Analiza przydatności wybranych standardów do modelowania architektury systemu informatycznego dla służby zdrowia

T. GÓRSKI
e-mail:gorski@wat.edu.pl, tomasz.gorski@rightsolution.pl

Instytut Systemów Informatycznych
Wydział Cybernetyki WAT
ul. S. Kaliskiego 2, 00-908 Warszawa

Artykuł przedstawia wyniki analizy standardów BPMN, GELLO, UML, OCL, XML oraz HL7 w kontekście ich przydatności do modelowania architektury systemu informatycznego. Istotnym jest, że rozpatrywane standardy oceniane są pod kątem możliwości ich zastosowania do modelowania systemu informatycznego budowanego dla służby zdrowia. W artykule przedstawiono podstawowe cechy i własności każdego z rozpatrywanych standardów. Wskazano także własności każdego z rozpatrywanych standardów istotne z punktu widzenia architektury systemu informatycznego. W podsumowaniu zawarto rekomendację zestawu standardów do zastosowania przy modelowaniu architektury systemu informatycznego dla służby zdrowia.

Słowa kluczowe: architektura systemu informatycznego

1. Wprowadzenie

W artykule przedstawiono wyniki analizy standardów informatycznych w kontekście ich przydatności do modelowania architektury systemu informatycznego dla służby zdrowia. Analizie poddano następujące standardy:

- BPMN – Business Process Modelling Notation
- GELLO – Object-Oriented Query and Expression Language for Clinical Decision Support
- UML – Unified Modelling Language
- OCL – Object Constraint Language
- XML – Extensible Markup Language
- HL7 – Health Level Seven.

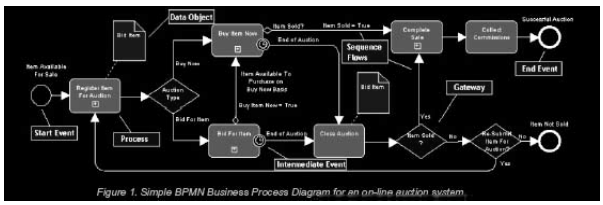
2. BPMN

BPMN jest standardem modelowania przebiegu procesów biznesowych, przepływów pomiędzy nimi oraz usług WWW. BPMN został opracowany przez organizację BPMI (ang. Business Process Management Institute). Jego głównym zadaniem jest dostarczenie sposobu opisu procesów i środowiska biznesowego zrozumiałego dla wszystkich użytkowników, od analityków biznesowych opracowujących początkowe zarysy procesów, aż po personel techniczny odpowiedzialny za wdrożenie technologii opracowanej na potrzeby tych procesów. Drugim, równie ważnym zadaniem, jest zapewnienie metody jednolitej wizualizacji języków wykonawczych procesów biznesowych wykorzystujących XML, przez zastosowanie wspólnej notacji.

BPMN opisuje jeden diagram procesów biznesowych zwany BPD (ang. Business Process Diagram). Został on opracowany do realizacji dwóch celów. Po pierwsze jest łatwy w zrozumieniu i stosowaniu. Można go wykorzystać do szybkiego modelowania procesów biznesowych i jest zrozumiały dla użytkowników pozbawionych umiejętności technicznych. Po drugie dostarcza możliwości modelowania skomplikowanych i złożonych procesów biznesowych oraz może być przełożony na język wykonawczy procesów biznesowych.

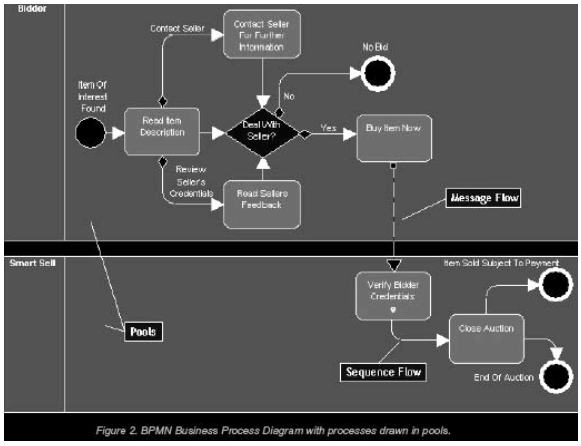
W procesie modelowania przebiegu procesu biznesowego wystarczy zamodelować zdarzenia powodujące rozpoczęcie procesu, następnie działania, które muszą zostać podjęte do realizacji procesu, a na końcu możliwe rezultaty przebiegu procesu. Decyzje oraz rozgałęzienia procesów modeluje się za pomocą węzłów decyzyjnych. Na rysunku rys.1. znajduje się przykład diagramu BPD.

Ponadto, proces może zawierać w sobie procesy zstępujące, które z kolei mogą być reprezentowane w innym diagramie BPD podłączonym za pomocą linku do dowolnego symbolu procesu macierzystego. Jeśli proces nie zawiera żadnych procesów zstępujących to jest uznawany za proces najniższego poziomu i nazywany zadaniem.



Rys. 1. Diagram BPD dla systemu aukcji on-line

Zagłębiając się dalej w analizę biznesową można umieszczać zdarzenia i procesy w cieniowanych obszarach zwanych pulami, określających kto realizuje proces (rys. 2.). Pula może być z kolei podzielona dalej na tory. Pula zazwyczaj reprezentuje organizację, podczas gdy tor, jej określony dział lub pion.



Rys. 2. Diagram BPD z procesami pogrupowanymi w pule

W trakcie modelowania procesów biznesowych niezbędne jest również modelowanie zdarzeń występujących w tych procesach i przed--stawienie sposobu, w jaki zdarzenia wpływają na przebieg procesów. Zdarzenie powoduje uruchomienie procesu, występuje w trakcie przebiegu procesu lub kończy proces. BPMN dostarcza wyraźnie różnicującej notacji dla każdego ze wspomnianych typów zdarzeń.

Przy modelowaniu bardziej złożonych przebiegów procesów, takich jak usługi WWW B2B pojawia się potrzeba modelowania bardziej skomplikowanych zdarzeń, takich jak wiadomości, zależności czasowe (ang. timers), zasady biznesowe (ang. business rules) i błędy. BPMN umożliwia określenie typu aktywatora zdarzenia. Określenie typu aktywatora zdarzenia nakłada określone ograniczenia na modelowany przebieg procesu. Przykładowo zależność czasowa nie może kończyć przebiegu procesu, a przepływ wiadomości może mieć miejsce wyłącznie pomiędzy zdarzeniami wiadomości. Narzędzie modelowania wspierające model BPMN powinno automatycznie wymusić te zależności modelowania, które są specjalnymi przypadkami reguł biznesowych.

Zdarzenia mogą występować w trakcie realizacji procesu, powodując przerwanie procesu i powodując uruchomienie nowego procesu. Ewentualnie proces kończy się powodując zdarzenie i pośrednio uruchomienie nowego procesu. Można modelować te zdarzenia pośrednie (ang. intermediate events) przez umieszczenie symbolu zdarzenia bezpośrednio na procesie, z którym jest ono związane.

