ва сети. В идеальном случае и сервер, и клиент (а, возможно, даже несколько клиентов) должны запускаться на одном и том же компьютере и взаимодействовать точно так же, как если бы они были запущены на различных. Благодаря технологии .NET Remoting, клиент и сервер можно запускать в том числе и на одном и том же компьютере.

6. ***Защита данных***. И клиентская, и серверная часть должны быть устойчивы к различного рода атакам. Клиентская часть требует защиты в первую очередь для внутренних данных, серверная — от атак прямым перебором паролей и «отказ в обслуживании» (DDOS атаки). Даже без применения криптографии сама структура программы делает невозможным большинство методов взлома. Узнать правильный ответ, работая с клиентом, невозможно в принципе: сравнение и анализ ответов производится только на сервере. Перехват трафика и попытка «обмануть» сервер поддельными пакетами возможны только в том случае, если известна структура запроса, то есть, фактически, если на руках у злоумышленника будут полные исходники клиентской и серверной части.

7. ***Мобильность***. Сеть любого университета или даже школы — сложная и хорошо отлаженная система. Программа должна быть проста в установке, настройке и апробировании. Кроме того, она не должна требовать существенного изменения текущих настроек сети. В нашем случае мобильность реализована за счёт использования технологии .NET Remoting, которая целиком абстрагирует разработчика от специфики данной сети.

8. ***Высокая отказоустойчивость***. Программа должна чётко и грамотно реагировать на сбои в работе сети, потерю пакетов, внезапный разрыв связи, уметь взаимодействовать с различными системами защиты (антивирусами, файрволлами, системами защиты на самих узлах связи). В случае ошибки и администратор, и пользователи должны чётко знать, что произошло и как устранить возникшую проблему. Благодаря системе исключений C# и отказу от технологии сетевого программирования через сокеты, система тестирования будет весьма устойчива. Так, в случае обрыва связи она может просто дождаться, когда связь восстановится и дать пользователю возможность продолжить тестирование.

ПРИМЕНЕНИЕ СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ НА ОСНОВЕ ПРЕЦЕДЕНТОВ В СИСТЕМАХ ДИАГНОСТИКИ ТЕХНИЧЕСКОГО СОСТОЯНИЯ МЕХАНИЗМОВ С ВРАЩАТЕЛЬНЫМ ДВИЖЕНИЕМ

Гончарова С. А.

Белорусский государственный университет информатики и радиоэлектроники, г. Минск

В настоящее время системы поддержки принятия решений (СППР) используются во многих сферах жизнедеятельности человека: в медицине для постановки диагноза, в промышленности для диагностирования технического состояния оборудования, в экономике для принятия управленческих решений и даже в философии для построения высказываний. Одним из способов организации системы поддержки принятия решений является построение системы на основе прецедентов.

Вывод на основе прецедентов – это метод, основанный на использовании знаний о ситуациях и случаях из предыдущего опыта (прецедентов). Прецедент в СППР представляет собой описание случая в виде набора информативно значимых признаков и их значений, совокупность действий, предпринятых в данных условиях (набор рекомендаций), а также возможно сохранение признака ошибочности решения проблемы, для исключения подобных ситуаций в будущем. Как правило, такие методы рассуждений включают в себя четыре основных этапа, образующие так называемый цикл рассуждения на основе прецедентов или CBR-цикл (Case-Based Reasoning) [1]:

- извлечение наиболее соответствующего (подобного) прецедента (или прецедентов) для сложившейся ситуации из библиотеки прецедентов (БП);

- повторное использование извлеченного прецедента для попытки решения текущей проблемы;
- пересмотр и адаптация в случае необходимости полученного решения в соответствии с текущей проблемой;
- сохранение вновь принятого решения как части нового прецедента.

Как правило, последний этап в описанном выше CBR-цикле исключается и выполняется лицом, принимающим решения (ЛПР).

В таких СППР чаще всего используются следующие два способа нахождения прецедентов: метод ближайшего соседа и кластеризация. В основе метода ближайшего соседа лежит измерение степени близости текущего случая и прецедента. Во втором методе база прецедентов подвергается кластеризации в пространстве признаков. При поиске аналогов для текущего случая его точка в этом пространстве сравнивается с расположением полученных кластеров. Близкими считаются прецеденты, принадлежащие кластеру, в который попадает случай.

Рассмотрим возможность применения СППР на основе прецедентов для оценки технического состояния механизмов с вращательным движением на основе вибрационных сигналов пуска-выбега. Существует множество способов оценки технического состояния механизмов по вибрационным параметрам, основанных на использовании различных типов вибрационных характеристик. Одной из важных характеристик является характеристика пуска-выбега, которая снимается во время останова или пуска объекта и показывает зависимость изменения вибрационного параметра от частоты вращения вала.

