UNIVERSIDADE FEDERAL DA BAHIA
ESCOLA POLITÉNICA
PROGRAMA DE PÓS GRADUAÇÃO EM MECATRÔNICA

MÁRCIO FREIRE CRUZ

SERVIDOR HL7-OPC PARA AQUISIÇÃO E INTEGRAÇÃO DE SINAIS VITAIS DE PACIENTES EM MONITORIZAÇÃO CLÍNICA

Salvador
2015

MÁRCIO FREIRE CRUZ

SERVIDOR HL7-OPC PARA AQUISIÇÃO E INTEGRAÇÃO DE SINAIS VITAIS DE PACIENTES EM MONITORIZAÇÃO CLÍNICA

Dissertação apresentada ao Programa de Pós-Graduação em Mecatrônica da Escola Politécnica da Universidade Federal da Bahia, como requisito para obtenção do grau de Mestre em Mecatrônica.

Orientador:
Prof. Dr. Carlos Arthur Mattos Teixeira Cavalcante

Salvador
2015

C957 Cruz, Márcio Freire.

Servidor HL7-OPC para aquisição e integração de sinais vitais de pacientes em monitorização clínica / Márcio Freire Cruz – Salvador, 2015.

104 f. : il.

Orientador: Prof. Dr. Carlos Arthur Mattos Teixeira Cavalcante.

Dissertação (mestrado) – Universidade Federal da Bahia. Escola Politécnica, 2015.

1. HL7. 2. OLE *for Process Control*. 3. Sistemas de informação - Medicina. 4. Interoperabilidade. I. Cavalcante, Carlos Arthur Mattos Teixeira. II. Universidade Federal da Bahia. III. Título.

CDD: 005.74

TERMO DE APROVAÇÃO

MÁRCIO FREIRE CRUZ

SERVIDOR HL7-OPC PARA AQUISIÇÃO E INTEGRAÇÃO DE SINAIS VITAIS DE PACIENTES EM MONITORIZAÇÃO CLÍNICA

Dissertação submetida ao corpo docente da coordenação do Programa de Pós-Graduação em Mecatrônica da Universidade Federal da Bahia como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências em Mecatrônica.

Aprovada em: 10 de abril de 2015.

Banca Examinadora:

Prof. Dr. Carlos Arthur Mattos Teixeira Cavalcante _____
Doutor em Engenharia de Produção, Universidade de São Paulo – USP.
Universidade Federal da Bahia – UFBA.

Prof. Dr. Alírio Santos de Sá _____
Doutor em Ciência da Computação – Universidade Federal da Bahia - UFBA
Universidade Federal da Bahia – UFBA.

Prof. Dr. Nei Yoshihiro Soma _____
Doutor em Applied and Computational Mathematics – University of Sheffield, Sheffield, UK.
Instituto Tecnológico de Aeronáutica – ITA.

Dedico este trabalho à minha mãe, Élida, meu irmão, André e minha irmã, Carolina, sem os quais eu não estaria aqui, por terem formado meu caráter e incentivado a minha vida acadêmica desde o princípio para que eu pudesse enfrentar as adversidades da vida.

A minha esposa Márcia e minha filha Maria Luiza, pela paciência, compreensão e apoio durante esta etapa extenuante, porém gratificante das nossas vidas, onde tivemos que abdicar do lazer em prol desta dissertação de mestrado.

AGRADECIMENTOS

A Deus, por me permitir chegar até aqui para que eu pudesse ter o livre arbítrio de escolher este programa de Pós-Graduação como ponto de partida para alçar novos voos.

Ao meu orientador Prof. Arthur, pelas ideias e redirecionamento, frutos do seu árduo ofício de orientar e estimular a execução correta das atividades necessárias a produção de um trabalho deste porte. Sem o seu esforço e paciência este trabalho não lograria êxito.

Ao meu grande amigo Sérgio, pela experiência e imaginação fértil na concepção da ideia inicial, além da cooperação crucial no desenvolvimento da solução e do conteúdo desta dissertação. O seu vasto conhecimento e apoio foram essenciais para o sucesso deste mestrado.

A minha amiga Luciana, pela perspicácia na identificação de detalhes e por disponibilizar seus conhecimentos em prol deste trabalho.

Ao meu amigo André, pelo apoio imprescindível na resolução das questões técnicas da solução desenvolvida neste trabalho.

Ao amigo Grinaldo, pelo bom humor, conselhos e apoio nos momentos das dificuldades tão comuns a este tipo de empreitada.

Ao Programa de Pós-graduação em Mecatrônica e aos professores, pela transmissão de seus conhecimentos, ensinamentos e convívio.

Aos funcionários da UFBA, pela dedicação e paciência ao lidar com os alunos.

A FIOCRUZ Bahia e meus colegas de trabalho, pela colaboração, aprendizado e oportunidade de finalizar o mestrado.

A todas as pessoas que, direta ou indiretamente, contribuíram para a execução desta Dissertação de Mestrado.

Muito obrigado!

Quem tiver talento, obterá o êxito na medida que lhe corresponda. Porém, apenas se persistir naquilo que faz.

Isaac Asimov

RESUMO

Hospitais e clínicas ao redor do mundo têm se beneficiado do uso da Tecnologia da Informação para atender pacientes com diversas enfermidades. Dentre os softwares e hardwares disponíveis, destacam-se, por sua importância na avaliação da condição clínica dos pacientes, os equipamentos de monitoramento de sinais vitais, tais como frequência cardíaca, temperatura, pressão arterial e saturação de oxigênio. Apesar da relevância destes dispositivos, observa-se uma lacuna no que diz respeito à sua interoperabilidade com diferentes sistemas. Normalmente, quando há funções de compartilhamento, são utilizados padrões ou protocolos de comunicação proprietários que dificultam o desenvolvimento de uma solução integrada. Entretanto, alguns padrões de interoperabilidade para área médica têm sido adotados com o objetivo de compatibilizar a utilização desses aparelhos em sistemas integrados. Dentre estes, destaca-se o *Health Level Seven* (HL7) que define um formato comum para transmissão, recepção e armazenamento de informações clínicas, independentemente do protocolo de comunicação adotado pelo fornecedor. Desta forma, o HL7 vem sendo utilizado no desenvolvimento de sistemas médicos centralizados que capturam e armazenam dados provenientes de equipamentos heterogêneos de monitoramento de sinais vitais. Contudo, quando se trata de armazenar dados de diversos pacientes por longos períodos (meses ou anos registrados minuto a minuto, por exemplo), este padrão revela-se inadequado. Padrões como o HL7 têm sido largamente utilizados em soluções que envolvem o armazenamento em bancos de dados tradicionais, apesar destes serem menos eficientes em armazenar e disponibilizar, em tempo real, grandes volumes de dados, diferentemente dos historiadores de dados industriais ou *Process Information Management System* (PIMS). O que essencialmente diferencia um PIMS de um banco de dados tradicional é a sua capacidade de armazenar grandes volumes de dados consumindo pouco espaço em disco, devido ao uso de filtros de compressão e exceção antes do efetivo armazenamento. Entretanto, para que dados no padrão HL7 sejam inseridos em um PIMS, é necessário convertê-los para um padrão de comunicação adequado. O *Plant Information* (PI) é um PIMS que opera no padrão OLE *for Process Control* (OPC) e tem sido largamente utilizado com sucesso na indústria de manufatura para resolver a interoperabilidade entre os diversos equipamentos e sistemas. Neste sentido, o presente trabalho propõe resolver este mesmo problema em sistemas da área médica ao desenvolver um servidor HL7-OPC que habilita o PI a capturar e armazenar longas séries históricas de sinais vitais. Consequentemente, além do tradicional monitoramento das condições de saúde, em tempo real ou histórico, os dados podem ser analisados por algoritmos de reconhecimento de padrões para, dentre outras finalidades, determinar com antecipação a probabilidade de ocorrências médicas em pacientes sob monitorização. Foi realizado um estudo detalhado dos padrões de comunicação utilizados na indústria e na área médica, seguido de uma análise da arquitetura do padrão OPC e da arquitetura geral dos PIMS, especialmente quanto a algoritmos de compressão e exceção. Com base nos elementos levantados, foi possível desenvolver o Servidor HL7-OPC proposto neste trabalho. A solução foi submetida à uma sequência de cenários de testes que comprovaram o seu pleno funcionamento e desempenho.

Palavras-chave: HL7; OLE *for Process Control*; Sistemas de informação - Medicina; Interoperabilidade.

ABSTRACT

The Hospitals and clinics worldwide have been benefited by using of Information Technology in order to assist their patients with many health issues. Within the available software and hardware, there are equipment that monitors vital sign, such as the heart frequency, body temperature, blood pressure and blood oxygenation, due to their important role on the evaluation of the patients' clinical condition. Despite of the relevance of these devices, there is a gap in its interoperability with different systems. Usually when there are functions for sharing, they use proprietary communication protocols that hinder the development of an integrated solution. However, some medical interoperability standards have been adopted aiming to use these devices in integrated systems. Among these, there is the Health Level Seven (HL7), which is a standard that defines a common format for transmission, reception and storage of clinical information, regardless of the communication protocol from the supplier. Thus, the HL7 has been applied to medical centralized systems that capture and store data from heterogeneous vital signs monitoring equipment. Nevertheless, in the case of storage multiple patient's data for long periods (e.g. months or years registered every minute), this communication standard seems inappropriate. Communication standards, such as HL7, are usually used to storage data into traditional databases that are not designed to efficiently store large amounts of data and provide real-time queries of them, as occurs in industrial data historians or Process Information Management System (PIMS). The point that essentially distinguishes a PIMS from a traditional database is the ability to store large amount of data in low disk space, due to the use of compression and exception filters before the storage. However, to insert HL7 data into a PIMS, it is necessary to submit them to an appropriate communication standard. The Plant Information (PI) is a PIMS that uses OLE for Process Control (OPC) standard, which has been successfully used in the manufacturing industry to address interoperability between its various equipment and systems. Therefore, this research proposes to solve this issue in medical systems by developing a HL7-OPC Server that enables PI to capture and store long historical series of vital signs. Consequently, besides the traditional real-time or historical monitoring of health conditions, pattern recognition algorithms can analyze the data to, among other purposes, determine the probability of medical occurrence in patients under monitorization. This study evaluated communication patterns used in industry and in medical field, followed by an analysis of OPC standard and of PIMS overall architecture, especially theirs compression and exception algorithms. Based on these elements, the HL7-OPC Server was developed. Several tests were performed made to prove that it was full operation and with great performance.

Key Words: HL7; OLE for Process Control; Information Systems - Medicine; Interoperability.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACK	Mensagem de reconhecimento geral do HL7.
ACR	American College of Radiology.
ADT	Admissão, alta e transferência de paciente.
ASC	Accredited Standards Committee.
ASC X12	Padrão para intercâmbio de dados definido pela ASC.
ASTM	American Society of Testing and Materials.
ASTM 1238.94	Padrão para registros de dados de laboratório definido pela ASTM.
BCBS	Boxcar/Backslope.
BD	Base de Dados.
BPM	Batimentos cardíacos por minuto.
C/C++	Linguagem de Programação C/C++.
CEN	European Committee for Standardization.
CLP	Controlador Lógico Programável.
COM	Common Object Model.
DCOM	Distributed Common Object Model.
DICOM	Digital Imaging and Communications in Medicine.
ECG	Eletrocardiograma.
EMR	Eletronic Medical Record.
HITSP	Healthcare Information Technology Standards Panel.
HL7	Health Level Seven.
HL7I	Health Level Seven International.
IDE	Integrated Development Environment.
IEEE	Institute of Electrical and Electronics Engineers.
IEEE P1157	Padrão de intercâmbio de dados definido pela IEEE.
IHE	Integrating the Healthcare Enterprise.
IP	Internet Protocol.
ISO	International Standards Organization.
MSH	Message Header Segment.
NEMA	National Electrical Manufacturers Association.
OBR	Order Information.
OBX	Observation Information.
ODBC	OLE for Data Base Controls.

OLE	Object Linking Embedding.
OO	Orientação a Objetos.
OPC	OLE (Object Linking & Embedding) for Process Control.
OPC AE	OPC Alarms & Events.
OPC DA	OPC Data Access.
OPC HAD	OPC Historian Data Access.
OPC UA	OPC Unified Architecture.
OPC XML-DA	OPC XML Data Access.
ORM	Mensagem de solicitação de material, exames ou procedimentos.
ORU	Observação Médica Não Solicitada.
OSI	Open System Interconnection.
OSISoft	OSISoft Incorporation.
PI	Plant Information – Sistema de Gerenciamento de Informações de Plantas Industriais da empresa OSISoft Incorporation.
PI SMT	PI System Management Tools.
PID	Patient identification.
PIMS	Process Information Management System.
RESP	Quantidade de Respirações.
SCADA	Supervisory Control and Data Acquisition.
SDT	Swinging Door Trending
SGBD	Sistemas de Gerenciamento de Bancos de Dados.
SI	Sistema de Informação.
SPO2	Pulse Oximeter Oxygen Saturation.
TCP	Transmission Control Protocol.
TCP/IP	Transmission Control Protocol/Internet Protocol.
TEMP	Temperatura.
TI	Tecnologia da Informação.
UTI	Unidade de Terapia Intensiva.
XML	Extensible Markup Language.

LISTA DE SÍMBOLOS

β - Coeficiente linear.

A - Amplitude máxima da curva senóide/cossenóide.

ω - Velocidade angular do sinal em radianos por segundo.

t - Tempo em segundos.

LISTA DE EQUAÇÕES

SUMÁRIO

1 INTRODUÇÃO.

Os Sistemas de Informação (SI) aplicados à área médica são cada vez mais importantes para a eficácia do atendimento de pacientes hospitalizados ou sob monitorização clínica. Dentre as diversas soluções de software disponibilizadas, destacam-se os sistemas voltados para o registro das informações relativas ao atendimento dispensado a pacientes, conhecidos como Eletronic Medical Record (EMR)[1], bem como os softwares embarcados em equipamentos que realizam o monitoramento em tempo real de sinais vitais de pacientes, tais como frequência cardíaca, temperatura corporal, pressão arterial, dentre outros sinais, seja de forma contínua ou pontual (HADZIC; DILLON; CHANG, 2007; ZAHARIA; DRAGAN, 2011).

O registro manual em meio eletrônico de dados médicos decorrentes da adoção de EMRs traz diversas vantagens em relação aos métodos "convencionais" de registro e acompanhamento das ações médicas realizadas, na medida em que o acesso aos dados é feito por meios eletrônicos, o que facilita o controle das interações medicamentosas realizadas e o gerenciamento do estado de saúde dos pacientes.

Entretanto, de acordo com Archer e Cocosila (2009), em muitos casos, os sistemas EMRs não são desenvolvidos com o objetivo de compartilhar com sistemas de terceiros os dados nele inseridos. Além disso, quando possuem alguma função de compartilhamento, a forma pela qual fazem o compartilhamento é geralmente específica e restrita à "linguagem" ou protocolo de comunicação desenvolvido pelo próprio fornecedor do sistema EMR. Este fato tem um impacto relevante na eficácia do atendimento médico, já que os dados de um mesmo paciente – atendido em diferentes momentos e tendo seus dados registrados em diferentes sistemas EMR –, não serão compartilháveis uns com os outros; quer os diferentes sistemas EMR estejam fisicamente instalados num mesmo local, quer estejam instalados em diferentes hospitais, clínicas, laboratórios, enfermarias, postos de saúde, dentre outros. Além disso, a falta de compartilhamento ou o compartilhamento precário (restrito ao protocolo de comunicação de cada sistema EMR), tem também um relevante impacto na evolução dos conhecimentos na área médica, na medida em que um histórico completo dos dados médicos e clínicos de um mesmo paciente ou de diferentes pacientes não fica facilmente (eletronicamente) acessível. Por esse motivo, alguns autores como Hadzic, Dillon e Chang (2007) apontam ser de suma importância

[1] Eletronic Medical Record (EMR) são sistemas que permitem o registro e monitoramento das interações medicamentosas, das alergias, das prescrições de exames, de agendamentos, dentre outras informações médicas de pacientes (HADZIC; DILLON; CHANG, 2007).

que sejam elaborados padrões que permitam a integração entre diferentes sistemas EMRs, de modo a viabilizar o compartilhamento de informações entre os vários profissionais da área médica, nos mais diversos hospitais, centros médicos, clínicas, laboratórios, enfermarias, postos de saúde, etc..

Os sistemas de monitoramento em tempo real de sinais vitais de pacientes também são uma realidade na maioria dos hospitais do mundo. Estes sistemas permitem o monitoramento em tempo real de diversas variáveis referentes ao estado clínico de pacientes, de forma contínua e simultânea. Tais variáveis são tão importantes para o acompanhamento das situações onde há risco de morte, como é o caso de pacientes internados em Unidades de Terapia Intensiva (UTIs), quanto para estudos e análises médicas destinadas ao diagnóstico e tratamento de pacientes, e bem assim para a realização de pesquisas posteriores em prol do avanço da Medicina (ROUSSELOT; DECOTIGNIE, 2009).

Dependendo da enfermidade e do estado clínico do paciente, diversas variáveis devem ser mensuradas e monitoradas, de forma contínua ou intervalar. Uma série de equipamentos de medição de sinais vitais são utilizados para mensurar e apresentar uma ou algumas das variáveis sob monitoração, geralmente utilizando-se de tecnologias específicas e proprietárias (ZAHARIA; DRAGAN, 2011). De fato, estes equipamentos, via de regra, não são projetados visando a sua utilização conjunta com sistemas informatizados de terceiros. Geralmente, cada fabricante implementa seu próprio protocolo de comunicação, o que dificulta o desenvolvimento de um sistema apto a comunicar-se com os diversos equipamentos heterogêneos (CHRIST et al., 2004).

Para que haja interoperabilidade entre estes equipamentos é necessário o uso de um padrão que permita a comunicação entre eles. De acordo com Bogdan et al. (2010), estes padrões tem como objetivo primordial definir um conjunto de especificações, incluindo palavras reservadas, formatos e sequência de transmissão, que permitam a livre comunicação e troca de dados, no intuito de eliminar ou reduzir as incompatibilidades.

Dentre os padrões para interoperabilidade entre sistemas e equipamentos médicos, o *Health Level Seven* (HL7) pode ser considerado um dos mais importantes. O HL7 possui um padrão específico de comunicação e um vocabulário aderente aos conceitos médicos que o torna apto a permitir o compartilhamento de dados entre sistemas clínicos, sistemas administrativos de hospitais e laboratórios, bem como entre equipamentos médicos (NAMLI; ALUC; DOGAC, 2009; SÁEZ et al., 2013).

Neste sentido, o uso exclusivo do HL7 no desenvolvimento de um sistema centralizado para registrar, visualizar e analisar os sinais vitais de pacientes monitorados através de equipamentos heterogêneos poderia até ser considerado adequado. Porém, devido ao grande volume de dados envolvido e a despeito do avanço da TI que vem construindo dispositivos com capacidades de armazenamento cada vez maiores, uma solução exclusivamente baseada no HL7 revela-se insuficiente, pois, em casos como este, surge a necessidade de se aplicar filtros de modo que apenas os dados considerados essenciais sejam armazenados. Corroborando com esta constatação, Cockshott et al. (1998) afirmam que o volume de dados armazenados em bases de dados tradicionais é dez ou mais vezes maior que o mínimo necessário.

Algumas soluções para o problema acima vêm sendo propostas. No ambiente computacional industrial já é largamente utilizado um tipo específico de sistema historiador de dados, comumente identificado no meio industrial por *Process Information Management System* (PIMS), em referência a todo sistema apto a capturar e armazenar, no menor espaço possível, e a consultar em tempo real, um número muito grande de medições de variáveis de interesse, como vazão, temperatura e pressão, dentre outros parâmetros considerados necessários ao monitoramento de plantas industriais (FRAS; DANG, 2004).

Similarmente ao que ocorre na área médica, na indústria de manufatura a questão da integração entre os diversos sistemas de monitoramento e controle de plantas industriais é um fator crítico, principalmente quando se trata do armazenamento de um grande volume de dados históricos (dados temporais). O padrão OLE *for Process Control* (OPC) vem sendo utilizado com sucesso na indústria de manufatura para a integração de vários tipos de fontes de dados, incluindo dispositivos, sensores e equipamentos industriais, bem como para a comunicação ou integração entre os diferentes tipos de bancos de dados por eles gerados, permitindo assim a integração de dados extraídos no nível operacional (produção), com as informações referentes às decisões necessárias à gestão estratégica do negócio (PATTLE; RAMISCH, 1997; SHIMANUKI, 1999).

Semelhantemente ao que já ocorre no setor industrial, o presente trabalho propõe o desenvolvimento de uma solução de software que permita integrar os dados de diferentes tipos de monitores de sinais vitais de pacientes, disponibilizados no padrão HL7, converte-los para o padrão OPC, constituindo assim um Servidor OPC, apto a enviar as séries históricas de sinais vitais para um PIMS , o que permitirá o armazenamento e a análise de grandes volumes de dados coletados por períodos extremamente longos (meses ou anos registrados minuto a minuto, por exemplo). Desta forma, sistemas de análise e de apoio a tomada de decisão médica

poderão ser desenvolvidos com o intuito de oferecer aos profissionais da área uma ferramenta tanto para a visualização e análise dos sinais vitais de pacientes durante todo o período em que estiverem hospitalizados sob monitoramento, quanto para compará-los com as medições de outras internações dos mesmos pacientes ou entre pacientes diferentes. Estes dados também poderão ser facilmente associados com as informações dos EMRs e consolidados em sistemas que detectam padrões de comportamento para que sejam encontradas correlações entre os sinais coletados e armazenados e os sinais típicos ou médios esperados.

1.1 MOTIVAÇÃO E JUSTIFICATIVA.

A coleta dos dados dos medidores de sinais vitais através do uso de padrões de interoperabilidade da área médica em conjunto com o padrão OPC, permitirá o armazenamento destes dados em um PIMS. Como explanado anteriormente, este tipo de sistema se caracteriza pela capacidade de armazenar grandes volumes de dados de forma rápida e eficiente, com baixa taxa de ocupação de espaço físico, bem como a realização de consultas otimizadas, permitindo assim o monitoramento em tempo real ou análise histórica das informações.

A decisão sobre qual procedimento médico deve ser adotado em cada caso, pode ser beneficiada pela solução proposta aqui, na medida em que os sinais vitais são analisados utilizando-se o maior tempo decorrido possível. Além disso, as medições de sinais vitais tais como frequência cardíaca, pressão arterial e temperatura podem ser cruzadas com informações sobre medicamentos utilizados, resultados de exames, dados sobre alergias, dentre outros, para auxiliar a equipe médica nos diagnósticos.

O sucesso da solução proposta permitirá também que este grande volume de dados seja analisado através de algoritmos especialmente concebidos com a finalidade de diagnosticar, com antecedência de horas ou minutos, a probabilidade de possíveis intercorrências que podem acometer um paciente em monitoramento e, deste modo, antecipar a realização das ações necessárias. Por exemplo, estudos realizados com base na identificação de padrões simultâneos de comportamento dos sinais de oxigenação sanguínea, de frequência cardíaca e de pressão arterial, entre outras variáveis, poderiam possibilitar a determinação da probabilidade de um paciente vir a ter uma intercorrência cardíaca com a antecedência de minutos ou horas.

No que diz respeito a produção de conhecimento, o estudo do padrão OPC como mecanismo de intercomunicação entre equipamentos e sistemas utilizados em plantas industriais é bem difundido no meio acadêmico-científico e tecnológico, porém na área médica ainda há um caminho longo a ser seguido. Há poucos trabalhos que lidam com o assunto e não foram encontradas patentes específicas sobre este tema[2]. Pretende-se, portanto, contribuir com resultados que venham a ser utilizados e expandidos científica e tecnologicamente.

1.2 OBJETIVOS.

O objetivo principal deste trabalho é contribuir para a integração dos sistemas médicos através do desenvolvimento de uma solução de software capaz de converter medições de sinais vitais no formato HL7 para o padrão OPC, possibilitando o armazenamento destes dados em um PIMS. Num sentido mais amplo, o objetivo deste trabalho é resolver concretamente uma limitação hoje existente na área médica que se manifesta na impossibilidade de integrar o longo histórico dos sinais vitais de pacientes com outros dados clínicos que estão armazenados em sistemas médicos, tais como os EMRs, com o intuito de viabilizar a análise e a tomada de decisão baseada no inter-relacionamento destas informações. O objetivo do trabalho é, assim, disponibilizar uma tecnologia prévia e necessária para a realização de futuras pesquisas visando o avanço dos sistemas médicos, de modo que os objetivos mais amplos deste trabalho estão além dos resultados aqui apresentados.

Estando disponíveis no formato OPC, os dados podem ser armazenados, visualizados e analisados através dos diversos sistemas, banco de dados e ferramentas que se comunicam através deste padrão, ao invés de ficarem restritos somente às ferramentas que se comunicam em formato HL7. Desta forma, a solução de software OPC, doravante identificada neste trabalho como Servidor HL7-OPC, permitirá que longas séries históricas de medições de sinais vitais possam ser armazenadas em um PIMS. Por conseguinte, o histórico destas medições poderá ser monitorado e analisado "em tempo real", local ou remotamente por profissionais da área médica. Além disso, estes dados podem ser acessados por softwares de diagnóstico, seja

[2] Foram realizadas pesquisas de patentes nos sites do INPI - Instituto Nacional de Propriedade Industrial (http://www.inpi.gov.br) e do *United States Patent and Trademark office* (http://www.uspto.gov)

em tempo real ou não, com o objetivo de estudar o efeito de ações realizadas e de prever possíveis ocorrências médicas nos pacientes monitorados.

É ainda um objetivo desse trabalho apresentar os resultados dos testes simulados de funcionamento e desempenho da solução de software desenvolvida, submetendo-a a situações limites quanto ao volume, concorrência e redução de tempos de acesso, explicitando a sua validade, eficácia e limitações, proporcionando assim uma visão crítica do que foi proposto e subsidiar eventuais trabalhos futuros.

1.3 METODOLOGIA.

Visando alcançar o objetivo principal de desenvolver a solução de software proposta, foi realizado, inicialmente, um estudo exaustivo sobre as especificações necessárias para o desenvolvimento de Servidores OPC e também sobre os padrões ou formatos de comunicação (HL7, DICOM, ASTM 1238.94, e outros) utilizados por diversos dispositivos e sistemas médicos.

Este estudo permitiu o entendimento dos principais conceitos que permeiam a arquitetura dos Servidores OPC e foi de extrema importância para a eficácia do desenvolvimento do Servidor HL7-OPC proposto neste trabalho. Já o estudo dos principais padrões de comunicação utilizados por dispositivos e sistemas na área médica permitiu a identificação da melhor alternativa para o desenvolvimento do algoritmo de captura dos dados no padrão selecionado (HL7).

Concluída a fase inicial, foram definidos os requisitos do software e a arquitetura da solução com o objetivo de nortear os passos seguintes. Foram pesquisadas e selecionadas duas bibliotecas de funções no intuito de abstrair o uso de funcionalidades básicas: uma para o desenvolvimento do Servidor HL7-OPC e outra para o desenvolvimento do mecanismo de comunicação com os dispositivos em formato HL7. Também foi definido um PIMS, bem como configurados os seus parâmetros de compressão e exceção, para permitir um eficiente armazenamento dos dados.

A partir da definição técnica e das bibliotecas de funções, foi desenvolvido o Servidor HL7-OPC com capacidade de obter mensagens no formato HL7, convertê-las para o padrão OPC e disponibilizá-las para leitura dos Clientes OPC. No caso específico da demonstração

pretendida com objetivos almejados por este trabalho, foi utilizado o Cliente OPC do PIMS selecionado.

Um gerador de sinais vitais em formato HL7, doravante denominado neste trabalho como GSV-HL7 foi também desenvolvido e utilizado com o objetivo de simular a comunicação via rede entre um conjunto de monitores de sinais vitais e o Servidor HL7-OPC. Cabe destacar que GSV-HL7 é um simulador confiável de dispositivos medidores de sinais vitais, uma vez que permite a configuração e geração de diversos sinais vitais, tais como batimento por minuto (BPM), eletrocardiograma (ECG), dentre outros, em intervalos de tempo determinados pelo usuário. O software também possibilita a configuração de parâmetros que são utilizados para a variar o valor das medições no tempo, oferecendo mais realismo ao trabalho.

Para realização dos testes para verificação de erros no funcionamento e desempenho, os softwares desenvolvidos tanto o Servidor HL7-OPC e o GSV-HL7, bem como o PIMS foram instalados em uma máquina virtual. O GSV-HL7 foi configurado para se comunicar com o Servidor HL7-OPC via *socket* TCP/IP, simulando uma rede. O Cliente OPC do PIMS também foi configurado para obter as medições disponibilizadas pelo Servidor HL7-OPC.

Por fim, para verificar o pleno funcionamento da solução em situações reais, sem que ocorram perdas, travamentos ou outros tipos de anomalias, foi simulado o envio de medições de sinais em sequência, a concorrência entre Clientes OPC, bem como a redução da granularidade do serviço, através da redução do tempo de varredura das variáveis provenientes das medições de sinais vitais simulados.

Com base na análise dos resultados dos testes realizados, foram comprovadas as possibilidades do uso do Servidor HL7-OPC proposto nesta pesquisa para as finalidades mais amplas apontadas, além de verificar e expor os seus limites de operação.