Istnieją trzy typy procesów – proces, proces zstępujący i zadanie.

Ponieważ diagram BPD w specyfikacji BPMN jest głównie zaprojektowany tak, aby był łatwy w odbiorze, więc aby ułatwić odbiorcom zrozumienie złożoności procesów, można graficznie ją zaprezentować przez umieszczenie miniaturek potomnych diagramów BPD w symbolach procesów. Dzięki temu można, oglądając diagram BPD określić, które procesy są złożone i dzielą się na procesy zstępujące niższych poziomów.

Uwidocznienie kolejności realizacji procesów osiąga się poprzez połączenie ich przepływami sekwencyjnymi (ang. sequence flow). Przepływ sekwencyjny jest stosowany do określenia kolejności działań w organizacji lub dziale. Zatem, jeśli diagram jest pogrupowany w pule i tory, to przepływy sekwencyjne łączą zdarzenia, procesy i punkty decyzyjne w obrębie torów i pul. BPMN udostępnia drugi rodzaj linii przepływu do modelowania, tzn. przepływ wiadomości (ang. message flow), który pozwala określić kolejność działań pomiędzy organizacjami lub pulami.

Jednym z celów opracowania specyfikacji BPMN jest umożliwienie modelowania przepływu informacji w relacjach B2B. W tym celu, diagram BPD oferuje metodę modelowania przepływów wiadomości. Tradycyjne diagramy przepływów procesów biznesowych pozwalają na modelowanie sekwencyjnych przepływów procesów od zdarzeń rozpoczynających proces do rezultatów końcowych. Diagram BPD wspiera przepływy sekwencyjne za pomocą przepływów wiadomości, co pozwala zobrazować wysyłanie i otrzymywanie wiadomości pomiędzy ludźmi czy organizacjami. Stanowi to istotny element obrazowania i rozumienia relacji między przedsiębiorstwami (B2B) i między przedsiębiorstwem a klientem (B2C).

Specyfikacja ta została zaprojektowana tak, aby umożliwić łatwe modelowanie typowych procesów biznesowych, a jednocześnie pozwolić na modelowanie procesów o dowolnie dużym stopniu złożoności.

3. GELLO

GELLO jest zorientowanym obiektowo językiem, który jest zbudowany na istniejących standardach. GELLO obejmuje zarówno zapytania jak i języki wyrażeń. GELLO wykorzystuje język Object Constraint Language (OCL), opracowany przez Object Management Group. Istotne składniki OCL zostały wybrane i zintegrowane w zapytaniach i językach wyrażeń GELLO, aby dostarczyć odpowiednich podstaw do manipulacji danymi medycznymi w celu podejmowania decyzji w ochronie zdrowia. Język GELLO może być używany do:

- budowania zapytań do wydobywania i manipulowania danymi z historii choroby
- konstruowania kryteriów decyzyjnych przez tworzenie wyrażeń dla szczególnych danych cecha/wartość – te kryteriach mogą być używane w bazach wiedzy systemów wspomagania decyzji takich jak te zaprojektowane by dostarczyć ostrzeżeń i przypomnień, wskazówek, albo inna zasad decyzyjnych
- tworzenia wyrażeń, formuł, i zapytań dla innych aplikacji.

Zapytania i języki wyrażeń współdzielą model danych obiektowych dopóki wyrażenia przetwarzają dane dostarczone przez zapytania. Język zapytań został zaprojektowany w kontekście modelu zaproponowanego w HL7 CDSTC. Model ten proponuje wykorzystanie vMR, który dostarcza standardowego interfejsu do różnych systemów historii choroby. Język wyrażeń może być używany dla określania kryteriów decyzji albo abstrakcyjnych lub dostarczonych zbiorczych wartości. Obiektowe podejście pozwala na uzyskanie cechy rozszerzalności tego języka.

W celu ułatwienia procesu kodowania i oceny wyrażeń a co ważniejsze, maksymalizowania możliwości współdzielenia zapytań i wyrażeń, GELLO obejmuje podstawowe wbudowane typy danych. Ponadto dostarcza koniecznych mechanizmów dostępu do zasadniczego modelu danych.

Główną przeszkodą do dzielenia się wiedzą medyczną jest brak wspólnego formatu dla kodowania i manipulacji danymi. Pomimo, że składnia Arden Syntax zajęła się tym problemem przez odizolowywanie odniesień do lokalnych danych w nawiasach klamrowych w MLMs, to wciąż nie dostarcza mechanizmów dostępu do danych w sposób niezależny od formatu.

Wirtualna historia choroby (vMR), to obiektowe podejście zgodne z HL7 RIM, używające typowego modelu danych, jako pośrednika do różnych systemów historii choroby. Pojęcie vMR zostało zaproponowane, jako podstawowy model dla przetwarzania danych pacjentów w kontekście systemów wspomagania decyzji. vMR jest udoskonaleniem modelu: Reference Information Model (RIM). Podejście to dostarcza pierwszego podejścia w kierunku standardu wymiany, zarządzania i integracji danych medycznych. Jednakże, nadejście obiektowej wirtualnej karty choroby vMR jest niekompatybilne ze składnią Arden Syntax, ponieważ może ona przetwarzać tylko atomowe typy danych.

Faktem jest, że istnieje potrzeba języka do formułowania zapytań i wyrażeń w celu wydobywania i manipulowania danymi medycznymi. Język taki powinien być:

- niezależny od dostawcy
- niezależny od platformy
- zorientowany obiektowo i kompatybilny z vMR
- łatwy w czytaniu i używaniu
- rozszerzalny.

Najważniejsze cele i własności za--proponowanego języka GELLO, jako niezależnego od platformy standardu zapytań i wyrażeń, dla współdzielenia i manipulowania wiedzą w medycznym kontekście, to:

- GELLO jest skierowany dla medycznych aplikacji, które muszą korzystać z zapytań i języków wyrażeń dla wspomagania decyzji
- GELLO jest niezależny od dostawcy przez wykorzystywanie specyfikacji języka, który nie jest zależny od dostawcy
- GELLO jest niezależny od platformy
- GELLO dostarcza mechanizmów dostępu do danych przez obiektowy model danych, za pomocą ściśle typowanych wyrażeń, przez ogólnego przeznaczenia języki zapytań i wyrażeń
- GELLO jest językiem deklaratywnym
- GELLO jest rozszerzalny przez dodawanie nowych klas definiowanych przez użytkownika do podstawowego obiektowego modelu danych
- wszystkie metody manipulowania danymi muszą być wyraźnie określone w obiektowym modelu danych
- przez używanie określonego obiektowego modelu danych, takiego jak vMR, każda zasada decyzji albo wskazówka nie potrzebuje oddzielnego mechanizmu translacji elementów danych do/z środowisk dostarczających dane

- podejście zorientowane obiektowo umożliwia modularność, hermetyzację i rozszerzalność.

