ШЕЕНОК
Дмитрий Александрович *- аспирант Красноярского института железнодорожного транспорта - филиала Иркутского государственного университета путей сообщения в г. Красноярске (КрИЖТ ИрГУПС)*
Адрес: 660028, г. Красноярск, ул. Ладо Кецховели, 89
e-mail: dmitryshkras@rambler.ru

Оптимизация программной архитектуры при разработке информационной системы Пенсионного фонда России

Программно-технический комплекс «Администрирование страховых взносов»

Разработанный в Отделении Пенсионного фонда Российской Федерации (ПФР) по Красноярскому краю программно-технический комплекс «Администрирование страховых взносов» (ПТК АСВ) автоматизирует деятельность по администрированию страховых взносов ПФР и почти 2500 его территориальных органов. На сегодняшний день этот комплекс применяется во всех 82 отделениях ПФР [1].

К разработке ПТК АСВ региональное Отделение ПФР приступило в 2009 году в рамках подготовки к принятию функции администрирования страховых взносов. С 1 января 2010 года во всех регионах страны данная система была внедрена как резервная, а с 28 июля того же года ПТК АСВ получил статус основной программы. ОПФР по Красноярскому краю занимается внедрением и сопровождением программного комплекса в других региональных отделениях ПФР.

ПТК АСВ обеспечивает возможность выполнения следующих задач:

- интеграция с ПТК «Страхователи» в части сведений о регистрации и постановке на учет (снятии с учета) в органах ПФР плательщиков страховых взносов в Пенсионный фонд Российской Федерации и Фонды обязательного медицинского страхования в рамках электронного обмена информацией с регистрирующими налоговыми органами по технологии «одно окно»;
- интеграция с ПТК «Страхователи» в части использования выписок из лицевого счета администратора доходов, платежных документов плательщиков, уведомлений о зачетах и уточнениях вида и принадлежности платежа страхователей, переданных органами Федерального казначейства (ОФК);
- прием и регистрация отчетности страхователей с соблюдением всех требований законодательства Российской Федерации, утвержденных форматов и методик проверки;
- обеспечение камеральных проверок отчетности с соблюдением всех требований законодательства Российской Федерации, утвержденных форматов и методик проверки;
- обеспечение механизма контроля над полнотой и своевременностью уплаты страховых взносов на ОПС и ОМС;
- взыскание страховых взносов, пеней и штрафов за счет денежных средств, имеющихся на счетах плательщика страховых взносов в кредитных учреждениях;
- организация механизма контроля над взысканием недоимки, пеней и штрафов с формированием документов аналитического и сводного учета по результатам взыскания;
- обеспечение зачета и возврата сумм, излишне уплаченных страховых взносов, пеней, штрафов;
- обеспечение возврата сумм излишне взысканных страховых взносов, пеней и штрафов;
- формирование утвержденных законодательством и органом контроля уплаты страховых взносов форм отчетности;
- взаимодействие с ПО, осуществляющим электронный документооборот по каналам связи с применением ЭЦП с участниками процесса администрирования страховых взносов, в рамках заключенного между участниками соглашения.

ПТК АСВ выполнен как единое комплексное web-приложение на платформе объектно-ориентированной СУБД *InterSystems Cache*.

ПТК АСВ взаимодействует с другими штатными системами: ПТК «Страхователь», СОУ «Эталон», ПК «Бесконтактной передачи информации», СЭД Федерального казначейства, СЭД Федеральной службы судебных приставов, ПТК «Система персонифицированного учета», ПК «Банковские счета», ПК «Выездные проверки». Также с помощью ПТК АСВ происходит взаимодействие с системами Федеральной налоговой службы, ЦБ РФ, Федеральной базы страхователей, Фонда обязательного медицинского страхования.

Среднее количество конечных пользователей, эксплуатирующих систему, составляет около 15 тыс. человек.

ПТК АСВ постоянно модернизируется, охватывая новые функции и задачи. Над модернизацией ПТК

АСВ работают специалисты компании «Сибирские интеграционные системы». В постановке задач и тестировании новых версий участвуют Отделения ПФР по г. Москве и Московской области, по г. Санкт-Петербургу и Ленинградской области и многие другие.

