*Barbara Kędzierska**, *Tadeusz Kędzierski***

INFORMATYCZNE ŚRODOWISKO TWORZENIA DIALOGÓW

O komputerach mowi się dziś niemal bez przerwy we wszystkich prawie środowiskach. Nie świadczy to jednak o stopniu wykorzystania komputera jako narzędzia pracy; bardzo często zdarza się, że jest to sprzęt, z którym nie wiadomo, co zrobić, poza - najczęściej - traktowaniem go jako inteligentnej maszyny do pisania. Dlaczego tak się dzieje?

Użytkownik (specjalista we własnej dziedzinie, nie informatyk) najczęściej nie zna oprogramowania komputera, często nie ma czasu, by się go nauczyć, czasem nie ma na to ochoty. Na pewno nie ma obowiązku studiowania różnych systemów i języków programowania, które umożliwiłyby mu wykorzystanie komputera do własnych celów. Zadaniem informatyka jest stworzenie systemu pośredniczącego między użytkownikiem a możliwościami komputera.

W dalszym ciągu obserwuje się olbrzymie zapotrzebowanie na systemy dialogowe, na generatory programów dialogowych, które umożliwią użytkownikowi w sposób prosty i szybki przygotowanie

* Instytut Fizyki i Informatyki, WSP w Krakowie
** Instytut Informatyki, Uniwersytet Jagielloński

i opracowanie dialogów, jakie powinien prowadzić w jego imieniu komputer. Brak tego typu oprogramowania daje się odczuć najbardziej wśród następujących grup użytkowników komputerów:

- nauczycieli, przygotowujących programowane jednostki lekcyjne bądź opracowujących testy sprawdzające
- lekarzy różnych specjalności, opracowujących wstępne wywiady lekarskie oraz próby automatycznego diagnozowania
- psychologów, logopedów, przygotowujących dialogi z pacjentami, stanowiące proces badania lub terapii
- wychowawczyń przedszkolnych, by w sposób możliwie najprostszy przedstawić dzieciom ich przyszłe narzędzie pracy, które obecnie jest dla nich nietypowym nauczycielem rachunków, ortografii czy towarzyszem gier logicznych.

Wymieniać można by długo. Wszędzie gdzie pojawia się komputer, potrzebne jest dobrze przygotowane oprogramowanie, które stanowić będzie pomost pomiędzy komputerem, a przeciętnym użytkownikiem. Im więcej powstawać będzie systemów tego rodzaju, tym szybciej komputer stanie się nieodzownym sprzymierzeńcem człowieka w jego działaniach.

Przykładem takiego generatora dialogów jest opisany dalej system YAMES.

CHARAKTERYSTYKA SYSTEMU

Podstawowym zadaniem systemu jest umożliwienie łatwego tworzenia scenariuszy dialogów przez użytkownika nie będącego informatykiem. W każdym dialogu operacją najczęściej wykonywaną jest wyświetlanie tekstu na ekranie monitora. Dla ustalenia uwagi

przyjmijmy, że użytkownikiem systemu jest nauczyciel. Nawet najprostsze języki programowania, jakich mógłby użyć nauczyciel, wymagają "obudowania' tekstu, który ma być wyświetlany tak, by powstała kompletna, poprawna instrukcja wyprowadzania informacji, w której wyświetlany tekst jest tylko jednym z parametrów. W naszym systemie nie ma osobnej instrukcji wyprowadzania informacji - cały tekst, który nie zostanie rozpoznany jako komenda YAMES-a jest wyświetlany na ekranie. Jest to bardziej naturalne podejście - nauczyciel, tworzący dialog po prostu pisze tekst, który ma się pojawić na ekranie.