4. UML

Unified Modeling Language jest językiem do: wizualizacji, specyfikacji, dokumentowania i budowania systemu informatycznego. Dzięki zastosowaniu UML można w sposób graficzny przedstawić różne spojrzenia na system: strukturalne, dynamiczne. Tworząc modele UML dokonujemy specyfikacji wymagań na system, szczególnie stosując model przypadków użycia i diagramy aktywności. Modelując system za pomocą diagramów UML jednocześnie dokumentujemy decyzje projektowe w jednoznaczny i zrozumiały dla całego zespołu sposób.

W ramach UML 2.0 występują następujące typy diagramów zachowania (przypadków użycia, aktywności, sekwencji, przeglądu interakcji, komunikacji, stanów, czasowy) oraz statyczne (klas, obiektów, pakietów, komponentów, struktury połączeń, wdrożeniowy).

Wersja UML 2.0 została opisana w precyzyjny i zwięzły sposób w [1].

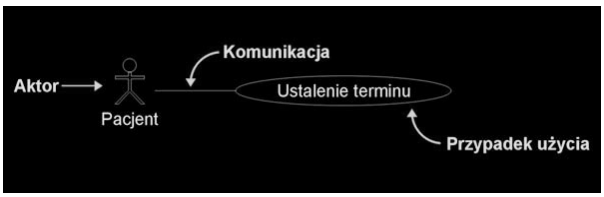
W kontekście architektury systemu informatycznego największe zastosowanie mają diagramy:

- Przypadków użycia
- Klas
- Sekwencji
- Komunikacji
- Aktywności
- Wdrożeniowy.

Diagramy przypadków użycia opisują, co robi system z punktu widzenia zewnętrznego obserwatora. Diagramy przypadków użycia pozostają w bliskim związku ze scenariuszami. Scenariusz to przykład tego, co się dzieje, kiedy ktoś wchodzi w interakcję z systemem. Oto scenariusz dla kliniki medycznej:

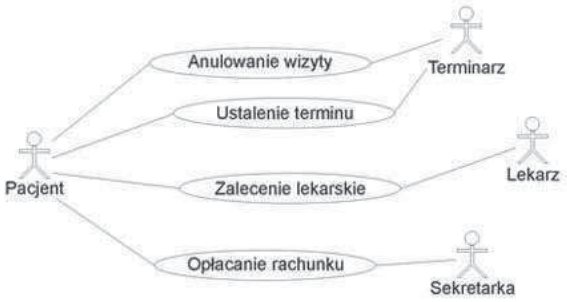
"Pacjent dzwoni do kliniki, aby umówić się na coroczne badania. Recepcjonistka znajduje najbliższy wolny termin w książce przyjęć i wyznacza badanie na tę datę".

Przypadek użycia to podsumowanie scenariuszy pojedynczego zadania lub celu. Aktor to ktoś albo coś, co inicjuje zdarzenia związane z tym zadaniem. Aktor po prostu określa rolę, którą odgrywa człowiek lub obiekt. Rys. 3. przedstawia przypadek użycia Ustalanie terminu badania w klinice medycznej. Aktor to Pacjent. Połączenie między aktorem a przypadkiem użycia to komunikacja.



Rys. 3. Przykład komunikacji między aktorem i przypadkiem użycia

Diagram przypadków użycia to zbiór aktorów, przypadków użycia i ich komunikacji (rys. 4.).



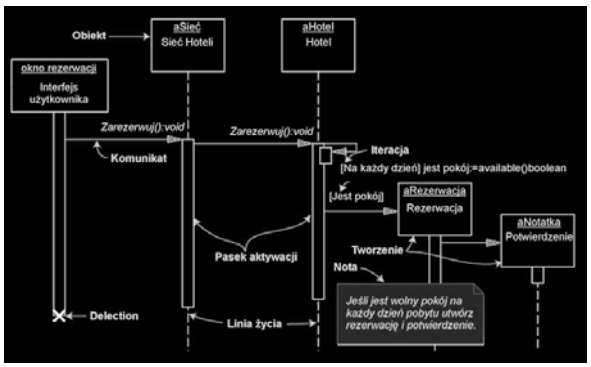
Rys. 4. Przykład diagramu przypadków użycia

Diagramy przypadków użycia mają trzy zastosowania:

- określanie funkcji (wymagań) - nowe przypadki użycia często generują nowe wymagania, kiedy system jest analizowany i projekt przybiera coraz wyraźniejszy kształt,
- komunikacja z klientami - prostota notacji sprawia, że diagramy przypadków użycia są dobrym sposobem porozumiewania się programistów z klientami,
- generowanie przypadków testowych - zbiór scenariuszy danego przypadku użycia może zasugerować sposoby testowania tych scenariuszy.

Diagram klas przedstawia ogólną panoramę systemu, pokazując klasy i ich wzajemne relacje. Diagramy klas są statyczne – pokazują elementy wchodzące w interakcje a nie działania zachodzące podczas interakcji.

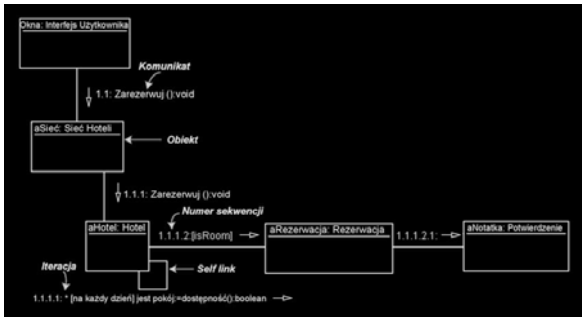
Diagram sekwencji to diagram interakcji, który szczegółowo pokazuje kolejność przekazywania komunikatów, a co za tym idzie, wywoływania operacji w obiektach. Na rys. 5. przedstawiono diagram sekwencji ilustrujący rezerwację hotelu. Obiektem inicjującym sekwencję komunikatów jest Okno rezerwacji.



Rys. 5. Przykład diagramu sekwencji

Każda przerywana pionowa linia to linia życia reprezentująca czas, przez który istnieje obiekt. Każda strzałka to przesłanie komunikatu. Strzałka zaczyna się od nadawcy, a kończy na pasku aktywności komunikatu na linii życia odbiorcy. Na powyższym diagramie Hotel wykonuje autowywołanie, aby sprawdzić, czy dysponuje wolnym pokojem. Jeśli tak, to Hotel tworzy Rezerwację i Potwierdzenie. Wyrażenie w nawiasie kwadratowym, [ ], to warunek.

Diagramy komunikacji to również diagramy interakcji. Dostarczają tych samych informacji co diagramy sekwencji, ale skupiają się na rolach obiektów, a nie na czasach przesyłania komunikatów. Przykład diagramu komunikacji znajduje się na rys. 6. Na diagramie sekwencji role obiektów są wierzchołkami, a komunikaty – liniami łączącymi wierzchołki.



