УДК 004

В. Бурлаченко, магістр гр. КІ-21М-1,4
Центральноукраїнський національний технічний університет

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ СЕРВІСУ ДІАГНОСТУВАННЯ ТА ТЕСТУВАННЯ СКЛАДОВИХ ПК

У статті розроблено програмне забезпечення, яке призначено для системи сервісу діагностування та тестування складових ПК. Метою розробки є дослідження та програмна реалізація системи сервісу діагностування та тестування складових ПК. Об'єктом дослідження є процес сервісу діагностування та тестування складових ПК. Предметом дослідження є методи сервісу діагностування та тестування складових ПК. Методи дослідження базуються на методах схемотехніки, методах математичної статистики, методах розробки програмного забезпечення. Результат роботи – програмна реалізація системи сервісу діагностування та тестування складових ПК. В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.
комп'ютерна інженерія, діагностування, тестування

Постановка проблеми. На перший погляд більшості домашніх користувачів зовсім не потрібно розбиратися в залізній начинці свого комп'ютера – але лише за умови, якщо хто-небудь інший візьме на себе рішення питань, пов'язаних з «залізом». А якщо таких не найдеться, то ознайомитися з деякою початковою інформацією із цієї області зовсім не зашкодить. Справа в тому, що рано або пізно вам доведеться проводити апгрейд вашого ПК, що далеко не завжди припускає банальну зміну одного системного блоку з начинкою на іншій, більш сучасний. Можливо, у багатьох випадках виявиться досить замінити (або доповнити) його окремі компоненти, наприклад просто докупити пам'ять. Але, щоб вибрати вірну стратегію апгрейду, варто чітко знати, яке апаратне забезпечення у вас установлено, і зрозуміти, чого конкретно не вистачає вашому комп'ютеру для більш швидкої роботи: потужності процесора, пам'яті, швидкості вінчестера й т.п. Але це лише одна із проблем. Інша, не менш важлива пов'язана з тим, що любе встаткування в якийсь момент починає працювати зі збоями або взагалі виходить із ладу. Сумні наслідки подібних ситуацій очевидні, тому набагато надійніше контролювати стан важливих апаратних систем, що дозволить із певною ймовірністю прогнозувати можливе поводження апаратури в найближчому майбутньому, вчасно виявити ті або інші неполадки з комп'ютерними компонентами й вчасно подбати про запобіжні заходи. І нарешті, припустимо, що ви придбали новий комп'ютер або провели апгрейд старого – очевидно, що вам необхідно швидко зорієнтуватися й зрозуміти, чи дійсно комп'ютер стабільно працює й чи відповідає начинка його системного блоку заявленій при покупці, причому не розбираючи сам блок, на якому може стояти пломба.

Аналіз останніх досліджень і публікацій. При аналізі останніх досліджень і публікацій [1-10] було виявлено певні прогалини у забезпеченні системи сервісу діагностування та тестування складових ПК.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи сервісу діагностування та тестування складових ПК.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем сервісу діагностування та тестування складових ПК.
– Дослідження системи сервісу діагностування та тестування складових ПК.

– Програмна реалізація системи сервісу діагностування та тестування складових ПК.

Об'єктом дослідження є процес сервісу діагностування та тестування складових ПК.

Предметом дослідження є методи сервісу діагностування та тестування складових ПК.

Методи дослідження базуються на методах схемотехніки, методах математичної статистики, методах розробки програмного забезпечення.

Виклад основного матеріалу. Windows API – найбільш важлива й потужна додаткова бібліотека функцій, доступна кожному програмістові. Багато хто з них, у тому числі й досвідчених розроблювачах працюють із ними, використовуючи прості готові рішення, почерпнуті в різних книгах і журналах, не дуже замислюючись про суть цієї технології. Такий підхід є достатнім при рішенні простих задач, але для серйозної роботи переважніше більш детально розібратися з основними принципами використання функцій Windows API. Чим ми зараз і займемося.

Windows API – набір функцій операційної системи

Абревіатура API для багатьох починаючих програмістів виглядає досить таємниче. Насправді ж Application Programming Interface (API) – це просто деякий готовий набір функцій, що можуть використовувати розроблювачі додатків. У загальному випадку дане поняття еквівалентно тому, що раніше частіше називали бібліотекою підпрограм. Однак найчастіше під API мається на увазі деяка особлива категорія таких бібліотек.