1.4 ESTRUTURA DO TRABALHO.

Este trabalho foi estruturado em nove capítulos. No Capítulo 1 são apresentados os assuntos introdutórios onde a pesquisa é contextualizada e são mostrados os principais fundamentos da dissertação: motivação, justificativa, objetivos e metodologia para o desenvolvimento da solução. No Capítulo 2, são descritos os conceitos referentes à integração e interoperabilidade, explicitando características e classificações propostas por diversos

autores. No Capítulo 3, a interoperabilidade entre sistemas industriais é descrita com ênfase no padrão OPC que é utilizado na solução de software deste trabalho. Em seguida, no Capítulo 4, são discutidas a arquitetura técnica, o funcionamento e algoritmos compressão de dados dos PIMS. Já no Capítulo 5, são descritos os principais padrões para interoperabilidade entre sistemas médicos. Neste Capítulo, o padrão HL7 é detalhado, objetivando a aquisição de conhecimentos para o desenvolvimento dos softwares propostos. No Capítulo 6 é explicitada a metodologia utilizada na construção do Servidor HL7-OPC e do GSV-HL7, incluindo linguagens de programação, documentação técnica necessária e produzida, componentes e bibliotecas utilizadas. A descrição detalhada dos testes realizados e resultados obtidos é realizada no Capítulo 7. No Capítulo 8, é apresentada a análise do trabalho realizado, dificuldades encontradas, limites de utilização da solução, bem como propostas para trabalhos futuros. Finalmente, no Capítulo 9, são apresentadas as referências utilizadas na pesquisa. O trabalho contém ainda cinco apêndices explicitando o código-fonte do software GSV-HL7, um exemplo de execução do GSV-HL7, o código fonte do Servidor HL7-OPC, um exemplo de execução do Servidor HL7-OPC e um extrato da planilha de evidência do primeiro teste de desempenho descrito no Capítulo 7.

2 INTEGRAÇÃO E INTEROPERABILIDADE ENTRE SISTEMAS

É de conhecimento geral que as organizações públicas e privadas têm implementado vários princípios e sistemas para melhorar a colaboração com seus parceiros externos e internos, permitindo compartilhar experiências, processos e ferramentas, e enfrentar os desafios decorrentes das necessidades de integração e interoperabilidade entre sistemas a fim de que possam, sobretudo, concentrar-se nos objetivos essenciais de negócio da organização ou empresa, sem perda de desempenho. A capacidade dos parceiros para interagir, ou seja, enviar e receber informações de forma legível entre todos eles, é um dos fatores chave para caracterizar e avaliar o desempenho global de um processo colaborativo (MALLEK; DACLIN; CHAPURLAT, 2012).

Ao mesmo tempo, o aumento exponencial dos dados disponibilizados pelos mais diversos dispositivos e o concomitante aumento do poder computacional, vêm permitindo que as organizações em todo o mundo possam aproveitar os benefícios da integração e da interoperabilidade e, bem assim, o aprimoramento dos processos de planejamento e tomada de decisões. Consequência deste crescimento é uma exigência cada vez maior para que os fluxos de informação se tornem cada vez mais orientados e organizados de modo a permitir a qualidade cada vez maior nos serviços, a redução de tarefas redundantes e a maior facilidade na realização de auditorias, dentre outros benefícios (DIPNALL et al., 2014).

O conceito de integração de informações entre organizações ou sistemas está diretamente relacionado com a interoperabilidade. Estes conceitos possuem diferenças sutis que precisam ser esclarecidas. De acordo com Chen, Doumeingts e Vernadat (2008), integração, dentre outras abordagens, é o processo de garantir o relacionamento entre entidades necessárias para alcançar um ou mais objetivos, podendo ser dividida em três níveis:

- Integração física: interligação entre dispositivos através de redes;
- Integração de aplicativos: integração entre aplicativos, sistemas e banco de dados;
- Integração de negócios: coordenação de funções para gerenciar, controlar e monitorar os processos de negócios.

Interoperabilidade, por outro lado, é a capacidade de dois sistemas entenderem um ao outro e compartilharem funcionalidades. Do ponto de vista da Tecnologia da Informação (TI), é a capacidade de dois sistemas heterogêneos funcionarem em conjunto e darem acesso aos seus

recursos de forma recíproca. Outros conceitos de interoperabilidade consideram seu aspecto multidimensional, o que amplia a complexidade do assunto. Por exemplo, a interoperabilidade também pode ser entendida como a capacidade promovida, mas não garantida, de um acordo de conformidade com um determinado conjunto de normas, que permite que equipamentos heterogêneos, geralmente construídos por vários fornecedores, possam trabalhar em conjunto em um ambiente de rede (CHEN; DOUMEINGTS; VERNADAT, 2008; REZAEI et al., 2014).

Panetto e Molina (2008) acrescentam que, diferentemente da interoperabilidade, a integração geralmente envolve o aspecto da dependência funcional. Enquanto sistemas interoperáveis podem funcionar de forma independente, uma solução que possui dois sistemas integrados não consegue executar uma funcionalidade se o fluxo de serviços for interrompido. Ou seja, sistemas integrados devem, necessariamente, ser interoperáveis, mas os sistemas interoperáveis não precisam ser integrados.

Alguns autores propõem que a interoperabilidade deve ser subdividida em níveis para sua melhor compreensão. Rezaei et al. (2014) sugerem a utilização dos níveis apresentados no Quadro 1.

Quadro 1 - Níveis de Interoperabilidade propostos por Rezaei et al. (2014).

Nível	Descrição
Interoperabilidade técnica	Associada aos componentes de hardware, software, sistemas e plataformas que permitem a comunicação máquina-a-máquina. Este tipo de interoperabilidade, muitas vezes, se concentra em protocolos de comunicação e infraestrutura necessária para os protocolos funcionarem.
Interoperabilidade sintática	Definida como a capacidade de trocar dados e é geralmente associada ao formato dos dados. Desta forma, as mensagens transferidas por protocolos de comunicação devem possuir uma sintaxe bem definida.
Interoperabilidade semântica	Capacidade de operar de acordo com entendimento semântico ou definição de conteúdo. Normalmente, lida com a interpretação que o ser humano dá ao conteúdo ou qual o significado comum daquela informação entre as pessoas.
Interoperabilidade organizacional	Capacidade das organizações para se comunicarem, utilizando de forma eficaz os diferentes tipos de sistemas de informação e suas infraestruturas, considerando também questões geográficas e culturais. Para ser bem-sucedida depende dos outros níveis anteriormente citados.

Fonte: (REZAEI et al., 2014).

Apesar dos esforços contínuos no avanço do domínio conceitual da interoperabilidade empresarial, Koussouris e Lampathaki (2011) afirmaram que a definição ainda necessitava de uma base científica, já que o nível alto de abstração dificulta aos pesquisadores e profissionais identificarem os problemas para oferecer soluções adequadas. Além disso, os níveis até então propostos não só se sobrepõem em muitos casos, mas também escondem importantes aspectos que lidam com tecnologias e métodos que abrangem todos os níveis. Neste contexto, os autores propuseram a categorização da interoperabilidade empresarial em níveis de granularidade que foram resumidas e organizadas por Rezaei et al (2014). A Figura 1 ilustra a proposta dos autores.

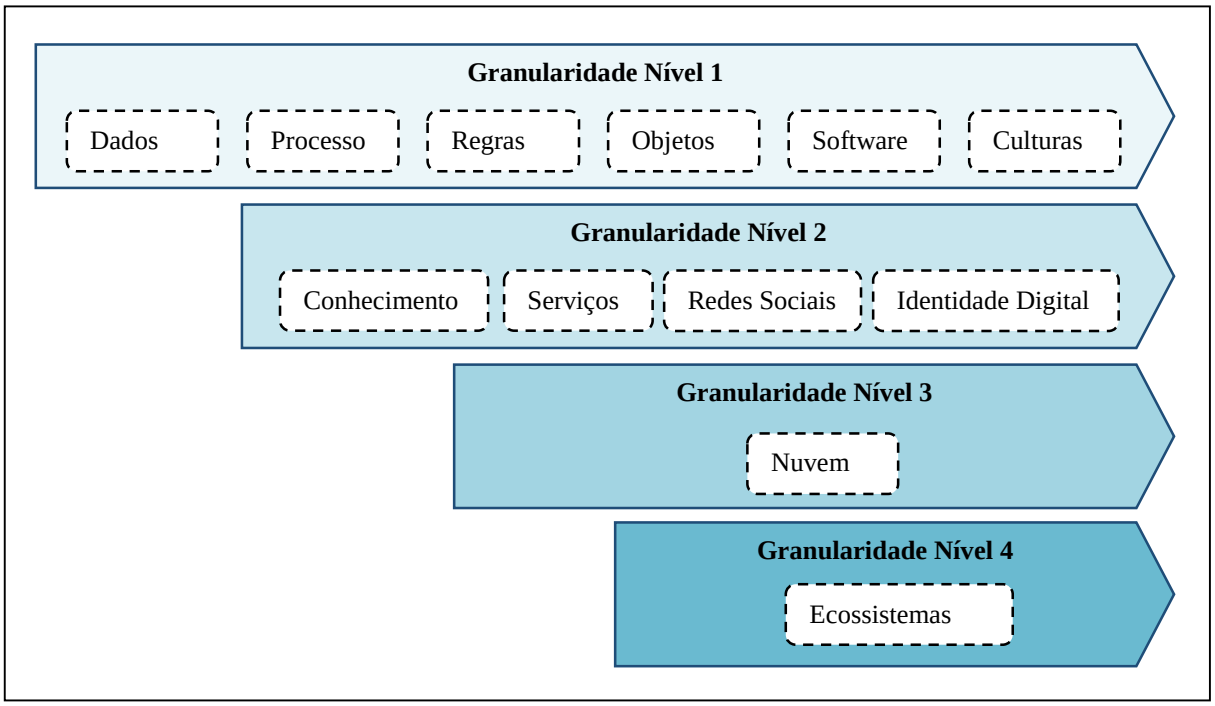


Figura 1: Níveis de Granularidade da Interoperabilidade Empresarial.
Fonte: Modificado de (KOUSSOURIS; LAMPATHAKI, 2011; REZAEI et al., 2014).

O primeiro nível de granularidade lida com aspectos relativos à interoperabilidade dos dados (documentos, multimídia ou recursos digitais), dos processos de negócio da empresa e das regras das transações de negócio, que garantem a segurança e a resolução de problemas entre as diferentes partes que se comunicam. Também diz respeito aos objetos interconectados através de redes, tais como dispositivos e componentes, à cultura organizacional (línguas e regionalismos), além dos softwares utilizados pela empresa (KOUSSOURIS; LAMPATHAKI, 2011; LUSK et al., 2006; MYKKÄNEN; TUOMAINEN, 2008).

O nível 2 de granularidade se concentra na interoperabilidade do conhecimento, ou seja, a habilidade de duas entidades de compartilharem seus ativos intelectuais, por exemplo através de repositórios de relatórios, artigos, patentes, comentários, invenções e etc. Este nível também retrata a capacidade da empresa de consumir serviços, através da internet, de um parceiro ou de um provedor qualquer, de utilizar redes sociais para colaboração e de utilizar identidades digitais em seus sistemas para autorizar transações internas e externas (KOUSSOURIS; LAMPATHAKI, 2011; PALFREY; GASSER, 2007).

Já os níveis 3 e 4 tratam, respectivamente, da capacidade dos serviços que são disponibilizados na nuvem interagirem entre si ou com outros sistemas, e da habilidade de diferentes setores de atividades empresariais (ecossistemas) de se interconectarem para o bem comum (KOUSSOURIS; LAMPATHAKI, 2011; SHETH; RANABAHU, 2010).

Independentemente do tipo de classificação ou do nível de granularidade, a interoperabilidade entre sistemas tem desafiado profissionais de TI e organizações. Consideráveis recursos vêm sendo investidos em iniciativas para alcançar eficácia nessas implementações, porém nem sempre os resultados são satisfatórios, causando o desperdício de recursos e desconfiança nas soluções.

2.1 O DESAFIO DA INTEGRAÇÃO

As organizações estão à procura de métodos realmente eficientes para a troca de informações e documentos, porém em muitas situações os procedimentos não são capazes de serem executados automaticamente e em formato eletrônico. Isto acontece, principalmente, devido a problemas de compatibilidade na representação de informação e aos métodos de integração adotados. Uma causa comum para esta dificuldade está relacionada aos investimentos anteriores em equipamentos e em softwares. Não é incomum que estes investimentos estejam focados na resolução de problemas específicos, desconsiderando, ao menos inicialmente, a necessidade de desenvolvimento de uma solução integrada. Esta abordagem focada na resolução de problemas específicos em geral causa a fragmentação de informações e inviabilizam ou dificultam sobremaneira a integração funcional posterior, devido à incompatibilidade de acesso e representação ou disponibilização de dados provenientes de

diferentes equipamentos, softwares e sistemas em um único sistema integrado (JARDIM-GONCALVES; GRILO; STEIGER-GARCAO, 2006).

Neste contexto, Avelar et al. (2014) acrescentam que uso da TI hoje é baseado em soluções comercias que propõem integrações complexas porque necessitam ser realizadas entre dispositivos que muitas vezes não podem interoperar com aqueles criados por diferentes fornecedores e mesmo entre dispositivos de um mesmo fabricante, comprometendo a escalabilidade[3] do sistema como um todo. Segundo os autores, um exemplo típico é o uso de diferentes dispositivos que, em geral, são de baixo custo, mas seguem normas específicas ou formatos proprietários quanto à estrutura ou linguagem com que os dados são disponibilizados e ou acessados.

No âmbito da automação industrial, os Sistemas de Supervisão e Aquisição de Dados, ou em inglês *Supervisory Control and Data Acquisition* (SCADA), são largamente utilizados para a medição, registro (armazenagem), monitoramento e controle de sistemas de manufatura, especialmente por empresas de eletricidade e de gás natural, água e esgoto, ferrovias, telecomunicações, dentre outras organizações (HONG; JIANHUA, 2006). Acoplado aos sistemas SCADA, é consagrado na indústria de manufatura a utilização de PIMS. Estes sistemas realizam a importante função de armazenar, em um banco de dados temporal, em intervalos que chegam a ser menores que um segundo, os valores de centenas ou milhares de variáveis registradas pelos mais diversos dispositivos (sensores de pressão, temperatura, vazão, etc.) (FRAS; DANG, 2004) .

Embora, originalmente, a maioria dos sistemas SCADA e seus PIMS acoplados utilizem formatos de comunicação não proprietários ou públicos para facilitar a interoperabilidade entre os dados provenientes de diferentes componentes e dispositivos, também entre eles há incompatibilidades que impedem ou dificultam que os dados registrados sejam utilizados por outros sistemas. Desta forma, apesar do avanço que apresentam em relação à área médica, também na área de automação da manufatura enfrenta-se o desafio de conceber padrões para que os dados coletados por componentes ou dispositivos diversos possam ser utilizados. Um dos padrões de maior sucesso, neste contexto, é o *OLE for Process Control* (OPC), cujo principal objetivo é oferecer uma estrutura padronizada de comunicação para resolver os

[3] Escalabilidade: propriedade de um sistema ou dispositivo que indica uma preparação prévia para sua evolução em termos de capacidade de processamento, armazenamento ou comunicação.

problemas incompatibilidade na troca de informações entre os mais heterogêneos dispositivos utilizados na área industrial (HONG; JIANHUA, 2006).

Desafio semelhante ocorre na TI para a área de médica ou de saúde. Sáez et al. (2013) afirmam que os sistemas implementados em organizações de saúde, tais como sistemas de apoio à decisão clínica, EMRs e sistemas de monitoramento de sinais vitais de pacientes também demandam um elevado grau de integração para serem efetivamente úteis, na prática.

Uma série de padrões de comunicação é utilizada para atingir este objetivo na área médica. Um dos mais difundidos é o *Health Level Seven* (HL7). Ele fornece uma base sintática e semântica para a interoperabilidade de dados clínicos e médicos entre sistemas desta natureza. A pesquisa realizada neste trabalho identificou que a maioria dos principais fabricantes de dispositivos leitores de sinais vitais, quando disponibilizam dados através de um padrão de comunicação não proprietário, utilizam também o padrão HL7. Por este motivo, foi o padrão escolhido para o desenvolvimento do Servidor HL7-OPC proposto por este trabalho.

Este trabalho realiza a junção dos conhecimentos sobre integração e interoperabilidade disponíveis na área de automação industrial e os desafios sobre integração e interoperabilidade na área médica. Neste contexto, o trabalho propõe o desenvolvimento de uma solução de software com o objetivo de obter dados de monitores de sinais vitais, formatados segundo o padrão HL7, e convertê-los para o padrão OPC, constituindo um Servidor HL7-OPC. Desta forma, as longas séries históricas de sinais vitais de diversos pacientes podem ser enviadas para sistemas que se comuniquem através do padrão OPC, como os PIMS. Uma vez armazenados em um PIMS, este grande volume de dados torna-se uma ferramenta de análise e pesquisa para os mais diversos profissionais da área médica.

3 INTEROPERABILIDADE ENTRE SISTEMAS INDUSTRIAIS

Nos primeiros anos de crescimento da automação industrial, um dos grandes problemas a ser resolvido foi a implementação de sistemas que promovessem a intercomunicação eficiente entre os diferentes tipos de hardware e software utilizados. Na medida em que as soluções para automação industrial se tornaram cada vez mais dependentes de softwares específicos providos por fornecedores com intenção de criar seus próprios padrões, surgiu a necessidade de se estabelecer uma estrutura que padronizasse a comunicação entre essas soluções (PATTLE; RAMISCH, 1997; ZHENG; NAKAGAWA, 2002).

Para tentar resolver tal situação, em meados dos anos 1990, um consórcio de empresas intitulado OPC Foundation definiu um padrão de comunicação baseado nos padrões *Object Linking & Embedding* (OLE) e *Distributed Common Object Model* (DCOM) [4], ambas pertencentes à Microsoft Corporation, chamada OLE *for Process Control* (OPC). O consórcio criou o OPC com o propósito de ser uma estrutura padrão para comunicação com várias fontes de dados, incluindo dispositivos industriais ou bancos de dados, permitindo assim a integração entre os dados extraídos do nível operacional (produção) com os sistemas de informação utilizados pela gestão do negócio (PATTLE; RAMISCH, 1997; SHIMANUKI, 1999).

Zheng e Nakagawa (2002) afirmam que, antes do OPC, a indústria de software utilizava drivers específicos para cada equipamento que se pretendia comunicar, incorrendo em diversos problemas:

- As aplicações necessitavam de um driver para cada dispositivo;
- Conflitos entre drivers de fornecedores diferentes;
- Mudanças no hardware poderiam causar problemas em alguns drivers;
- Dois softwares diferentes não poderiam acessar um dispositivo específico ao mesmo tempo, pois utilizavam drivers independentes.

Esta situação pode ser visualizada na Figura 2.

[4] A tecnologia OLE/DCOM permite encapsular componentes de software como interfaces padronizadas para serem utilizadas em outros softwares independentes (FONSECA, 2002).

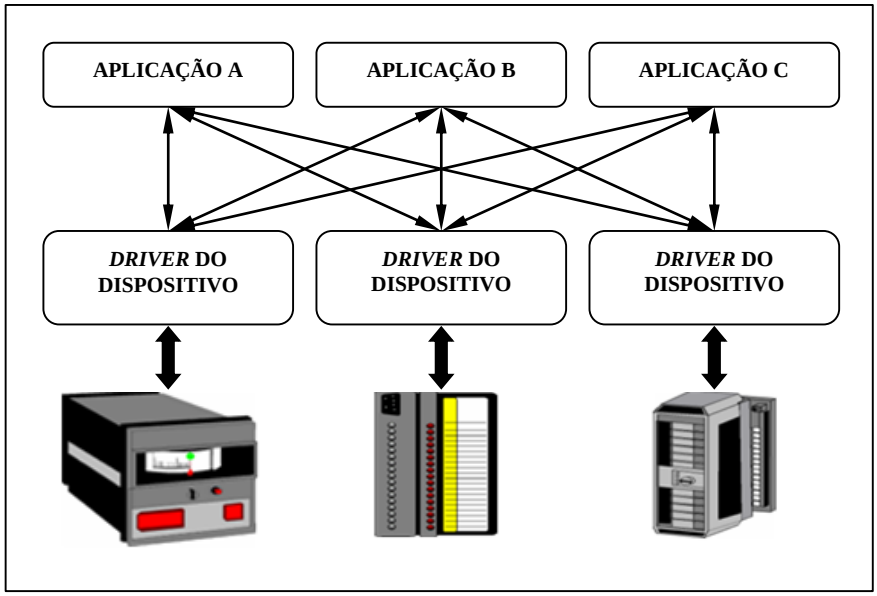


Figura 2: Arquitetura antes do OPC.
Fonte: Modificado de (PATTLE; RAMISCH, 1997; ZHENG; NAKAGAWA, 2002).

A indústria de hardware tentou resolver os problemas apontados acima construindo novos drivers, porém, como os protocolos de comunicação utilizados pelas aplicações clientes normalmente eram diferentes, as incompatibilidades continuaram. Com o OPC (Figura 3), os drivers dos dispositivos são interconectados por uma camada de software chamada Servidor OPC que padroniza o protocolo de comunicação com as diversas aplicações, independente do fornecedor do hardware (ZHENG; NAKAGAWA, 2002).

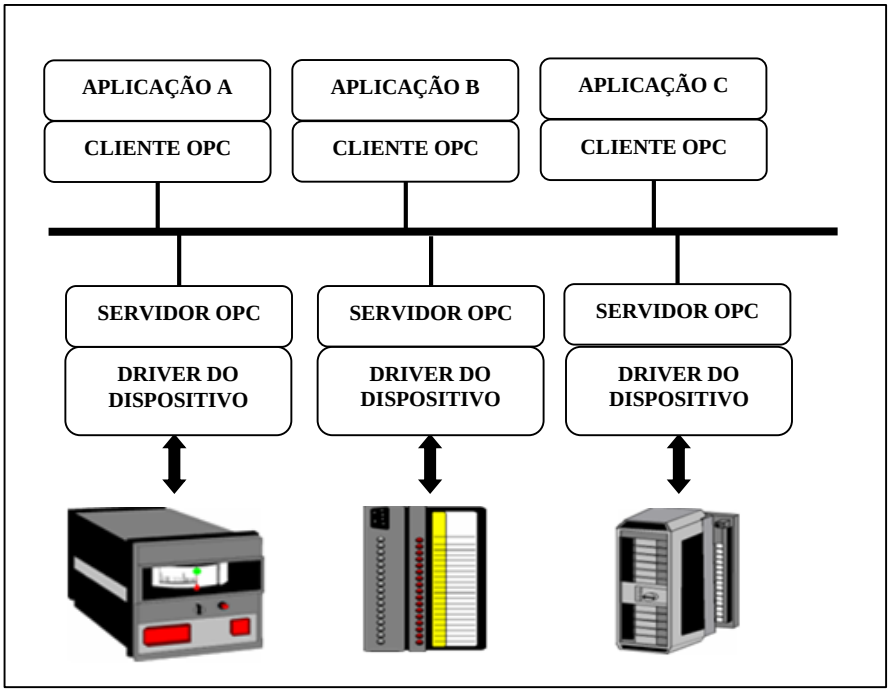


Figura 3: Arquitetura com o OPC.
Fonte: Modificado de (PATTLE; RAMISCH, 1997; ZHENG; NAKAGAWA, 2002).

Shimanuki (1999) afirma que a arquitetura OPC permite, então, que um fabricante desenvolva um servidor reutilizável e otimizado para que qualquer aplicação possa se comunicar com o seu hardware de forma eficiente.

Desde o seu início em 1995, a OPC Foundation elaborou uma série de especificações com o objetivo de descrever a arquitetura OPC para que os fornecedores pudessem utilizar o padrão em suas soluções. Um resumo das principais especificações pode ser visto no Quadro 2 (ZHIHAO LING; WEIBIN CHEN; JINSHOU YU, 2004).

Quadro 2 - Lista de Especificações dos OPCs.

Especificação	Descrição
OPC Data Access (OPC DA)	Utilizada para obter dados de equipamentos industriais de forma contínua e em tempo real e exibi-los em aplicações cliente. Atualmente em sua 3ª versão.
OPC Historical Data Access (OPC HDA)	Enquanto o OPC DA fornece acesso em tempo real e contínuo, o OPC HDA provê acesso uniforme a dados históricos já armazenados.
OPC Alarms & Events (OPC AE)	Provê notificações de alarmes de processos, ações de operadores, mensagens genéricas e de auditoria sob demanda.

Especificação	Descrição
OPC XML-Data Access (OPC XML-DA)	Primeira especificação OPC que se propôs a ser independente de plataforma. Objetivo era substituir a arquitetura COM/DCOM, utilizada somente com o sistema operacional Microsoft Windows, por outra baseada em XML e Web Services que poderia ser implementada em qualquer sistema operacional (MAHNKE et al., 2009).
*OPC Unified Architecture (*OPC *UA)*	Criada a partir dos conceitos de independência de plataforma do OPC XML-DA, porém com a intenção de manter o desempenho do OPC DA (MAHNKE et al., 2009).

Fonte: Modificado de (ZHIHAO LING; WEIBIN CHEN; JINSHOU YU, 2004).

Esta pesquisa enfoca o uso OPC DA como instrumento arquitetural da solução que realiza a conversão das informações recebidas dos monitores de sinais vitais de pacientes para sejam capturadas por um servidor OPC.

3.1 OPC DATA ACCESS (DA)

Como visto anteriormente, o OPC DA foi a primeira especificação definida pela OPC *Foundation* para ser um padrão de interoperabilidade entre softwares. O OPC DA define objetos COM/DCOM e um conjunto de interfaces[5] de servidores OPC, cuja principal função é permitir o acesso em tempo real aos dados gerados pelos dispositivos de campo, através da leitura, escrita e monitoramento das variáveis dispostas no processo (OPC FOUNDATION, 2003; ZHIHAO LING; WEIBIN CHEN; JINSHOU YU, 2004).

A especificação OPC DA, de acordo com OPC FOUNDATION (2003), detalha como devem ser desenvolvidos os componentes COM/DCOM do cliente e do servidor, independente do fornecedor que irá implementar a solução. Além disso, a especificação tem como objetivo facilitar o desenvolvimento de Servidores OPC nas linguagens de programação C e C++, porém as aplicações cliente podem utilizar outras linguagens.

[5] Interface: No contexto do padrão OPC, interfaces são funções e propriedades capazes de promover a comunicação entre o Servidor OPC e o Cliente OPC ou entre o Servidor OPC e o dispositivo.

O servidor OPC tem a função de realizar a aquisição dos dados e deixá-los disponíveis para os clientes OPC (FONSECA, 2002; OPC FOUNDATION, 2003). A arquitetura de um Servidor OPC é composta por:

- Um Objeto Servidor OPC: mantem informações sobre o servidor em si e também é utilizado como um repositório de objetos do tipo grupo;
- Objetos Grupo OPC: mantem informações próprias e proveem um mecanismo para organizar logicamente objetos do tipo Item OPC;
- Objetos Item OPC: representam as variáveis que serão disponibilizadas para leitura ou escrita. Um Item OPC, desta forma, pode ser um valor de temperatura, pressão, vazão ou qualquer outro.

Já o cliente OPC DA se conecta com o servidor OPC para realizar leitura, escrita ou monitoramento dos dados representados pelos Itens OPC. O cliente seleciona os Itens necessários e estabelece a comunicação com o servidor através da criação do Objeto Servidor OPC (MAHNKE; LEITNER; DAMM, 2009). O cliente OPC enxerga o servidor como uma estrutura de Grupos com seus respectivos Itens e um objeto que representa o Servidor propriamente dito. A relação entre cliente e servidor OPC está ilustrada na Figura 4.

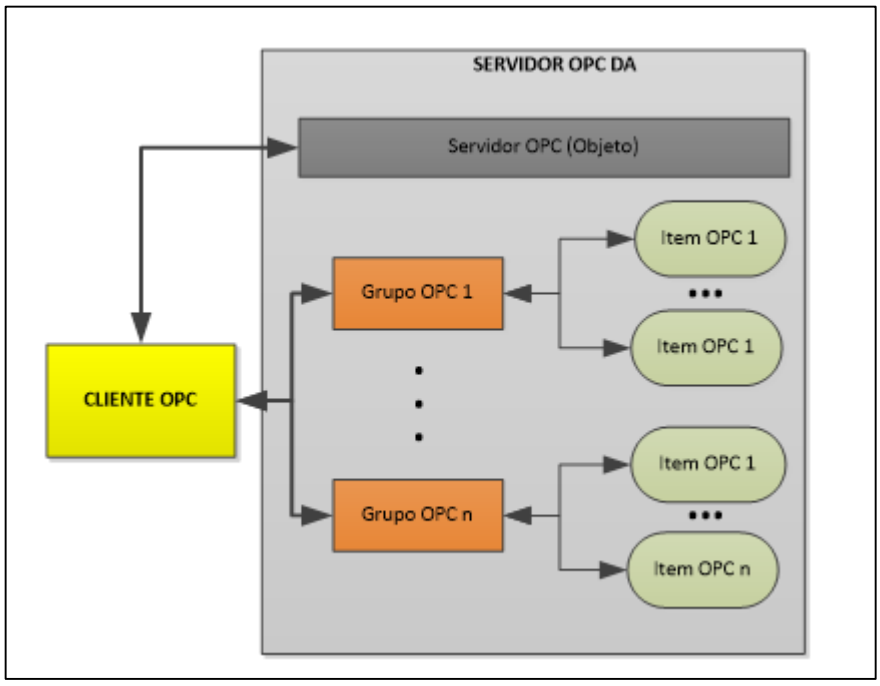


Figura 4: Relação entre Cliente e componentes do Servidor OPC.
Fonte: Modificado de (FONSECA, 2002).