Общий размер кода системы - более 573 тыс. строк, что позволяет отнести ПТК АСВ к классу крупномасштабных проектов. Поэтому при сопровождении системы необходимо наиболее полное документирование всех этапов ее жизненного цикла [2]. В процессе разработки системы в специальной системе отслеживания ошибок *Atlassian JIRA* фиксируются отчеты тестировщиков о выявленных дефектах, затраченном времени тестирования, времени устранения дефектов разработчиками и др. Процесс разработки носит итерационный характер, и обнаруженные ошибки устраняются до достижения необходимого уровня надежности.

Программная архитектура

В любой современной методологии разработки программного обеспечения выделяется этап разработки программной архитектуры. Уже на этапе формирования требований начинается проектирование архитектуры будущей программной системы. Системный архитектор определяет общую структуру каждого архитектурного представления, декомпозицию представлений и интерфейсы взаимодействия элементов. Таким образом, происходит разбиение большой системы на более мелкие части (модули), в соответствии с определенным уровнем абстракции. Поэтому архитектурный компонент может быть определен по-разному в зависимости от архитектурного подхода и степени подробности описания архитектуры.

Модульность построения приводит к использованию иерархической структуры взаимодействия модулей программы. Иерархическая схема, отражая функции модулей, одновременно показывает структуру связей между ними. Иерархические структуры системы характеризуются, с одной стороны, вертикальным управлением, когда модули верхнего уровня имеют право вмешательства и координирования работы модулей нижнего уровня. С другой стороны, действия модулей верхнего уровня зависят от информации, полученной в результате функционирования модулей нижних иерархических уровней. Таким образом, сверху вниз идут в основном управляющие воздействия, а снизу вверх - информация о соответствующих решениях и переменных. Число архитектурных уровней в модели архитектуры ПО зависит от частного проекта системы.

Известно, что чем позже будет обнаружена ошибка проектирования, тем дороже обойдется ее исправление разработчику программного продукта. Поэтому для соблюдения требований к надежности разрабатываемой программной системы необходимо уже на стадии дизайна архитектуры тщательно прорабатывать связи между компонентами и устанавливать иерархию их взаимодействия. Компоненты, которые наиболее часто используются или архитектурно связанные с множеством других компонентов, оказывают наибольшее влияние на надежность системы. Зависимости компонентов позволяют неисправности распространяться из компонента, в котором она происходит, к другим компонентам. Обнаружение отказов во время разработки зависит от тщательности этапа тестирования. Одной из наиболее перспективных и уже положительно зарекомендовавших себя методологий обеспечения высокой надежности и отказоустойчивости программного обеспечения является введение программной избыточности. Согласно данной методологии предполагается, что возникновение сбоя в функционально эквивалентных модулях (версиях) на одних и тех же входных данных происходит в различных точках исполнения [3].

Существует два основных подхода к реализации программной избыточности [4]:

1. *NVP (N-version programming* - *N*-версионное программирование) - версии выполняются параллельно во времени. Результат их работы определяется с помощью какого-либо алгоритма голосования. Алгоритмы голосования могут быть разными в зависимости от задачи. Обычно используется выбор результата по абсолютному большинству (эквивалентных выходов больше половины) или по согласованному большинству (самая большая группа эквивалентных выходов). При этом выходные значения являются идентичными при заданной погрешности.

2. *RB (Recovery Block* - блок восстановления) - версии выполняются последовательно. После выполнения первой версии программного компонента - приемочный тест, который по результатам выполнения принимает результаты вычисления либо запускает следующую версию компонента.

Надежность функционирования программного компонента зависит от глубины программной избыточности (количества версий v), а также выбранного подхода к ее реализации p:

$$R = F(v, p, R_{v1},..., R_{vn}),$$

где R - надежность функционирования программного компонента с программной избыточностью, F - функция расчета надежности компонента с избыточностью, v - количество версий, p - подход к реализации программной избыточности *(NVP* или *RB)*, R_{vn} - надежность функционирования n-й версии компонента.