Samo pisanie tekstów bardzo szybko przestaje wystarczać. Jako naturalne pojawiają się potrzeby wczytania odpowiedzi ucznia, jej analizy i warunkowego wykonania partii dialogu. W tym miejscu musieliśmy zdefiniować język komend używanych podczas tworzenia scenariusza dialogu. Nasze wcześniejsze doświadczenia z systemami dialogowymi [1] oraz z nauczaniem języków programowania skłoniły nas do przyjęcia następujących założeń:

"kategoryczność" formatu komend - każda komenda zawsze jest pisana od pierwszej kolumny i zawsze zaczyna się od znaku wyróżniającego komendę - w naszym przypadku jest to kropka. Doświadczenie uczy, że reguły typu "komendy pisz zawsze od pierwszej kolumny i poprzedzaj kropką" są - dla użytkownika nie będącego informatykiem - czytelniejsze, niż reguły typu "komenda może być pisana w dowolnym miejscu linii, może zaczynać się dowolnym ze słów kluczowych", które powodują zwykle, że użytkownik zaczyna się zastanawiać, czy miejsce, w którym zaczął pisać komendę jest "dowolne".

język komend - słowami kluczowymi, tworzącymi komendę, są wyrazy języka angielskiego podobne do wyrazów używanych w językach programowania. Próby wprowadzania polskich słów kluczowych do języków

programowania - czego nie należy mylić z polskojęzyczną komunikacją program ⟺ użytkownik - dowiodły, że nie jest to dobre rozwiązanie. Użytkownicy (w tym miejscu może lepiej byłoby zastosować termin "początkujący programiści") zaczęli je traktować zbyt dosłownie - odmieniając przez przypadki słowa kluczowe i starając się zapisywać instrukcję tak, by była ona zgodna z regułami gramatycznymi języka polskiego, a nie regułami gramatyki instrukcji języka programowania. Lepszym rozwiązaniem jest uświadomienie użytkownikowi (nauczycielowi), że tworzony dialog, oprócz tekstu wyświetlanego uczniowi na ekranie zawiera również - zakodowane - pewne polecenia dla komputera, realizującego taki dialog. Kody tych poleceń powinny być zawsze tak samo wpisywane do dialogu. Nieznajomość języka angielskiego przez tworzącego dialog nauczyciela nie jest przeszkodą - nauczyciel otrzymuje słowniczek komend, które ma wstawić do swego dialogu wraz z opisem akcji, jakie komendy te spowodują.

Język angielski został wybrany z kilku powodów. Słowa w tym języku są zwykle krótsze niż w języku polskim, język ten jest dość powszechnie znany i używany w większości języków programowania - ich znajomość może być pomocna, choć nie jest konieczna, podczas tworzenia dialogu. Język angielski - w świadomości większości użytkowników - był zawsze związany z komputerem i najczęściej w tym języku zapisywane są komendy systemu operacyjnego. Sentymenty językowe twórców programu również odegrały tutaj swoją rolę.

niestrukturalność języka komend - jest to świadoma decyzja autorów programu. Analiza sposobów tworzenia dialogów przez nauczycieli wykazała, że na pewno strukturalna notacja dialogu nie byłaby notacją naturalną. Nauczyciel przygotowujący lekcję (dialog) zwykle stara się zapisać do końca jedną ze ścieżek logicznych dialogu,

a dopiero potem przechodzi do uzupełnienia - w ten sam sposób - pozostałych, alternatywnych ścieżek. Bardzo często zdarza się również, że wiele alternatywnych dróg dialogu prowadzi do tego samego miejsca. W tej sytuacji użycie instrukcji goto i etykiet wydaje się bardziej naturalne. Brak konieczności tłumaczenia nauczycielowi reguł tworzenia instrukcji strukturalnych powinien znacznie przyspieszyć poznawanie nowego narzędzia, jakim zawsze jest program do tworzenia lekcji, oraz zmniejszyć opory przed użyciem takiego programu, eliminując postawy typu, "jeżeli po to, by zacząć używać tego programu, muszę stracić tyle czasu, by się go nauczyć, to nie jest on mi potrzebny". Bardziej "zdradliwe" i efektywniejsze wydaje się dostarczenie użytkownikowi programu, którego może zacząć używać prawie od razu, po paruminutowych wyjaśnieniach - pisanie prostych, bezwarunkowych dialogów bardzo szybko przestanie wystarczać i użytkownik zacznie sam szukać możliwości ich rozbudowania.

STAN OBECNY

Pierwsza wersja systemu miała umożliwić prowadzenie prostych "rozmów" ze studentami logopedii, umożliwiając zarówno prezentację kolejnych etapów stawiania diagnozy, jak i przeprowadzanie "diagnoz testowych", w których studentowi prezentowana byłaby lista objawów, a jego (studenta) zadaniem byłoby postawienie diagnozy. System został zaimplementowany jako klasyczny interpreter z bardzo ograniczoną liczbą komend. Nawet w tej postaci możliwe jest tworzenie znacznie rozbudowanych dialogów.