A Figura 4 demonstra que os Grupos OPC são responsáveis por atender às solicitações do Cliente. Eles fornecem a infraestrutura básica para que os dados (itens) sejam organizados. Por exemplo, um grupo pode conter itens que serão visualizados em um relatório específico ou que possuírem as mesmas características. Apesar dos Grupos OPC pertencerem ao Servidor OPC, eles são criados através da requisição do cliente. Dentro de cada grupo, o cliente pode definir um ou mais Itens OPC (OPC FOUNDATION, 2003).

Os Itens OPC representam conexões com as fontes de dados (dispositivos) dentro do servidor. Não existe interface externa definida por um Item OPC. Ou seja, todo o acesso a um Item OPC é realizado através de um Grupo OPC que contém virtualmente o referido Item.

A especificação OPC DA afirma que cada Item deve possuir três parâmetros associados:

1) Valor: contém o valor da variável no dispositivo;
2) *Timestamp*: Data e hora da leitura do valor no dispositivo ou geração interna;
3) Qualidade: representa o estado daquele valor que foi lido do dispositivo, podendo ser definido de três formas:
    a. *Good*: quando o valor é considerado correto;
    b. *Bad*: quando não há link comunicação com o dispositivo;
    c. *Uncertain*: quando há link, porém o dispositivo está fora de operação.

3.2 ARQUITETURA E COMPONENTES DO OPC DA

O documento principal da especificação do OPC DA detalha dois tipos de interfaces, uma intitulada de *Custom*, que se propõe a permitir que aplicações Cliente OPC, desenvolvidas através da linguagem de programação C/C++, se comuniquem com o Servidor OPC, e a *Automation*, que lida com aplicações Clientes OPC desenvolvidas em outras linguagens de programação, tais como Visual Basic. Desta forma, o Servidor OPC realiza o canal de comunicação entre os Clientes OPC e o dispositivo (GUO; XIE; NI, 2012; OPC FOUNDATION, 2003). A Figura 5 abaixo, ilustra o exposto.

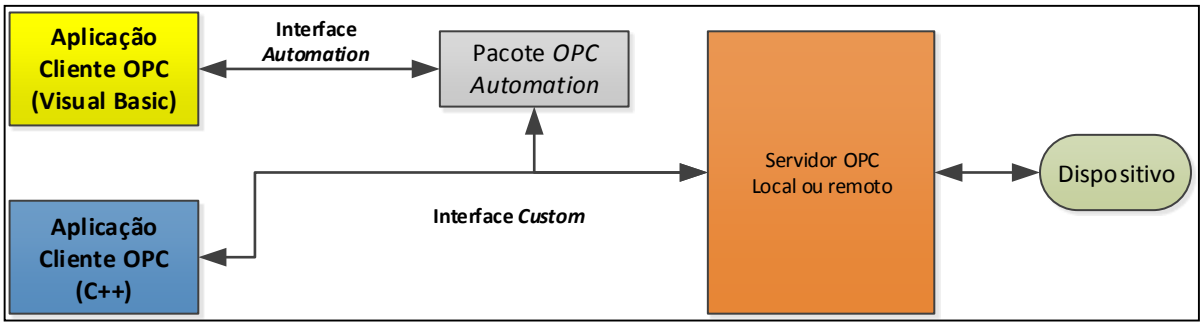


Figura 5: Método de Comunicação OPC
Fonte: Modificado de (GUO; XIE; NI, 2012; OPC FOUNDATION, 2003).

O Servidor OPC deve ser capaz de otimizar e consolidar acessos provenientes de mais de um Cliente, através de suas interfaces. Existem três formas de obter dados de um Servidor OPC (GUO; XIE; NI, 2012; SOUZA; SEIXAS FILHO; PENA, 1998):

- Modo síncrono: o Cliente lê ou escreve valores e atributos em uma única transação executada imediatamente pelo Servidor;
- Modo Assíncrono: o Cliente requisita uma leitura ou escrita e não necessita aguardar o retorno do Servidor. Quando o Servidor processa o pedido, ele retorna ao Cliente em uma transação de retorno independente;
- Modo de Subscrição: o Cliente requisita ao servidor o envio de mensagens de atualização de valores periodicamente ou quando acontecer alguma mudança de estado.

A OPC FOUNDATION (2003) esclarece que, como em todas as implementações de objetos COM/DCOM, a arquitetura do OPC é um modelo cliente-servidor onde o componente Servidor somente fornece uma interface para os objetos OPC e os gerencia. Ressalta também que os documentos disponibilizados para aqueles que desejem desenvolver um Servidor OPC são somente especificações, portanto determinam exclusivamente o comportamento esperado das interfaces para que o cliente OPC possa acessa-lo e não como estas interfaces devem ser desenvolvidas pelos fornecedores de soluções.

3.3 ESPECIFICAÇÃO DAS INTERFACES DOS OBJETOS

A especificação OPC DA detalha de forma criteriosa as interfaces disponibilizadas pelos objetos Servidor e Grupo com a intenção de deixar claro o que deve estar contido na implementação de um Servidor OPC. Como explanado anteriormente, os objetos Item não possuem interfaces externas, já que o acesso a este tipo de objeto é realizado através das interfaces do Grupos.

O objeto Servidor OPC é primeiro que se torna disponível para o Cliente. As interfaces definidas, basicamente, permitem controlar o estado do Servidor e administrar os Grupos. No Quadro 3 podem ser vistos os nome das principais interfaces deste objeto e suas descrições (OPC FOUNDATION, 2003).

Quadro 3 - Principais Interfaces do objeto Servidor OPC.

Interface	Descrição
IOPCServer	É a principal interface. Permite ao cliente adicionar e remover grupos, além de controlar o comportamento geral do Servidor.
IOPCBrowse	Permite ao cliente navegar pela configuração do servidor, além de obter as propriedades dos itens.
IOPCItemIO	Interface que provê a Clientes uma forma mais fácil de obter dados do OPC. Permite escrever valores de *timestamp* e qualidade nos servidores que possuem tal funcionalidade.

Fonte: Modificado de (OPC FOUNDATION, 2003).

Ainda de acordo com o mesmo autor, as interfaces de um Grupo OPC definem os mecanismos, principalmente, para solicitações de leitura/escrita, além da adição, remoção e alteração de Itens. A seguir o Quadro 4 resume as principais interfaces deste objeto:

Quadro 4 - Principais Interfaces do objeto Grupo OPC.

Interface	Descrição
IOPCGroupStateMgt	Permite ao cliente administrar as propriedades do grupo e seu estado geral do grupo.
IOPCSyncIO	Permite ao cliente requisitar leitura e escrita síncrona.
IOPCAsyncIO	Permite ao cliente requisitar leitura e escrita assíncrona.
IOPCItemMgt	Permite ao cliente adicionar, remover e alterar itens e controlar o comportamento dos itens no grupo.

Fonte: Modificado de (OPC FOUNDATION, 2003).

Como relatado anteriormente, o padrão OPC é utilizado largamente na indústria para permitir a interoperabilidade entre diversos sistemas. Dentre eles, há os PIMS, que se caracterizam por armazenar, ocupando o mínimo de espaço possível, os valores de variáveis registradas por diversos dispositivos industriais, tais como sensores de pressão, temperatura, vazão, dentre outros. A presente pesquisa propõe que um PIMS seja utilizado para inserir os valores de sinais vitais de pacientes em monitorização clínica.

4 SISTEMAS HISTORIADORES DE DADOS INDUSTRIAIS OU *PROCESS INFORMATION MANAGEMENT SYSTEMS*

Sistemas de controle de processos industriais de manufatura, sistemas automatizados de armazenamento, e outros sistemas de automação da produção em larga escala são, normalmente, orientados a dados. Esses sistemas em geral possuem muitos componentes distribuídos e heterogêneos que operam separadamente, porém de forma simultânea. Os dados gerados por estes componentes muitas vezes são usados para controlar automaticamente todo o sistema ou para permitir a tomada de decisões pela gerência (EREN, 2012).

A expansão do uso destes sistemas pelas organizações tem gerado grandes quantidades de dados que, para serem analisados, precisam ser armazenados, gerando um problema crescente de espaço. Diante do volume e da importância destes dados, a necessidade de utilizar um meio de armazenamento eficiente se torna cada vez mais presente (NETO et al., 2014). Os Sistemas Historiadores de Dados Industriais ou, em inglês, *Process Information Management Systems* (PIMS) vem sendo utilizados como solução para estas demandas.

Um PIMS pode ser entendido como uma coleção de módulos de software que obtém, armazena, contextualiza e agrega dados de forma eficiente com o propósito de realizar cálculos, garantir a qualidade, analisar estatísticas, monitorar e arquivar (EREN, 2012). A sua arquitetura típica é composta por uma base de dados que permite o armazenamento em tempo real e de um núcleo que realiza as principais funções do sistema, além das interfaces externas e módulos assessórios. A Figura 6 mostra simplificadamente a sua composição.

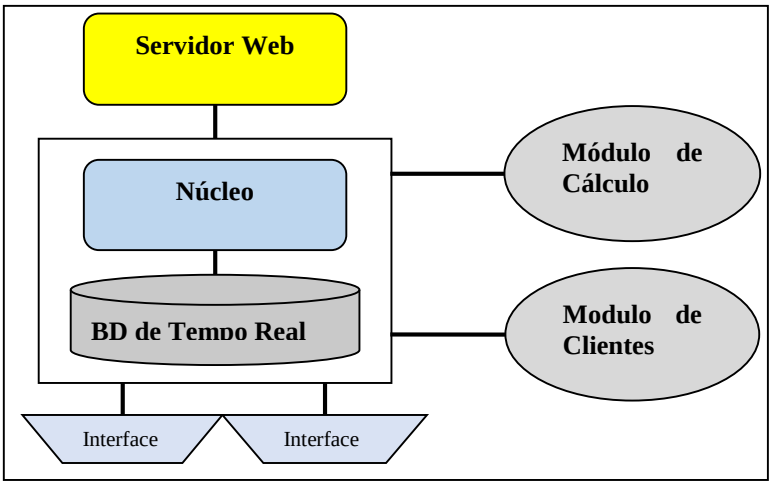


Figura 6: Arquitetura Simplificada de um PIMS.
Fonte: Modificado de (FRAS; DANG, 2004).

Fras e Dang (2004) afirmam que as interfaces utilizam drives para obtenção de dados de dispositivos e sistemas utilizados em plantas industriais tais como:

- Controladores Lógicos Programáveis (CLPs);
- Sistemas de Laboratório;
- Sistemas SCADA (Supervisory Control and Data Acquisition);
- Outros Sistemas e Bancos de Dados não industriais também podem ser utilizados como origem.

Os autores afirmam ainda que um PIMS é construído tipicamente para capturar um número muito grande de dados ao mesmo tempo (cerca de 50.000 pontos por segundo), utilizando o menor espaço possível de armazenamento. Além disso, os algoritmos de consulta são otimizados para que as requisições de usuários não sofram queda de desempenho com o aumento do volume de dados.

Os projetos de implantação de PIMS permitem que as empresas tenham à disposição informações que até então não estavam disponíveis ou que eram de difícil acesso em tempo real, permitindo assim a revisão das operações e resolução de problemas. O armazenamento centralizado de informações também pode ser considerado positivo neste caso, pois facilita a análise de desempenho das unidades produtivas (ARMSTRONG, 1998; SÁ BARRETTO, 2009).

Para viabilizar o armazenamento eficaz desses grandes volumes de dados, os PIMS utilizam pelo menos uma das técnicas de compressão e exceção antes de proceder o armazenamento no banco de dados. Estas duas técnicas complementares foram inicialmente apresentadas nos anos 1980 e se propõem a criar uma base de dados temporal com as informações de origem, considerando uma precisão predefinida. A técnica de exceção reduz o fluxo de dados provenientes dos sistemas de origem. Já a compressão é utilizada depois da exceção para minimizar o espaço em disco consumido (BARR, 1994).

4.1 EXCEÇÃO DE DADOS

Para reduzir o fluxo que chega ao banco de dados, esta técnica define limites para cada fonte de origem de dados. Na Figura 7 podem-se verificar os limites de exceção sendo

representados pelos valores $-d$ e d na ordenada do gráfico, além dos tempos $Tmin$ e $Tmax$ na abscissa (BARR, 1994).

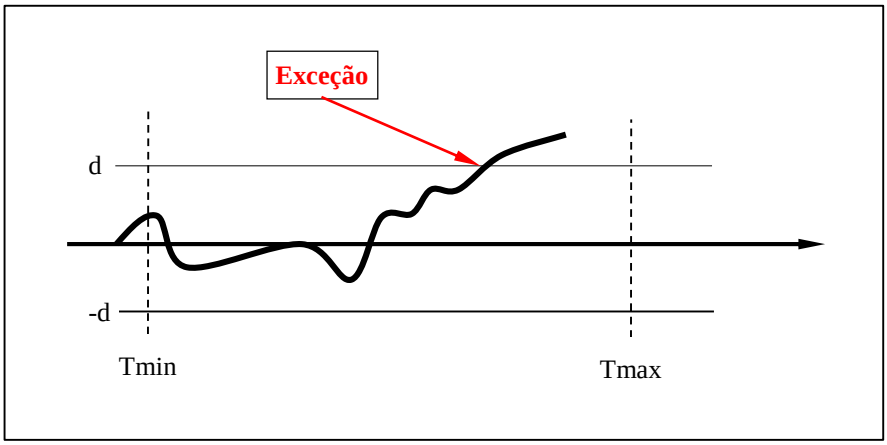


Figura 7: Exceção de Dados.
Fonte: Modificado de (BARR, 1994).

Os limites de exceção d (superior) e $-d$ (inferior) normalmente são definidos como um percentual onde os valores que estão sendo obtidos no fluxo de dados podem variar sem que haja prejuízo na análise. Já o $Tmin$ e o $Tmax$ representam o período em que os valores são analisados para determinar se houve a exceção ou não. O primeiro valor é sempre enviado para verificação pelo algoritmo de compressão. Se o valor seguinte estiver dentro dos limites de exceção (-d, d), ele é descartado. Do contrário, ele é enviado para compressão. Por fim, caso não haja um evento de exceção no tempo ($Tmax - Tmin$), um valor é enviado e um novo ciclo é realizado.

4.2 COMPRESSÃO DE DADOS

Após exceção de dados, o PIMS utiliza o processo de compressão para verificar se aquele valor que não foi descartado deve ser gravado fisicamente ou não no banco de dados, diferentemente do processo de compressão de arquivos onde todos dados são mantidos, apesar da compressão. Considerando o grande número de valores disponíveis para inserção durante um curto período de tempo, há a necessidade real de se minimizar o espaço em disco, mantendo

a precisão desejada (BARR, 1994). Dois dos algoritmos mais comuns são o *Boxcar/Backslope* (BCBS) e o *Swinging Door Trending* (SDT).

4.2.1 Algoritmo de compressão *Boxcar/Backslope*

O BCBS é um algoritmo de filtro em tempo real, ou seja, considerando um instante "n", quando um valor medido é disponibilizado, é decidido se deve ser armazenado ou não. O BCBS não contém mecanismos para pós-processar os dados com o objetivo de alterar os valores já armazenados (PETTERSSON; GUTMAN, 2004).

O algoritmo de BCBS, ilustrado na Figura 8, utiliza o conceito de desvio de compressão $(-d, d)$ e requer dois pontos armazenados para iniciar a sequência. Uma linha é traçada através destes dois pontos e duas bandas são desenhadas em cada lado da linha formando um paralelogramo (*boxcar*), cuja largura corresponde ao dobro do desvio de compressão. A cada novo valor, o paralelogramo é estendido, até que surja um fora faixa. Este valor é, então, armazenado e uma nova linha é desenhada entre o previamente armazenado e este último. A partir disso, é verificado se todos os pontos entre eles estão dentro do desvio de compressão, procedimento chamado de *backslope* (linhas pontilhadas). Ressalta-se que um valor de tempo máximo e mínimo, nos mesmos moldes do algoritmo de exceção, também é estabelecido neste caso (BARR, 1994; PETTERSSON; GUTMAN, 2004).

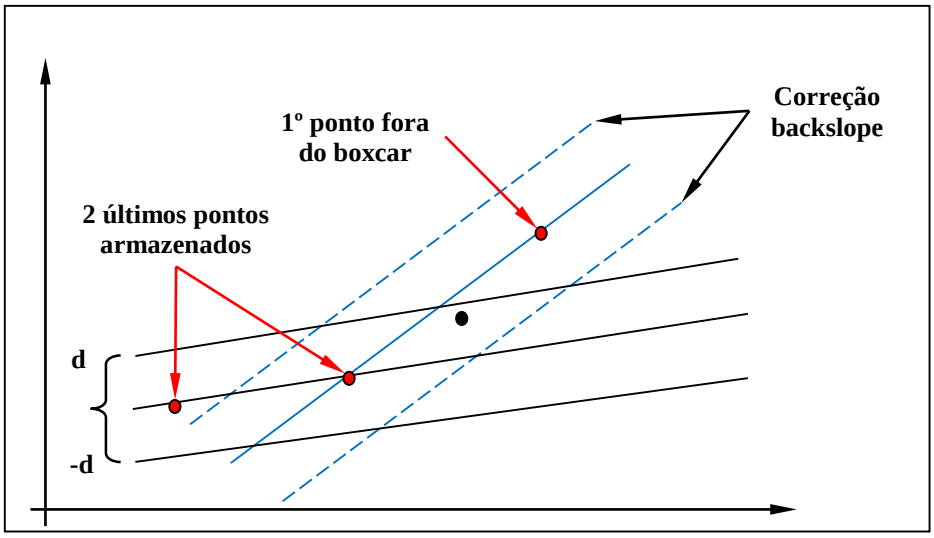


Figura 8: Algoritmo *Boxcar/Backslope*.
Fonte: Modificado de (BARR, 1994).

4.2.2 Algoritmo *Swinging Door Trending* (SDT)

Assim como o BCBS, este método tenta estabelecer um gráfico de tendência mais longo possível, ao determinar os pontos inicial e final de cada linha a partir valores a serem armazenados. No SDT, o ponto inicial é estabelecido como primeiro intervalo de compressão e o ponto final atua como inicial do próximo intervalo (XIAODONG et al., 2002; ZHANG et al., 2014).

O algoritmo SDT também prevê três parâmetros: Tempo mínimo, tempo máximo e desvio de compressão. A taxa de compressão é diretamente proporcional ao tempo máximo de compressão e ao desvio de compressão. O algoritmo define uma faixa em formato de paralelogramo entre o último valor armazenado e o atual, cuja altura é igual ao dobro do desvio de compressão. Caso um dos pontos presentes entre o último valor armazenado e o valor atual esteja fora do paralelogramo, todos os pontos são descartados, com exceção do penúltimo ponto recebido, que é armazenado no banco de dados por estar fora da faixa. Caso o tempo máximo seja ultrapassado desde que um valor foi armazenado, o último valor recebido é armazenado. A Figura 9 ilustra o processo em duas situações distintas. Na Figura 9a, à esquerda, os valores intermediários entre o último armazenado e o último recebido estão dentro da faixa, logo todos eles serão descartados. Já a Figura 9b apresenta a situação onde um dos valores intermediários fica fora da faixa. Neste caso, o penúltimo valor é armazenado (NETO et al., 2014).

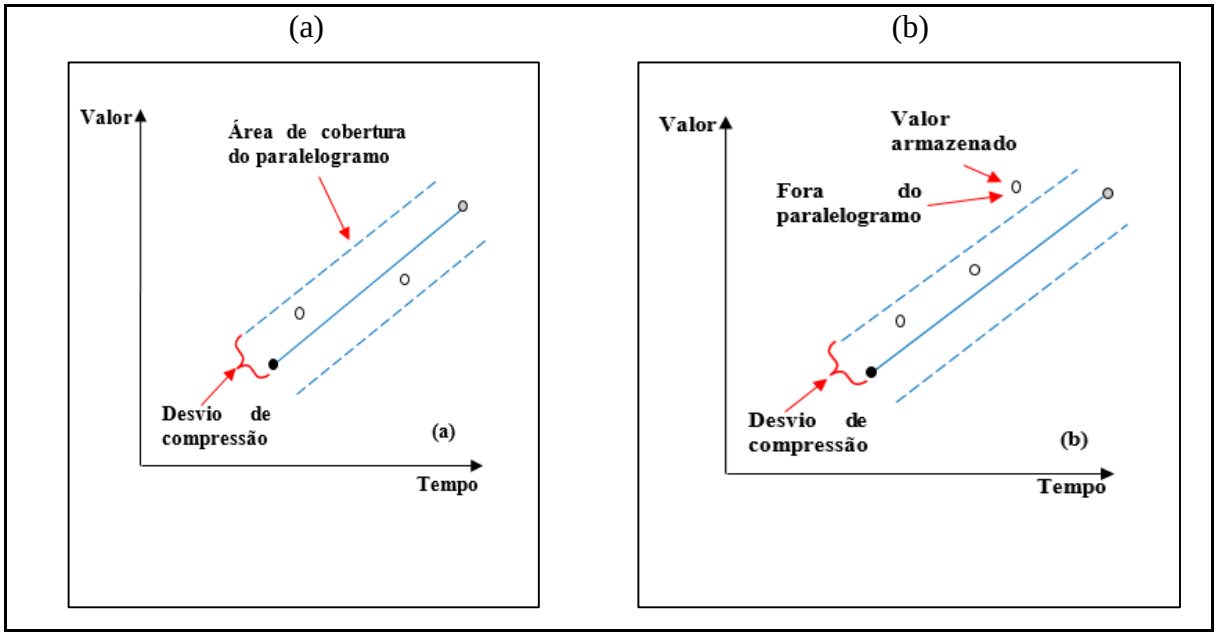


Figura 9: Algoritmo SDT.
Fonte: Adaptado de (NETO et al., 2014).

Os PIMS normalmente são utilizados para armazenar séries de dados industriais, porém existem outros tipos de séries que podem se beneficiar das características já descritas destes sistemas. O histórico dos sinais vitais de pacientes medidos através de monitores em UTIs representa uma série de dados grande e complexa que pode ser armazenada neste tipo de sistema. Esta pesquisa propõe o desenvolvimento de um servidor OPC para que seja viabilizada a obtenção desta série e o seu armazenamento no PIMS Plant Information (PI), que utiliza um algoritmo baseado no SDT para realizar a compressão dos dados (OSISOFT, 2009).

5 INTEROPERABILIDADE ENTRE SISTEMAS E DISPOSITIVOS MÉDICOS

Nos dias atuais considera-se impraticável possuir um ambiente hospitalar de médio ou grande porte sem o auxílio dos sistemas informatizados e equipamentos que realizem a medição de sinais vitais de pacientes.

O aumento do uso de TI no ambiente médico também vem demandando novos protocolos para facilitar a transferência de dados entre os diferentes hardwares e softwares, além do armazenamento destes dados em um formato padrão. Muitas organizações nos Estados Unidos, tais como a *National Electrical Manufacturers Association* (NEMA), a *American College of Radiology* (ACR), *Health Level Seven International* (HL7I) e *Integrating the Healthcare Enterprise* (IHE) vêm fomentando e especificando padrões para atender a estas demandas (BOGDAN et al., 2010).

No intuito de criar um padrão para armazenamento de imagens obtidas por tomografia, ressonância magnética, raios-X digital, ultrassom, dentre outros dispositivos de imagem médica, a ACR e a NEMA publicaram em 1985 a especificação do *Digital Imaging and Communications in Medicine* (DICOM). Várias versões foram lançadas desde então, tornando-se um dos padrões mais utilizados nesta indústria. O DICOM facilita a interoperabilidade entre equipamentos médicos de imagem através da padronização de protocolos que devem ser utilizados pelos dispositivos e da definição do formato das informações transmitidas através do protocolo. Para isso, o padrão propõe a associação de diversas informações do paciente e do exame de imagem realizado (BOGDAN et al., 2010; SAHU; VERMA, 2011).

No âmbito dos dispositivos médicos que realizam a medição dos sinais vitais de pacientes, os monitores que medem concomitantemente a frequência cardíaca, o eletrocardiograma (ECG), a temperatura corporal, a respiração e a saturação de oxigênio no sangue, são considerados os mais importantes. A heterogeneidade de fabricantes destes equipamentos dificulta consideravelmente o acesso simultâneo e automático aos dados gerados com o objetivo de oferecer aos profissionais da instituição uma visão geral das informações dos seus pacientes em uma interface única. Apesar de atualmente permitirem comunicação via cabo serial (RS232), ethernet e também wireless, em alguns casos, estes monitores sozinhos normalmente não fornecem uma solução *plug-and-play* para a interoperabilidade com outros sistemas, pois utilizam protocolos de comunicação proprietários ou pouco documentados (TANG; ZOU, 2010).

Para realizar a interoperabilidade entre tais dispositivos e Sistemas de Apoio à Decisão Clínica, Prontuário Eletrônico e Sistemas de Monitoramento de Sinais Vitais de Pacientes, alguns padrões têm sido criados desde o início da década de 1990. Destaca-se nesse cenário o padrão *Health Level Seven* (HL7) que, em conjunto com o DICOM, formam uma dupla praticamente obrigatória quando se pretende integrar sistemas heterogêneos da área médica. Enquanto o DICOM, como apresentado anteriormente, é dedicado principalmente à imagiologia médica, o HL7 abrange aspectos mais gerais do processamento de dados clínicos e de gestão, relacionados à transmissão de dados dos prontuários dos pacientes, arquivos e outros documentos associados (BOGDAN et al., 2010).

5.1 O PADRÃO HEALTH LEVEL SEVEN (HL7)

Nos últimos anos, o HL7 se tornou um dos principais padrões para compartilhamento de dados relativos ao atendimento médico. Gaion et al. (2010) afirmam que o HL7 é definido e divulgado pela organização *Health Level Seven International* (HL7I), que possui afiliados em diversos países do mundo. A organização é credenciada pela *International Standards Organization* (ISO) através do *Healthcare Information Technology Standards Panel* (HITSP) e tem como principal objetivo promover padrões para intercâmbio, gerenciamento e integração de dados que suportam o atendimento clínico de pacientes bem como o gerenciamento dos serviços de saúde.

O número sete do padrão se refere à camada de aplicação do modelo *Open System Interconnection* (OSI)[6]. Por se tratar de um protocolo de camada sete, o HL7 deve utilizar protocolos que forneçam estrutura de transporte para ser operacionalizado, sendo o mais comumente utilizado o TCP/IP (TANG; ZOU, 2010)

De acordo com Menezes (2011), o HL7 se tornou o padrão de interoperabilidade entre sistemas médicos mais utilizado do mundo. Para conquistar este espaço, a HL7I estreitou relações com as principais organizações de padronização existentes, principalmente com o *Technical Committee* 251 (TC) do *European Committee for Standardization* (CEN), que vem proporcionando modificações em seu padrão CEN 13606 com o objetivo de permitir o

[6] Modelo OSI é um padrão para comunicação entre sistemas e equipamentos heterogêneos que define sete camadas hierárquicas: física, enlace, rede, transporte, sessão, apresentação e aplicação.

intercâmbio de informações com o HL7. Além disso, o ISO/TC215, que trabalha com a padronização de informações de saúde e tecnologias de comunicação associadas para permitir interoperabilidade entre sistemas médicos, reconheceu o HL7 como um padrão internacional. Estas e outras relações conquistadas pela HL7I estão ilustradas na Figura 10.