У ході розробки практично будь-якого досить складного додатка (MyAppication) для кінцевого користувача формується набір специфічних внутрішніх функцій, використовуваних для реалізації даної конкретної програми, що називається MyApplication API. Часто виявляється, що ці функції можуть ефективно використовуватися також для створення інших додатків, у тому числі іншими програмістами. У цьому випадку автори виходячи зі стратегії просування свого продукту повинні вирішити питання – чи відкривають вони доступ до цього набору для зовнішніх чи користувачів ні? При позитивній відповіді на нього в описі програмного пакета, як його переваго, з'являється фраза про те, що "комплект включає відкритий набір API-функцій" (але іноді за додаткові гроші).

Таким чином, найчастіше під API мається на увазі набір функцій, що є частиною одного додатка, але при цьому доступних для використання в інших програмах. Наприклад, Excel крім інтерфейсу для кінцевого користувача має набір функцій Excel API, що може використовуватися, зокрема, при створенні додатків за допомогою Delphi.

Відповідно, Windows API – це набір функцій, що є частиною самої операційної системи й у той же час – доступної для будь-якого іншого додатка, у тому числі написаного за допомогою Delphi. І в цьому плані цілком виправдана аналогія з набором системних переривань BIOS/DOS, що фактично являє собою DOS API

Відмінність полягає в тім, що склад функцій Windows API, з однієї сторони значно ширше, у порівнянні з DOS, з іншого боку – не включає багато засобів прямого керування ресурсами комп'ютера, які були доступні програмістам у попередньої ОС. Крім того, звертання до Windows API виконується за допомогою звичайних процедурних обігів, а виклик функцій DOS – через спеціальну машинну команду процесора, що називається Interrupt ("переривання").

Далі під терміном API буде матися на увазі Win API і більше того, за замовчуванням – Win64 API.

Навіщо потрібний Win API

Незважаючи на те, що Delphi має величезну множину різноманітних функцій, при більш-менш серйозній розробці виявляється, що їхніх можливостей часто не вистачає для рішення необхідних задач. При цьому програмісти-новачки часто починають скаржитися на недоліки Delphi і подумувати про зміну інструмента, не підозрюючи, що на їхньому комп'ютері є величезний набір засобів і потрібно тільки вміти ним скористатися.

При знайомстві з Win API виявляється, що багато вбудованих Delphi-функцій – не що інше, як звертання до відповідних системних процедур, але тільки реалізовані у вигляді синтаксису даної мови. З огляду на це, необхідність використання API визначається наступними варіантами:

1. API-функції, які повністю реалізовані у вигляді убудованих Delphi-функцій. Проте, іноді й у цьому випадку буває корисним перейти до застосування API, тому що це дозволяє часом істотно підвищити продуктивність (зокрема, за рахунок відсутності непотрібних перетворень переданих параметрів).

2. Вбудовані Delphi-функції реалізують лише окремий випадок відповідної API-функції. Це досить звичайний варіант. Наприклад, API-функція CreateDirectory має більше широкі можливості в порівнянні з убудованим Delphi-оператором MkDir.

3. Величезне число API-функцій взагалі не мають аналогів в існуючому сьогодні варіанті мови Delphi. Наприклад, видалити каталог не можна засобами Delphi – для цього потрібно використовувати функцію DeleteDirectory.

Варто також підкреслити, що деякі API-функції (їхня частка в Win API досить незначна) не можуть викликатися з Delphi-програм через ряд обмежень мови, наприклад відсутність можливості роботи з адресами пам'яті. Але в ряді випадків можуть допомогти нетривіальні прийоми програмування (зокрема, у випадку з тими ж адресами).

Як вивчати Win API

Це не таке просте питання, якщо врахувати що число функцій Win64 API оцінюється величиною порядку 10 тисяч (точної величини ніхто не знає, навіть Microsoft).

До складу Delphi (версій 4-6) входить файл із описом оголошень Win API – WIN64API.TXT (про його застосування ми розповімо докладніше пізніше). Але, по-перше, з його допомогою можна одержати відомості про призначення тої або іншої функції і її параметрів тільки по використовуваних мнемонічних іменах, а по-друге – перелік функцій у цьому файлі далеко не повний. У свій час в Delphi були спеціальні довідкові файли, з описом функцій Win16 API. Вичерпна інформація про Win64 API перебуває в довідковій системі Platform Software Development Kit, що зокрема перебуває на компакт-дисках MSDN Library, включених до складу Delphi 5.0 і 6.0 Enterprise Edition і Office 2000 Developer Edition. Однак потрібно зізнатися, що знайти там потрібну інформацію й розібратися в ній зовсім не просто. Не говорячи вуж про те, що всі описи там приводяться стосовно до мови С.

