROFIsafe não ajuda na forma como este deve ser integrado ao restante do sistema. Dessa forma, não basta apenas conceber a implementação do protocolo. É necessário considerar o restante do sistema, para obter o isolamento de software exigido pela IEC 61508. Na especificação do protocolo existe pouca informação sobre como ele deve ser acionado. Essa omissão é suprida nos kits de desenvolvimento, onde é definida uma API de acionamento do protocolo.

As recomendações impostas pela IEC 61508 implicam em uma mudança de cultura para desenvolvedores acostumados com sistemas de mais alto nível sem restrições de segurança. Seguir as normas é uma tarefa difícil e não intuitiva. Especificação, planejamento de testes (validação), desenvolvimento, integração, operação, manutenção (ou seja, procedimentos para modificação) e testes devem fazer parte do ciclo de vida de qualquer bom projeto de software, mesmo aqueles sem qualquer requisito de segurança. A diferença entre sistemas a serem certificados pela IEC 61508 e os demais é que a aplicação das técnicas e medidas recomendadas para cada fase do ciclo de vida deve ser justificada e demonstrada para uma equipe externa de certificadores através de documentação adequada.

A norma preconiza o estabelecimento de um padrão de codificação que deve ser obedecido pelos desenvolvedores. Segundo a IEC 61508, o padrão de codificação deve:

- Definir boas práticas de programação,

- Proibir características não seguras da linguagem,
- Promover facilidades para entender o código,
- Facilitar verificação e teste,
- Especificar procedimentos para documentação do código fonte,
- Garantir que no mínimo o código fonte deva conter: entidade legal, descrição, entradas e saídas, histórico da gerência de configuração.

O padrão de codificação proposto incorpora a experiência com o desenvolvimento do protótipo e consiste de uma lista de restrições e alternativas. Uma parte da lista é mostrada na Tabela 1.

Tabela 1 – Restrições e alternativas para a codificação de sistemas seguros

Não usar recursão
Não usar ponteiros
Não usar variáveis e objetos dinâmicos
Não usar conversão automática de tipos
Não usar desvios incondicionais em linguagens de alto nível
Usar abordagem modular
Usar bibliotecas de módulos e componentes de software confiáveis / verificados quando disponíveis
Limitar o número de parâmetros para as rotinas
Usar apenas um ponto de entrada e saída por rotina
Usar interfaces completamente definidas
Não usar linguagens orientadas a objeto
Evitar linguagens fracamente tipadas
Não usar estratégias de detecção de erros baseadas em inteligência artificial
Facilitar rastreabilidade
Favorecer testabilidade

As regras podem parecer muito restritivas e comprometer a produtividade de geração de código. Novas técnicas, métodos e estratégias para o desenvolvimento de software visando aumentar a produtividade aparecem continuamente e são rapidamente adotados pelas empresas desenvolvedoras. Métodos e técnicas estão sujeitos a um processo seletivo bem acentuado. Os melhores sobrevivem após algum tempo.

Muitos métodos de desenvolvimento promissores não resistem à passagem do tempo, e são substituídos por outros mais eficientes. Algumas técnicas inovadoras aplicadas ao software mostram posteriormente fragilidades em relação à robustez do código gerado. Assim o tempo é um importante aliado na escolha dos métodos e técnicas mais eficientes para a área de desenvolvimento de software relacionado à segurança. Métodos e técnicas que não podem ser provados seguros ou que geram software com maior probabilidade de conter falhas residuais não devem ser aplicados no desenvolvimento de software que executa funções de segurança. Métodos e técnicas mais antigos, que na prática já mostraram capacidade de gerar produtos robustos, são preferidos nesta área. Esse é o motivo porque as regras de codificação que aparecem na Tabela 1 parecem tão restritivas. Elas sozinhas não garantem a segurança do código, mas permitem gerar código com menor probabilidade de erros.

A maior parte das restrições não é novidade para os desenvolvedores. Apesar disso, convencer desenvolvedores a não usar ponteiros e evitar linguagens orientadas a objetos e até linguagens fracamente tipadas é uma tarefa difícil. Mas evidências baseadas na experiência prometem ser uma boa motivação.

6. Conclusão

O protocolo de comunicação segura escolhido atende as recomendações de segurança da norma IEC 61508, tendo sua especificação sido certificada para aplicações de segurança até SIL 3. Utiliza o conceito de *Black Channel*, no qual os recursos necessários para transportar as mensagens do protocolo seguro não são considerados para fins de avaliação da segurança. Além disso, o protocolo apresenta mecanismos de controle de falhas adequados aos modelos de falhas descritos na IEC 61508. Esses mecanismos são o número de sequência, *watchdog*, identificadores únicos e CRC.