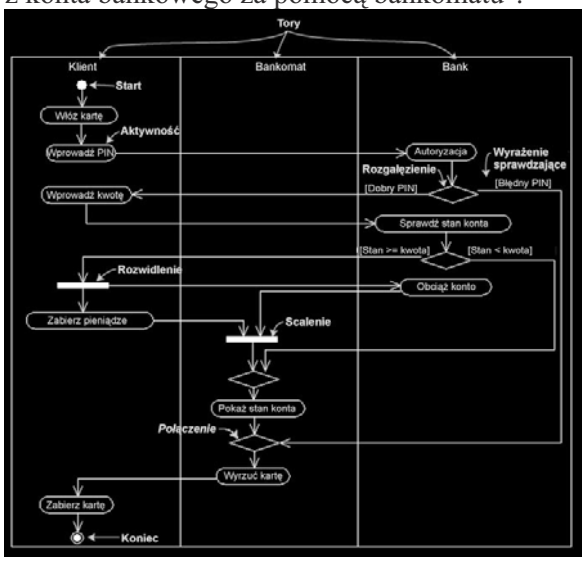
Rys. 6. Przykład diagramu komunikacji

Prostokąty opisujące rolę obiektu są oznaczone nazwami klas lub obiektów (albo obiema nazwami). Nazwy klas są poprzedzone dwukropkiem (:).

Każdy komunikat na diagramie komunikacji ma numer sekwencji. Komunikaty najwyższego poziomu mają numer 1. Komunikaty na tym samym poziomie (wysyłane podczas tego samego wywołania) mają ten sam przedrostek dziesiętny oraz przyrostki 1, 2 itd. w zależności od tego, kiedy występują.

Diagram aktywności to inaczej diagram przepływu. Diagram aktywności skupia się na obiekcie przechodzącym pewien proces (albo na procesie traktowanym jak obiekt). Diagram aktywności pokazuje wzajemne zależności między tymi operacjami.

Na rys. 7. przedstawiono przykład diagramu aktywności dla procesu "Pobieranie pieniędzy z konta bankowego za pomocą bankomatu".

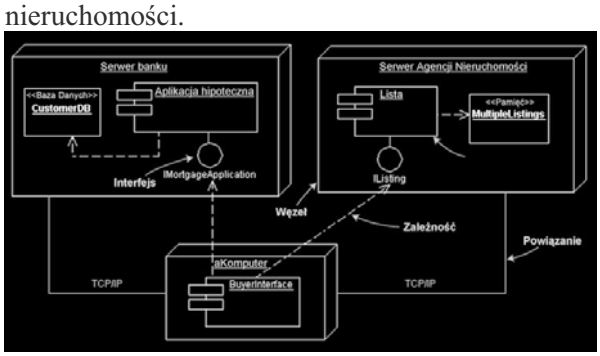


Rys. 7. Przykład diagramu aktywności

Diagramy aktywności można dzielić na tory obiektów. Tory określają, który obiekt jest odpowiedzialny za daną aktywność. Od każdej aktywności odchodzi przejście, łącząc ją z kolejną aktywnością.

Przejście może rozgałęziać się na dwa lub więcej wzajemnie wykluczających się przejść. Wyrażenia sprawdzające (w nawiasach kwadratowych [ ]) opisują przejścia wychodzące z rozgałęzienia. Rozgałęzienie i jego późniejsze scalenie, które wskazuje koniec rozgałęzienia, pojawiają się na diagramie jako puste romby.
Przejście może rozwidlać się na dwie lub więcej równoległych aktywności. Rozwidlenie i późniejsze połączenie wątków wychodzących z rozwidlenia jest przedstawione, na diagramie, jako grube poziome linie.

Diagramy wdrożeń pokazują fizyczną konfigurację oprogramowania i sprzętu.
Diagram wdrożenia (rys. 8.) pokazuje związek między komponentami programowymi i sprzętowymi związanymi ze sprzedażą nieruchomości.



Rys. 8. Przykład diagramu wdrożeniowego

Fizyczny sprzęt składa się z węzłów. Każdy komponent należy do węzła. Komponenty są przedstawione, jako prostokąty z dwoma zakładkami w lewym górnym rogu.

5. OCL

OCL jest językiem wyrażeń, umożliwiającym formułowanie ograniczeń dla modeli obiektowych i innych artefaktów powstałych w czasie modelowania obiektowego. Ograniczenie to restrykcja nałożona na jedną lub więcej wartości (części) modelu lub systemu obiektowego.

Bertrand Meyer, jeden z pionierów stosowania ograniczeń, nazywa ograniczenia asercjami (ang. assertion) i definiuje, jako „wyrażenie znaczenia elementu". Asercje w rozumieniu Meyera występują w trzech odmianach: warunki początkowe, warunki końcowe i niezmienniki.

Ian Graham używa zarówno terminu asercja, jak i reguła. Asercje to w jego rozumieniu „postać niezmiennika, warunku początkowego i warunku końcowego, który musi być spełniony, odpowiednio, gdy metoda jest wykonywana, wywoływana i zakończona". Natomiast pojęcie reguły określa sposób wyrażania wzajemnego oddziaływania obiektów".

James Rumbaugh definiuje ograniczenie, jako „zależność funkcjonalną między bytami modelu obiektowego". W opracowanej przez niego metodzie analizy i projektowania (OMT) ograniczenia zawężają zbiór możliwych wartości bytów modelu obiektowego.

Grady Booch także używa pojęcia ograniczenia, rozumiejąc je, jako „wyrażenie opisujące warunek znaczeniowy, który musi być spełniony". Podkreśla, że ograniczenie może być spełnione tylko wtedy, kiedy system jest w stanie stabilnym. Może dojść do przejściowych sytuacji, kiedy ograniczenia nałożone na system nie będą obowiązywały.

Język OCL pomaga wyrazić wspólny element wszystkich definicji i ustalić zrozumiały i łatwy w użyciu standard umożliwiający projektantowi specyfikowanie tego co niezbędne.

Za pomocą warunków początkowych i końcowych można skutecznie określać operacje i metody. Używając wyrażeń OCL wewnątrz modelu UML, możemy wyspecyfikować warunki początkowe i końcowe operacji oraz metod dla wszystkich klas, typów i interfejsów. Zasada kryjąca się za takim sposobem postępowania nosi nazwę „projektowania według umowy" (ang. design by contract). Może być stosowania w dowolnej metodzie obiektowej. Jest ponadto jednym z podstawowych paradygmatów inżynierii oprogramowania.

W terminologii obiektowej umowa jest sposobem jasnego i jednoznacznego przydzielania obiektowi odpowiedzialności (ang. responsibilities). Obiekt jest zobowiązany do wykonania usługi wtedy i tylko wtedy, kiedy są spełnione warunki. Umowa jest dokładną specyfikacją interfejsu obiektu, nazywanego dostawcą (ang. supplier). Natomiast obiekty, korzystające z oferowanych przez dostawcę usług, są nazywane klientami (ang. clients) lub konsumentami (ang. consumers).

Na interfejs obiektu składa się pewna liczba wykonywalnych operacji. Prawa obiektu-dostawcy są określane w postaci warunków początkowych, które muszą być spełnione tuż przed wywołaniem operacji. Obowiązki specyfikują warunki końcowe, które muszą być prawdziwe zaraz po zakończeniu operacji. Niedotrzymanie warunków początkowego lub końcowego oznacza naruszenie umowy. Warunki początkowe i końcowe są uzupełniane trzecim rodzajem ograniczenia: niezmiennikiem – restrykcją ściśle związaną z klasami, typami i interfejsami.