В случае сложных комплексных объектов таких, как турбины на энергетических предприятиях, может иметь место большое количество возможных неполадок, например: неуравновешенность ротора, дефект соединительной муфты, автоколебания ротора в подшипниках, дефект системы смазки и т.д. В таких случаях применение СППР на основе прецедентов нецелесообразно из-за сложности накопления вибрационных характеристик объектов с различными неполадками. Тем не менее, остается возможность применения СППР данного типа как дополнения к основной СППР, основанной на сравнении текущей характеристики с эталонной. При этом в основной СППР текущая характеристика пуска-выбега сравнивается с эталонной и выдается заключение об их степени близости, и о причинах их расхождений на основе концептуальных и экспертных знаний. Затем текущая характеристика сравнивается с базой прецедентов и выдается заключение - имела ли место подобная ситуация в прошлом или нет.

Приведем структуру отношения для хранения прецедентов в базе данных СППР оценки технического состояния механизмов по вибрационным характеристикам пуска-выбега.

Исходные данные для построения характеристик пуска-выбега накапливаются в результате работы измерительно-вычислительного комплекса «Лукомль-2001» и после предварительной обработки сохраняются в текстовых файлах.

```
Турбина 1, Подшипник 1,  14. 12. 1999 г.
Групповой просмотр. Направление вертикальное
Фаза 2 об
Оборот|Подш.1|Подш.2|Подш.3|Подш.4|Подш.5|Подш.6|Подш.7|Подш.8|Подш.9|Час|Мин|Сек|
 475.0|225.75|352.11|205.75|189.21| 23.44|127.85|154.08|147.56|166.27|  7| 23| 23|
 500.0|310.10| 23.41|272.58|243.50| 40.72|162.07|132.37|131.05|139.74|  7| 22| 51|
 525.0|285.19| 21.28|244.35|242.27| 55.80| 55.92|170.67|154.01|151.84|  7| 22| 21|
 550.0|274.86|117.67|279.77|274.37| 24.09|116.10|211.60|206.97|189.37|  7| 21| 52|
 575.0|332.17|217.76|304.08|327.40| 23.65|137.45|214.31|170.81|186.87|  7| 21| 24|
```

Рисунок 1. Пример файла данных

Каждый файл данных характеризуется такими параметрами, как предприятие, объект и точка получения характеристик пуска-выбега, дата проведения испытаний, направление контроля, тип процесса и самой характеристики.

Файлы представляют собой таблицу данных, в первой колонке которой указывается частота вращения вала, а в остальных - значение соответствующего вибрационного параметра и время (рис. 1) [2].

Структура отношения в СППР для хранения информации о характеристиках имеет следующий вид (рис. 2):

Characteristic	
PK	**CharacteristicID**
	Enterprise
	Object
	TestDate
	Number
	Process
	Direction
	Type
	FilePath
	FileName

Рисунок 2. Структура отношения Characteristic,

где поле CharacteristicID – уникальный идентификатор характеристики;

Enterprise – название предприятия, на котором проводились измерения;

Object – объект или механизм исследования;

TestDate – дата сбора данных;

Number – номер опыта;

Process – тип исследуемого процесса (пуск или выбег);

Direction – направление измерения (горизонталь, вертикаль, осевое);

Type – тип снятой характеристики (первая оборотная, вторая оборотная, фаза первой оборотной, фаза второй оборотной);

FilePath – путь размещения файла данных;

FileName – имя файла, содержащего характеристики пуска-выбега.

Для расширения БД для хранения информации о прецедентах необходимо добавить поле для хранения признака характеристики (прецедент или эталонная) и ввести новое отношение для дополнительной информации о прецеденте (рис. 3), где:

Kind – тип характеристики (эталонная или прецедент);

Problem – тип неполадки;

Solution – действия, предпринятые для решения данной проблемы;

Result – результативность решения.

Связь между данными отношениями «один ко многим», т.к. возможна ситуация, когда возникает несколько неполадок, либо есть несколько вариантов решения проблемы.