Win API і Dynamic Link Library (DLL)

Набір Win API реалізований у вигляді динамічних DLL-бібліотек. У такий спосіб далі зараз ми будемо фактично говорити про технологію використання DLL у середовищі Delphi на прикладі бібліотек, що входять до складу Win API. Однак говорячи про DLL необхідно зробити кілька важливих зауважень.

У цьому випадку під DLL ми маємо на увазі традиційний варіант двійкових динамічних бібліотек, які забезпечують прямий обіг додатків до потрібних процедур – підпрограмам або функціям (приблизно також як це відбувається при виклику процедур усередині Delphi-проекту). Такі бібліотеки можуть створюватися за допомогою різних інструментів – VC++, Delphi, Fortran, Assembler. Delphi може робити Active DLL, доступ до яких виконується через інтерфейс OLE Automation.

Звичайно файли динамічних бібліотек мають розширення .DLL, але це зовсім не обов'язково. Для Win16 часто застосовувалося розширення .EXE, драйвери зовнішніх пристроїв позначаються за допомогою .DRV.

Як ми вже відзначали, визначити точне число API-функцій Windows і файлів їх утримуючих – досить складно (але всі вони перебувають у системному каталозі). У цьому плані краще виділити склад бібліотек, що становлять ядро операційної системи, і основних бібліотек із ключовими додатковими функціями.

Бібліотеки Win64 API ядра операційної системи Windows 95/98:

–KERNEL64.DLL: низькорівневі функції керування пам'яттю, задачами й іншими ресурсами системи;

–USER64.DLL: тут в основному перебувають функції керування користувальницьким інтерфейсом;

–GDI64.DLL: бібліотека Graphics Device Interface – різноманітні функції висновку на зовнішні пристрої;

–COMDLG64.DLL: функції, пов'язані з використанням діалогових вікон загального призначення.

Основні бібліотеки з функціями розширення:

–COMCTL64.DLL: набір додаткових елементів керування Windows, у тому числі Tree List і Rich Text;

–MAPI64.DLL: функції роботи з електронною поштою;

–NETAPI64.DLL: елементи керування й функції роботи з мережею;

–ODBC64.DLL: функції цієї бібліотеки потрібні для роботи з різними базами даних через протокол ODBC;

–WINMM.DLL: операції доступу до системних засобів мультимедіа.

Програми написані на Win API мають більшу продуктивність і невеликий розмір. За допомогою функцій Win API можна одержати доступ до різних об'єктів Windows.

Функція CreateWindowEx

Створює вікно із заданими властивостями.

Функція CreateWindowEx виглядає так:

```
function CreateWindowEx(
dwExStyle: DWORD;
lpClassName: PChar;
lpWindowName: PChar;
dwStyle: DWORD;
X, Y, nWidth, nHeight: Integer;
hWndParent: HWND;
hMenu: HMENU;
hInstance: HINST;
lpParam: Pointer
): HWND;
```

Параметри CreateWindowEx:

dwExStyle– визначає розширений стиль вікна, може бути комбінацією (за допомогою оператора or):

– WS_EX_ACCEPTFILES– на вікно можна перетаскувати файли із Провідника, коли користувач опускає на вікно з таким стилем файли, посилає повідомлення WM_DROPFILES;

– WS_EX_APPWINDOW– на панелі задач для вікна з таким стилем з'являється кнопка;

– WS_EX_CLIENTEDGE– навколо клієнтської частини буде рамка, причому клієнтська частина буде втиснена усередину;

– WS_EX_CONTEXTHELP– у заголовку вікна з'явиться кнопка контекстної допомоги, це прапор не може використовуватися одночасно з WS_MAXIMIZEBOX і WS_MINIMIZEBOX;

– WS_EX_CONTROLPARENT;

– WS_EX_DLGMODALFRAME;

– WS_EX_LEFT;

– WS_EX_LEFTSCROLLBAR;

– WS_EX_LTRREADING;