Niezmiennik jest ograniczeniem, które musi być zawsze spełnione przez wszystkie egzemplarze klas, typów lub interfejsów. Warunki początkowe i końcowe muszą być prawdziwe tylko w określonej chwili – odpowiednio przed i po wykonaniu operacji.

W języku UML powyższe trzy postacie ograniczeń zdefiniowano, jako standardowe stereotypy: <<invariant>>, <<precondition>>, <<postcondition>>.

Ograniczenia są jednoznaczne i zwiększają dokładność modelu lub systemu, do którego się odnoszą.

Gdy stosujemy język OCL do wyrażania ograniczeń, możemy używać analizatora składni OCL, aby upewnić się, co do poprawności sformułowania ograniczeń nałożonych na model. Taka weryfikacja pomaga opracować spójny i precyzyjny model lub system.

Ograniczenia wyrażane w języku deklaratywnym nie mają wpływu na stan systemu. Innymi słowy, stan systemu nie ulega zmianie z powodu takiej a nie innej wartości wyrażenia.

Dzięki kontroli typów wyrażenia OCL mogą być wartościowane podczas modelowania, przed uruchomieniem systemu.

Ograniczenia OCL służą podczas modelowania i specyfikowania, lecz nie są

opisem implementacji. Określają tylko, jakie twierdzenia są prawdziwe w wyidealizowanym, bezbłędnym systemie. Język OCL nie umożliwia wyrażania działań, które powinny zostać podjęte w wypadku naruszenia ograniczenia w niepoprawnej realizacji systemu.

6. XML

XML jest językiem utworzonym przez World Wide Web Consortium (W3C). W XML można tworzyć własne znaczniki, budując nowe języki. W HTML wszystkie znaczniki są narzucone. XML jest metajęzykiem znaczników, gdyż pozwala tworzyć nowe języki znacznikowe.
XML jest popularny z wielu powodów. Najważniejszym z nich jest łatwość obsługi i wymiany danych za pośrednictwem XML.

W XML dane i znaczniki przechowywane są w postaci tekstu, którego postać można określić. Do tworzenia dokumentów XML można używać edytorów XML. Dane nie są też kodowane w żaden sposób objęty patentami czy innymi ograniczeniami, więc są łatwiej dostępne.

Dokumenty XML same się opisują. Wykorzystując nazwy nadane poszczególnym elementom, możemy się domyślić, jakie dane zawierają poszczególne fragmenty dokumentu. Oznacza to, że dokumenty XML w znacznej mierze same się dokumentują (niezależnie od tego możliwe jest wstawianie do plików XML komentarzy).

Kolejną zaletą XML jest możliwość określenia nie tylko danych, ale także ich struktury. Jest to ważne szczególnie wtedy, gdy mamy do czynienia ze złożonymi, ważnymi danymi. Można na przykład długą transakcję bankową zapisać, jako HTML, ale w XML można także zapisać reguły semantyczne opisujące strukturę dokumentu, aby można było sprawdzić poprawność takiego dokumentu.

W XML na poprawność dokumentów kładziony jest duży nacisk. Przeglądarki XML przeprowadzają dwie kontrole: pierwsza polega na sprawdzeniu, czy dokument jest poprawnie sformułowany, druga kontrola nazywana jest walidacją.

Dokument, który jest poprawnie sformułowany musi spełniać wymagania składniowe stawiane przez utworzoną przez W3C specyfikację XML 1.0. Każdy element musi być całkowicie zamknięty w elementach nadrzędnych względem niego.

Większość przeglądarek sprawdza, czy dokumenty są poprawnie sformułowane, niektóre natomiast przeprowadzają jeszcze walidację. Dokument XML można walidować, jeśli związana jest z nim definicja typu dokumentu (DTD) i kiedy dokument jest z nią zgodny. DTD dokumentu określa jego prawidłową składnię.

Do określenia wyglądu dokumentu XML można wykorzystać dwa narzędzia: arkusze CSS lub XSL. Standard CSS używany jest z HTML i obsługiwany przez liczne narzędzia. Za jego pomocą można określić formatowanie poszczególnych elementów, stworzyć klasy stylów, definiować czcionki, wybierać kolory, a nawet określać rozmieszczenie elementów na stronie.

Z kolei XSL jest zdecydowanie lepszy do obsługi XML, gdyż jest znacznie silniejszym narzędziem (zresztą same arkusze XSL są poprawnie sformatowanymi dokumentami XML). Dokumenty XSL składają się z reguł dotyczących dokumentów XML. Jeśli wzorzec reguły XSL pasuje do elementu XML, reguła ta przekształca dopasowany fragment kodu na inny. W ten sposób można nawet przekształcić kod XML na HTML.

Znaczniki w dokumencie określają jego strukturę. Są to:

- znaczniki początkowe
- znaczniki końcowe
- znaczniki elementów pustych
- odwołania do encji
- odwołania do encji znakowych
- komentarze
- ograniczniki sekcji CDATA
- deklaracje typu dokumentu
- instrukcje przetwarzania.

Zatem dane znakowe w dokumencie XML to wszystkie napisy niebędące znacznikami.

Strukturę dokumentu XML określa się za pomocą znaczników wyznaczających elementy. Element XML składa się ze znacznika początkowego, znacznika końcowego oraz treści, czyli tego, co się między tymi znacznikami znajduje. Wyjątkiem są elementy puste, które mogą składać się tylko z jednego znacznika o specyficznej konstrukcji.

Poprawnie sformułowany dokument XML musi zawierać jeden element, który będzie zawierał wszystkie inne elementy – jest to element główny. Element ten jest w dokumentach XML bardzo ważny, szczególnie z punktu widzenia programisty, gdyż parsowanie zawsze zaczyna się od niego.

Zgodnie ze specyfikacją XML 1.0 nazwy atrybutów podlegają tym samym regułom, które dotyczą nazw elementów.

Znaczniki zawsze są tekstem, więc i wartości atrybutów także są tekstem. Nawet,

jeśli atrybutowi przypisana jest liczba, to będzie ona traktowana, jako napis, który należy podawać w cudzysłowie.

Jednym z wymagań specyfikacji XML jest to, że nazwa atrybutu nie może pojawić się więcej niż raz w jednym znaczniku początkowym, lub znaczniku elementu pustego.
W XML istnieje pięć predefiniowanych odwołań do encji. Odwołanie takie jest umieszczane tam, gdzie wystąpić ma w przetwarzanym dokumencie sama encja.

W XML istnieje pięć predefiniowanych encji ogólnych podmienianych podczas parsowania dokumentu na odpowiednie znaki:

- &amp - znak &
- &lt - znak mniejszości <
- &gt - znak większości >
- &apos - apostrof '
- &quo - cudzysłów podwójny prosty ".