Obecnie w systemie YAMES dostępne są niżej wymienione komendy:

.comment komentarz pozwala na wpisanie komentarza, całowicie ignorowanego przez system.

.author komentarz pozwala na wpisanie tekstu, traktowanego przez system jak komentarz; tekst ten jest wyświetlany w nagłówku ekranu na początku dialogu

.label etykieta definiuje etykietę.

.goto etykieta powoduje bezwarunkowy skok do wskazanej etykiety.

.cls spacjuje ekran, ustawia kursor w pierwszej linii i pierwszej kolumnie ekranu.

.beep powoduje wyemitowanie krótkiego sygnału dźwiękowego.

.delay generuje (około) sekundowe opóźnienie. Może być użyta do wyświetlenia tekstu na ekranie przez pewien okres czasu, wyświetlenie komunikatu o błędzie z automatycznym przejściem do obsługi tego błędu po upływie wygenerowanego opóźnienia itp.

.read powoduje wczytanie i zapamiętanie odpowiedzi ucznia.

.if .answer = 'text' .then .goto etykieta, jeżeli ostatnia odpowiedz ucznia zaczyna się od podanego tekstu - skocz do wskazanej etykiety; jeżeli nie - przejdź do następnej linii dialogu.

.if 'text' .in .answer .then .goto etykieta, jeżeli ostatnia odpowiedź ucznia zawiera -w dowolnym miejscu- podany tekst - skocz do wskazanej etykiety; jeżeli nie - przejdź do następnej linii dialogu.

.if .not 'text' .in .answer .then .goto etykieta, jeżeli w ostatniej odpowiedzi ucznia nie występuje podany tekst - skocz do wskazanej etykiety; jeżeli nie - przejdź do następnej linii dialogu.

.quit zakończ działanie programu, skończ dialog.

.end ta linia jest traktowana jako znacznik końca szablonu dialogu.

System działa dwuetapowo. W pierwszym etapie rozpoznawane i kodowane są komendy występujące w szablonie dialogu. Sygnalizowane są ewentualne błędy składniowe oraz badana jest kompletność dialogu (obecnie ograniczona do sprawdzenia, czy zdefiniowane są wszystkie etykiety, do których odwołują się instrukcje skoków). Tworzony jest listing opracowywanego szablonu dialogu wraz z ewentualnymi komunikatami o wykrytych błędach. Przetworzony szablon dialogu jest zapamiętywany w postaci upakowanej (kody komend, zamiast ich tekstów) w wewnętrznej tablicy programu, a następnie inicjowany jest etap drugi - prowadzenie dialogu. W etapie tym system sekwencyjnie wykonuje instrukcje sterujące dialogiem, wyświetla zapamiętane teksty oraz wczytuje odpowiedzi ucznia i wartościuje związane z nimi komendy warunkowe.

PRZEWIDYWANE ROZSZERZENIA

Pierwsze próby praktycznego tworzenia dialogów przy użyciu systemu YAMES dowiodły jego przydatności, ukazując jednocześnie kierunki zmian i rozszerzeń systemu. Do najważniejszych należeć będą:

rozbudowa komend Konieczne jest rozbudowanie języka komend. Obecny zestaw jest zestawem minimalnym powstałym po to, by można było jak najszybciej praktycznie tworzyć i sprawdzać dialogi. Zestaw ten musi być uzupełniony o pełne wyrażenia regularne, z dużą ilością operacji na łańcuchach tekstowych. Konieczne jest również dołożenie instrukcji umożliwiających wykorzystywanie zapisanych na pliku dialogów jako rodzaju procedur - wspólnych "poddialogów", dołączanych do dialogu głównego. Rozbudowane mogą również zostać instrukcje sterujące wyświetlaniem

informacji na ekranie (pozycjonowanie kursora, łatwe tworzenie ramek, negacja, rozjaśnianie i zmiana koloru tekstu, itp.) oraz instrukcje "dźwiękowe" (dodanie możliwości łatwego zapisu melodyjek). Nie przewidujemy obecnie rozszerzenie YAMES-a o operacje graficzne - powodowało by to znaczną komplikację całego systemu.