– WS_EX_MDICHILD;

– WS_EX_NOINHERITLAYOUT;

– WS_EX_NOPARENTNOTIFY;

– WS_EX_OVERLAPPEDWINDOW;

– WS_EX_PALETTEWINDOW;

– WS_EX_RIGHT;

– WS_EX_RIGHTSCROLLBAR;

– WS_EX_RTLREADING;

– WS_EX_STATICEDGE;

– WS_EX_TOOLWINDOW– створюється вікно з маленьким заголовком, як у панелі інструментів;

– WS_EX_TOPMOST– створюване вікно буде перебувати поверх інших;

– WS_EX_TRANSPARENT;

– WS_EX_WINDOWEDGE.

Визначення інших констант ви зможете знайти в довідці по Win64api.

lpClassName– Ім'я класу вікна. Ви можете створювати свої класи за допомогою функції RegisterClassEx або використовувати визначені: edit, button, static, scrollbar, combobox і інші;

lpWindowName– текст, що з'явиться в заголовку вікна (якщо вікно із заголовком), на кнопці (якщо клас вікна button), у поле уведення тексту (якщо клас вікна edit);

dwStyle– список основних стилів вікна. Містить кілька наступних констант, з'єднаних оператором or:

– WS_BORDER– вікно буде мати тонку рамку;

– WS_CAPTION– вікно буде мати заголовок;

– WS_CHILD або WS_CHILDWINDOW– вікно буде дочірнім, тобто цілком розташовуватися усередині деякого іншого вікна;

– WS_CLIPCHILDREN– площа займана дочірніми вікнами не буде перемальовуватися;

– WS_CLIPSIBLINGS– перемальовування одного дочірнього вікна не впливає на інші;

– WS_DISABLED– вікно створюється недоступним, його можна розблокувати за допомогою функції EnableWindow;

– WS_DLGFRAME– створюється вікно з рамкою як у діалогових вікон;

– WS_GROUP– для дочірнього вікна (зі стилем WS_CHILD) визначає перший елемент у групі, при натисканні на Tab саме він одержить фокус, група простирається до наступного дочірнього вікна з тим же стилем, усередині групи можна переміщатися за допомогою клавіш керування курсором;

– WS_HSCROLL– створюється вікно з горизонтальною смугою прокручування;

– WS_ICONIC або WS_MINIMIZE– створюване вікно споконвічно мінімізоване;

– WS_MAXIMIZE– створюване вікно споконвічно максимізоване;

– WS_MAXIMIZEBOX– створюване вікно має кнопку максимізації;

– WS_MINIMIZEBOX– створюване вікно має кнопку мінімізації;

– WS_OVERLAPPED– створюється вікно, що перекривається, має заголовок і рамку;

– WS_OVERLAPPEDWINDOW– комбінація прапорів WS_OVERLAPPED, WS_CAPTION, WS_SYSMENU, WS_THICKFRAME, WS_MINIMIZEBOX і WS_MAXIMIZEBOX;

– WS_POPUP– створюється вікно не має споконвічно рамки й заголовка, не може використовуватися зі стилем WS_CHILD;

– WS_SIZEBOX або WS_THICKFRAME– створюється вікно, розмір якого можна змінювати;

– WS_SYSMENU– створюється вікно зі значком системного меню, повинен уживатися із прапором WS_CAPTION;

– WS_TABSTOP– створюється дочірнє вікно, що може одержувати фокус уведення при натисканні на Tab;

– WS_TILEDWINDOW– комбінація прапорів WS_OVERLAPPED, WS_CAPTION, WS_SYSMENU, WS_THICKFRAME, WS_MINIMIZEBOX і WS_MAXIMIZEBOX;

– WS_VISIBLE– створюється вікно, що споконвічно видиме. Якщо ви не вкажете це прапор для вікна, то ви його ніколи не побачите (якщо тільки не скористаєтеся функцією ShowWindow);

– WS_VSCROLL– створюване вікно буде мати вертикальну смугу прокручування;

– X-X– горизонтальна координата верхнього лівого кута вікна; якщо ви хочете надати windows можливість розташувати вікно за замовчуванням, укажіть тут CW_USEDEFAULT, у цьому випадку наступний параметр ігнорується;

– Y-Y– вертикальна координата верхнього лівого кута вікна.