W XML wprawdzie jest tylko pięć predefiniowanych encji, ale można definiować nowe. Dokonuje się tego wykorzystując DTD.
W XML łatwo można definiować nowe znaczniki, jednak w miarę powstawania coraz liczniejszych aplikacji, pojawiać się będzie problem nieprzewidziany przez twórców pierwotnej wersji specyfikacji XML: konflikty nazw znaczników.

Rozwiązaniem tego problemu są przestrzenie nazw. Umożliwiają one uniknięcie konfliktów między poszczególnymi zbiorami znaczników.

7. HL7

Standard opracowywany przez grupę wolontariuszy skupionych w Health Level Seven (HL7) obejmuje swym zakresem wymianę informacji wewnątrzszpitalnej. Profesjonalnie przygotowane opracowanie standardu, przy niezbyt długim okresie funkcjonowania wskazuje na HL7 jako wzorcową grupę standaryzującą. Nazwa grupy związana jest z modelem Open System Interconnection (OSI) opracowanym przez International Standard Organization (ISO) – poziom 7.

Standard HL7 obejmuje swym zakresem definicje komunikatów przesyłanych pomiędzy poszczególnymi modułami systemu szpitalnego wskutek zaistnienia zdarzenia, o którym powinny zostać poinformowane pozostałe moduły.

Poszczególne podrozdziały dokumentu opisują następujące zagadnienia:

- ogólna struktura i zasady dotyczące przesyłanych komunikatów w standardzie HL7
- przyjęcie pacjenta, wypisanie, przeniesienie oraz rejestrację
- obsługa zleceń
- system rozliczania kosztów
- obserwacja kliniczna pacjenta
- ogólny interfejs służący do synchronizacji wspólnych zbiorów danych, np. zbiór lekarzy, zestawów badań, testów laboratoryjnych, etc.

Przy opracowywaniu zestawu wspólnych komunikatów grupa Health Level Seven założyła następujące cele:

- standard powinien wspomagać wymianę danych bez względu na wykorzystywane środowisko techniczne
- standard powinien wspomagać zarówno pełen (teoretycznie) model OSI do 7. poziomu jak i prymitywną wymianę na poziomie połączenia „punkt do punktu” za pomocą złącza RS-232C
- natychmiastowy transfer pojedynczych transakcji musi być wspomagany plikiem transferującym wiele transakcji
- standard powinien dawać możliwość uwzględniania specyfiki danego miejsca (segment typu Z w standardzie HL7)
- opracowane rozwiązanie nie może blokować naturalnego wzrostu wymagań środowiska medycznego
- standard powinien być opracowany zgodnie z obowiązującymi i występującymi na rynku protokołami transmisji bez szczególnego faworyzowania jakiegoś rozwiązania
- priorytetowym działaniem jest współpraca z innymi grupami standaryzacyjnymi np. ACR/NEMA DICOM, ASC X12, ASTM, IEEE/MEDIX, NCPDP na bazie grupy ANSI HISPP (Health Information Systems Planning Panel).

Można przedstawić następujący paradygmat odpowiedzi (w procesie wymiany uczestniczą aplikacje A i B):

- aplikacja A wysyła komunikat do B, który bez problemów odbiera go, o czym powiadamia moduł wysyłający A
- aplikacja B, po otrzymaniu komunikatu od A, stwierdza błąd aplikacji, o czym powiadamia moduł A
- aplikacja B może odrzucić komunikat, o czym także powiadamia bezzwłocznie moduł wysyłający komunikat A
- moduł odbierający B może wstrzymać wykorzystanie komunikatu od A, wysyła wówczas dwa powiadomienia, jedno o fakcie wstrzymania, a drugie o fizycznym wykorzystaniu komunikatu.

Z pojęciem przesyłanych komunikatów związane są tzw. abstrakty, które są zapisem transakcji pomiędzy modułami. Abstrakty zawierają specyfikację typu przesyłanych informacji, kiedy zostały wysłane oraz wyspecyfikowane zostają warunki wystąpienia błędów.

Informacją nadrzędną jest komunikat, który jest:

- zestawem danych transferowanych pomiędzy systemami
- każdy komunikat jest grupą złożoną z segmentów w ściśle określonym porządku
- każdy komunikat posiada swój trzyznakowy identyfikator określający jego typ.

Zdefiniowane są typy komunikatów, np.:

- ACK General acknowledgement message (ogólne potwierdzenie)
- ADT message (komunikaty przyjęciowo – wypisowe)
- ARD Ancillary RPT (display) (podrzędny RPT)
- BAR Add/change billing account (dodanie/zmiana konta rachunku)
- DFT Detail financial transaction (transakcja finansowa)
- DSR Display response (podanie odpowiedzi)
- MCF Delayed acknowledgement (opóźnione potwierdzenie)
- ORF Observation result/record resp. (wynik/zapis badania)
- ORM Order message (komunikat zlecenia),
- ORR Order acknowledgement message (potwierdzenie zlecenia)
- ORU Observation result/unsolicited (wynik dobrowolnego badania)
- OSQ Order status query (kolejka stanu zleceń)
- RAR Pharmacy Administration information (apteka – administracja)
- RAS Pharmacy Administration message (apteka – komunikaty)
- RDE Pharmacy encoded order message (apteka – zlecenia)
- QRY Query (kolejka)
- UDM Unsolicited display message (dobrowolne komunikaty).

Segmentem nazywany jest logicznie pogrupowany zestaw pól danych, który:

- może być opcjonalny lub obligatoryjny
- może wystąpić jeden raz lub wiele (dopuszczalne powtórzenia)
- każdy segment ma swój trzyznakowy identyfikator, np. dla typu ADT:
  - MSH nagłówek komunikatu
  - EVN typ zdarzenia
  - PID identyfikator pacjenta.

Każdy segment składa się z atrybutów (pól), które są:

- uporządkowanym łańcuchem (ciągiem) znaków
- o każdym polu wysyłane są następujące (przykładowo) informacje:
  - położenie w segmencie
  - unikalna, globalna nazwa
  - identyfikator (small integer)
  - maksymalna długość pola
  - opcjonalność
  - powtarzalność.

Standard HL7 definiuje zdarzenia związane z ADT, które generują przepływ komunikatów, np.:

- A01 Przyjęcie pacjenta
- A02 Przeniesienie pacjenta
- A03 Wypisanie pacjenta
- A04 Rejestracja pacjenta
- ...
- A28 Dodanie informacji personalnych pacjenta
- ...
- A37 Rozłączenia połączenia.

Postać komunikatu zaprezentowana została na przykładzie zdarzenia o kodzie A01 (przyjęcie pacjenta):

MSH nagłówek komunikatu
EVN typ zdarzenia
PID identyfikacja pacjenta
[{ NK1 }] najbliższy krewny
PV1 wizyta pacjenta
[PV2] wizyta pacjenta – uzupełnienia
[{ OBX }] informacje o stanie zdrowia
[{ AL1 }] alergie
[{ DG1 }] diagnoza
[{ PR1 }] procedury
[{ GT1 }] dane poręczyciela
[
{ IN1 informacje o ubezpieczeniu
[IN2] informacje o ubezpieczeniu
[IN3] informacje o ubezpieczeniu
}
]
[ACC]dane o wypadku
[UB1] dane kosztowe – rachunki
[UB2] dane kosztowe – rachunki

Do prezentacji użyta została notacja BNF, w której jeżeli identyfikator występuje w nawiasach kwadratowych, wówczas jego wystąpienie jest opcjonalne, gdy w klamrowych może wystąpić jedno lub więcej powtórzeń, bez nawiasów – musi wystąpić).