Трудозатраты на разработку программного компонента также зависят от глубины избыточности:

$$T = G(v, T_v),$$

где T - трудозатраты на разработку компонента с программной избыточностью, G - функция расчета трудозатрат на разработку компонента с избыточностью, v - количество версий, T_v - трудозатраты на разработку одной версии программного компонента.

В связи с масштабом внедрения и сложностью программной системы к ее качеству предъявляются высокие требования, в частности, к надежности функционирования. Для обеспечения высокой надежности требуется больший вклад трудовых и финансовых ресурсов со стороны исполнителя. Поэтому при любой модернизации системы перед разра-

ботчиком возникает задача построения оптимальной архитектуры программного обеспечения при минимальных трудозатратах [3].

Архитектура ПТК АСВ была проанализирована до определенного уровня детализации и представлена в виде 1030 взаимодействующих программных компонентов. На основании анализа статистики системных журналов были произведены расчеты вероятностей сбоя для всех программных компонентов. Компоненты, разработанные достаточно давно, функционируют намного надежнее, чем разработанные относительно недавно.

При выполнении одного из этапов модернизации системы было определено, что необходимо реализовать 36 новых программных компонентов, по 15-ти из которых возникла неопределенность в размере трудозатрат для достижения заданного уровня надежности. Для каждого из 15 компонентов была дана оценка возможных уровней надежности и соответствующих им трудозатрат из 3 вариантов с использованием модели роста надежности *SGRM* [4]. Таким образом, для каждого компонента был задан массив значений:

$(R_i, T_i),..,(R_i, T_i)$,

где i - номер варианта надежности компонента и соответствующих трудозатрат на ее достижение.

В архитектуре системы было выбрано 2 наиболее важных компонента, в которых возможно введение программной избыточности не более чем из двух версий.

По остальному 21 новому компоненту спрогнозированные трудозатраты и надежность были детерминированы.

Надежность и трудозатраты каждого нового архитектурного компонента влияют на значения интегральных показателей коэффициента готовности системы S и трудоемкости разработки T_s. Функции расчета данных показателей заданы алгоритмически и зависят от значений надежности каждого компонента и трудозатрат на ее достижение [3]:

$S(R_1,..,\ R_n)$,

$T_s(T_1,..,\ T_n)$,

где $R_{1..n}$ - значения надежности компонентов, $T_{1..n}$ - значения трудозатрат на разработку компонентов.

Оптимизация характеристик архитектурных компонентов

Чтобы выбрать оптимальные характеристики реализуемых программных компонентов, необходимо решить следующую задачу оптимизации:

$S \rightarrow max,\ S \geq S^0$,

$T_s \rightarrow min,\ T_s \leq T_s^0$,

где S - критерий оценки коэффициента готовности системы, T_s - критерий оценки трудоемкости разработки системы, S^0, T_s^0 - предельные допустимые уровни критериев.

По оценочным данным было определено, что самое надежное и затратное архитектурное решение имеет T_s = 720.2 чел.-час. и S = 0.9904613, а самое ненадежное и дешевое - T_s = 403.2 чел.-час. и S = 0.9883245. Для выполнения данной модернизации системы заданы следующие ограничения:

$S \geq 0.99$,

$T_s \leq 650$ чел.-час.

Пространство оптимизации такой задачи составляет $7.03 \cdot 10^8$, поэтому решение с помощью полного перебора невозможно [3]. Дополнительную сложность в решение задачи вносит наличие двух критериев оптимальности и алгоритмическое задание функций для их расчета. Особую эффективность при решении задач многокритериальной оптимизации показали генетические алгоритмы (ГА) [5].

Генетический алгоритм представляет собой метод оптимизации, основанный на концепциях естественного отбора и генетики. В этом подходе переменные, характеризующие решение, представлены в виде генов в хромосоме. Генетический алгоритм оперирует конечным множеством решений (популяцией) - генерирует новые решения как различные комбинации частей решений популяции, используя такие операторы, как отбор, рекомбинация (кроссинговер, скрещивание) и мутация. Новые решения позиционируются в популяции в соответствии с их положением на поверхности исследуемой функции.