kompilator Użycie klasycznego, dwuetapowego interpretera komend sterujących dialogiem jest również rozwiązaniem przejściowym. W wersji finalnej przewidujemy rozdzielenie obu etapów - wydzielony jako osobny program translator dialogów, działający w oparciu o gramatykę komend dialogowych będzie badał poprawność szablonu dialogu, diagnozując błędy oraz tworząc wewnętrzną, skompilowaną postać szablonu dialogu. Niezależny program uruchomieniowy, działający na wytworzonej przez translator postaci wewnętrznej dialogu, prowadziłby dialog z uczniem.

otoczenie programowe systemu YAMES Oprócz programów tworzących system YAMES użytkownik powinien mieć do swojej dyspozycji edytor, pozwalający na tworzenie szablonów dialogów. Obecnie można w tym celu użyć dowolnego edytora tekstowego, natomiast celowe wydaje się stworzenie otoczenia programowego, zbliżonego do otoczenia języków Turbo firmy Borland - zintegrowany edytor, oparty o uproszczoną wersję WordStara, z elementarnymi funkcjami redakcji tekstu (justyfikacja, zmiana koloru lub atrybutów tekstu zapisywanego dialogu, ramki wokół tekstu itp.). Edytor powinien umożliwiać automatyczne wpisywanie szkieletów komend sterujących dialogiem, uzupełnianych następnie przez nauczyciela definiującego szablon dialogu, połączony z translatorem szablonów dialogu, umożliwiającym łatwą lokalizację i poprawę błędów oraz interpreterem dialogu wyposażonym w elementarne funkcje śledzenia i testowania dialogów. Wszystko to powinno

umożliwić nauczycielowi łatwe stworzenie, wytestowanie i translację szablonu dialogu, który byłby potem wykonywany przez program uruchomieniowy.

Program uruchomieniowy, oprócz prowadzenia samego dialogu, powinien umożliwić dokumentowanie prowadzonych dialogów -- selektywne pamiętanie wyświetlanych tekstów i odpowiedzi, udzielanych przez ucznia lub pacjenta.

egzaminy testowe i automatyczna diagnoza Oprócz definiowania dialogów system powinien umożliwić łatwe tworzenie egzaminów testowych i dialogów prowadzących do automatycznej diagnozy. Nauczyciel lub lekarz definiowałby nazwę zmiennej, której wartościowanie decydowałoby o wyniku egzaminu lub diagnozy oraz pytania zadawanego przez system i odpowiedzi, jakich mógłby udzielić uczeń lub pacjent wraz z informacją, jakie wartości nadawane byłyby wtedy wartościowanej zmiennej.

Przykład takiej definicji przedstawiamy poniżej:

```
.define gorączka
.value 'tak'
.question 'Czy masz podwyższoną temperaturę?'
.answer 'tak'
.define głowa
.value 'ból'
.question 'Czy boli Cię głowa?'
.answer 'tak'
.value 'zawroty'
.question 'Czy masz zawroty głowy lub kłopoty z utrzymaniem
          równowagi?'
.answer 'tak','czasem'
.if .gorączka = 'tak' .and. .głowa  =  'ból'  .then .goto grypa
```

Edytor systemu YAMES powinien ułatwiać tworzenie tego typu definicji, a sam system - na ich podstawie - generować pytania zadawane uczniowi lub pacjentowi. Kolejność generowanych pytań definiują warunki, umieszczane przez tworzącego szablon dialogu - - w powyższym przykładzie instrukcja ".if .goraczka = 'tak' .and. .głowa = 'ból' .then .goto grypa" spowodowałaby zadanie pacjentowi dwóch pytań - "Czy masz podwyższoną temperaturę?" i "Czy boli Cię głowa?". Ten sposób zapisu egzaminu testowego lub automatycznej diagnozy pozwala na łatwe generowanie tego typu dialogów oraz na tworzenie systemów "podpowiadających", bazujących na wiadomościach zgromadzonych przez nauczyciela lub lekarza.