nWidth– ширина вікна (в одиницях пристрою, для монітора– у пікселях), якщо ви хочете надати Windows вибрати положення вікна, то виставите тут CW_USEDEFAULT, у цьому випадку наступний параметр ігнорується;

nHeight– висота вікна (в одиницях пристрою, для монітора– у пикселах);

hWndParent– описувач батьківського вікна, якщо вікно створюється зі стилем WS_CHILD, то тут обов'язково повинен стояти коректний описувач;

для дочірнього вікна (із прапором стилю WS_CHILD) визначає ідентифікатор цього дочірнього вікна, для звичайного вікна визначає описувач головного меню вікна (якщо дорівнює нулю, то використовується меню з визначення класу вікна);

hInstance– описувач додатка, з яким вікно зв'язується (для Windows NT/2000/XP ігнорується);

lpParam– покажчик на що-небудь, він передається в повідомленні WM_CREATE при створенні вікна, і в ньому можуть бути переправлені які-небудь дані для копії вікна.

Функція ShowWindow

Ця функція показує або ховає вікно.

```
function ShowWindow(
      hWnd: HWND;
      nCmdShow: Integer
): BOOL;
```

Параметри функції:

hWnd– Описувач потрібного вікна;

nCmdShow– Константа, що визначає, що буде зроблено з вікном:

– SW_HIDE– вікно буде приховано;

– SW_SHOWNORMAL– вікно буде показане й активоване, якщо вікно було мінімізовано або максимізоване, то воно буде відновлено у вихідну позицію й розмір;

– SW_SHOWMINIMIZED– активізує й звертає (мінімізує) вікно;

– SW_SHOWMAXIMIZED– активізує й максимізує вікно;

– SW_MAXIMIZE– максимізує вікно;

– SW_SHOWNOACTIVATE– те ж саме, що SW_SHOWNORMAL, тільки вікно не активізується;

– SW_SHOW– відображає вікно в його поточній позиції;

– SW_MINIMIZE– мінімізує вікно й активізує наступне по Z-Списку;

– SW_SHOWMINNOACTIVE– те ж саме, що й SW_SHOWMINIMIZED, тільки вікно не активізується;

– SW_SHOWNA– те ж саме, що SW_SHOW, тільки вікно не активізується;

– SW_RESTORE– відновлює вікно з максимізованого або мінімізованого стану;

– SW_SHOWDEFAULT– відображає вікно так, як воно було відображено при старті відповідного додатка;

– SW_MAXIMIZE– максимізує вікно.

Структура типу TWndClassEx

Структура типу TWndClassEx має такий вигляд:

```
tagWNDCLASSEXA = packed record
cbSize: UINT;
style: UINT;
lpfnWndProc: TFNWndProc;
cbClsExtra: Integer;
cbWndExtra: Integer;
hInstance: HINST;
hIcon: HICON;
```

```
hCursor: HCURSOR;
hbrBackground: HBRUSH;
lpszMenuName: PAnsiChar;
lpszClassName: PAnsiChar;
hIconSm: HICON;
end;
```

Обробка повідомлень в Win API

Повідомлення – інформація про деяку зміну в користувальницькому інтерфейсі, наприклад переміщення вікна або натискання клавіші на клавіатурі. Повідомлення також можуть розсилатися іншими додатками.

Цикл обробки повідомлень виглядати так:

```
while GetMessage(Mmsg,0,0,0) do
begin
  TranslateMessage(Mmsg);
  DispatchMessage(Mmsg);
end;
```

TranslateMessage– ця функція переводить повідомлення віртуальних клавіш у символьні повідомлення.

DispatchMessage– ця функція повідомлення віконному оброблювачеві подій. Як віконний оброблювач подій служить функція WindowProc.

Бібліотека візуальних компонентів (VCL)

Бібліотека візуальних компонентів (VCL) – об’єктно-орієнтована бібліотека для розробки програмного забезпечення, розроблена компанією «Borland» для підтримки принципів візуального програмування. VCL входить у комплект поставки «Delphi», «C++ Builder» і «Borland Developer Studio» і є, по суті, частиною середовища розробки, хоча розробка додатків цих середовищах можлива й без використання VCL.

VCL представляє величезну кількість готових до використання компонентів для роботи в самих різних областях програмування, таких, наприклад, як інтерфейс користувача (екранні форми й т.зв. «контроли»), робота з базами даних, взаємодія з операційною системою, програмування мережевих додатків та інше.