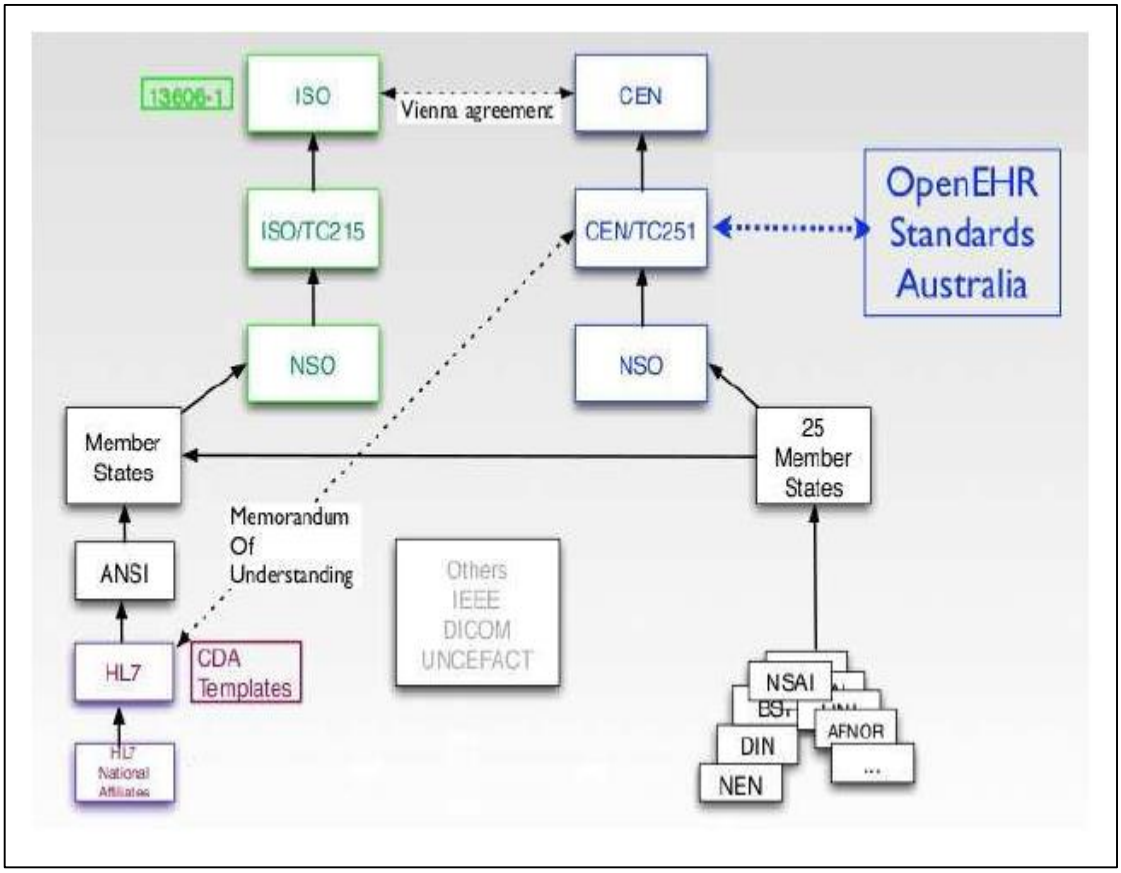


Figura 10: Relação entre o HL7 e outras organizações.
Fonte: (MENEZES, 2011).

Além dos relacionamentos já citados, a HL7I continua criando alianças com instituições responsáveis por outros padrões afins com o intuito de expandir a sua gama de atuação e atender o mais plenamente possível as necessidades da área médica no que se diz respeito à interoperabilidade entre sistemas. A seguir uma lista com algumas das principais alianças (HEALTH LEVEL SEVEN INTERNATIONAL, 2011):

- ACR/NEMA DICOM: A HL7I mantém uma relação estreita com o DICOM para que soluções que usam os dois padrões sejam mais eficientes;
- ASC X12: família de padrões que fornece especificação para o intercâmbio de dados para uma série de soluções em diversas áreas. As regras de codificação do

HL7 são modeladas segundo as diretrizes do X12. Em 2005, HL7 e X12 assinaram um acordo de longa duração para criar e estender padrões abrangentes para comunidade de saúde;

- ASTM 1238.94: padrão para registros e descrição de dados de laboratório. Um intercâmbio entre o HL7 e a ASTM permitiu pequenas mudanças na especificação ASTM para melhorar a compatibilidade com o padrão HL7.
- IEEE P1157 ("MEDIX"): o MEDIX é um padrão para intercâmbio de dados médicos assim como o HL7. Apesar de o objetivo final ser o mesmo, a HL7I mantém uma relação estreita com comitê MEDIX, almejando a melhoria constante dos dois padrões.

De acordo com Bogdan et al. (2010), o padrão HL7 produz um conjunto de especificações para permitir a livre comunicação e troca de dados entre softwares da área médica, no intuito de eliminar ou reduzir a incompatibilidade entre eles. Para alcançar este objetivo, as seguintes propostas foram estabelecidas:

- A norma deve apoiar o intercâmbio de informações entre sistemas desenvolvidos através de uma grande variedade de linguagens de programação em ambientes de desenvolvimento e de comunicação diversos;
- Deve atender ao mais alto grau de normatização, considerando a formatação dos seus elementos e atendendo as necessidades específicas de cada área médica;
- Deve lidar com as evoluções tecnológicas;
- A evolução do padrão deve ser realizada com base na experiência já adquirida na especificação de protocolos de comunicação existentes e novos;
- A norma não deve estar relacionada à arquitetura técnica específica de qualquer sistema médico.

O padrão HL7 é direcionado aos desenvolvedores de software e fabricantes de equipamentos médicos, no sentido de unificar a forma como as informações presentes em unidades médicas e instituições é transmitida ou armazenada, com base em um formato comum, que deve ser negociado com todas as partes envolvidas. Como apresentando previamente, existem outras normas dedicadas ao setor médico, cada uma com um domínio muito bem definido: farmácia, dispositivos médicos, imagens médicas e seguros. HL7 é dedicado ao processamento e gestão de dados administrativos e clínicos (BOGDAN et al., 2010). De acordo

com a especificação do padrão, o HL7 concentra-se nas seguintes áreas (HEALTH LEVEL SEVEN INTERNATIONAL, 2011):

- Admissão, alta e transferência de paciente (ADT);
- Consultas médicas;
- Recursos (quartos, camas, aparelhos, etc.);
- Agendamento de pacientes;
- Agendamento de procedimentos médicos e resultados;
- Ensaios e exames;
- Administração Financeira;
- Documentos médicos;
- Registos médicos;
- Tratamentos médicos.

Desde a sua primeira versão, o padrão HL7 vem evoluindo suas características com o objetivo de atender de forma mais abrangente possível à interoperabilidade entre sistemas médicos. A seguir um histórico da sua versão mais popular, a 2.x. Esta série aumentou consideravelmente o escopo de atuação do padrão, porém sempre mantendo o princípio de preservar a compatibilidade com as anteriores (BENSON, 2010).

Quadro 5 - Evolução do padrão HL7 v2.x

Período	Evento
1987	Estudos par simples troca de informações sobre ADT. Lançamento do HL7 v1.0
1988	Publicação do HL7 v2.0 estendendo o padrão para atuar no intercâmbio de requisições e relatórios para exames e tratamentos médicos, baseados no padrão E.1238.88 da American *Society of Testing and Materials* (ASTM).
1991	Publicação da versão 2.1 que se tornou a primeira a ser utilizada em larga escala.
1996	Publicação da versão 2.2 em fevereiro.
1997	Publicação da versão 2.3 em maio.
1999	Publicação da versão 2.3.1 em abril.
2000	Publicação da versão 2.4 em outubro.
2003	Publicação da versão 2.5 em julho.
2007	Publicação das versões 2.5.1 em fevereiro e 2.6 em outubro.
2011	Publicação da versão 2.7, a última até o momento.

Fonte: (BENDER; SARTIPI, 2013; BENSON, 2010; HEALTH LEVEL SEVEN INTERNATIONAL, 2011).

A versão 3 do padrão HL7 é a mais recente. Embora a numeração superior proponha uma sequência da versão 2.x, a sua elaboração começou ainda na década de 1990, portanto antes da publicação das versões mais atuais da 2.x. O HL7 v3 não foi uma melhoria incremental da v2 e sim uma ruptura arquitetural. A grande questão a ser resolvida era que a versão anterior

permitia múltiplas formas de desenvolver a mesma solução. Neste sentido, foram introduzidos um processo de desenvolvimento próprio e uma estrutura mais detalhada para os elementos da linguagem, oferecendo assim maior rigidez e congruência no desenvolvimento das soluções (BENDER; SARTIPI, 2013; BENSON, 2010).

Apesar do advento da versão 3.0, o HL7 2.x é, de acordo com a HL7I, o padrão de interoperabilidade entre sistemas médicos mais utilizados no mundo. Mais de 35 países possuem soluções baseadas na versão 2.x, além disso cerca de 95% das instituições dos Estados Unidos utilizam esta mesma versão. A HL7I ainda se compromete a manter o suporte e realizar atualizações para versão 2.x, promovendo assim a garantia de continuidade para instalações atuais e futuras (HEALTH LEVEL SEVEN INTERNATIONAL, 2014).

Independentemente da versão do HL7, considerando-se a grande variedade de aplicações envolvidas na atuação médica e a exigência da troca eficiente de mensagens contendo informações vitais entre elas, é óbvio que muitas dessas interfaces de comunicação se beneficiariam em usar uma abordagem padronizada. O HL7 resolve este problema fornecendo uma estrutura precisa para a realização comunicação, como pode ser visto no exemplo representando pela Figura 11. A aplicação médica que usa o padrão HL7 envia para os sistemas A e B uma mensagem HL7, gerada como resultado de algum evento médico: admitir paciente - A01, transferir paciente - A02, a alta do paciente - A03 e etc. Se o aplicativo receber as mensagens de confirmação de acordo com as normas HL7, então não haverá falha na comunicação. Desta forma, todas as informações recebidas serão interpretadas da maneira certa e uma resposta adequada será emitida de volta, tornando a troca de informações coerente e eficiente (BOGDAN et al., 2010).

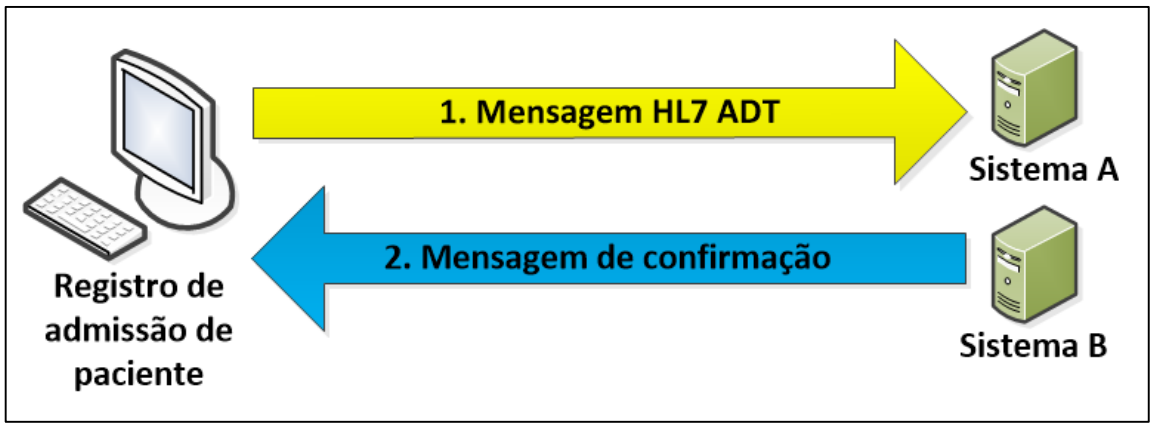


Figura 11: Exemplo de troca de mensagens HL7
Fonte: Adaptado de (BOGDAN et al., 2010; HEALTH LEVEL SEVEN INTERNATIONAL, 2011).

Ressalta-se que o padrão, na verdade, pressupõe que um evento ocorrido no mundo real criará a necessidade de transferência de informações entre sistemas. O documento de especificação do padrão HL7 detemina que este evento deve chamado de *trigger* ou gatilho (HEALTH LEVEL SEVEN INTERNATIONAL, 2011). No exemplo acima, o evento do registro do paciente gerou um *trigger* que enviou a mensagem ADT para as aplicações médicas que estão instaladas em outros equipamentos da rede de computadores.

Para entender bem o padrão, deve-se conhecer a sintaxe e os tipos de dados existentes nas mensagens. A sintaxe define a estrutura geral e como as partes são reconhecidas (BENSON, 2010). Na seção seguinte, serão detalhados os componentes dos diversos tipos de mensagens existentes no HL7 2.x. Esta versão é utilizada nesta pesquisa para simular o envio de dados (mensagens) de dispositivos de medição de sinais vitais para o Servidor HL7-OPC.

5.1.1 Mensagens HL7 2.x

Para o HEALTH LEVEL SEVEN INTERNATIONAL (2011), uma mensagem é uma unidade atômica de dados transferidos entre sistemas. Considerando o aspecto estrutural, uma mensagem HL7 v2.x é uma composição de segmentos ordenados, sendo que cada segmento possui campos que podem ser subdivididos em componentes e subcomponentes. No Quadro 6 pode ser visualizado um exemplo de mensagem HL7 que descreve a informação relativa a um relatório de exame bioquímico de um paciente chamado McDonald. O doutor Steves entregou a solicitação ao laboratório no dia 15/05/2010 às 14:30h, que examinou a creatinina do paciente e emitiu o relatório às 15:45h (LU et al., 2010).

Quadro 6 - Exemplo de mensagem HL7 v2.x.

MSH|^~\&|LIS|1021|HIS|1034|200705151727|ORU^R01|MSG00001|P|2.5<cr>

PID||0001||| **McDonald** ||19810401|M||| XISHIKUSt.^^Beijing^^ 100034^ CHN<cr>

OBR|1|P0001||200705151430|**201005151445**|10^ml||||**^Stevens** |||**201005151545**<cr>

OBX||NM|1000764^**CREATININE**^^^CREA|0001|1.2|mg/dL|.7-1.4||||F|||**201005151545**||284<cr>

Fonte: (LU et al., 2010).

Detalhando um pouco mais a estrutura geral e disposição das diferentes partes, cada mensagem HL7 v2.x é composta de segmentos em uma sequência especificada, que possuem campos também em uma sequência especificada. Os campos, por sua vez, são classificados por tipos de dados. Os tipos de dados são os blocos de construção dos campos e podem ser simples, com um único valor, ou complexos, com múltiplos componentes. Os componentes também têm seus tipos de dados, que podem ser simples ou complexos, levando aos subcomponentes (BENSON, 2010). A Figura 12 apresenta os conceitos chaves que formam uma mensagem HL7 desta versão.

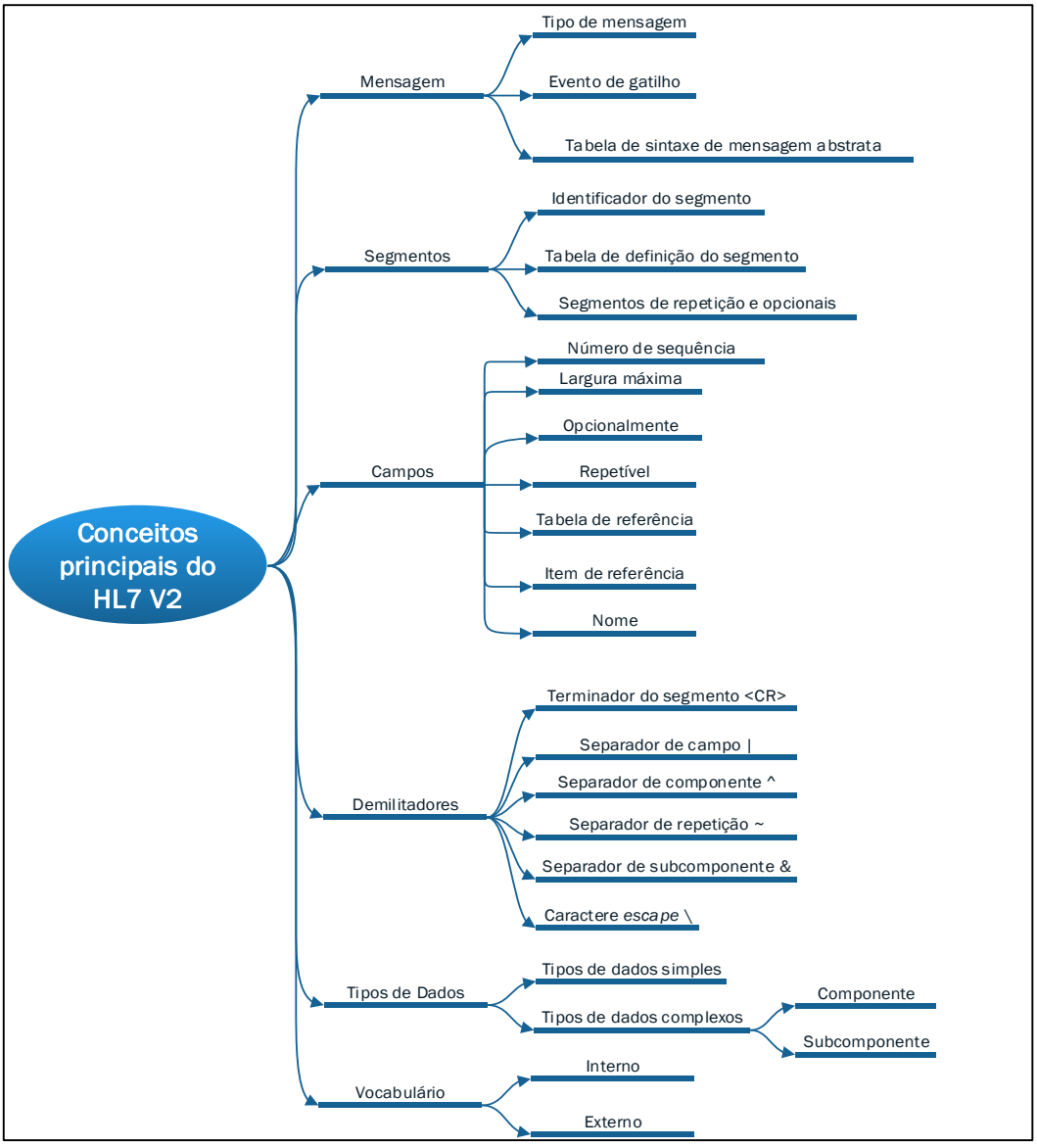


Figura 12: Conceitos chaves do HL7 v2.x
Fonte: Adaptado de (BENSON, 2010).

Como visto anteriormente, mensagens HL7 v2.x são enviadas como respostas a eventos. O nome da mensagem é derivado do tipo de mensagem e do evento de disparo. O tipo de mensagem é a categoria geral que mensagem pertence (BENSON, 2010; HEALTH LEVEL SEVEN INTERNATIONAL, 2011). No Quadro 7, há alguns tipos de mensagens previstas no HL7. A lista completa encontra-se disponível na especificação do padrão, de acordo com a versão desejada.

Quadro 7 - Tipos de mensagens HL7.

Código	Descrição
ACK	Mensagem de reconhecimento geral.
ADT	Mensagem de admissão, alta ou transferência.
ORM	Mensagem de solicitação (material, exames, procedimentos e etc.).
ORU	Resultado de alguma observação médica (pulsação, pressão arterial e etc.).

Fonte: Adaptado de (BENSON, 2010; HEALTH LEVEL SEVEN INTERNATIONAL, 2011).

O evento de disparo indica o que causou a geração da mensagem. Por exemplo, a mensagem do tipo ADT pode ser disparada por um evento A01 – Admissão/visita, A02 – transferência, A03 – Alta/fim de visita, A04 – registro do paciente, dentre outros. Retornando ao exemplo de mensagem HL7 visto no Quadro 6, a termo ORU^R01 indica que a mensagem é um resultado de observação médica não solicitada (LU et al., 2010).

O HEALTH LEVEL SEVEN INTERNATIONAL (2011) afirma que os segmentos que compõem uma mensagem são identificados por um código único de três caracteres chamado de *Segment ID*. A estrutura geral e o conteúdo permitido para cada mensagem são definidos em uma tabela que lista os segmentos na ordem em que devem ocorrer. No exemplo de mensagem do Quadro 6 há quatro segmentos:

- MSH – *Message Header Segment*
- PID – *Patient identification*
- OBR – *Order Information*
- OBX – *Observation Information*

Cada segmento possui seus campos específicos definidos no padrão e são separados pelo caractere delimitador "|". Por exemplo, o segmento PID é composto pelos campos *Patient ID, Patient Name, Sex Code*, dentre outros (LU et al., 2010):

Na mensagem de exemplo do Quadro 6, o campo *External Patient ID* possui valor igual a 0001 e *Patient Name* é igual a “McDonald”. Benson (2010) elaborou uma lista de segmentos mais utilizados no padrão HL7 v2.x e seus respectivos campos conforme se vê na Figura 13.

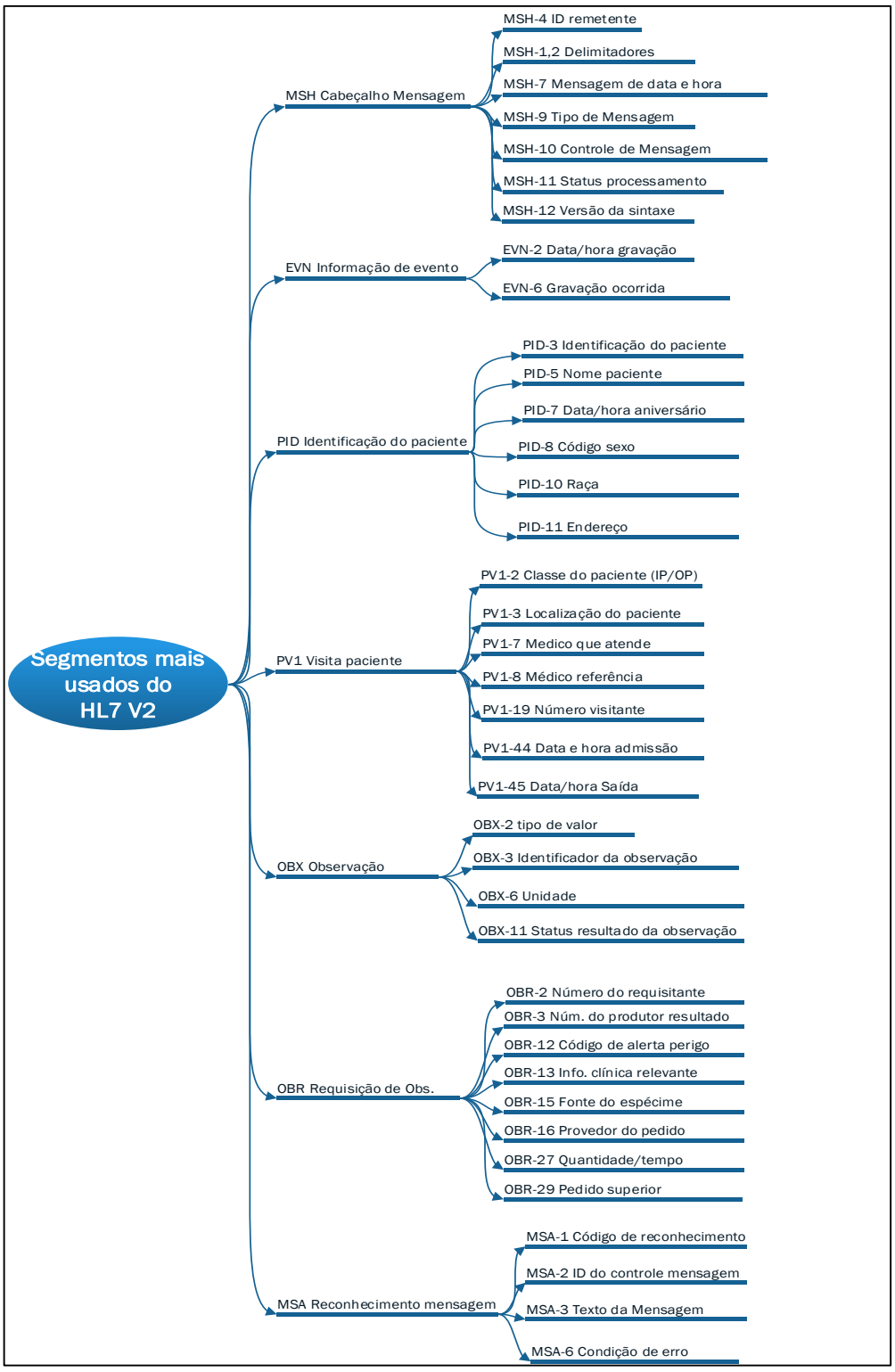


Figura 13: Segmentos mais utilizados no HL7 v2.x.
Fonte: Adaptado de (BENSON, 2010).

Os campos também podem ser subdivididos em componentes separados pelo caractere delimitador “^”. No exemplo do Quadro 6, o campo *Patient Address* do segmento PID é igual a “XISHIKUSt.^^Beijing^^ 100034^ CHN”. Percebe-se uma composição do endereço em rua, cidade, CEP e etc. (LU et al., 2010).

A cada versão do padrão os componentes e subcomponentes das mensagens HL7 são estendidos. Novos tipos de mensagens são introduzidos e componentes alterados, porém os antigos são sempre mantidos para razões de retro compatibilidade (HEALTH LEVEL SEVEN INTERNATIONAL, 2011).

6 IMPLEMENTAÇÃO DO SERVIDOR HL7-OPC

Nos capítulos anteriores deste trabalho, pôde-se demonstrar que a interoperabilidade entre sistemas heterogêneos permanece como um desafio enfrentado por empresas de diversos setores. As soluções de software encontradas por hospitais e clínicas perpassam pela utilização de padrões como o HL7. Já na indústria de manufatura o padrão OPC é uma alternativa concreta, principalmente, no que diz respeito à disponibilização de grandes volumes de dados de medições em tempo real.

Esta pesquisa utilizou estes dois padrões para simular um possível ambiente computacional real onde a série histórica de sinais vitais de pacientes, que são capturados por uma ou mais Centrais de Monitoramento de UTI ou enfermaria, é coletada por um Servidor OPC e armazenada em um PIMS. Considerando a capacidade dos PIMS de armazenar um grande volume de dados de forma eficiente, ou seja, ocupando o menor espaço possível, será possível capturar e armazenar, para análise em tempo real ou não, o histórico de medições de sinais vitais de pacientes durante todo o tempo em que estiveram sob monitoramento. Esta situação está ilustrada na Figura 14.

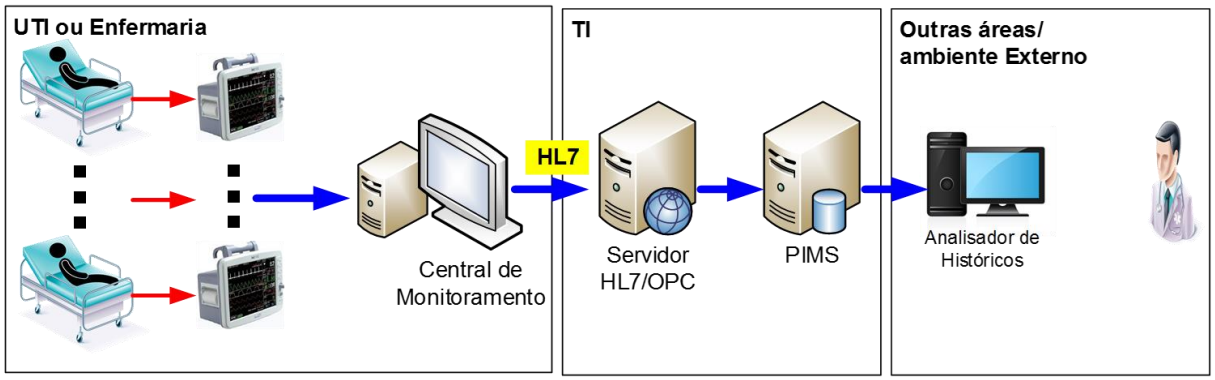


Figura 14: Situação real considerada.
Fonte: Elaboração própria.

Vale ressaltar que esta análise pode ser realizada através da simples visualização dos gráficos dos sinais vitais, bem como utilizando algoritmos de reconhecimento de padrão que demonstrem tendências. As diversas possibilidades serão discutidas em uma seção específica desta dissertação.

O experimento realizado possui a arquitetura ilustrada na Figura 15. Resumidamente, a solução é formada pelos seguintes componentes de software:

- Gerador de Sinais Vitais em HL7 (GSV-HL7): Este software gera uma série de medições de sinais vitais de pacientes, formata no padrão de mensagens HL7 v2.4 e encaminha para o servidor HL7/OPC, através de conexão *socket*.
- Servidor HL7/OPC: Recebe as mensagens HL7 via *socket*, converte para o padrão OPC e submete internamente ao servidor OPC, que disponibiliza o *timestamp* e valor de cada medição de sinal vital ao PIMS.
- PIMS Plant Information: Recebe as medições e armazena. Este sistema foi inserido para que fosse possível analisar os limites operacionais do experimento.

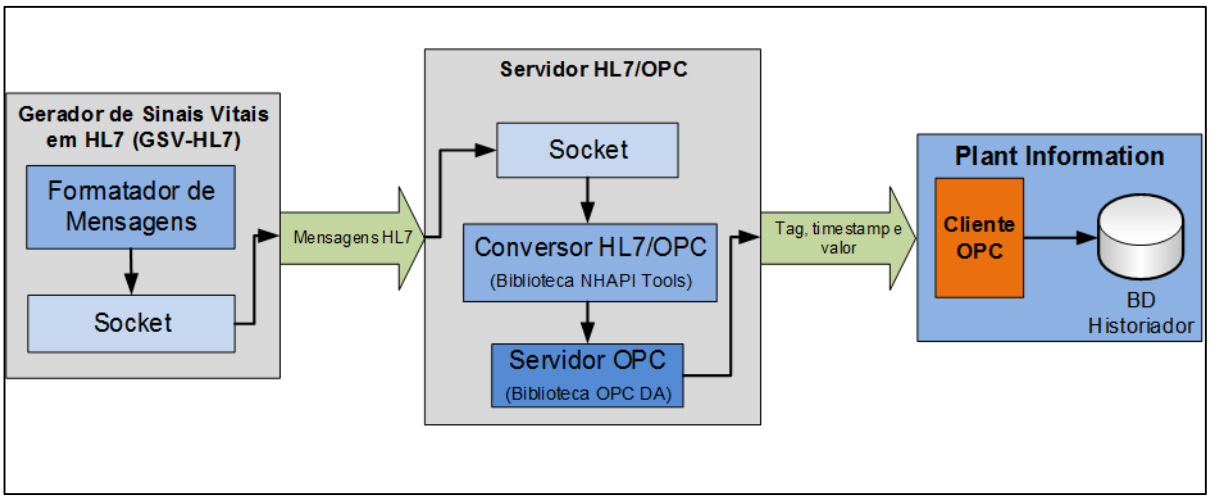


Figura 15: Arquitetura do experimento.
Fonte: Elaboração própria.