PODSUMOWANIE

Możliwości rozwoju systemu YAMES jest sporo. Sposób rozszerzania częściowo związany jest z dziedziną, w jakiej ma być eksploatowany ten system. Bez względu jednak na to, już teraz z powodzeniem służyć on może zarówno nauczycielowi do opracowywania programowanych jednostek lekcyjnych, jak i lekarzowi do przygotowywania dialogu z pacjentem. Poniższy przykład prostego dialogu uczącego dzieci jest tego dobrą ilustracją.

```
.comment Dialog przykładowy uczący dzieci kilku pojęć.
.label początek
 Proponuję małą zgadywankę. Czy wiesz, czym różnią się słowa HOMAR
i HOMER? Naciśnij na klawiaturze literę  a,  b lub  c  w zależności
od tego, którą odpowiedź uznasz za prawidłową.

a: HOMAR to nazwisko, a HOMER to zwierzę
b: HOMAR to gruba gałąź, a HOMER to mały rak
c: HOMAR to rodzaj raka, a HOMER to nazwisko poety
```

```
.read
.if .answer = 'a' .then .goto l1
.if .answer = 'b' .then .goto l1
.if .answer = 'c' .then .goto l2
--- Nacisnąłeś niewłaściwy klawisz. Naciśnij jeden z klawiszy
    a, b lub c.
.beep
.delay
.cls
.goto początek
.label l1
Niestety, źle. Prawidłową odpowiedzią była odpowiedź c.
.beep
.goto lab1
.label l2
BRAWO! To właśnie oznaczają te słowa.
.label lab1
    A teraz druga próba. Co oznaczją słowa KOMIK i KOMIKS?

a: KOMIK to zwierzę podobne do myszki, a KOMIKS - to człowiek,
    który wszystkich rozśmiesza
b: KOMIK to człowiek, z którego się śmiejemy, a KOMIKS to historia
    obrazkowa
c: KOMIK to zwierzę, a KOMIKS to książka

Wybierz prawidłową odpowiedź:
.read
.if .answer = 'a' .then .goto l3
.if .answer = 'b' .then .goto l4
.if .answer = 'c' .then goto l3
--- Nacisnąłeś niewłaściwy klawisz. Naciśnij jeden z klawiszy
    a, b lub c.
```

```
.beep
.delay
.goto lab1
.label 13
Niestety źle. Prawidłowa była odpowiedź b. Może teraz Ci się uda?
.beep
.goto lab2
.label 14
BRAWO! I to jest to!
.label lab2
    A teraz trzecia i ostatnia runda. HOMOFON i DOMOFON to:

a: HOMOFON to rodzaj mikrofonu, a DOMOFON to domowy głośnik
b: HOMOFON to wyraz brzmiący identycznie z innym wyrazem, a DOMOFON
    to prosty telefon
c: HOMOFON to człowiek rozmawiający ze sobą, a DOMOFON to domowy
    mikrofon

I która odpowiedź jest prawidłowa?

.if .answer = 'a' .then .goto 15
.if .answer = 'b' .then .goto 16
.if .answer = 'c' .then .goto 15
--- Naciśnij klawisz a, b lub c.
.beep
.cls
.goto lab2
.label 15
    Niestety, to nie tak!. HOMOFON to wyraz brzmiący identycznie
z innym wyrazem, a DOMOFON to prosty telefon. Musisz trochę więcej
czytać! Następnym razem na pewno uda Ci się lepiej.
    Do zobaczenia!
```

.quit

Tak! Dobrze! Jeżeli wszystkie Twoje odpowiedzi były prawidłowe, to masz niemały zasób słów. Gratuluję!

Do zobaczenia!

.quit

.end

BIBLIOGRAFIA

1. Kędzierska B., *DIALMED - komputerowy system dialogowego prowadzenia wywiadu lekarskiego, wspomagający proces diagnostyczny*, praca magisterska, Uniwersytet Jagielloński, 1981.
2. *The human side of information processing*, praca zbiorowa, North-Holland, 1980.
3. *Methodology of interaction*, praca zbiorowa, North-Holland 1980.

Abstract

YAMES, system which allows dialogues preparation for the various kind of the user (students, patients and so on) is presented in this paper. System structure, current implementation and possible extensions are also covered.