Розробка структурної схеми

На рисунку 1 зображена структурна схема системи тестування та діагностики вузлів ЕОМ, яка була розроблена у результаті виконання проекту. З цієї схеми ми бачимо, щ о система складається з наступних структурних модулів:

– Панель «Головне вікно програми», з яким зв’язані усі інші панелі виконані у вигляді закладок;
– Панель «Загальне»;
– Панель «Процесор»;
– Панель «Диски»;
– Панель «Принтери»;
– Панель «Пам'ять»;
– Панель «Клавіатура»;
– Панель «Шрифти»;
– Панель «Відеокарта»;
– Панель «Діагностика».

Усі панелі, крім останньої відображають інформацію, яка відноситься до параметрів відповідного структурного блоку.

Остання панель відображає інформацію діагностування, й включає в себе наступні підпрограми тестування:

– Перевірка CD/DVD-ROM;
– Тестування пам’яті;
– Тестування процесора.

Крім того програма надає інформацію про автора програми та місце, де вона створена.

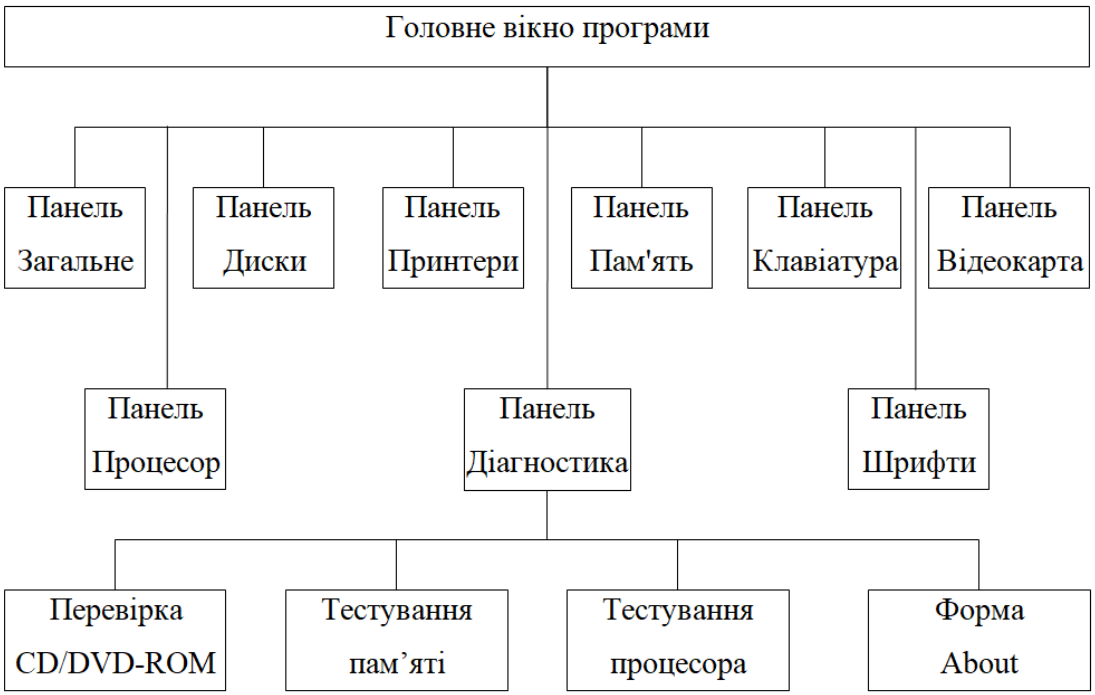

Рисунок 1 – Структурна схема системи

Висновки. У статті наведені теоретичне узагальнення й рішення наукового завдання дослідження методів сервісу діагностування та тестування складових ПК. Рішення даного завдання полягало у вирішенні наступних задач: Був проведений огляд існуючих систем сервісу діагностування та тестування складових ПК. Досліджена система сервісу діагностування та тестування складових ПК. На основі отриманих результатів досліджень створена програмна реалізація системи сервісу діагностування та тестування складових ПК. Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання сервісу діагностування та тестування складових ПК. Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Список літератури