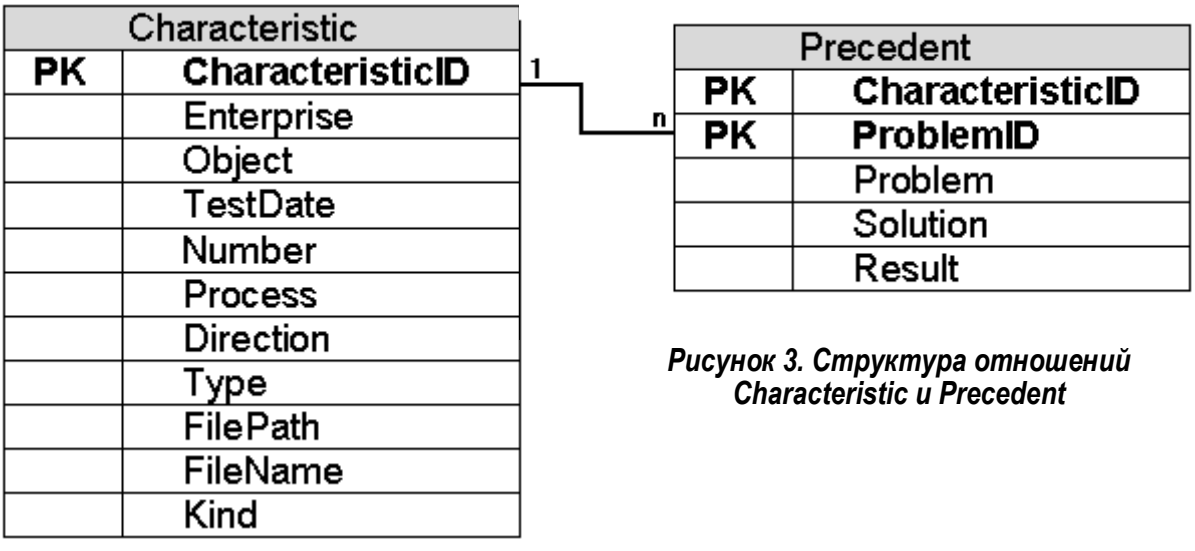

Рисунок 3. Структура отношений Characteristic и Precedent

Для возможности использования двойной обработки характеристики (сравнения с эталонной характеристикой и с базой прецедентов) необходимо:

- ввести величину максимального отклонения текущей характеристики от эталонной, что является признаком начала процесса сравнения характеристики с базой прецедентов;
- ввести признак максимального отклонения от наиболее близкого прецедента для выдачи заключения о том, что данная ситуация в прошлом не возникала.

Также в данной системе необходимо реализовать возможность изучения новых проблемных ситуаций и занесений их в базу прецедентов.

Реализация дополнительного модуля вывода на основе прецедентов в СППР оценки технического состояния механизмов по вибрационным характеристикам пуска-выбега позволит повысить эффективность работы системы за счет увеличения количества методов анализа характеристики.

Литература

1. Aamodt A., Plaza E. Case-based reasoning: foundational issues, methodological variations, and system approaches // AI Communications. IOS Press. Vol. 7: 1. 1994. – P. 39-59.
2. Бранцевич П.Ю., Гончарова С.А. Программная система обработки и анализа вибрационных характеристик выбега/ «Известия Белорусской инженерной академии» №1(15)/2 – Минск, 2003. - с. 26-28.

РАЗРАБОТКА ПАКЕТА ПРИКЛАДНЫХ ПРОГРАММ СИНТЕЗА СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ НА БАЗЕ UML

Жиляк Н. А.

Белорусский государственный технологический университет, г. Минск

Одной из важнейших задач при разработке системы становится уменьшение количества ошибок, вносимых в нее. При этом значительно возрастает роль стадий проекта, предшествующих его непосредственной реализации.

Кризис разработки сложного программного обеспечения, а особенно корпоративных информационных систем (КИС), как правило, выражается в следующих обстоятельствах: проекты не сдаются в срок; существенно перерасходуется бюджет проектов; проекты не удовлетворяют заданным спецификациям; модификация проектов становится чрезвычайно трудоемкой и рискованной.

Преодоление этого кризиса связывают с объектно-ориентированной технологией программирования. Теперь говорят о начале "революции в программировании". При этом имеется в виду уже не объектно-ориентированное программирование, а разработка проектов КИС на основе визуального моделирования компонент (CBD – Component Based Development).

Современные инструментальные средства визуальной CBD-разработки характеризуются следующими свойствами:

1. Поддержка генерации кода и обратного проектирования (т.е. восстановление визуальной модели по программному коду) сразу для нескольких языков, включая: Visual Basic, C++, Java, PowerBuilder, CORBA Interface Definition Language (IDL), а также Data Definition Language для большинства СУБД [1].
2. Поддержка визуального объектно-ориентированного моделирования и полная совместимость с языком UML (Unified Modeling Language), который, начиная с 1997 года, задает стандарт для языков описания моделей.

В настоящее время широко известны два подхода к проведению декомпозиции системы при построении ее визуальной модели:

1. Основанная на упорядочении событий/потоков данных алгоритмическая декомпозиция (структурный анализ/проектирование).