Tanto do GSV-HL7 quanto o Servidor HL7-OPC foram desenvolvidos, segundo os conceitos da programação Orientação a Objetos (OO), utilizando a linguagem C# .Net. O ambiente de programação, ou em inglês *Integrated Development Environment* (IDE), escolhido foi o Microsoft Visual Studio versão 2013. Nas próximas seções serão detalhados cada um dos componentes.

6.1 O GERADOR DE SINAIS VITAIS EM HL7 (GSV-HL7)

Para gerar a série de sinais vitais, simulando uma situação real, este componente de software inicia sua execução a partir da leitura de um arquivo texto contendo um modelo de mensagem HL7 2.4 do tipo ORU^R01 - observação médica não solicitada (Figura 16).

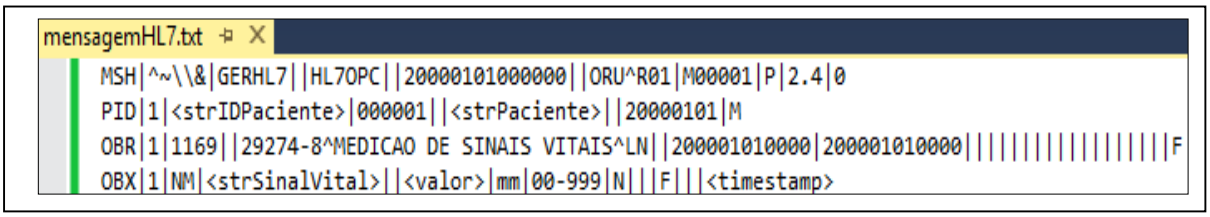


Figura 16: Modelo de mensagem HL7 2.x.
Fonte: Elaboração própria.

Este modelo é utilizado como parâmetro para gerar uma sequência de sinais, variando o paciente, o tipo de sinal vital, a hora (*timestamp*) e o valor medido. Os outros campos não são considerados nesta simulação, deste modo são mantidos valores padrão para cada um deles. No texto da mensagem, as variáveis são representadas pelos termos abaixo:

- <strIDPaciente>: Código de Identificação do paciente
- <strPaciente>: Nome do paciente
- <strSinalVital>: tipo de sinal vital
- <valor>: valor medido do sinal vital
- <*timestamp*>: Data e hora da medição

Somente os segmentos *Patient identification* (PID), *Order Information* (OBR), *Observation Information* (OBX) e *Message Header Segment (*MSG) foram utilizados no intuito de simplificar os dados contidos nas mensagens. Além do modelo de mensagem, o GSV-HL7 lê uma lista de parâmetros (Figura 17).

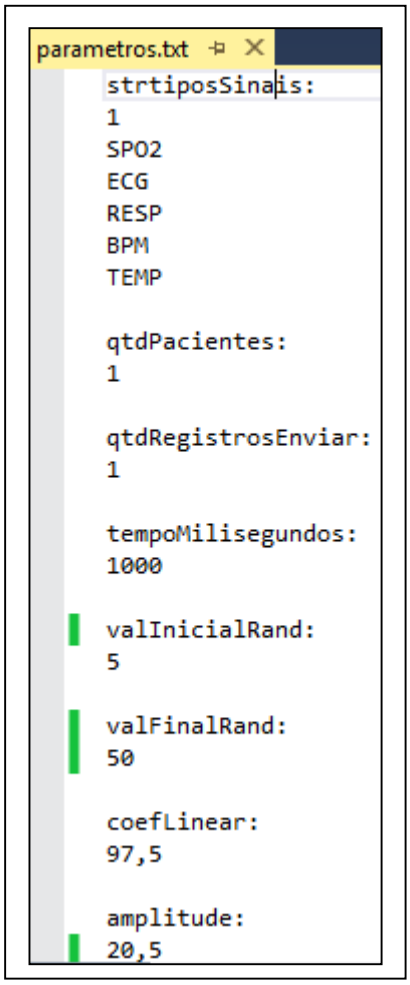


Figura 17: Principais parâmetros do GSV-HL7
Fonte: Elaboração própria.

Os parâmetros são necessários para configurar a quantidade e composição das mensagens, o formato da transmissão e como os valores medidos de sinais vitais serão calculados. Os principais parâmetros são:

- strtiposSinais: representa quais medições de sinais vitais serão geradas para cada paciente. Foram considerados, inicialmente, os seguintes sinais vitais, porém outro podem ser inseridos:
  - ECG: Eletrocardiograma
  - RESP: Respiração
  - SPO2: Pulse Oximeter Oxygen Saturation (Saturação de Oxigênio no Sangue)
  - BPM: Batimentos cardíacos
  - TEMP: Temperatura

- qtdPacientes: quantidade de pacientes que terão leituras de sinais vitais. Se, por exemplo, forem definidos os tipos de sinais BPM e TEMP e a quantidade de pacientes for 10, haverá 20 medições ao todo em cada grupo de mensagens;
- qtdRegistrosEnviar: quantidade de mensagens enviadas em uma conexão socket ao mesmo tempo;
- tempoMilissegundos: tempo de espera para envio de uma nova mensagem;
- valInicialRand: valor inicial de uma faixa randômica usada para gerar o próximo valor do ângulo a ser aplicado à função seno ou cosseno para calcular a medição;
- valFinalRand: valor final de uma faixa randômica usada para gerar o próximo valor do ângulo a ser aplicado à função seno ou cosseno para calcular a medição;
- coefLinear: coeficiente linear da curva seno ou cosseno utilizada para gerar o próximo valores das medições;
- amplitude: amplitude máxima da curva seno ou cosseno utilizada para gerar o próximo de valores das medições.

Para cada sinal vital é definido no início da execução se será utilizada a função seno ou cosseno. O ângulo começa em zero e aumenta a cada "tempoMilissegundos" de acordo com um valor randômico entre os parâmetros "valInicialRand" e "valFinalTag". O ângulo, então é submetido a função seno ou cosseno, como explicado anteriormente. Vale observar que estas funções foram arbitradas para definir a dinâmica do sinal, porém qualquer outro tipo de função poderia ser utilizado sem prejuízo ao experimento. A Equação 1 abaixo demonstra o cálculo do próximo valor da medição a ser enviada.

$$v = \beta + A \cdot \sin(\omega \cdot t) \quad ou \quad v = \beta + A \cdot \cos(\omega \cdot t)$$

Onde:

$\beta =$ Coeficiente linear.

$A =$ Amplitude máxima da curva senóide/cossenóide.

$\omega =$ Velocidade angular do sinal em radianos por segundo.

$t =$ Tempo em segundos.

Equação 1 - Cálculo do próximo valor da medição do sinal vital
Fonte: Elaboração própria.

A Figura 18 representa um exemplo de uma senóide gerada a partir dos seguintes parâmetros: Amplitude, 20,5; Coeficiente Linear, 97,5 e valores randômicos do próximo ângulo variando a cada 500 milissegundos entre 5 e 50.

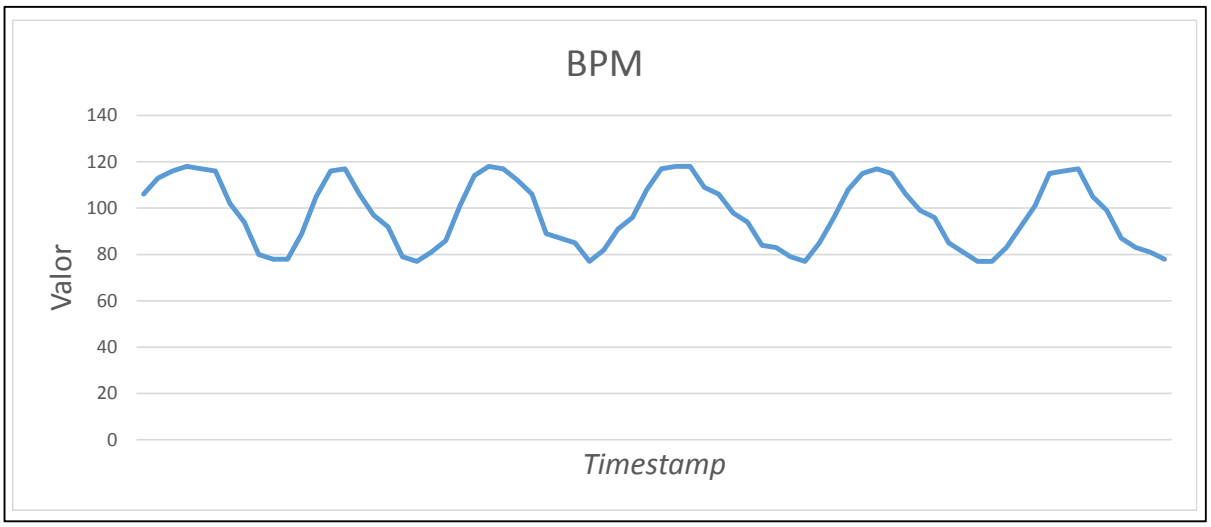


Figura 18: Exemplo de Gráfico de Sinais Vitais.
Fonte: Elaboração própria.

O software é composto pela classe GeradorMsgHL7 ilustrada na Figura 19. Basicamente, a ordem lógica de execução começa com a leitura dos parâmetros, método “leParametros”. Na sequência, são geradas as mensagens HL7, através do método “geraStringsHL7”. A partir deste ponto e de forma cíclica, o método “enviaMensagemHL7” encaminha para o servidor HL7/OPC as mensagens via conexão socket, variando sempre o *timestamp*, o valor da medição e o paciente. O código fonte dos principais métodos deste componente encontra-se no APÊNDICE A.

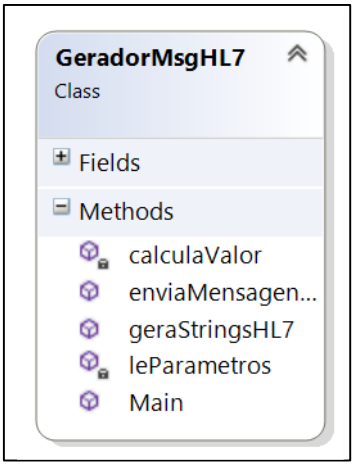


Figura 19: Classe GeradorMsgHL7.
Fonte: Elaboração própria.

Um exemplo de execução onde o GSV-HL7 se conecta com o Servidor HL7/OPC e envia dois valores de sinais vitais dos tipos RESP e ECG referentes a dois pacientes pode ser visualizado no APÊNDICE B.

6.2 O SERVIDOR HL7/OPC

Este é o principal componente de software do experimento, pois realiza a conversão da mensagem HL7 proveniente do GSV-HL7 para o padrão OPC e a disponibiliza, através de uma interface, para leitura dos clientes OPC.

A execução inicia com a leitura de um arquivo de parâmetros (Figura 20), onde são definidos os tipos de sinais vitais que serão recebidos e a quantidade máxima de pacientes envolvidos. Estes parâmetros são usados para gerar os nomes dos Itens OPC que poderão ser lidos pelos Clientes OPC que vierem a se conectar com este Servidor. O nome dos Itens OPC é formado pelo paciente seguido do sinal vital, gerando nomes do tipo: PAC0000-ECG, PAC0001-BPM e etc. Observa-se que Itens OPC são comumente denominados *tags* no ambiente industrial. De acordo Silveira e Santos (1998), *tag* é o termo utilizado para denominar uma variável que representa um valor medido por um instrumento existente no processo produtivo de uma indústria.

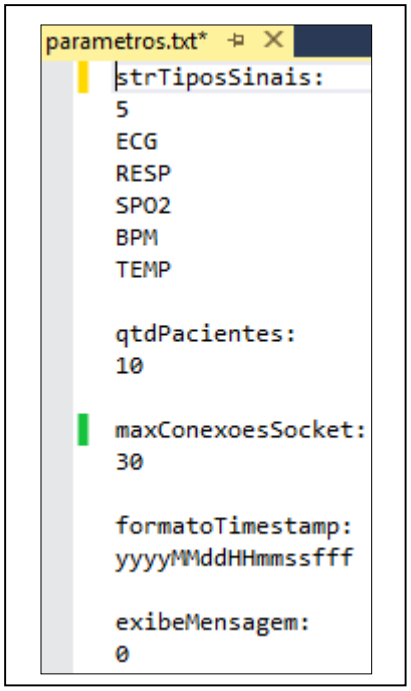


Figura 20: Principais parâmetros de configuração do Servidor HL7/OPC.
Fonte: Elaboração própria.

Neste arquivo também são definidos o número máximo de conexões *socket* suportados ao mesmo tempo, o formato do *timestamp*, que deve ser igual ao do GSV-HL7, além de outro parâmetro usado para exibir na tela ou não a mensagem recebida.

Este componente de software possui sete classes que devem ser levadas em consideração para o pleno entendimento das suas funcionalidades. A dependência entre estas classes está ilustrada na Figura 21. A principal delas é a HL7_OPC, onde é realizada a captura das mensagens HL7, conversão e escrita na interface OPC. As restantes são responsáveis pela criação do servidor OPC em si e são provenientes da biblioteca *OPC Classic Development Toolkit*[7]. Observa-se que a classe Servidor OPC desta biblioteca foi customizada para atender aos requisitos deste projeto. O código fonte das classes HL7_OPC e ServidorOPC pode ser visualizado no APÊNDICE C.

[7] Biblioteca de classes disponibilizada pela empresa *Softing AG*, através de seu site http://industrial.softing.com. A única restrição adotada pela empresa é o tempo de execução do servidor OPC que deve ser de no máximo 90 minutos a cada vez que for utilizado.

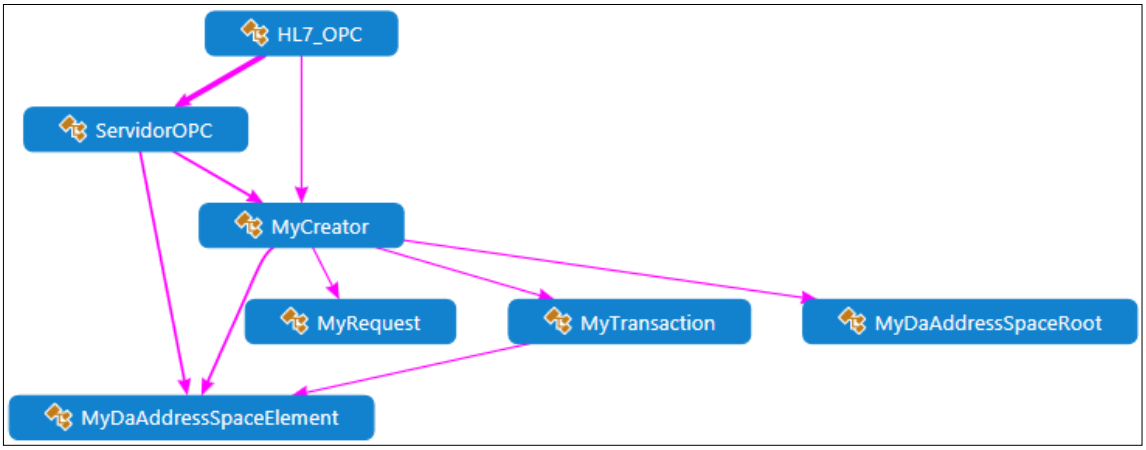


Figura 21: Dependência entre as classes do Servidor HL7/OPC
Fonte: Elaboração própria.

As classes HL7_OPC e Servidor OPC (Figura 22), em linhas gerais, executam os seguintes passos para obter as mensagens HL7 e disponibilizar ao Servidor OPC:

1) Após a leitura dos parâmetros de configuração, o método "iniciaServidorOPC" cria a instância do objeto Servidor OPC, faz o registro do servidor no Windows e inicia a thread que controlará a execução;
2) A partir deste momento, o método "obtemHL7ConverteOPC" estabelece os parâmetros, inicia conexão socket e permanece em estado de espera de uma conexão a ser realizada pelo GSV-HL7;
3) Quando a conexão for estabelecida, as mensagens HL7 enviadas são separadas e cada uma é submetida ao método "converteHL7Tag". Este método utiliza a biblioteca classes *NHAPI Tools*[8] para obter o nome da *tag*, o valor da medição e o *timestamp;*
4) Na sequência, o método "AtualizaValorTag" escreve na interface do servidor OPC o valor e o *timestamp* para aquela tag. Estes passos são executados para todas as mensagens HL7 obtidas naquela conexão;
5) Antes de iniciar o novo ciclo, a interface do servidor OPC bloqueia a thread atual até receber um sinal de positivo do sistema operacional Windows para continuar. Este mecanismo garante a execução correta do processo.

[8] Biblioteca de classes *open source* que disponibiliza uma série de métodos para manipulação de mensagens HL7. Pode ser obtida através do endereço: https://www.dib0.nl/code/504-nhapitools-1-1-release.

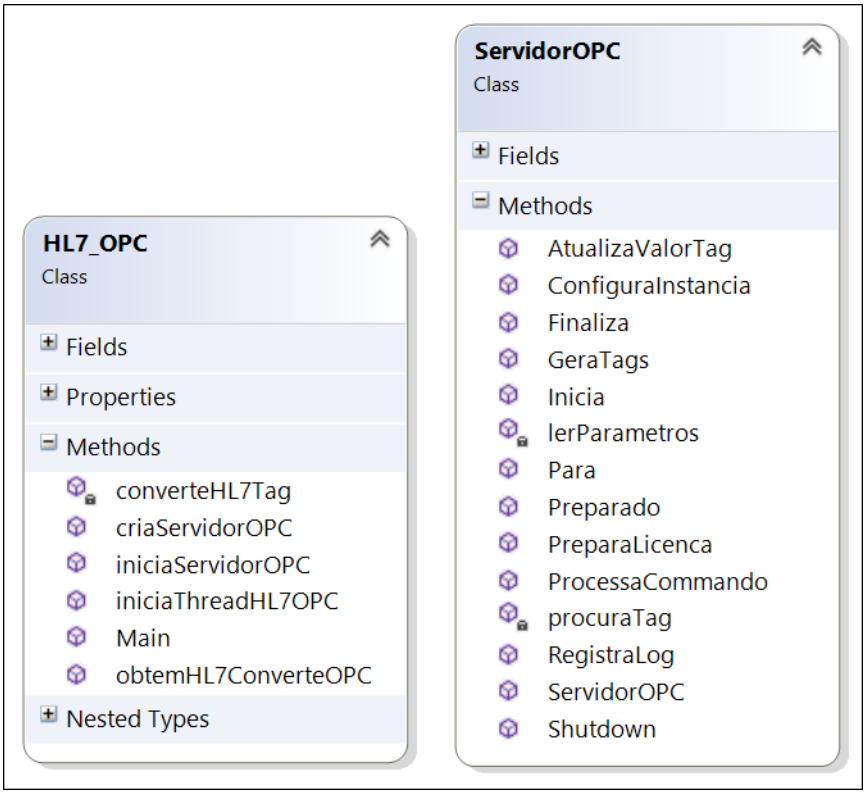


Figura 22: Estrutura das Classes HL7_OPC e Servidor OPC.
Fonte: Elaboração própria.

Um exemplo de execução do Servidor HL7/OPC, onde são recebidos valores de sinais vitais dos tipos RESP e ECG referentes a dois pacientes pode ser visualizado no APÊNDICE D.

6.3 O PIMS

Para avaliar os limites operacionais da proposta deste trabalho, houve a necessidade instalar e configurar um PIMS, como explicado anteriormente. Neste sentido, foram considerados alguns dos principais PIMS que utilizam o padrão OPC para aquisição de dados e armazenamento: o *IHistorian*, pertencente a GE Fanuc; *o InfoPlus.21*, da AspenTech; o *Plant Information* (PI) da empresa OSISoft e *Uniformance PHD*, da empresa Honneywell. Por questões de disponibilidade de acesso e afinidade com suas ferramentas, foi selecionado *Plant Information*, porém qualquer um deles poderia ser utilizado nesta pesquisa.

O PI foi instalado na mesma máquina virtual onde estavam o GSV-HL7 e o Servidor HL7/OPC. O cliente OPC do PI foi também configurado para realizar a interface com o Servidor HL7/OPC no intuito de garantir a coleta de dados. A partir deste momento, pôde-se

estabelecer a comunicação entre o servidor e cliente OPC para que os valores das medições de sinais vitais fossem inseridos no PI.

Foram configuradas 50 *Tags* no PI para conter os valores correspondentes ao possível envio de cinco tipos de medições de sinais vitais para 10 pacientes. No PI, esta configuração é realizada através da ferramenta *PI System Management Tools* (PI SMT) que lê uma planilha com as informações relativas às *Tags* elaborada através do software Microsoft Excel. A Figura 23 ilustra um exemplo desta planilha, onde estão sendo definidos os nomes, as descrições, os desvios, tempos mínimos e máximos de compressão e exceção das *Tags* relativas aos sinais vitais do paciente PAC0000. Outras informações tais como unidade de medida, tipo de dado e forma de aquisição também constam na planilha. Observa-se que os valores cadastrados para configuração de cada *Tag*, na prática, devem ser determinados em conjunto com o especialista da área em questão. Na indústria de manufatura os engenheiros definem estes valores. Já na situação proposta por esta pesquisa, médicos devem ser acionados.

	B	F	G	H	I	J	O	S	T	U	V
1	Tag	compdev	compdevpercent	compmax	compmin	compressing	descriptor	excdev	excdevpercent	excmax	excmin
2	PAC0000-TEMP	0	2	60	0	0	Temperatura do paciente 0	0	1	60	0
3	PAC0000-ECG	0	2	60	0	0	ECG do paciente 0	0	1	60	0
4	PAC0000-SPO2	0	2	60	0	0	Oximetria do paciente 0	0	1	60	0
5	PAC0000-BPM	0	2	60	0	0	BPM do paciente 0	0	1	60	0
6	PAC0000-RESP	0	2	60	0	0	Respiração do paciente 0	0	1	60	0

Figura 23: Exemplo de Planilha de Configuração de *Tags*.
Fonte: Elaboração própria.

Outro parâmetro importante é Classe de Varredura das *Tags*, pois estabelece o ciclo de tempo no qual o PI irá verificar se há algum valor a ser capturado no Cliente OPC. Estes e outros parâmetros, tais como os nomes computador *host* e do servidor OPC, são configurados no arquivo “opcint.bat”, que é utilizando para iniciar a execução o Cliente OPC do PI.

Através da ferramenta PI-SMT é possível verificar se os valores estão sendo lidos e inseridos no Banco de Dados corretamente. Na Figura 24, pode ser visualizado um exemplo de uso do PI-SMT para exibir os valores da *Tag* PAC0000-ECG. O PI também instala um complemento no software Microsoft Excel que é muito utilizado para analisar o histórico de valores das *Tags*.

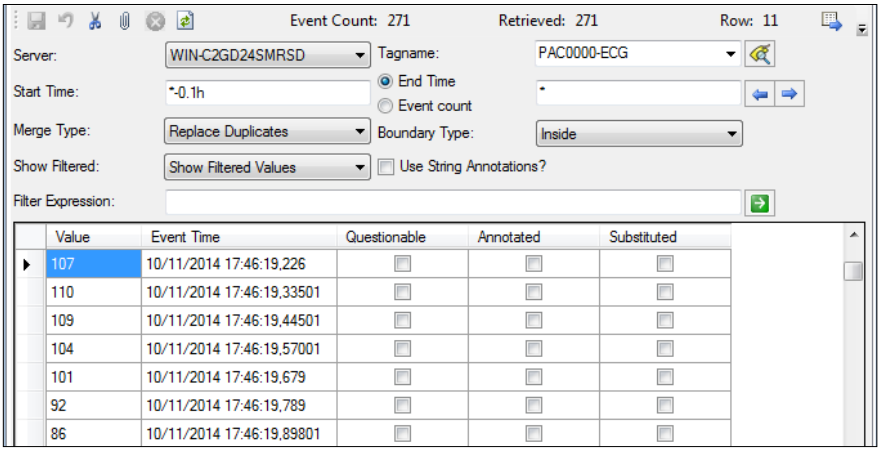


Figura 24: Valores da tag PAC0000-ECG no PI-SMT
Fonte: Elaboração própria.

Vale observar que as *Tags* do PI, por padrão, são configuradas no modo de leitura por subscrição. Desta forma, quando o seu Cliente OPC é iniciado, o PI passa a "escutar" o Servidor OPC, se prontificando a capturar as medições que forem enviadas. Neste caso, uma medição só é disponibilizada pelo Servidor OPC quando houver uma mudança no valor.

7 RESULTADOS

Como explicado anteriormente, esta pesquisa realiza a junção entre os conceitos relativos ao padrão HL7 e o padrão OPC. Com isso, os dados provenientes de equipamentos que medem sinais vitais de pacientes podem ser capturados, armazenados e visualizados através de sistemas que possuem interface OPC, como os PIMS, utilizados largamente na indústria de manufatura.

Nesta seção, serão demonstrados os resultados dos testes realizados com a solução de software desenvolvida no sentido de validar as suas principais funcionalidades, além de verificar o seu desempenho. Para realizar os referidos testes, uma máquina virtual com a configuração abaixo foi preparada para que os componentes de software e o PIMS fossem instalados.

- Sistema Windows 7 Professional de 64bits;
- 1 GB de memória RAM;
- Processador Intel Core 2 Quad de 2.66GHz;
- 15 GB de disco rígido.

Embora a configuração da máquina virtual seja modesta, os resultados obtidos foram satisfatórios como poderá ser visto nas próximas seções. O plano de testes envolveu a definição de dois cenários. No primeiro, é realizado um teste para verificar se os componentes da solução de software proposta executam as funções de acordo com os requisitos especificados. O objetivo neste primeiro momento não é analisar desempenho e sim a funcionalidade. Já no segundo cenário, foram estabelecidos valores específicos para os parâmetros de configuração dos componentes de software para que a solução fosse submetida a situações limites quanto a volume de dados, concorrência e redução de tempos de acesso. Com isso, pretendeu-se oferecer uma visão crítica do desempenho da proposta para que seja avaliada sua utilização em trabalhos futuros.

Os cenários propostos envolveram alguns conceitos que precisam ser definidos no contexto desta pesquisa:

- Granularidade: representa o tempo em milissegundos ocorrido entre duas medições. Neste caso, quanto maior a granularidade, maior o tempo ocorrido;
- Filtro: representa os valores definidos para os parâmetros da exceção e compressão no PI. Ou seja, se a execução for realizada sem filtro, os parâmetros estão definidos como zero;

- Percentual de Armazenamento: percentual de medições armazenadas no Banco de Dados do PI em relação à quantidade real de medições gerada.

7.1 CENÁRIO 1: DOIS CLIENTES OPC CAPTURANDO MEDIÇÕES DO SERVIDOR HL7/OPC

Este primeiro cenário foi estabelecido com a intenção de verificar o funcionamento pleno da solução, ou seja, se todas as mensagens HL7 geradas estão, realmente, sendo enviadas para o Servidor HL7/OPC, convertidas para o padrão OPC, enviadas para o Cliente OPC do PI e, efetivamente, armazenadas no Banco de Dados.

Esta verificação poderia ser realizada simplesmente através da contabilização das mensagens HL7 na origem e comparada com a quantidade armazenada no PI. Entretanto, este teste não seria suficiente para avaliar o fluxo desde a geração até a inclusão no Banco de Dados. Desta forma, foi implementado o cenário ilustrado na Figura 25, onde o GSV-HL7 envia mensagens para o Servidor HL7/OPC, que disponibiliza as medições para o PI e também para outro Cliente OPC, que salva as medições em um arquivo texto (*log*). Este último foi desenvolvido com a biblioteca *OPC Classic Development Toolkit*, a mesma empregada no Servidor HL7/OPC. Todos os softwares foram instalados na máquina virtual já relatada. Vale observar que na prática o Servidor OPC, o Cliente OPC e o PI, normalmente, são instalados em computadores diferentes.