Każdy z wykorzystanych w zdarzeniu A01 identyfikatorów kryje w sobie segment danych,

np. segment PID wykorzystywany przez wszystkie aplikacje, jako podstawa do transferu danych o pacjencie składa się z atrybutów (pól). Atrybut opisany jest przez informacje o jego długości, typie, ewentualnych ograniczeniach, co do czytania, może składać się z sekwencji pojedynczych pól oddzielonych separatorami.
W ramach segmentu PID występuje atrybut: Patient Name (PN) o wewnętrznym kodzie 00108, który składa się z następujących elementów:
<nazwisko>^<imię>^<wewnętrzny inicjał lub imię>^<sufix>^ prefix>^ <stopień naukowy>
Przykład komunikatu o przyjęciu pacjenta na oddział (zdarzenie A01) w standardzie Health Level Seven:
MSH|^~\&|REGADT|MCM|LABADT|MCM|19
9512171205|SECURITY|ADT^A01|MSG00001|
P|2.2|<cr>
EVN|01|199512171205||<cr>
PID|||PATID1234^5^M11||MALINOWSKI^JA
N^A^JR||19610615|M||C|1200 N ELM
STREET^ ...GREENSBORO^NC^27401
1020|GL|(919)379 1212|(919)271 3434||S||
...PATID12345001^2^M10|123456789|987654^
NC|<cr>
NK1|KOWALSKA^ANNA^K|WIFE|<cr>
PV1|1|I|2000^2012^01||||004777^NOWAK^KA
ZIMIERZ^J.|||SUR||||ADM|A0|<cr>
co oznacza:
„Pacjent Jan A. Malinowski, Jr. został przyjęty 17 grudnia 1995 roku, o godzinie 12.05 przez lekarza Kazimierza Nowaka (o identyfikatorze 004777) na oddział chirurgiczny (SUR). Został umieszczony w pokoju nr 2012, na łóżku 01, na odcinku pielęgniarskim 2000".

Istotnym elementem każdego standardu wymiany międzymodułowej jest jego warstwa transportowa, na którą w uproszczeniu nałożono następujące warunki:

- wyeliminowanie błędów podczas transportu
- gwarancja konwersji typów kodowania
- zagwarantowana dostępność komunikatów o nieograniczonej długości.

Standard HL7 określa trzy poziomy transportowe w zależności od tego, w jakim środowisku został on zaimplementowany. Do tej pory analizowany był poziom komunikatów (najwyższy), poniżej jest poziom zasad kodowania (ang. encoding rules) a najniższy to poziom protokołu (ang. lower level protocol).
Warstwa komunikatów HL7 została opisana powyżej, warto dodać listę typów danych wykorzystywanych przez standard:

- ST łańcuch
- TX dane tekstowe
- FT formatowany tekst
- NM wartości numeryczne
- DT data
- TM czas
- TS symbol czasu
- PN pacjent
- TN telefon
- AD adres
- CK złożony identyfikator (z liczbą kontrolną)
- CN złożony identyfikator (numer + nazwa),
- CE zakodowany element
- RP wskaźnik.

Dodatek A Standardu HL7 dokładnie prezentuje listę wszystkich wykorzystywanych słowników.
Zasady kodowania komunikatów Standardu HL7 określają:

- zasady kodowania komunikatów do postaci ciągu znaków ASCII
- zasady dekodowania komunikatu przez aplikację odbierającą
- zmienną długość pól bazującą na znakach specjalnych.

HL7 3.0 wprowadza szereg modeli jednostek i aktywności występujących w służbie zdrowia i na tej podstawie odwzorowuje je na wiadomości. Podstawowym, opracowanym przez HL7, referencyjnym modelem jest obiektowy model informacji w służbie zdrowia o nazwie RIM (ang. Reference Information Model). Model ten przez zastosowanie języka modelowania obiektowego UML przedstawia 123 klasy obiektów, możliwe stany oraz relacje pomiędzy obiektami. Uproszczeniem tego modelu jest model informacyjny wiadomości – MIM (ang. Message Information Model) ujmujący te klasy, których obiekty są odwzorowywane na wiadomości. Dla każdego obiektu można prześledzić trasę jego powiązań poczynając od obiektu-korzenia. Wszystkie relacje dla danego typu obiektu tworzą określone drzewo (diagram) atrybutów i stanów związanych z modelowanym zjawiskiem (np. z pacjentem). Stworzony w ten sposób diagram jest określanym mianem MOD (ang. Message Object Diagram). Bazując na diagramie tworzony jest hierarchiczny zapis atrybutów i relacji obiektu (HMD – Hierarchical Message Description), który jest wprost rzutowany na elementy wiadomości.

Rys. 9. Proces formowania wiadomości w HL7 wersja 3.0

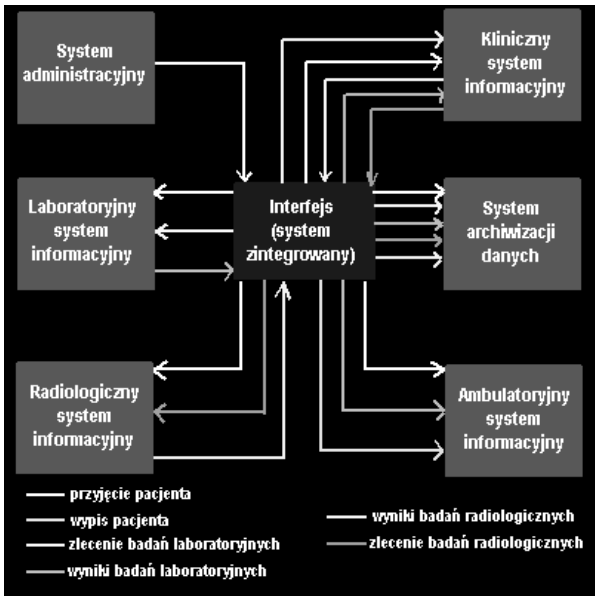
Cały proces od określenia potrzeb i stanu zjawisk poprzez modelowanie do implementacji obiektów w formie wiadomości określany jest w HL7 MDF (Message Development Framework). HL7 opracowując wersje 3.0 pragnie powiązać organizację gromadzenia i reprezentacji danych w służbie zdrowia z wymianą danych. Opracowany model referencyjny RIM ma więc charakter uniwersalny i może być wykorzystywany w tworzeniu systemów informacyjnych służby zdrowia. Co więcej, samo kodowanie treści wiadomości może odbywać się za pomocą różnych metod, jak np. ER7 (encoding rules 7), XML (eXtended Markup Language) czy EDIFACT.