1. Кожанова А.С. Обґрунтування необхідності створення систем технічної діагностики інтегрованих інформаційних систем / О.А. Смірнов, А.С. Кожанова, О.В. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2013. – Вип. 6(113). – С. 255-257.
2. Коваленко А.С. Задачи распознавания ситуаций в ERP системах / А.В. Коваленко ., А.А. Смірнов, А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2014. – Вип. 4(120). – С. 161-164.
3. Коваленко А.С. Підсистема технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко , О.А.Смірнов, О.В. Коваленко // Системи озброєння і військова техніка.– Х.: ХУПС, 2014. – № 1(37). – С. 126-129.
4. Коваленко А.С. Анализ эффективности использования экспертной системы технической диагностики с традиционной структурой / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 2(38). – С. 106-108.
5. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.
6. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.
7. Коваленко А.С. Структура системи технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Кіровоградського національного

технічного університету / техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Кіровоград: Вид-во КНТУ, 2014. – Вип. 27. – С. 245-251.
8. Коваленко А.С. Дослідження будови інтегрованої інформаційної системи та її елементів / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 4(40). – С. 85-88.
9. Коваленко А.С. Розробка структури бази даних для обліку технічного стану елементів інтегрованої інформаційної системи з урахуванням вимог споживачів інформації / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2015. – Вип. 1(126). – С. 75-79.
10. Коваленко А.С. Обґрунтування набору даних для оцінки технічного стану інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2015. – Вип. 1(42). – С.39-41.
11. Коваленко А.С. Експертна система технічного діагностування інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2015. – № 1(41). – С. 106-111.
12. Коваленко А.С. Удосконалення методу технічного обслуговування об’єктів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко, О.П. Доренський // Системи озброєння і військова техніка. – Х.: ХУПС, 2016. – № 2(46). – С. 109-114.
13. Kovalenko A.S. Information model and its element for displaying information on technical condition of objects of integrated information system / A.S. Kovalenko, A.A. Smirnov, A.V. Kovalenko, A.P. Dorensky // International Journal of Computational Engineering Research (IJCER). – India: Delhi, 2016. – Volume 6, Issue 1. – P. 21-27.
14. Кожанова А.С. Система технічної діагностики інтегрованих інформаційних систем – обґрунтування необхідності створення, визначення понятійного апарату та напрямів досліджень / А.С. Кожанова, О.А. Смірнов, М.П. Савченко, Д.М. Ізосімов, В.В. Мороз // Створення та модернізація озброєння і військової техніки в сучасних умовах: Тринадцята наук.-техн. конф., 5-6 вер. 2013 р., м. Феодосія: тези доп. – Феодосія: ДНВЦ, 2013. – С. 187-188.
15. Кожанова А.С. Визначення основних напрямків досліджень щодо створення системи технічної діагностики інтегрованих інформаційних систем / А.С. Кожанова, О.А. Смірнов, А.В. Челпанов // Проблемні питання розвитку озброєння та військової техніки Збройних Сил України: IV наук.-техн. конф., 16-20 груд. 2013 р., м. Київ: зб. тез. – Київ: ЦНДІ ОВТ ЗСУ, 2013. – С. 293.
16. Коваленко А.С. Обґрунтування необхідності створення систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Інформатика та системні науки : V Всеукр. наук.-практ. конф., 13–15 бер. 2014 р., м. Полтава : зб. тез. – Полтава: ПУЕТ, 2014. – С. 292-294.
17. Коваленко А.С. Створення систем технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Проблеми і перспективи розвитку ІТ-індустрії: VI між нар. наук.-практ. конф., 17-18 квіт. 2014 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2014. – С. 241.
18. Коваленко А.С. Визначення понятійного апарату та напрямів досліджень для синтезу систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Комп’ютерне моделювання у наукоємних технологіях (КМНТ-2014): наук.-техн. конф. з міжнар. участю, 28-31 трав. 2014 р., м. Харків: зб. наук. праць. – Харків: ХНУ, 2014. – С. 190-193.
19. Коваленко А.С. Розробка структури бази даних інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Проблеми і перспективи розвитку ІТ-індустрії: VII міжнар. наук.-практ. конф., 17-18 квіт. 2015 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2015. – С. 15.
20. Коваленко А.С. Дослідження елементів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Комбінаторні конфігурації та їх застосування: XVII між нар. наук.-практ. сем., 17-18 квіт. 2015 р., м. Кіровоград: зб. тез – Кіровоград: КНТУ, 2015. – С. 5.