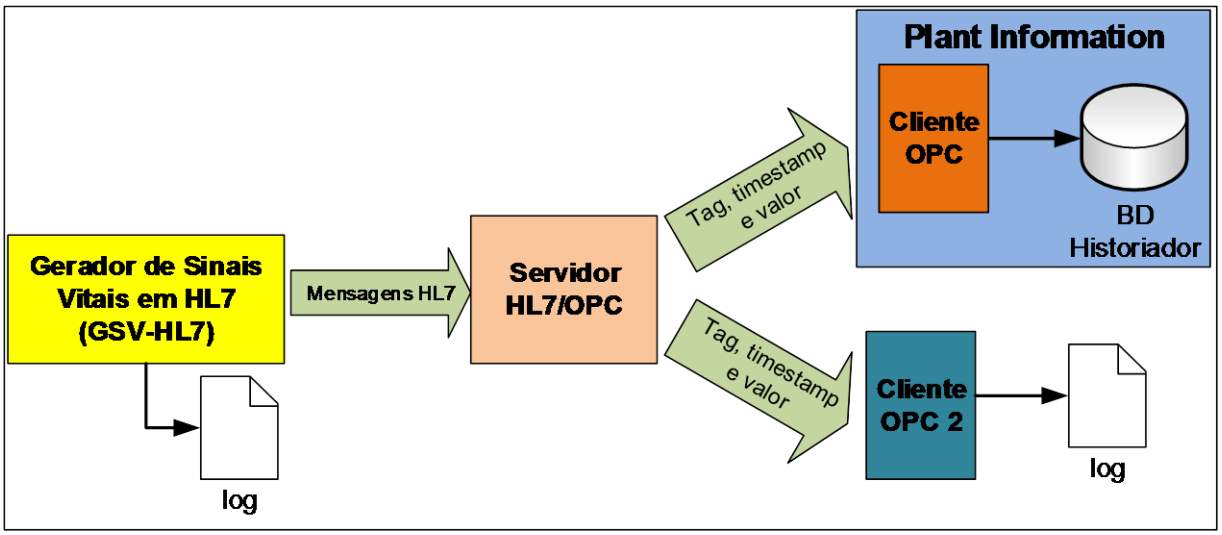


Figura 25: Cenário com dois Clientes OPC
Fonte: Elaboração própria.

Para avaliar a solução, o conteúdo do *log* do GSV-HL7, do Banco de Dados e do *log* do Cliente OPC 2 devem ter seus conteúdos compatíveis. Os dois últimos devem ser exatamente iguais, pois fazem as aquisições de forma subscritiva, ou seja, aguardando que o Servidor HL7/OPC disponibilize a medição quando houver mudança no valor. Já o *log* do GSV-HL7 deve possuir a mais, justamente, os registros com valores duplicados em sequência, como explicado na seção 6.3.

O GSV-HL7 foi configurado para enviar medições do sinal vital ECG para o paciente PAC0000, com granularidade de 5 segundos. O coeficiente linear e amplitude máxima foram definidos em 90 e 20, respectivamente, para que os valores variassem entre 70 e 110. Os ângulos submetidos à função seno foram calculados a partir de um valor randômico entre 10 e 30, ou seja, o próximo ângulo seria no mínimo 10 e no máximo 30 graus maior que o anterior. Deste modo, os valores das medições, calculados a partir da Equação 1, não seriam próximos a cada tempo. Na sequência, a *Tag* correspondente, a PAC0000-ECG, foi configurada no PI para não filtrar valores, portanto com desvios de compressão e exceção iguais a zero.

Partindo da configuração dos parâmetros iniciais, o cenário foi colocado à prova através da execução dos softwares na sequência correta: servidor HL7/OPC, Cliente OPC do PI, Cliente OPC 2 e, finalmente, o GSV-HL7.

A análise deste teste e dos seguintes foi realizada através da mesma planilha Excel utilizada para visualizar as medições do PI. Nela também foram copiados e organizados os conteúdos das outras duas fontes, o que pode ser conferido no extrato da planilha constante no APÊNDICE E. Os resultados obtidos demonstram que o teste logrou êxito, pois as quantidades de medições armazenadas no Banco de Dados do PI, no *log* do Gerador e no *log* do Cliente OPC 2 são compatíveis (Tabela 1). Vale ressaltar que, devido ao formato de aquisição através de subscrição, as medições com valores iguais ao imediatamente anterior (duplicidade) não foram disponibilizadas pelo Servidor HL7/OPC, por conseguinte não foram capturadas tanto pelo PI quanto pelo Cliente OPC 2. Além disso, não foram encontrados também registros com mesmo *timestamp* e valor diferente na comparação linha a linha entre as fontes, confirmando que a execução permaneceu dentro do esperado.

Tabela 1 - Resultados dos testes no cenário 1.

Fonte	Medições	Quantidade
GSV-HL7	Geradas e enviadas (A)	506
	Com valores iguais ao imediatamente anterior (B)	19

Fonte	Medições	Quantidade
	A – B	**487**
Plant Information	Armazenadas no PI	**487**
	Filtradas	0
Cliente OPC 2	Armazenadas pelo Cliente OPC 2	**487**

7.2 CENÁRIO 2: VERIFICAÇÃO DO LIMITE DE GRANULARIDADE COM USO DE FILTRO

Após confirmação do funcionamento correto da solução de software proposta, foi verificada a necessidade de identificar qual seu comportamento diante de situações limites quanto ao tempo de geração e disponibilização das mensagens. O ponto de partida desta análise foi a preparação de um cenário onde os valores correspondentes a um sinal vital são enviados para o Servidor HL7/OPC e deste para o PI. Os filtros de compressão e exceção foram estabelecidos para oferecer mais realismo ao processo. Os parâmetros do GSV-HL7 foram os mesmos do cenário anterior. Ressalta-se que ao manter os parâmetros originais, os valores das medições não serão próximos a cada tempo, o que não condiz com a realidade de um sinal vital, onde normalmente se tem valores iguais ou extremamente próximos em sequência. Esta decisão se justifica na medida em que os resultados deste cenário serão comparados com uma situação mais realista demonstrada na seção 7.2.3 deste trabalho. A execução foi realizada em etapas, onde a granularidade foi reduzida gradativamente até que um limite de capacidade de processamento ou erro fosse identificado.

7.2.1 Etapa 1: granularidade de 2 segundos

Para a primeira etapa foi utilizada a *Tag* PAC0000-ECG configurada no PI de acordo com parâmetros de desvios de compressão e exceção recomendados pelo fornecedor do software (OSISOFT, 2009). A parametrização pode ser visualizada no Quadro 8. O arquivo de parâmetros do GSV-HL7 foi configurado para gerar sinais vitais tipo ECG do paciente PAC0000 a cada 2 segundos.

Quadro 8 – Parâmetros de configuração da Tag no cenário 2 - etapa 1.

Parâmetro	Valor
Desvio de compressão	2%
Tempo máximo compressão	60 segundos
Tempo mínimo compressão	0 segundos
Desvio de exceção	1%
Tempo máximo exceção	60 segundos
Tempo mínimo exceção	0 segundos

A solução foi executada durante 5 minutos e foram obtidos os resultados contidos na Tabela 2. Neste teste pôde-se observar que todas as medições que foram escritas na interface do Servidor OPC foram lidas pelo Cliente OPC do PI.

Tabela 2 - Resultados do cenário 2 - etapa 1.

Medições	**Quantidade**
Enviadas pelo Gerador	150
Recebidas pelo Servidor OPC/HL7	150
Não enviadas pelo Servidor OPC	6
Armazenadas no PI	121
Filtradas pelo PI	23
Perdidas (não lidas pelo PI)	0
Percentual de Armazenamento (Armazen./Geradas)	80,7%

Na Tabela 2 também pode-se observar que algumas medições não foram gravadas no PI, devido à configuração dos filtros de exceção e compressão, gerando um Percentual de Armazenamento de 80,7%. Este alto percentual era esperado, pois os parâmetros de configuração do GSV-HL7 garantem que os valores das medições não sejam muito próximos, sequencialmente, como já explicado. Vale recordar que a decisão de manter os parâmetros desta forma objetiva a geração de resultados a serem comparados com uma situação mais realista (seção 7.2.3), onde o Percentual de Armazenamento será consideravelmente menor.

Outras medições não chegaram a ser enviadas pelo Servidor HL7/OPC, já que possuíam valores iguais ao imediatamente anterior e a *tag* foi configurada como subscrição. Ressalta-se que, diferentemente do Percentual de Armazenamento, este fato contribui para tornar o cenário mais condizente com a realidade, pois além da subscrição ser a configuração padrão recomendada pelo fabricante OSISoft, o resultado vai ao encontro da necessidade de armazenar somente os valores que interessam para representar a curva das medições. A Tabela 3 exibe

alguns dos períodos onde a medição não passou pelos filtros de Exceção e Compressão ou não foram enviadas.

Tabela 3 - Medições filtradas no cenário 2 - etapa 1

Timestamp	Δt	**Valor**	**Situação**
...		...	...
24-11-2014 12:42:53,825	-	71	Armazenado
24-11-2014 12:42:55,853	00:00:02,028	70	**Filtrado**
24-11-2014 12:42:57,869	00:00:02,016	70	**Não enviado**
24-11-2014 12:42:59,885	00:00:02,016	72	Armazenado
24-11-2014 12:43:01,885	00:00:02,000	74	Armazenado
24-11-2014 12:43:03,900	00:00:02,015	76	Armazenado
24-11-2014 12:43:05,932	00:00:02,032	79	Armazenado
24-11-2014 12:43:07,947	00:00:02,015	83	Armazenado
24-11-2014 12:43:09,978	00:00:02,031	86	Armazenado
24-11-2014 12:43:11,978	00:00:02,000	90	Armazenado
24-11-2014 12:43:14,010	00:00:02,032	94	Armazenado
24-11-2014 12:43:16,025	00:00:02,015	99	Armazenado
24-11-2014 12:43:18,041	00:00:02,016	103	Armazenado
24-11-2014 12:43:20,072	00:00:02,031	106	Armazenado
24-11-2014 12:43:22,072	00:00:02,000	108	Armazenado
24-11-2014 12:43:24,088	00:00:02,016	110	Armazenado
24-11-2014 12:43:26,103	00:00:02,015	110	**Não enviado**
24-11-2014 12:43:28,135	00:00:02,032	110	**Não enviado**
24-11-2014 12:43:30,166	00:00:02,031	108	Armazenado
...		...	...

O que ocorre no caso acima é que, embora o cálculo do valor seguinte de uma medição utilize em sua fórmula um aspecto randômico, como foi demonstrado na seção 6.1, a sua curva será sempre em formato de uma senóide ou cossenóide. Deste modo, a variação dos valores próximos aos picos (amplitudes) será muito pequena ou nula, gerando valores muito próximos ou iguais. Esta situação está representada na Figura 26.

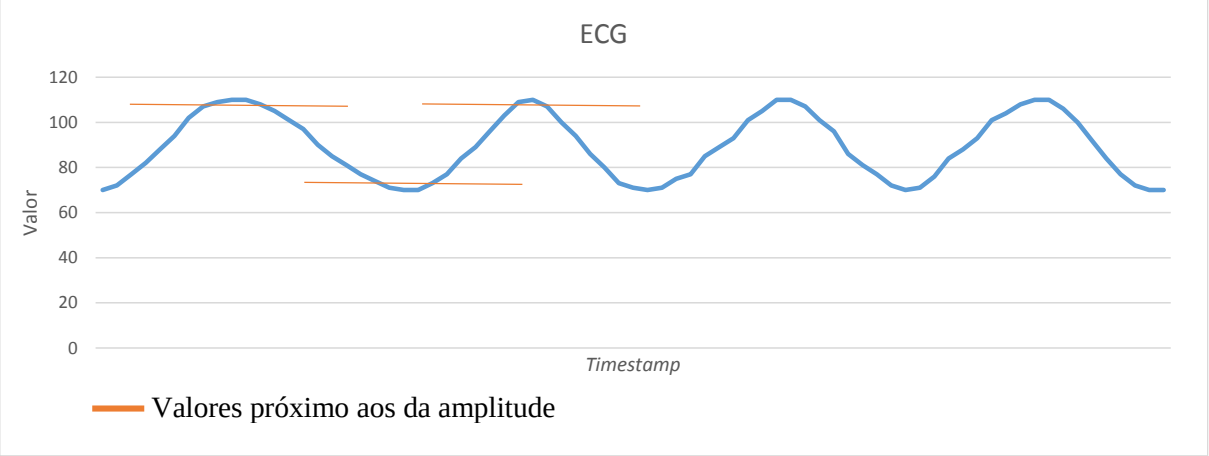


Figura 26: Valores próximos à amplitude da curva senóide
Fonte: Elaboração própria.

Outro aspecto importante que pode ser observado na Tabela 3 é ocorrência de intervalos com valores diferentes entre uma medição e a seguinte (Δt), além dos 2 segundos previstos pela granularidade. Por exemplo, entre a primeira medição e a segunda, houve um acréscimo de 0,028 segundos. Este fato se justifica através tempo de execução da construção da mensagem no GSV-HL7. A disponibilidade do processador e da memória RAM da máquina virtual no momento da execução influenciam diretamente neste caso. Como a diferença é proporcionalmente pequena em relação a granularidade estabelecida, não houve impacto significativo no resultado final, já que as 150 medições previstas foram geradas. Entretanto, em situações nas quais o tempo de execução da construção da mensagem seja superior à granularidade estabelecida e se perceba que esta situação influenciará a análise dos dados, deve-se rever a configuração do computador utilizado no sentido de ampliar sua capacidade de armazenamento e processamento.

7.2.2 Etapa 2: granularidade de 500 milissegundos

Na segunda etapa, a granularidade foi reduzida para 500 milissegundos e mantidos os outros parâmetros de configuração. O tempo de execução foi de 5 minutos e 172 milissegundos. Observou-se que o comportamento da solução de software foi exatamente o mesmo, variando somente as quantidades de mensagens geradas e armazenadas, como pode ser visualizado na Tabela 4.

Tabela 4 - Resultados do cenário 2 - etapa 2.

Medições	**Quantidade**
Enviadas pelo Gerador	598
Recebidas pelo Servidor HL7/OPC	598
Não enviadas pelo Servidor HL7/OPC	24
Armazenadas no PI	527
Filtradas pelo PI	45
Perdidas (não lidas pelo PI)	0
Percentual de Armazenamento (Armazens/Geradas)	88,1%

Outro resultado importante a ser esclarecido é a quantidade de mensagens enviadas pelo Gerador, que desta vez foi menor que a esperada. As duas mensagens que faltaram não foram geradas durante os 5 minutos de execução, devido ao tempo de processamento entre mensagens, que em alguns momentos ultrapassa a granularidade estabelecida, como explicado anteriormente. As medições que chegaram até o PI também foram filtradas, considerando os parâmetros de exceção e compressão definidos. A Tabela 5 exemplifica as observações acima.

Tabela 5 - Medições filtradas no cenário 2 - etapa 2.

Timestamp	Δt	**Valor**	**Situação**
28-11-2014 22:35:22,121	-	97	Armazenado
28-11-2014 22:35:22,652	00:00:00,531	105	Armazenado
28-11-2014 22:35:23,168	00:00:00,516	107	Armazenado
28-11-2014 22:35:24,871	00:00:01,703	110	Armazenado
28-11-2014 22:35:25,371	00:00:00,500	110	**Não enviada**
28-11-2014 22:35:25,871	00:00:00,500	108	Armazenado
28-11-2014 22:35:26,371	00:00:00,500	105	Armazenado
28-11-2014 22:35:26,871	00:00:00,500	98	Armazenado
28-11-2014 22:35:27,371	00:00:00,500	94	Armazenado
28-11-2014 22:35:27,871	00:00:00,500	91	Armazenado
28-11-2014 22:35:28,402	00:00:00,531	84	Armazenado
28-11-2014 22:35:28,902	00:00:00,500	80	Armazenado
28-11-2014 22:35:29,418	00:00:00,516	75	Armazenado
28-11-2014 22:35:29,918	00:00:00,500	71	Armazenado
28-11-2014 22:35:30,418	00:00:00,500	70	**Filtrado**
28-11-2014 22:35:30,918	00:00:00,500	72	Armazenado
28-11-2014 22:35:31,418	00:00:00,500	77	Armazenado
28-11-2014 22:35:31,933	00:00:00,515	82	Armazenado
28-11-2014 22:35:32,433	00:00:00,500	88	Armazenado
...	...	...	...

7.2.3 Etapa 3: granularidade de 10 milissegundos

Nesta etapa do teste, a granularidade foi reduzida drasticamente para 10 milissegundos, o limite de classe de varredura configurada no arquivo "opcint.bat" do PI, como explicado na seção 6.3 deste trabalho. Durante os 5 minutos de execução percebeu-se que a granularidade configurada nunca foi alcançada, devido ao tempo de processamento na geração das mensagens no Gerador. O menor valor conseguido foi de 15 milissegundos, chegando a até 688 milissegundos, revelando a necessidade de ampliar a especificação técnica do equipamento utilizado para executar o cenário. Apesar deste fato negativo, a solução funcionou dentro do esperado com relação aos outros requisitos, como pode ser visto na Tabela 6.

Tabela 6 - Resultados do cenário 2 - etapa 3.

Medições	**Quantidade**
Enviadas pelo Gerador	16346
Recebidas pelo Servidor HL7/OPC	16346
Não enviadas pelo Servidor HL7/OPC	592
Armazenadas no PI	14185
Filtradas pelo PI	1569
Perdidas (não lidas pelo PI)	0
Percentual de Armazenamento (Armazens/Geradas)	86,8%

Com esta quantidade de medições armazenadas no PI, pode-se considerar a economia de espaço em disco desta solução se comparada com um Banco de Relacional, como o Microsoft SQL Server. Supondo que cada *Tag* fosse uma tabela neste Banco de Dados, contendo um campo *Timestamp* do tipo *Datetime* com 8 bytes e outro campo chamado Valor do tipo Real com 4 bytes, cada medição ocuparia 12 bytes. A partir dos resultados obtidos na Tabela 6 acima, pode-se inferir que a economia em somente 5 minutos de processamento seria de (1569 + 592) * 12 bytes, perfazendo um total de 25932 bytes (cerca de 25 MBytes). Se o tempo de processamento for extrapolado para 24 horas, haveria uma redução próxima de 7,2 GB em espaço.

É importante observar que a Percentual de Armazenamento do cenário em questão não é condizente com a realidade, pois os valores de um sinal vital não variam de acordo com uma senóide ou cossenóide. Estes tipos de medição, normalmente, possuem muitos valores iguais ou próximos sequencialmente que não passariam pelos filtros de exceção e compressão, gerando uma economia ainda maior.

Para medir o desempenho em uma situação mais próxima da realidade, a solução foi submetida a um cenário onde a saturação de oxigênio no sangue (SPO2) é medida a cada segundo e seus valores variam dentro dos parâmetros considerados normais[9]. Foi estabelecido, então, que os valores das medições gerados variariam entre 95% e 100%. Assim, na prática, este cenário gera um grande número de valores iguais ou muito próximos em sequência, promovendo um percentual de Armazenamento bem menor que os anteriores. A Tabela 7 exibe o resultado da execução durante 5 minutos, onde pode ser visto um percentual de Armazenamento de 21,18%.

Tabela 7 - Resultados do cenário 2 – situação real

Medições	Quantidade
Enviadas pelo Gerador	298
Recebidas pelo Servidor HL7/OPC	298
Não enviadas pelo Servidor HL7/OPC	221
Armazenadas no PI	63
Filtradas pelo PI	14
Perdidas (não lidas pelo PI)	0
Percentual de Armazenamento (Armazens/Geradas)	21,14%

Comparando gráficos parciais dos valores gerados (Figura 27) com o dos armazenados (Figura 28), pôde-se observar que ainda são mantidas as amplitudes, variações e inclinações próximos dos valores originais, entretanto com uma quantidade de valores cerca de 80% menor. Este é um dos objetivos deste tipo de solução, já que a análise dos dados não é prejudicada pela redução dos valores disponíveis.

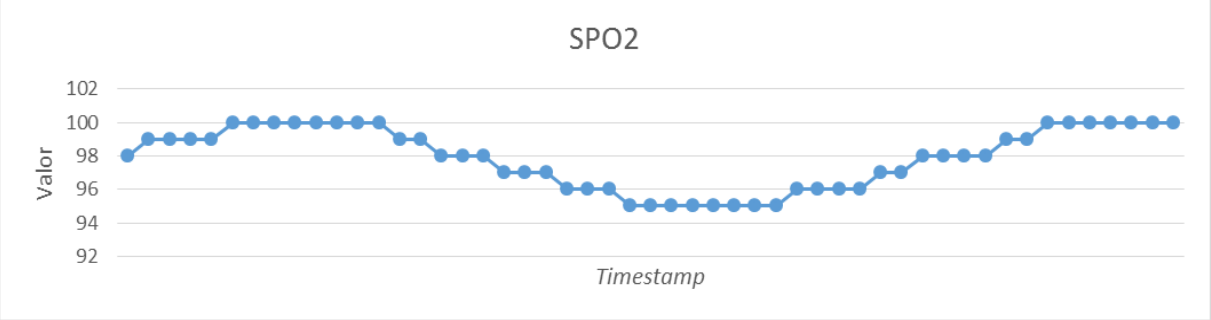


Figura 27: Gráfico parcial dos valores gerados
Fonte: Elaboração própria.

[9] A British Thoracic Society (BTS) publicou um guia para administração de oxigênio em pacientes adultos que recomenda que o SPO2 deve permanecer entre 94 a 98% para a maioria dos pacientes adultos (SMITH et al., 2012).

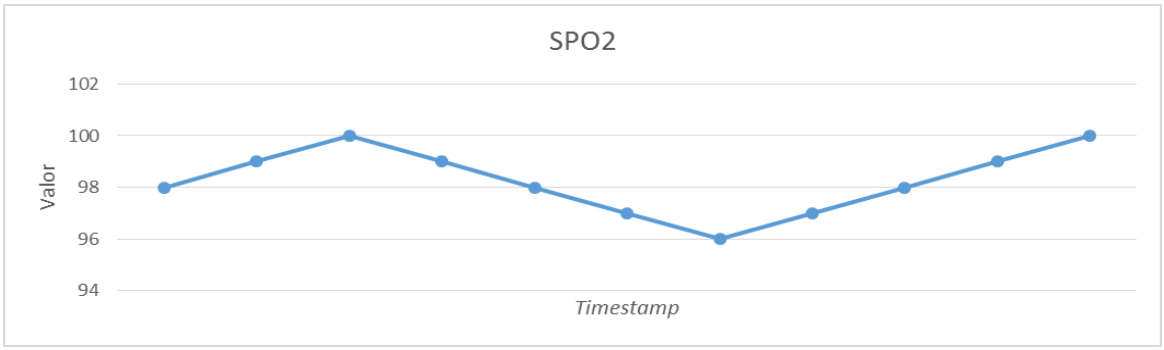


Figura 28: Gráfico parcial dos valores armazenados no PI
Fonte: Elaboração própria.

Embora seja importante a redução da quantidade de valores armazenados para o sucesso do tipo de solução apresentada aqui, deve-se tomar certo cuidado com o gráfico que resulta desta redução, pois este deve ser sempre suficiente para a tomada de decisão. Desta forma, os parâmetros de exceção e compressão de cada *Tag* devem ser definidos em conjunto com um especialista da área.

7.2.4 Etapa 4: granularidade de 5 milissegundos

Para alcançar esta granularidade, foi necessário reduzir a classe de varredura configurada do PI, pois não seria possível capturar medições vindas do Servidor HL7/OPC em uma granularidade menor do que este parâmetro. Porém, ao executar o cliente OPC do PI, o uso da CPU da máquina virtual foi a 100% e permaneceu desta forma, impedindo que qualquer outro software pudesse ser utilizado. Portanto, não foi possível executar o GSV-HL7, fato que impossibilitou a continuidade do teste.

8 CONSIDERAÇÕES E TRABALHOS FUTUROS

Diante dos testes realizados, pôde-se perceber que a solução de software proposta para integrar sistemas que realizam medições de sinais vitais de pacientes hospitalizados com sistemas que armazenam dados históricos eficientemente, funcionou dentro esperado com algumas ressalvas.

Os cenários aos quais os softwares foram submetidos revelaram que há limites que precisam ser respeitados, relacionados, principalmente, com a capacidade de processamento dos computadores utilizados. A granularidade não foi limitada pelo desempenho na conversão e transmissão do Servidor HL7-OPC e sim pelo tempo de execução da construção da mensagem e pelo desempenho do Cliente OPC do PIMS no ambiente computacional utilizado. A maior granularidade que se revelou possível, sem perda de dados, foi de 10 milissegundos. Ressalta-se que a granularidade alcançada se mostra suficiente para permitir a análise do histórico de qualquer sinal vital, de modo que a solução proposta tem pleno potencial para atender aos objetivos. Já para o monitoramento remoto em tempo real, a eficácia da solução aqui apresentada dependerá também da granularidade de transmissão dos dados do monitor de sinais vitais, que pode ser maior que a granularidade de visualização através do próprio monitor.

Observou-se que todas as mensagens enviadas para o Servidor HL7/OPC foram capturadas, convertidas e disponibilizadas para o PIMS, além disso o Cliente OPC do PIMS se comunicou a todo momento com o Servidor HL7/OPC, sem perdas na captura dos valores. Estes fatos aliados à comprovação da extrema capacidade do PIMS de filtrar valores, de acordo com os parâmetros de exceção e compressão, para reduzir o espaço de armazenamento, evidenciou a vantagem da utilização deste tipo de sistema em soluções que envolvem a análise de grandes volumes de históricos de sinais vitais.

A solução também se revelou factível de ser implementada mesmo em máquinas com configuração e desempenho extremamente modestas. Portanto, a sua instalação em máquinas com a configuração minimamente superior à adotada no experimento, desde que se observe o volume de dados a ser armazenado no PIMS, é seguramente livre de sobrecargas. Supondo que a maioria dos computadores hoje existentes em grande parte das clinicas e hospitais possui configuração superior à proposta neste experimento, a instalação do Servidor HL7-OPC não implica na necessidade de grandes investimentos adicionais na infraestrutura de TI atualmente existentes. Entretanto, observa-se que a seleção do PIMS pode impactar consideravelmente nos custos da solução, porém, como o Servidor HL7-OPC pode se comunicar com qualquer PIMS

que possua Cliente OPC, a lista de opções se estende a praticamente todos os sistemas desta categoria, o que facilitará a seleção por aquele que estiver ao alcance financeiro do projeto.

A implementação da solução de software Servidor HL7-OPC desenvolvida nesta pesquisa é um primeiro passo para o desenvolvimento de sistemas médicos avançados a partir da utilização do padrão OPC e, por consequência, do armazenamento das medições dos sinais vitais em PIMS. Já que os PIMS também permitem a implementação de sistemas que consultem, em tempo real, as suas bases de dados, a aplicação básica desta nova abordagem seria a visualização do histórico destas informações por meio de qualquer computador, *tablet*, *smartphone* ou celular, conectados tanto através da rede interna do hospital quanto da internet. Dentre outras possibilidades, os profissionais da área médica poderão monitorar remotamente seus pacientes, ser avisados automaticamente quando ocorrerem situações de emergência, além de visualizar o histórico instantâneo de um ou mais pacientes no sentido de comparar com a situação atual.

Os sistemas que realizam a consulta às bases de dados do PIMS poderão também integrar estes dados com outras informações pertinentes à internação do paciente, contidas na história da moléstia atual (HMA), bem como nos antecedentes médicos e familiares, medicamentos em uso, nutrição, resultados de exames prévios e atuais, registro de alergias, dados antropométricos, exame físico, intercorrências médicas durante o internamento, dentre outros, constituindo um sistema completo de suporte à decisão clínica. Atualmente, os hospitais e clínicas possuem EMRs que permitem o registro destas informações, porém, quando são integrados aos dados de sinais vitais, não disponibilizam o seu longo histórico, limitando a decisão clínica. O Servidor HL7-OPC e o PIMS permitirão que sejam consideradas nas análises todo o tempo e todas as vezes em que o paciente esteve internado.

Outras pesquisas poderão ser derivadas do presente trabalho, considerando que um grande volume de medições de sinais vitais com granularidade de tempo reduzida e a integração dos dados anteriormente citada estejam disponíveis. Além da análise através de estatística tradicional, presente em diversas pesquisas na área médica, técnicas de Mineração de Dados[10], poderão ser implementadas com o objetivo de reconhecer automaticamente padrões, até então desconhecidos, que correlacionem informações médicas entre si e também com as enfermidades que acometeram os pacientes.

[10] Mineração de Dados ou, em inglês Data Mining, é uma técnica que se propõe a analisar profundamente grandes bases de dados com o objetivo de descobrir regras, tendências, padrões ou relacionamentos que não sejam óbvios.

Em um estágio mais avançado no que diz respeito à aplicação de técnicas computacionais, a Mineração de Dados poderá ser conjugada a algoritmos de Redes Neurais Artificiais[11], especificamente redes *Neuro-Fuzzy*[12], para realizar um estudo da possibilidade de prever intercorrências médicas. Deste modo, uma rede neural devidamente treinada a partir do grande volume de dados disponível, anteciparia uma eminente intercorrência que poderia acometer um paciente sob monitoramento.

Ainda a partir do grande volume de dados disponível, propõe-se também a criação e melhoria de modelos matemáticos de pacientes, através da aplicação de técnicas de Identificação de Sistemas. Nesta abordagem, os pacientes seriam considerados sistemas, possuindo variáveis que os afetam e também que os caracterizam. Em seguida, um ou mais métodos de Identificação de Sistemas poderiam ser utilizados para identificar a relação entre as variáveis para constituir um modelo matemático a ser explorado em estudos de casos, simulações e treinamentos de profissionais da área de saúde.

As propostas de utilização de algoritmos de: Mineração de dados, Identificação de Sistemas ou Redes Neurais Artificiais, podem ser beneficiadas pela adoção, ou desenvolvimento, de métodos de Qualificação de Amostras que determinam se os conjuntos de dados a serem analisados, são válidos, ou não, antes de serem submetidos aos algoritmos citados. Um exemplo destes métodos foi desenvolvido no Laboratório de Sistemas Integrados de Produção (LABSIP) da Universidade Federal da Bahia a partir das aquisições de dados do processo produtivo de uma Unidade Termelétrica da Petrobras. Normalmente, as séries de dados armazenadas em um PIMS contêm lacunas, ou seja, dados com valores inexistentes, ou não numéricos, gerados em decorrência da indisponibilidade operacional de equipamentos, falhas no sensoriamento ou oscilações de comunicação entre a fonte de dados e o PIMS. A depender da quantidade e distribuição dos valores inválidos constantes em uma amostra, esta pode ser considerada desqualificada para fins de identificação de sistemas ou mineração de dados (SÁ BARRETTO, 2009).