A experiência no desenvolvimento do protótipo gerou um padrão de codificação exigido pela norma. O protótipo permitirá também aplicar as estratégias de teste previstas na norma. Testes iniciais sob falhas já foram conduzidos. Uma experiência mais abrangente com testes do protótipo será repassada à empresa interessada para ser aplicado no desenvolvimento de seus produtos. Caso a empresa opte por desenvolver sua própria codificação do protocolo de comunicação, o protótipo será usado para comparação dos resultados entre as duas implementações.

Agradecimentos

O projeto foi apoiado financeiramente pelo programa de formação de recursos humanos da ANP para o setor de Petróleo e Gás - PRH PB-217 (Petrobras) e o projeto Rio-SIL (Finep).

Referências

Alemzadeh, Homa, Jai Raman, Zbigniew Kalbarczyk, and Ravishankar Iyer. 2013. "Analysis of Safety-Critical Computer Failures in Medical Devices." *IEEE Security & Privacy* 99 (1): 1.

Bell, Ron. 2006. "Introduction to IEC 61508." In *Proceedings of the 10th Australian Workshop on Safety Critical Systems and software-Volume 55*, 3–12. Australian Computer Society, Inc.

———. 2011. "Introduction and Revision of IEC 61508." In *Advances in Systems Safety*, edited by Chris Dale and Tom Anderson, 273–91. Springer London.

Bilich, Carlos, and Zaijun Hu. 2009. "Experiences with the Certification of a Generic Functional Safety Management Structure According to IEC 61508." In *Computer Safety, Reliability, and Security*, edited by Bettina Buth, Gerd Rabe, and Till Seyfarth, 5775:103–17. Lecture Notes in Computer Science. Springer Berlin / Heidelberg.

Dunn, W.R. 2003. "Designing Safety-critical Computer Systems." *Computer* 36 (11): 40– 46.

Esposito, Christian, Domenico Cotroneo, and Nuno Silva. 2011. "Investigation on Safety-Related Standards for Critical Systems." In *Software Certification*

(WoSoCER), 2011 First International Workshop On, 49–54. IEEE.

Fowler, Derek, and Phil Bennett. 2000. "IEC 61508 — A Suitable Basis for the Certification of Safety-Critical Transport-Infrastructure Systems ??" In *Computer Safety, Reliability and Security*, edited by Floor Koornneef and Meine van der Meulen, 1943:250–63. Lecture Notes in Computer Science. Springer Berlin / Heidelberg.

Gall, Heinz, and Joachim Wen. 2010. "Functional Safety IEC 61508 and Sector Standards for Machinery and Process Industry the Impact to Certification and Users Including IEC 61508 2nd Edition." In *23rd International Congress on Condition Monitoring and Diagnostic Engineering Management, COMADEM 2010, June 28, 2010 - July 2, 2010*, 73–81.

Grottke, Michael, and Kishor S. Trivedi. 2007. "Fighting Bugs: Remove, Retry, Replicate, and Rejuvenate." *Computer* 40 (2): 107–9.

Hardy, Terry. 2013. "Case Studies in Process Safety: Lessons Learned From Software-Related Accidents."

IEC. 1999. *IEC 61508: Functional Safety of Electrical, Electronic/Programmable Electronic Safety-Related Systems*.

Leveson, Nancy G. 1995. *Safeware: System Safety and Computers*. 1 edition. Reading, Mass: Addison-Wesley Professional.

Malik, Robi, and Reinhard Mühlfeld. 2003. "A Case Study in Verification of UML Statecharts: The PROFIsafe Protocol." *J. UCS* 9 (2): 138–51.

Neumann, Peter. 2007. "Communication in Industrial automation—What Is Going On?" *Control Engineering Practice* 15 (11): 1332–47.

Schonbeck, Martin, Marvin Rausand, and Jan Rouvroye. 2010. "Human and Organisational Factors in the Operational Phase of Safety Instrumented Systems: A New Approach." *Safety Science* 48 (3): 310–18.

Thomesse, J.-P. 2005. "Fieldbus Technology in Industrial Automation." *Proceedings of the IEEE* 93 (6): 1073–1101.

Wassyng, Alan, Tom Maibaum, Mark Lawford, and Hans Bherer. 2011. "Software Certification: Is There a Case Against Safety Cases?" In *Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems*, 206–27. Springer.

Zhivich, Michael, and Robert K. Cunningham. 2009. "The Real Cost of Software Errors." *Security & Privacy, IEEE* 7 (2): 87–90.