Struktura wiadomości posiada odmienną formę od tych formowanych do wersji 2.3.1 włącznie. Pojawiają się zupełnie nowe segmenty (np. ENC – encounter – spotkanie) oraz oznaczania list (BL – begin list – początek listy, EL – end list – koniec listy) grup (BG – begin group – początek grupy, EG – end group – koniec grupy) i innych elementów składni.

Znaczny wysiłek prac w grupie HL7 jest skierowany na stworzenie implementacji wiadomości opracowywanej nowej normy HL7 3.0 w formie dokumentów XML. Możliwe do stworzenia definicje typów dokumentów (DTD – Document Type Definitions) stanowią oddzielne względem plików XML formy dokumentów. Możliwe jest stworzenie różnych form dokumentów np. dla różnych wiadomości.

Wraz z opracowywaniem nowego standardu wiadomości, modelu referencyjnego oraz typów dokumentów XML dla wiadomości HL7 postanowiło opracować architekturę elektronicznych kart pacjenta. Bazując na modelu referencyjnym RIM opracowano szereg form dokumentów DTD oraz procedur ich wymiany. Przedsięwzięcie to nazwano PRA – Patient Record Architecture. Ponieważ dokumenty PRA są implementacją XML stanowią opracowania niezależne od urządzeń, czy rozwiązań sprzętowo-programowych.

HL7 wersja 3.0, formy dokumentów DTD, PRA i inne opracowania związane z nowym sposobem reprezentacji zjawisk w medycynie są przygotowane i czekają na ostateczne ich zaakceptowanie jako norm. Istniejące i tworzone rozwiązania Szpitalnych Systemów Informacyjnych umożliwiać mogą różnorodne formatowanie wiadomości w procesie ich wymiany np. zgodnie z HL7 2.x, albo HL7 3.0 lub w formie PRA. Od możliwości interpretacji wymienianych wiadomości (interfejsy) zależy więc funkcjonalność całego systemu informacyjnego.



Rys. 10. Przykład przepływu wiadomości między systemami informacyjnymi służby zdrowia.

8. Podsumowanie

BPMN został opracowany do realizacji dwóch celów. Po pierwsze jest łatwy w zrozumieniu i stosowaniu. Można go wykorzystać do szybkiego modelowania procesów biznesowych i jest łatwy w zrozumieniu dla użytkowników pozbawionych umiejętności technicznych. Po drugie dostarcza możliwości modelowania skomplikowanych i złożonych procesów biznesowych i może być bez problemu przełożony na dowolny język wykonawczy procesów biznesowych. Istnieje możliwość przekształcenia modelu BPMN na model UML i wykorzystania go do budowy systemu informatycznego.

W kontekście architektury systemu informatycznego za pomocą języka UML modelujemy zarówno statyczny jak i dynamiczny aspekt systemu.

Z wykorzystaniem języka UML modelujemy złożone systemy za pomocą małego zbioru prawie niezależnych modeli. Tworzymy modele, które mogą być konstruowane i przetwarzane oddzielnie, ale są nadal wzajemnie powiązane.

Aby zrozumieć architekturę systemów zorientowanych obiektowo, potrzebujemy połączyć kilka uzupełniających się widoków: widok przypadków użycia, widok logiczny, widok procesów, widok implementacyjny, widok wdrożeniowy.

UML jest używany także do modelowania procesów biznesowych i reprezentowania struktur organizacyjnych.

Za pomocą języka wyrażeń OCL możemy uszczegóławiać modele UML, dodając warunki początkowe, warunki końcowe i niezmienniki dla klas i metod w modelu.

Język XML jest standardem, który w dużym stopniu stał się medium dla opisywania, komunikowania i realizowania strategii zarządzania informacją.

XML jest popularny z wielu powodów. Najważniejsza z nich jest łatwość obsługi i wymiany danych za pośrednictwem XML. Język XML wykorzystywany jest do strukturyzacji danych. Tak przygotowane dane mogą być zapisywane w postaci plików (dokumentów) i wykorzystywane w innych systemach informatycznych. XML możemy także zastosować do strukturyzacji danych wysyłanych w komunikatach między systemami informatycznymi lub ich modułami.

HL7 w wersji 3.0 to zarówno standard kodowania wiadomości (także w języku XML) jak i model referencyjny RIM i dokumenty PRA. Opracowany model referencyjny RIM ma charakter uniwersalny i może być wykorzystywany w tworzeniu systemów informacyjnych służby zdrowia. Wraz z opracowywaniem nowego standardu wiadomości, modelu referencyjnego oraz typów dokumentów XML dla wiadomości, HL7 opracowało architekturę „elektronicznych kart pacjenta". Bazując na modelu referencyjnym RIM opracowano szereg form dokumentów DTD oraz procedur ich wymiany. Przedsięwzięcie to nazwano PRA – Patient Record Architecture. Ponieważ dokumenty PRA są implementacją XML stanowią opracowania niezależne od urządzeń, czy rozwiązań sprzętowo-programowych.

Język GELLO może być używany do budowania zapytań do wydobywania i manipulowania danymi z historii choroby w systemach służby zdrowia. GELLO służy do konstruowania kryteriów decyzyjnych przez tworzenie wyrażeń dla szczególnych danych cecha/wartość. Te kryteriach mogą być używane w bazach wiedzy systemów wspomagania decyzji takich jak te, zaprojektowane by dostarczyć ostrzeżeń i przypomnień, wskazówek, albo innych zasad decyzyjnych. GELLO jest specjalizacją języka OCL, dzięki czemu można go wykorzystać do uszczegóławiania modeli UML. GELLO wykorzystuje model referencyjny RIM standardu HL7.

Po analizie rozpatrywanych standardów celowym wydaje się zastosowanie UML, XML, HL7 oraz GELLO do modelowania architektury systemu informatycznego dla służby zdrowia.

9. Bibliografia

[1] Fowler M., *UML Distilled 3rd Edition*, Addison-Wesley, Kanada, 2005.

[2] Johnston S., *UML 2.0 Profile for Software Services*, 2005, (www.ibm.com).

[3] Johnston S. *Rational UML Profile for Business Modeling*, 2004, (www.ibm.com).

[4] Rozanski N., *Software Systems Architecture*, Pearson Education Inc., Crawfordsville, USA, 2005.

[5] Warmer J., Kleppe A., *OCL precyzyjne modlowanie w UML,* WNT, Warszawa, 2003.

[6] Holzner S., *XML Vademecum profesjonalisty,* Helion, Gliwice, 2001.

[7] Pender T., *UML Bible,* Wiley Publishing, Inc., Indianapolis, 2003.

[8] *Business Process Modeling Notation version 1.2,* Object Management Group, 2009, (www.omg.org/spec/BPMN/1.2).

[9] Sordo M., Ogunyemi O., Boxwala A., Greenes R., *Software Specifications for GELLO: An Object-Oriented Query and Expression Language for Clinical Decision Support,* Decision Systems Group, Harvard Medical School, 2003.