Este mesmo laboratório produziu um trabalho que utilizou algoritmos de agrupamento de séries temporais (*clustering*), a partir de funções de similaridade ou distância entre curvas, com o objetivo de estabelecer curvas padrões de operação para partidas de turbinas a gás

[11] Redes Neurais Artificiais são algoritmos que simulam as conexões sinápticas do cérebro, permitindo o aprendizado de alguma tarefa ou conhecimento, bem como o reconhecimento de padrões.
[12] Redes Neuro-Fuzzy combinam os algoritmos das redes neurais com conceitos de lógica fuzzy que permitem a utilização de valores intermediários entre os valores 1 e 0, ou entre o "Sim e o "Não", possibilitando a análise de itens incertos ou não quantificáveis.

consideradas normais ou com falha. Os padrões gerados serviram de base para a configuração de um sistema *fuzzy* (PEREIRA et al., 2014). Este método pode também ser incorporado ao presente trabalho na medida em que as séries de medições de sinais vitais podem ser segregadas em situações com e sem intercorrência médica e submetidas a algoritmos de agrupamento que identificariam os respectivos padrões de curva. Após a finalização desta pesquisa, algumas sugestões e críticas, relacionadas exclusivamente às diversas possibilidades de desenvolvimento e testes dos softwares envolvidos na solução, foram levantadas no sentido de aprimorar os resultados. A primeira delas diz respeito à realização de testes de desempenho através do aumento gradativo da quantidade de pacientes e de sinais vitais. Os testes também poderiam considerar um número maior de clientes OPC simultaneamente para simular a situação onde vários PIMS adquirem e armazenam as medições. Estes tipos de avaliação podem contribuir para identificar quais os limites operacionais nestes aspectos específicos.

Ainda sobre os testes de desempenho, é conveniente que sejam também realizados em máquinas virtuais com diferentes configurações. Estes testes podem demonstrar a existência de eventuais correlações entre as diversas configurações e a eficiência dos componentes de software na geração de mensagens HL7, conversão para OPC e armazenamento no PIMS.

A implementação da versão 3.x do HL7, tanto na geração das mensagens quanto na aceitação pelo Servidor HL7-OPC, também foi indicada como complemento deste trabalho. Apesar de não haver ganho acadêmico aparente ao demonstrar que a solução proposta também funcionaria com este padrão de mensagem e mesmo este ainda não sendo o mais adotado, a pesquisa passaria a utilizar também um padrão mais novo, o que seria importante para outros trabalhos que seguissem a mesma linha.

Outro ponto discutido foi a possibilidade de desenvolver também um Servidor HL7-OPC baseado no padrão *OPC Unified Architecture* (UA) com o intuito de comparar o seu desempenho com o padrão OPC-DA, utilizado nesta pesquisa. Esta abordagem se mostra interessante na medida que a intenção do OPC UA é se tornar um padrão independentemente de sistema operacional, porém com a mesma qualidade e velocidade de transmissão e recepção que o OPC DA. Muitos trabalhos acadêmicos fizeram esta análise comparativa nos últimos anos, entretanto não foram encontrados artigos, dissertações ou teses que o fizessem a partir da obtenção de dados no padrão HL7, portanto o uso do OPC UA nestas condições também geraria contribuição acadêmico-científica.

Houve também a proposta para a implementação de uma arquitetura genérica e aberta para interoperabilidade entre monitores de sinais vitais e o PIMS, independente de padrões para

comunicação. Esta abordagem foi elaborada por um trabalho realizado no Laboratório de Sistemas Distribuídos (LASID) da UFBA, que concebeu uma plataforma que disponibiliza componentes e ferramentas básicas para as atividades de aquisição de dados, controle e supervisão para ser utilizada em ambientes computacionais da indústria (ANDRADE, 2006).

Alguns outros itens de melhoria, considerados mais simples, também foram propostos com o intuito de facilitar a execução do experimento e obtenção dos resultados. Um exemplo são as telas no estilo formulário do sistema operacional Windows para configuração, ativação e monitoramento tanto do GSV-HL7 quanto do Servidor HL7-OPC. Não obstante a questão estética, estas telas proporcionariam, dentre outros benefícios, a visualização intuitiva da contagem de mensagens geradas, enviadas, lidas e transmitidas, a possibilidade de iniciar, pausar ou parar a execução dos softwares, além de poder ser utilizadas para exibir os resultados dos diversos cenários de testes implementados durante a pesquisa.

9 REFERÊNCIAS

ANDRADE, S. S. **Sistemas distribuídos de supervisão e controle baseados em componentes de tempo real**. [s.l.] Universidade Federal da Bahia, 2006.

ARCHER, N.; COCOSILA, M. **Improving EMR System Adoption in Canadian Medical Practice: A Research Model**. World Congress on Privacy, Security, Trust and the Management of e-Business. **Anais**...p. 121–132. IEEE, 2009.

ARMSTRONG, J. A mill's experience with a plant wide information system. **Conference Record of 1998 Annual Pulp and Paper Industry Technical Conference**, p. 145–151. , 1998.

AVELAR, E. et al. Interoperability issues on heterogeneous wireless communication for smart cities. **Computer Communications**, 2014.

BARR, D. C. **The use of a data historian to extend plant life.** International Conference on Life Management of Power Plants. **Anais**...v. 1994, p. 35–39. Edinburgh: IET, 1994.

BENDER, D.; SARTIPI, K. **HL7 FHIR: An Agile and RESTful approach to healthcare information exchange**. Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems. **Anais**...p. 326–331. IEEE, 2013.

BENSON, T. **Principles of Health Interoperability HL7 and SNOMED.** New York: Springer London Dordrecht Heidelberg, . v. 6.p. 287, 2010.

BOGDAN, O. et al. Integrated medical system using DICOM and HL7 standards. In: LAZINICA, A. (Ed.). . **New Advanced Technologies**. [s.l: s.n.]. p. 231–247, 2010.

CHEN, D.; DOUMEINGTS, G.; VERNADAT, F. Architectures for enterprise integration and interoperability: Past, present and future. **Computers in Industry**, v. 59, n. 7, p. 647–659, 2008.

CHRIST, R. E. R. et al. **Sistema de Monitoração Remota de Pacientes em Tempo-Real Através da Intranet do Hospital.** Congresso Brasileiro de Informática em Saúde. **Anais**...Ribeirão Preto, 2004.

COCKSHOTT, W. P. et al. **Data compression in database systems**. Proceedings. IDEAS'98. International Database Engineering and Applications Symposium (Cat. No.98EX156). **Anais**...p. 111–120. IEEE Comput. Soc, 1998.

DIPNALL, J. F. et al. Data Integration Protocol In Ten-steps (DIPIT): A new standard for medical researchers. **Methods (San Diego, Calif.)**, 2014.

EREN, H. **Assessing the health of sensors using data historians.** IEEE Sensors Applications Symposium Proceedings. **Anais**...p. 1–4. IEEE, 2012.

FONSECA, M. D. O. **Comunicação OPC–Uma abordagem prática.** VI Seminário de Automação de Processos. **Anais**...Vitória, 2002.

FRAS, A.; DANG, T. **Improving industrial application ' s performances with an Historian.** IEEE International Conference on Industrial Technology. **Anais**...v. 2, p. 718–721. IEEE, 2004.

GAION, S. et al. **Design of a domain model for clinical engineering within the HL7 Reference Information Model.** 2010 IEEE Workshop on Health Care Management (WHCM). **Anais**...p. 1–6. IEEE, 2010.

GUO, Z. X.; XIE, X. Q.; NI, Z. G. **The application of OPC DA in factory data acquisition**. IEEE International Conference on Computer Science and Automation Engineering (CSAE). **Anais**...v. 2, p. 209–212. Zhangjiajie, China: IEEE, 2012.

HADZIC, M.; DILLON, T.; CHANG, E. **Use of Digital Ecosystem and Ontology Technology for Standardization of Medical Records**. 2007 Inaugural IEEE-IES Digital EcoSystems and Technologies Conference. **Anais**...p. 595–601. IEEE, 2007.

HEALTH LEVEL SEVEN INTERNATIONAL. **HL7 Messaging Standard version 2.7.** Estados Unidos, 2011.

HEALTH LEVEL SEVEN INTERNATIONAL. **HL7 Version 2 Product Suite**. Disponível em: <http://www.hl7.org/implement/standards/product_brief.cfm?product_id=185>. Acesso em: 10 out. 2014.

HONG, X.; JIANHUA, W. Using standard components in automation industry: A study on OPC Specification. **Computer Standards & Interfaces**, v. 28, n. 4, p. 386–395, 2006.

JARDIM-GONCALVES, R.; GRILO, A.; STEIGER-GARCAO, A. Challenging the interoperability between computers in industry with MDA and SOA. **Computers in Industry**, v. 57, n. 8-9, p. 679–689, 2006.

KOUSSOURIS, S.; LAMPATHAKI, F. Digging into the real-life enterprise interoperability areas definition and overview of the main research areas. **Proceedings of CENT**, p. 19–22, 2011.

LU, X. et al. **Research and implementation of transmitting and interchanging medical information based on HL7.** The 2nd International Conference on Information Science and Engineering. **Anais**...p. 457–460. Hangzhou, China: IEEE, 2010.

LUSK, E. et al. An Interoperability Approach to System Software, Tools, and Libraries for Clusters. **International Journal of High Performance Computing Applications**, v. 20, n. 3, p. 401–407, 2006.

MAHNKE, W.; LEITNER, S.-H.; DAMM, M. **OPC unified architecture.** Berlim: Springer-Verlag Berlin Heidelberg, 2009.

MALLEK, S.; DACLIN, N.; CHAPURLAT, V. The application of interoperability requirement specification and verification to collaborative processes in industry. **Computers in Industry**, v. 63, n. 7, p. 643–658, 2012.

MENEZES, A. L. **Uma abordagem, baseada na integração de arquétipos a mensagens hl7, para a comunicação de aplicações ubíquas no cuidado de saúde pervasivo**. [s.l.] Universidade Federal de São Carlos, 2011.

MYKKÄNEN, J. A.; TUOMAINEN, M. P. An evaluation and selection framework for interoperability standards. **Information and Software Technology**, v. 50, n. 3, p. 176–197, 2008.

NAMLI, T.; ALUC, G.; DOGAC, A. An interoperability test framework for HL7-based systems. **IEEE Transactions on Information Technology in Biomedicine**, v. 13, n. 3, p. 389–399, 2009.

NETO, E. J. M. et al. Adaptive swinging door trending: um algoritmo adaptativo para compressão de dados em tempo real. **Congresso Brasileiro de Automática**, p. 852–858, 2014.

OPC FOUNDATION. **OPC data access custom interface standard version 3.0**, 2003.

OSISOFT. **PI Server System Management Guide.** San LeandroOSIsoft, 2009.

PALFREY, J.; GASSER, U. **Interoperability DI. Digital Identity Interoperability and eInnovation**. [s.l.] Berkman Center Research Publication, . p. 45, 2007.

PANETTO, H.; MOLINA, A. Enterprise integration and interoperability in manufacturing systems: Trends and issues. **Computers in Industry**, v. 59, n. 7, p. 641–646, 2008.

PATTLE, R.; RAMISCH, J. **OPC the de facto standard for real time communication.** Proceedings of 5th International Workshop on Parallel and Distributed Real-Time Systems and 3rd Workshop on Object-Oriented Real-Time Systems. **Anais**...p. 289–294. IEEE Comput. Soc, 1997.

PEREIRA, O. J. et al. Fuzzy rule-based classifier for Fault Prediction in a Thermoelectric Unit. **Brazilian Journal of Operations & Production Management**, v. 10, n. 2, p. 79–90, 2014.

PETTERSSON, J.; GUTMAN, P.-O. Automatic tuning of the window size in the Box Car Back Slope data compression algorithm. **Journal of Process Control**, v. 14, n. 4, p. 431–439, 2004.

REZAEI, R. et al. Interoperability evaluation models: A systematic review. **Computers in Industry**, v. 65, n. 1, p. 1–23, 2014.

ROUSSELOT, J.; DECOTIGNIE, J.-D. **Wireless communication systems for continuous multiparameter health monitoring.** IEEE International Conference on Ultra-Wideband. **Anais**...p. 480–484. Vancouver, BC: IEEE, 2009.

SÁ BARRETTO, S. **Desenvolvimento de metodologia para atualização em tempo real de modelos matemáticos de processos decisórios.** [s.l.] Universidade Federal da Bahia, 2009.

SÁEZ, C. et al. An HL7-CDA wrapper for facilitating semantic interoperability to rule-based Clinical Decision Support Systems. **Computer methods and programs in biomedicine**, v. 109, n. 3, p. 239–49, 2013.

SAHU, B. K.; VERMA, R. **DICOM search in medical image archive solution e-Sushrut Chhavi**. 3rd International Conference on Electronics Computer Technology. **Anais**...v. 6, p. 256–260. IEEE, 2011.

SHETH, A.; RANABAHU, A. Semantic Modeling for Cloud Computing, Part 1. **IEEE Internet Computing**, v. 14, n. 3, p. 81–83. , 2010.

SHIMANUKI, Y. **OLE for process control (OPC) for new industrial automation systems**. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028). **Anais**...v. 6, p. 1048–1050. IEEE, 1999.

SILVEIRA, P. R.; SANTOS, W. E. **Automação e Controle Discreto**. 5.ed. ed. Tatuapé: Érica, 1998.

SMITH, G. B. et al. S(p)O(2) values in acute medical admissions breathing air implications for the British Thoracic Society guideline for emergency oxygen use in adult patients resuscitation, v. 83, n. 10, p. 1201–5, 2012.

SOUZA, L. C. A.; SEIXAS FILHO, C.; PENA, R. T. **Padrão de acesso a dados opc e sua implementação em um driver OPC-MODBUS**. II Congresso Mineiro De Automação, V Simpósio Regional de Instrumentação da ISA-BH/GRINST-MG. **Anais**...p. 157–164. Belo Horizonte, 1998.

TANG, J.; ZOU, G. **The application study of middleware based on HL7 in multi-parameter monitor**. ICENT 2010 - 2010 International Conference on Educational and Network Technology. **Anais**...n. Icent, p. 167–169. Qinhuangdao: IEEE, 2010.

XIAODONG, F. et al. **An improved process data compression algorithm**. Proceedings of the 4th World Congress on Intelligent Control and Automation (Cat. No.02EX527). **Anais**...p. 2190–2193. IEEE, 2002.

ZAHARIA, V.; DRAGAN, F. **International Conference on Advancements of Medicine and Health Care through Technology**. (S. Vlad, R. V. Ciupa, Eds.)International Conference on Advancements of Medicine and Health Care through Technology. **Anais**...: IFMBE Proceedings.v. 36, p. 44–47. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

ZHANG, F. et al. Application of a Real-Time Data Compression and Adapted Protocol Technique for WAMS. **IEEE Transactions on Power Systems**, p. 1–10., 2014.

ZHENG, L.; NAKAGAWA, H. **OPC (OLE for process control) specification and its developments**. Proceedings of the 41st SICE Annual Conference. SICE 2002. **Anais**...v. 2, p. 917–920. Soc. Instrument & Control Eng. (SICE), 2002.

ZHIHAO LING; WEIBIN CHEN; JINSHOU YU. **Research and implementation of OPC server based on data access specification**. Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No.04EX788). **Anais**...v. 2, p. 1475–1478. IEEE, 2004.

APÊNDICE A – CÓDIGO FONTE DO SOFTWARE GSV-HL7

```
public class GeradorMsgHL7
{

   static Int32 tempoMilisegundos = 5000; //Padrão 5 segundos
   static string[] strHL7; //Mensagens HL7 padrão para cada paciente
   static string[] strTag; //Nome da Tag que esta sendo enviada
   static string strLinhaHL7;
   static int[] grauCalculo; //grau atual para calculo do valor a ser enviado para cada medição
   static int qtdPacientes = 10; //Padrão 10 pacientes
   static int qtdRegistros; //Quantidade total de registros
   static int qtdRegistrosEnviar = 10; //Quantidade de registro a enviar de uma vez para o servidor
   static int ValInicialRand = 5; //Valor inicial para calculo do valor da Tag
   static int ValFinalRand = 20; //Valor final para calculo do valor da Tag
   static string strFormatoTimestamp = "{0:yyyyMMddHHmmss}"; //formato do timestamp
   static string[] strtiposSinais;
   static Int16 logArquivo = 0;
   static Int16 exibeMensagem = 1;
   static double amplitude = 20;
   static double coefLinear = 90;


   //Ler o arquivo de parâmetros
   private static void leParametros()
   {

      string linha;
      int i = 0, qtdTiposSinais;

      //Ler mensagem HL7
      System.IO.StreamReader arquivoHL7 = new System.IO.StreamReader(@"mensagemHL7.txt");
      strLinhaHL7 = arquivoHL7.ReadToEnd();
      arquivoHL7.Close();

      //Ler parametros
      System.IO.StreamReader arquivo = new System.IO.StreamReader(@"parametros.txt");
      while ((linha = arquivo.ReadLine()) != null)
      {
         switch (linha)
         {
            case "strtiposSinais:":
               qtdTiposSinais = Convert.ToInt32(arquivo.ReadLine()); //ler quantidade de registros
               strtiposSinais = new string[qtdTiposSinais];
               // ler os registros
               for (i = 0; i < qtdTiposSinais; i++)
               {
                  strtiposSinais[i] = arquivo.ReadLine();
               }
               break;
            case "qtdPacientes:":
               qtdPacientes = Convert.ToInt32(arquivo.ReadLine());
               break;
            case "qtdRegistrosEnviar:":
               qtdRegistrosEnviar = Convert.ToInt32(arquivo.ReadLine());
               break;
            case "tempoMilisegundos:":
               tempoMilisegundos = Convert.ToInt32(arquivo.ReadLine());
               break;
            case "ValInicialRand:":
               ValInicialRand = Convert.ToInt32(arquivo.ReadLine());
               break;
            case "ValFinalRand:":
               ValFinalRand = Convert.ToInt32(arquivo.ReadLine());
               break;
            case "formatoTimestamp:":
               strFormatoTimestamp = arquivo.ReadLine();
               strFormatoTimestamp = "{0:" + strFormatoTimestamp + "}";
               break;
            case "logArquivo:":
               logArquivo = Convert.ToInt16(arquivo.ReadLine());
               break;
```

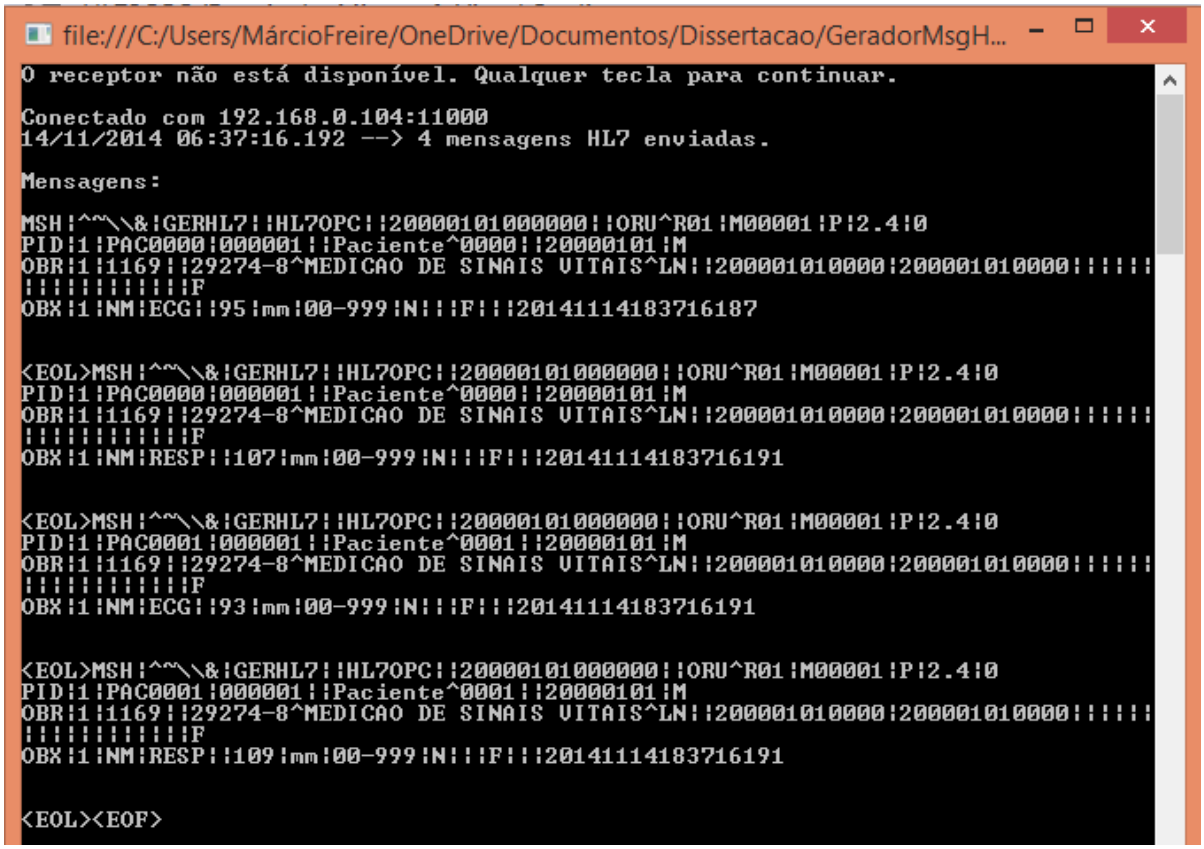
```
            case "exibeMensagem:":
                exibeMensagem = Convert.ToInt16(arquivo.ReadLine());
                break;
            case "amplitude:":
                amplitude = Convert.ToDouble(arquivo.ReadLine());
                break;
            case "coefLinear:":
                coefLinear = Convert.ToDouble(arquivo.ReadLine());
                break;
        }

    }

    arquivo.Close();
}


public static void geraStringsHL7()
{
    string strPaciente, strIDPaciente;

    int i = 0, j = 0;

    qtdRegistros = qtdPacientes * strtiposSinais.Length;

    strHL7 = new string[qtdRegistros]; // Define a quantidade de registros do array de mensagens para os pacientes
    grauCalculo = new int[qtdRegistros]; // Define a quantidade de registros do array de graus para os pacientes
    strTag = new string[qtdRegistros]; //Define quantidade de Tags

    while (i < qtdPacientes)
    {
        strPaciente = "Paciente^" + i.ToString("0000");
        strIDPaciente = "PAC" + i.ToString("0000");

        foreach (string strSinalVital in strtiposSinais)
        {
            strHL7[j] = strLinhaHL7.Replace("<strIDPaciente>", strIDPaciente); //Inclui o ID do paciente
            strHL7[j] = strHL7[j].Replace("<strPaciente>", strPaciente);       //Inclui o nome do paciente
            strHL7[j] = strHL7[j].Replace("<strSinalVital>", strSinalVital);   //Inclui o Sinal vital
            strHL7[j] = strHL7[j] + "<EOL>";

            strTag[j] = strIDPaciente + "-" + strSinalVital; //Obtem o nome da tag para logar

            j++;
        }
        i++;
    }
}

public static void enviaMensagensHL7()
{

    string strLog;
    byte[] bytes = new byte[1024]; //Buffer
    string timestamp;
    int  i = 0, j = 0;
    int valor = 0;
    string strMensagem;
    DateTime agora;
    Random rnd = new Random(); //valor randomico para aumento do grau;


    IPHostEntry InfoHost = Dns.Resolve(Dns.GetHostName());
    IPAddress EnderecoIP = InfoHost.AddressList[0];

    // Usa porta 11000 e o IP local.
    IPEndPoint EndPointRemoto = new IPEndPoint(EnderecoIP, 11000);

    //Abre arquivo de log
    //System.IO.StreamWriter arquivo = new System.IO.StreamWriter(@"log.csv", false);


    //Envia valores de medição para o servidor
    while (i < qtdRegistros)
    {
        try
```

```
        {
            // Cria o socket.
            Socket transmissor = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

            // Conecta com o servidor e envia os dados.
            try
            {
                transmissor.Connect(EndPointRemoto);

                Console.WriteLine("");
                Console.WriteLine("Conectado com {0}",
                transmissor.RemoteEndPoint.ToString());

                // Formata a mensagem HL7 com os registros
                strMensagem = "";
                strLog = "";
                agora = DateTime.Now;
                for (j = 0; j < qtdRegistrosEnviar; j++)
                {
                    //Aumenta em X graus para calculo do valor da Tag
                    grauCalculo[i] += rnd.Next(ValInicialRand, ValFinalRand);

                    //define o valor da medicao a partir da senoide ou cossenoide para um Paciente/Sinal Vital
                    valor = calculaValor(i);

                    //Obtem a hora no formato correto
                    timestamp = String.Format(strFormatoTimestamp, agora);

                    // Atualiza o valor na mensagem
                    strMensagem = strMensagem + strHL7[i].Replace("<valor>", Convert.ToString(valor));

                    // Atualiza o valor na mensagem
                    strMensagem = strMensagem.Replace("<timestamp>", timestamp);

                    if (logArquivo == 1)
                    {
                        strLog = strLog + strTag[i] + ";" + agora.ToString("dd/MM/yyyy") + ";" + agora.ToString("HH:mm:ss.fff") + ";" +
Convert.ToString(valor) + Environment.NewLine;
                    }

                    //verifica se chegou ao limite de registros
                    if (i == qtdRegistros - 1)
                        break;
                    else
                        i++;
                }
                strMensagem += "<EOF>"; //Insere o final de mensagem

                // Envia mensagem
                byte[] msg = Encoding.ASCII.GetBytes(strMensagem);
                int bytesSent = transmissor.Send(msg);

                Console.WriteLine(DateTime.Now.ToString("dd/MM/yyyy HH:mm:ss.fff") + " --> " + qtdRegistrosEnviar + " mensagem(ns)
HL7 enviada(s).");

                if (exibeMensagem == 1)
                {
                    Console.WriteLine();
                    Console.WriteLine("Mensagens: ");
                    Console.WriteLine();
                    Console.WriteLine(strMensagem);
                }
                //Abre arquivo de log
                if (logArquivo == 1)
                {
                    System.IO.File.AppendAllText(@"log.csv", strLog);
                    //arquivo.WriteLine(strLog);
                }

                // Libera a conexão para reinício.
                transmissor.Shutdown(SocketShutdown.Both);
                transmissor.Close();

                //Reinicia o vetor de mensagens
                if (i == qtdRegistros - 1)
                    i = 0;
```

```
            }

            // Tratamento de erro padrão para socket
            catch (ArgumentNullException ane)
            {
               Console.WriteLine("ArgumentNullException : {0}", ane.ToString());
            }
            catch (SocketException se)
            {
               if (se.NativeErrorCode == 10061)
               {
                  Console.WriteLine("O receptor não está disponível. Qualquer tecla para continuar.");
                  Console.ReadKey();
               }
               else
                  Console.WriteLine("SocketException : {0}", se.ToString());
            }
            catch (Exception e)
            {
               Console.WriteLine("Unexpected exception : {0}", e.ToString());
            }
         }
         // Tratamento de erro padrão
         catch (Exception e)
         {
            Console.WriteLine(e.ToString());
         }

         //Aguarda o tempo para nova execução
         Thread.Sleep(tempoMilisegundos);
      }

      //arquivo.Close();
   }

   //Calcula o próximo valor de ângulo utilizando função seno ou cosseno
   private static int calculaValor(int i)
   {

      double angRad; //Angulo em radianos

      angRad = Math.PI * grauCalculo[i] / 180;

      //Seleciona a funçao seno ou cosseno se indice par ou impar
      if ((i % 2) == 0)
         return (Convert.ToInt32(Math.Round(coefLinear + amplitude * Math.Sin(angRad))));
      else
         return (Convert.ToInt32(Math.Round(coefLinear + amplitude * Math.Cos(angRad))));
   }


   public static int Main(String[] args)
   {

      if (args.Length > 0)
      {
         tempoMilisegundos = Int32.Parse(args[0]) * 1000;
         qtdPacientes = Int32.Parse(args[1]);
      }
      leParametros();
      geraStringsHL7();
      enviaMensagensHL7();
      return 0;
   }
}
```

APÊNDICE B – EXEMPLO DE EXECUÇÃO DO SOFTWARE GSV-HL7

Exemplo de execução onde o GSV-HL7 se conecta com o Servidor HL7/OPC e envia dois valores de sinais vitais dos tipos RESP e ECG referentes aos pacientes PAC0000 e PAC0001.



APÊNDICE C – CÓDIGO FONTE DO SOFTWARE SERVIDOR HL7/OPC

```
public class HL7_OPC
{

   //Atributo para notificar para a thread que um evento ocorreu
   public static AutoResetEvent EventoServidorOPC = new AutoResetEvent(false);

   public static bool End = false;

   // Dados
   public static string dados = null;

   #region Atributos privados

   //-------------------------
   private static bool m_FimTag = false;

   //Mecanismo para sincronização entre processos
   private static Mutex m_mutex = new Mutex();

   //Servidor OPC
   private static ServidorOPC m_ServidorOPC = null;

   //Tag, timestamp e valor
   private static string tag, timestampTag, valTag;

   //--
   #endregion



   public static int Main(String[] args)
   {
      iniciaServidorOPC();
      return 0;
   }

   //-------------------------
   public static bool TagFinal
   {
      get
      {
         return m_FimTag;
      }
      set
      {
         m_FimTag = value;
      }
   }

   public static Mutex Mutex
   {
      get
      {
         return m_mutex;
      }
   }

   public static ServidorOPC ServidorOPC
   {
      get
      {
         return m_ServidorOPC;
      }
   }
   //--


   //cria um servidor OPC
   public static void criaServidorOPC()
   {
      if (m_ServidorOPC == null)
```

```
    {
        m_ServidorOPC = new ServidorOPC(ref EventoServidorOPC);
    }
}


//Obtem as mensagens HL7 enviadas pelo cliente via Socket, converte para OPC e escreve no servidor OPC
public static void obtemHL7ConverteOPC()
{
    string msgHL7="";
    int pos=0;
    bool FimServidorOPC = false;
    CultureInfo invC = CultureInfo.CurrentCulture;

    byte[] bufferDadosObtidos = new Byte[1024]; // buffer para obter dados


    // Aguarda o momento seguro para iniciar a Thread
    m_mutex.WaitOne();


    // Estabelece os parâmetros para receber a conexão via socket.

    IPHostEntry InfoHost = Dns.Resolve(Dns.GetHostName());
    IPAddress EnderecoIP = InfoHost.AddressList[0];
    IPEndPoint EndPointLocal = new IPEndPoint(EnderecoIP, 11000);

    // Cria o socket
    Socket receptor = new Socket(AddressFamily.InterNetwork,SocketType.Stream, ProtocolType.Tcp);

    try
    {
        // inicia conexão socket
        receptor.Bind(EndPointLocal);
        receptor.Listen(ServidorOPC.maxConexoesSocket);

        //Enquanto o servidor OPC estiver disponível executa o laço
        while (!m_FimTag && !FimServidorOPC)
        {
            Console.WriteLine("");
            Console.WriteLine(".....................................................");
            Console.WriteLine("Aguardando conexão...");


            // Aguarda o transmissor conectar e enviar dados
            Socket handler = receptor.Accept();
            dados = null;

            Console.WriteLine("Conexão estabelecida.");
            // Conexão estabelecida. Obtem as mensagens HL7 enviadas
            while (true)
            {
                bufferDadosObtidos = new byte[1024];
                int bufferDadosObtidosRec = handler.Receive(bufferDadosObtidos);
                dados += Encoding.ASCII.GetString(bufferDadosObtidos, 0, bufferDadosObtidosRec);
                if (dados.IndexOf("<EOF>") > -1)
                {
                    break;
                }
            }

            //Libera a conexão
            handler.Shutdown(SocketShutdown.Both);
            handler.Close();

            Console.WriteLine("");
            Console.WriteLine("Mensagens recebidas em : {0}", DateTime.Now.ToString("dd/MM/yyyy hh:mm:ss.fff"));

            //Exibe a mensagem recebida se o parâmetro permitir
            if (ServidorOPC.exibeMensagem == 1)
                Console.WriteLine("Mensagens : {0}", dados);

            //Excluindo o EOF do final das mensagens HL7
            dados = dados.Substring(0, dados.Length - 5);

            //Separa cada mensagem HL7, obtem tag, timestamp e valor e envia para o Servidor OPC
```

```
            Console.WriteLine("");
            Console.WriteLine("Escrevendo na interface OPC...");
            while (dados.Length > 0)
            {
               //obtem posição final de uma mensagem HL7 (primeira)
               pos = dados.IndexOf("<EOL>");

               //separa a mensagem HL7
               msgHL7 = dados.Substring(0, pos);

               //Converte a mensagem HL7 em Tag, timestamp e valor
               converteHL7Tag(msgHL7);

               //Escreve a Tag, o valor e o timestamp na interface do servidor OPC
               DateTime timestamp = DateTime.ParseExact(timestampTag, ServidorOPC.strFormatoTimestamp, invC,
DateTimeStyles.AssumeLocal);
               m_ServidorOPC.AtualizaValorTag(tag,valTag, timestamp);
               Console.WriteLine("Tag={0}, timestamp={1}, valor={2}", tag, timestamp.ToString("dd/MM/yyyy hh:mm:ss.fff"), valTag);


               //Bloqueia a thread atual até receber um sinal de OK por um intervalo de espera
               FimServidorOPC = EventoServidorOPC.WaitOne(ServidorOPC.tempoMilisegundos, true);

               //Exclui a mensagem já convertida
               dados = dados.Substring(pos + 5);
            }
         }

         if (FimServidorOPC)
         {
            End = true;
         }

         // Libera a sincronização
         m_mutex.ReleaseMutex();
      }
      catch (Exception e)
      {
         Console.WriteLine(e.ToString());
      }
   }


   //Utiliza a biblioteca NHPITools para converter a mensagem HL7 em Tag, timestamp e valor
   private static void converteHL7Tag(string m)
   {

      //Define um atributo para realizar a analise da mensagem
      EnhancedModelClassFactory emch = new EnhancedModelClassFactory();
      PipeParser analisador = new PipeParser(emch);
      emch.ValidationContext = analisador.ValidationContext;

      //Mensagem HL7
      IMessage im = analisador.Parse(m);

      //Obtem a estrutura da mensagem e insere em um pacote genérico independente de versão HL7
      string estruturaMSG = im.GetStructureName();
      GenericMessageWrapper MensagemGenerica = im as GenericMessageWrapper;

      if (MensagemGenerica != null)
      {

         //Obtem o ID do paciente
         ISegment pid = MensagemGenerica.GetSegment<ISegment>("PID");
         CX cx = (CX)pid.GetField(2, 0);
         string sid = cx.ID.ToString();

         //Obtem o ID Observation Identifier (codigo) da medição
         ISegment pidMedicao = MensagemGenerica.GetSegment<ISegment>("OBX");
         CE ce = (CE)pidMedicao.GetField(3, 0);
         string sIDMedicao = ce.Identifier.ToString();

         //Formata o nome da Tag
         tag = sid + "-" + sIDMedicao;

         //Obtem o valor da medição
```

```
        Varies ce2 = (Varies)pidMedicao.GetField(5, 0);
        valTag = ce2.Data.ToString();

        //Obtem o timestamp da medição
        TS ts = (TS)pidMedicao.GetField(14, 0);
        timestampTag = ts.TimeOfAnEvent.ToString();

    }
}


//Inicia a thread para obtenção das mensagens, conversão e escrita no servidor OPC
public static void iniciaThreadHL7OPC()
{
    Thread TagThread = new Thread(new ThreadStart(obtemHL7ConverteOPC));
    TagThread.Start();
}

//Inicia o Servidor OPC
public static void iniciaServidorOPC()
{
    try
    {

        MyWin32.HandlerRoutine handlerRoutine = new MyWin32.HandlerRoutine(MyWin32.Handler);
        MyWin32.SetConsoleCtrlHandler(handlerRoutine,true);


        //    Cria a instância Servidor OPC
        criaServidorOPC();
        ServidorOPC servidor = ServidorOPC;
        servidor.ConfiguraInstancia();

        //Prepara o servidor OPC, se não conseguir finaliza
        MyCreator creator = new MyCreator();
        if (!ResultCode.SUCCEEDED(servidor.PreparaLicenca(creator)))
        {
            servidor.Finaliza();
            servidor = null;
            return;
        }

        //    Tenta fazer o registro ou desregistro do servidor OPC, se digitado no prompt de comando
        int result = (int)EnumResultCode.S_OK;
        string comando = Environment.CommandLine;
        result = servidor.ProcessaCommando(comando);

        if (result != (uint)EnumResultCode.S_OK)
        {
            if (result == (uint)EnumResultCode.S_FALSE)
            {
                //registro OK
                servidor.RegistraLog(
                    EnumTraceLevel.INF,
                    EnumTraceGroup.USER1,
                    "Console::Main",
                    "Registro realizado com sucesso.");
            }
            else
            {
                servidor.RegistraLog(
                    EnumTraceLevel.INF,
                    EnumTraceGroup.USER1,
                    "Console::Main",
                    "Faha no registro");
            }
            servidor.Finaliza();
            servidor = null;
            return;
        }

        //    Inicia o mecanismo de I/O do servidor OPC
        if (ResultCode.SUCCEEDED(servidor.Inicia()))
        {
            //                Gera as Tags do servidor
            servidor.GeraTags();
```

```
            //                Servidor OPC OK para receber conexões
            servidor.Preparado();
        }

        //   Inicia a Thread
        if (ResultCode.SUCCEEDED(result))
        {
            iniciaThreadHL7OPC();
        }
        Console.WriteLine("Pressione Ctrl-C para sair.\n");

        //Aguarda a chamada para thread
        while (!End)
        {
            Thread.Sleep(1);
        }

        // Finaliza
        TagFinal = true;
        Mutex.WaitOne();
        Mutex.ReleaseMutex();

        servidor.Para();
        servidor.Finaliza();
        servidor = null;
    }
    catch (Exception exc)
    {
        Console.WriteLine(exc.ToString());
    }
}


//Classe que representa a janela do console e controla os comandos do teclado.
public class MyWin32
{
    // Utiliza a função SetConsoleCtrlHandler para receber comandos
    [DllImport("Kernel32")]
    public static extern Boolean SetConsoleCtrlHandler(
        HandlerRoutine Handler,
        Boolean Add);

    public delegate Boolean HandlerRoutine(CtrlTypes CtrlType);

    // Tipos de mensagens de controle
    public enum CtrlTypes
    {
        CTRL_C_EVENT = 0,
        CTRL_BREAK_EVENT,
        CTRL_CLOSE_EVENT,
        CTRL_LOGOFF_EVENT = 5,
        CTRL_SHUTDOWN_EVENT
    }

    // Trata os comandos
    static public Boolean Handler(MyWin32.CtrlTypes CtrlType)
    {
        string message = string.Empty;

        switch (CtrlType)
        {
            case MyWin32.CtrlTypes.CTRL_C_EVENT:
            case MyWin32.CtrlTypes.CTRL_BREAK_EVENT:
            case MyWin32.CtrlTypes.CTRL_CLOSE_EVENT:
            case MyWin32.CtrlTypes.CTRL_LOGOFF_EVENT:
            case MyWin32.CtrlTypes.CTRL_SHUTDOWN_EVENT:
                message = "Finalização devido ao comando CTRL";
                End = true;
                break;
        }
        System.Console.WriteLine(message);
        return true;
    }
}
}
```

```
//Classe OPC Server baseada no OPC Toolkit da Softing OPC
public class ServidorOPC
{

        #region Construtor
        //-----------------
        public ServidorOPC(ref AutoResetEvent appEndEvent)
        {
                ServidorOPC.EventoFim = appEndEvent;
        }

        #endregion

  #region Atributos Protected
  //--------------------------


  //protected ShutdownHandler m_shutdownRequest = null; //Atributo para receber solicitações de shutdown
  //protected Random m_random = new Random(1);

  protected MyDaAddressSpaceElement[] ElementoTag = null; //Array de Tags
        protected static AutoResetEvent EventoFim = new AutoResetEvent(false); //Representa o evento de fim
        //--
        #endregion

        #region Atributos Publicos
        //----------------------

  public static int qtdPacientes = 10; //Quantidade de Pacientes para geração das Tags
  public static string[] strTiposSinais; //Tipos de Sinais vitais para geração das Tags
  public static int tempoMilisegundos = 1; //Tempo de espera para escrever um novo valor no Servidor OPC
  public static int maxConexoesSocket = 30; //Numero maximo de conexoes via socket
  public static string strFormatoTimestamp = "yyyyMMddHHmmss"; //formato do timestamp
  public static int exibeMensagem=0; //Se exibe as mensagens recebidas ou não


        #endregion

  #region Metodos Publicos
  //-----------------------

  //Shutdown no servidor
        public static int Shutdown()
        {
                ServidorOPC.EventoFim.Set();
                return (int)EnumResultCode.S_OK;
        }


  //Inicia a instância do servidor OPC e seus parâmetros
        public int ConfiguraInstancia()
        {
                try
                {
                        Application.Instance.VersionOtb = 440;
                        Application.Instance.AppType = EnumApplicationType.EXECUTABLE;

                        Application.Instance.ClsidDa = "{613FA801-78A4-46E7-A97F-AB7A1A4182F4}";
                        Application.Instance.ProgIdDa = "Optvision.HL72OPC.DA.1";
                Application.Instance.VerIndProgIdDa = "Optvision.HL72OPC.DA";
                Application.Instance.Description = "Optvision HL72OPC OPC Server";
                Application.Instance.MajorVersion = 4; //Maior número de versão do servidor
                Application.Instance.MinorVersion = 40; //Menor número de versão do servidor
                        Application.Instance.BuildNumber = 0;
                Application.Instance.VendorInfo = "Optvision";
                Application.Instance.MinUpdateRateDa = 1; //Mínimo tempo de atualização para um grupo (ms)
                Application.Instance.ClientCheckPeriod = 30000; //Periodo em ms para monitorar as conexões com os clientes
                        Application.Instance.AddressSpaceDelimiter = '.';
                        Application.Instance.PropertyDelimiter = '/';

                        Application.Instance.ShutdownRequest += new Softing.OPCToolbox.Server.ShutdownHandler(Shutdown);
                }
                catch(Exception exc)
```

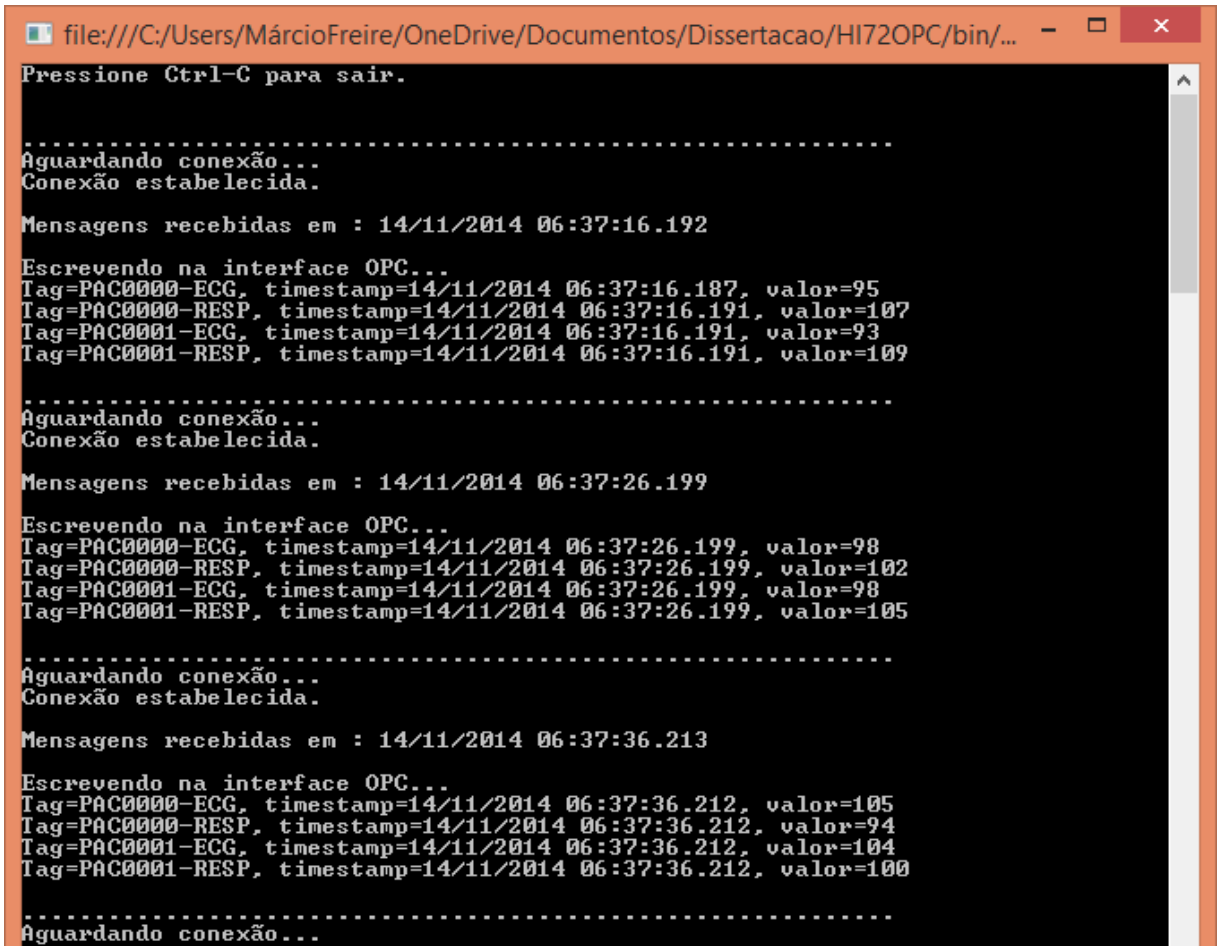
```
                {
                        RegistraLog(
                                EnumTraceLevel.ERR,
                                EnumTraceGroup.USER1,
                                "OpcServer::ConfiguraInstancia",
                                exc.ToString());

                        return (int)EnumResultCode.E_FAIL;
                }
                return (int)EnumResultCode.S_OK;
        }

    //Inicia servidor
        public int Inicia()
        {
                return Application.Instance.Start();
        }

    //Pára o servidor
        public int Para()
        {
                return Application.Instance.Stop();
        }

    //Servidor preparado
        public int Preparado()
        {
                return Application.Instance.Ready();
        }

    //Finaliza o servidor
        public int Finaliza()
        {
                return Application.Instance.Terminate();
        }


    //Ativa a licença adquirida, caso exista
        public int PreparaLicenca(MyCreator aMyCreator)
        {
                int resultado = (int)EnumResultCode.S_OK;

      //Ativa a licença adquirida, caso exista
                //      resultado = Application.Instance.Activate(EnumFeature.DA_SERVER, "XXXX-XXXX-XXXX-XXXX-
XXXX");
                if (!ResultCode.SUCCEEDED(resultado))
                {
                        return resultado;
                }

      resultado = Application.Instance.Initialize(aMyCreator);

                if (ResultCode.SUCCEEDED(resultado))
                {
                        Application.Instance.EnableTracing(
                                EnumTraceGroup.ALL,
                                EnumTraceGroup.ALL,
                                EnumTraceGroup.SERVER,
                                EnumTraceGroup.SERVER,
                                "Trace.txt",
                                1000000,
                                0);
                }

                return resultado;
        }

    //Processa um comando na linha de comando
        public int ProcessaCommando(string comando)
        {
                return Application.Instance.ProcessCommandLine(comando);
        }


    //Ler os parâmetros utilizados pelo programa
    private static void lerParametros()
```

```
{

   string linha;
   int i = 0, qtdTiposSinais;

   //Ler parametros
   System.IO.StreamReader arquivo = new System.IO.StreamReader(@"parametros.txt");
   while ((linha = arquivo.ReadLine()) != null)
   {
      switch (linha)
      {
         case "strTiposSinais:":
            qtdTiposSinais = Convert.ToInt32(arquivo.ReadLine()); //le quantidade de registros
            strTiposSinais = new string[qtdTiposSinais];
            // le os tipos de sinais
            for (i = 0; i < qtdTiposSinais; i++)
            {
               strTiposSinais[i] = arquivo.ReadLine();
            }
            break;

         case "qtdPacientes:":
            qtdPacientes = Convert.ToInt32(arquivo.ReadLine());
            break;
         case "tempoMilisegundos:":
            tempoMilisegundos = Convert.ToInt32(arquivo.ReadLine());
            break;
         case "maxConexoesSocket:":
            maxConexoesSocket = Convert.ToInt32(arquivo.ReadLine());
            break;
         case "formatoTimestamp:":
            strFormatoTimestamp = arquivo.ReadLine();
            break;
         case "exibeMensagem:":
            exibeMensagem = Convert.ToInt32(arquivo.ReadLine());
            break;
      }

   }

   arquivo.Close();
}


//Gera as Tags do servidor OPC
public int GeraTags()
         {
   string strIDPaciente;
   int i = 0, j = 0, qtdRegistros;

   //Ler os parâmetros utilizados
   lerParametros();

   //Quantidade de resgistros
   qtdRegistros = qtdPacientes * strTiposSinais.Length;

   try
                  {
                           MyCreator CriadorTags = (MyCreator)Application.Instance.Creator;

                           //              Elemento DA
                           DaAddressSpaceRoot DA = Application.Instance.DaAddressSpaceRoot;

      //Lista de Tags
      ElementoTag = new MyDaAddressSpaceElement[qtdRegistros];

      //Gera as Tags no formato PAC9999 - TipoSinalVital
      while (j < qtdPacientes)
      {
         strIDPaciente = "PAC" + j.ToString("0000");

         foreach (string strSinalVital in strTiposSinais)
         {
            ElementoTag[i] = (MyDaAddressSpaceElement)CriadorTags.CreateMyDaAddressSpaceElement();
            ElementoTag[i].Name = strIDPaciente + "-" + strSinalVital;
            ElementoTag[i].AccessRights = EnumAccessRights.READWRITEABLE; //Tag de escrita ou leitura
```

```
                ElementoTag[i].Datatype = typeof(System.Int32);
                ElementoTag[i].IoMode = EnumIoMode.POLL; //Mode de IO Síncrono ou assincrono de acordo com o cliente
                DA.AddChild(ElementoTag[i]);

                //Define a descrição da Tag
                DaProperty property = new DaProperty();
                property.Id = (int)EnumPropertyId.ITEM_DESCRIPTION;
                property.Name = "Descrição";
                property.Description = strSinalVital + " do paciente " + j.ToString("0000");
                property.ItemId = property.Name;
                property.Datatype = typeof(string);
                property.AccessRights = EnumAccessRights.READABLE;
                ElementoTag[i].AddProperty(property);

                i++;
              }
              j++;
            }

                        }
                        catch(Exception exc)
                        {
                                RegistraLog(
                                        EnumTraceLevel.ERR,
                                        EnumTraceGroup.USER1,
                                        "OpcServer:GeraTags",
                                        exc.ToString());
                                return (int)EnumResultCode.E_FAIL;
                        }

                        return (int)EnumResultCode.S_OK;
                }

        //Atualiza o valor da Tag
        public void AtualizaValorTag(string tag, object valor, DateTime timestamp)
        {
           int i = procuraTag(tag); //procura a Tag na lista de Tags do servidor
           if (i>=0)
              ElementoTag[i].ValueChanged(new ValueQT(valor,EnumQuality.GOOD,timestamp));

        }

        //Procura a Tag na lista de Tags do servidor
        private int procuraTag(string tag)
        {
           for (int i=0;i < ElementoTag.Length; i++)
              if (ElementoTag[i].Name == tag)
                 return (i);
           return (-1);
        }

        public void RegistraLog(
                        EnumTraceLevel traceLevel,
                        EnumTraceGroup traceGroup,
                        string objectID,
                        string message)
                {
                        Application.Instance.Trace(
                                traceLevel,
                                traceGroup,
                                objectID,
                                message);
                }
                #endregion

}
```

APÊNDICE D – EXEMPLO DE EXECUÇÃO DO SOFTWARE SERVIDOR HL7-OPC

Exemplo de execução onde são recebidos, por três vezes, dois valores de sinais vitais dos tipos RESP e ECG referentes aos pacientes PAC0000 e PAC0001. As mensagens são convertidas e os valores correspondentes as *tags* PAC0000-RESP, PAC0000-ECG, PAC0001-RESP e PAC0001-ECG são escritos na interface OPC juntamente com seus respectivos *timestamps*.



APÊNDICE E – PLANILHA DE EVIDÊNCIA DE TESTE DO CENÁRIO 1

Extrato da planilha de evidência de teste realizado no cenário 1 que ilustra a forma de tabulação dos dados utilizado no procedimento utilizado de validação da solução proposta.

Compativo Aquisições entre Cliente OPC, Plant Information e Gerador de Mensagens HL7

Granularidade:	5 seg
Tempo de Medição:	00:42:07
Qtd Medições:	487

Cliente OPC				Plant Information PAC0000-ECG		Gerador Msg HL7 SEM DUPLICADAS				Comparação valores
Tag	Timestamp		Valor	Timestamp	Valor	Tag	Timestamp		valor	Se A=B=C
PAC0000-ECG	27/11/2014	14:17:03	99	27-11-2014 14:17:03	99	PAC0000-ECG	27/11/2014	02:17:03	99	Valores são iguais
PAC0000-ECG	27/11/2014	14:17:08	104	27-11-2014 14:17:08	104	PAC0000-ECG	27/11/2014	02:17:08	104	Valores são iguais
PAC0000-ECG	27/11/2014	14:17:13	106	27-11-2014 14:17:13	106	PAC0000-ECG	27/11/2014	02:17:13	106	Valores são iguais
PAC0000-ECG	27/11/2014	14:17:18	110	27-11-2014 14:17:18	110	PAC0000-ECG	27/11/2014	02:17:18	110	Valores são iguais
PAC0000-ECG	27/11/2014	14:17:28	106	27-11-2014 14:17:28	106	PAC0000-ECG	27/11/2014	02:17:28	106	Valores são iguais
PAC0000-ECG	27/11/2014	14:17:33	104	27-11-2014 14:17:33	104	PAC0000-ECG	27/11/2014	02:17:33	104	Valores são iguais
PAC0000-ECG	27/11/2014	14:17:38	98	27-11-2014 14:17:38	98	PAC0000-ECG	27/11/2014	02:17:38	98	Valores são iguais
PAC0000-ECG	27/11/2014	14:17:43	92	27-11-2014 14:17:43	92	PAC0000-ECG	27/11/2014	02:17:43	92	Valores são iguais
PAC0000-ECG	27/11/2014	14:17:48	83	27-11-2014 14:17:48	83	PAC0000-ECG	27/11/2014	02:17:48	83	Valores são iguais
PAC0000-ECG	27/11/2014	14:17:53	76	27-11-2014 14:17:53	76	PAC0000-ECG	27/11/2014	02:17:53	76	Valores são iguais
PAC0000-ECG	27/11/2014	14:17:58	71	27-11-2014 14:17:58	71	PAC0000-ECG	27/11/2014	02:17:58	71	Valores são iguais
PAC0000-ECG	27/11/2014	14:18:03	70	27-11-2014 14:18:03	70	PAC0000-ECG	27/11/2014	02:18:03	70	Valores são iguais
PAC0000-ECG	27/11/2014	14:18:08	71	27-11-2014 14:18:08	71	PAC0000-ECG	27/11/2014	02:18:08	71	Valores são iguais
PAC0000-ECG	27/11/2014	14:18:13	75	27-11-2014 14:18:13	75	PAC0000-ECG	27/11/2014	02:18:13	75	Valores são iguais
PAC0000-ECG	27/11/2014	14:18:18	81	27-11-2014 14:18:18	81	PAC0000-ECG	27/11/2014	02:18:18	81	Valores são iguais
PAC0000-ECG	27/11/2014	14:18:23	85	27-11-2014 14:18:23	85	PAC0000-ECG	27/11/2014	02:18:23	85	Valores são iguais
PAC0000-ECG	27/11/2014	14:18:28	91	27-11-2014 14:18:28	91	PAC0000-ECG	27/11/2014	02:18:28	91	Valores são iguais
PAC0000-ECG	27/11/2014	14:18:33	100	27-11-2014 14:18:33	100	PAC0000-ECG	27/11/2014	02:18:33	100	Valores são iguais
PAC0000-ECG	27/11/2014	14:18:38	107	27-11-2014 14:18:38	107	PAC0000-ECG	27/11/2014	02:18:38	107	Valores são iguais
PAC0000-ECG	27/11/2014	14:18:43	109	27-11-2014 14:18:43	109	PAC0000-ECG	27/11/2014	02:18:43	109	Valores são iguais
PAC0000-ECG	27/11/2014	14:18:48	110	27-11-2014 14:18:48	110	PAC0000-ECG	27/11/2014	02:18:48	110	Valores são iguais
PAC0000-ECG	27/11/2014	14:18:58	108	27-11-2014 14:18:58	108	PAC0000-ECG	27/11/2014	02:18:58	108	Valores são iguais
PAC0000-ECG	27/11/2014	14:19:03	105	27-11-2014 14:19:03	105	PAC0000-ECG	27/11/2014	02:19:03	105	Valores são iguais
PAC0000-ECG	27/11/2014	14:19:08	98	27-11-2014 14:19:08	98	PAC0000-ECG	27/11/2014	02:19:08	98	Valores são iguais
PAC0000-ECG	27/11/2014	14:19:13	91	27-11-2014 14:19:13	91	PAC0000-ECG	27/11/2014	02:19:13	91	Valores são iguais
PAC0000-ECG	27/11/2014	14:19:18	87	27-11-2014 14:19:18	87	PAC0000-ECG	27/11/2014	02:19:18	87	Valores são iguais
PAC0000-ECG	27/11/2014	14:19:23	84	27-11-2014 14:19:23	84	PAC0000-ECG	27/11/2014	02:19:23	84	Valores são iguais