Таблица 1

Параметры запусков генетического алгоритма

№ запуска	Количество популяций и особей	Параметры кроссинговера	Количество уникальных решений	Количество удовлетворительных недоминируемых решений
1	15 поколений по 20 особей	3-точечный	207	1
2	30 поколений по 10 особей	3-точечный	48	1
3	15 поколений по 20 особей	3-точечный с вероятностью разрыва связанных генов 0.5.	170	4
4	30 поколений по 10 особей	3-точечный с вероятностью разрыва связанных генов 0.5.	61	1
5	15 поколений по 20 особей	равномерный	254	3
6	30 поколений по 10 особей	равномерный	145	0

Таблица 2

Характеристики оптимальных вариантов модернизации

№ варианта	Коэффициент готовности системы	Трудозатраты на модернизацию, чел.-час.
1	0.9900439	549,41
2	0.9900628	561,94
3	0.9901376	564,37
4	0.9903010	578,16

Для новых компонентов архитектуры, в которых возможно введение программной избыточности, может быть изменено количество версий *v*, подход к реализации программной избыточности *p*, номер варианта надежности версий компонента и трудозатрат на ее достижение *i*. Для компонентов без программной избыточности может быть изменен только номер варианта надежности и трудозатрат на ее достижение *i*.

Для решения данной задачи автором был разработан генетический алгоритм для многокритериальной условной оптимизации, основанный на методе с независимой селекцией Шаффера *VEGA (Vector Evaluated Genetic Algorithm)* [6, с. 33].

Алгоритм должен находить решение за приемлемое время. Расчет коэффициента готовности и трудозатрат для архитектуры такого проекта на ПК средней производительности составляет около 3 минут, поэтому алгоритму будет дано не более 300 расчетов. Таким образом, будет просчитано не более $4.3 \cdot 10^{-5}$ % поискового пространства.

По результатам работы алгоритма на тестовых задачах были определены следующие наилучшие параметры:

1. Вероятность скрещивания - 0.95;
2. Вероятность мутации особи - 0.15;
3. Вероятность мутации гена - 0.1.

В **таблице 1** приведены индивидуальные параметры ГА по шести запускам.

Наибольшее количество удовлетворительных решений было найдено при использовании модифицированного 3-точечного оператора скрещивания с вероятностью разрыва связанных генов 0.5.

Длительность выполнения каждого запуска алгоритма в среднем составила 5 часов. Характеристики найденных недоминируемых вариантов модернизации системы приведены в **таблице 2.**

Такое низкое количество найденных решений объясняется жесткими заданными ограничениями и лимитом на количество расчетов функций критериев. При больших временных ресурсах на выполнение ГА могут быть найдены еще более лучшие решения.

Заключение

Таким образом, была проанализирована архитектура программной системы Пенсионного фонда Российской Федерации и выявлены компоненты, которые необходимо разработать для одного из этапов ее модернизации. С помощью генетического алгоритма за приемлемое время были найдены оптимальные варианты модернизации системы, при ограничениях на надежность ее функционирования и необходимые трудозатраты.

Поэтому можно сделать вывод, что использование генетического алгоритма позволяет эффективно находить оптимальные характеристики архитектурных компонентов программного обеспечения при разработке крупномасштабных проектов.

Литература:

1. Основная программа по администрированию // Издание Пенсионного фонда Российской Федерации «Я работаю в ПФР». - 2011. [Электронный ресурс]. - URL: http://files.pfrf.ru/userdata/presscenter/gazeti/2011/gaz_co_201110.pdf (дата обращения 15.03.2012).

2. Липаев В.В. Сопровождение и управление конфигурацией сложных программных средств. - М.: СИНТЕГ, 2006. - 372 с.

3. Жуков В.Г., Шеенок Д.А, Терсков В.А. Повышение надежности программного обеспечения сложных систем // Вестник СибГАУ. Вып. 5(45). - Красноярск, 2012. - С. 28-33.

4. Русаков М.А. Многоэтапный анализ архитектурной надежности в сложных информационно-управляющих системах: Дис. канд. техн. наук: Красноярск, 2005. - 168 с.

5. Шеенок Д.А. Генетический алгоритм поиска оптимальной архитектуры программного обеспечения / Актуальные вопросы современной науки: Материалы IV Международной научной конференции 14-15 декабря 2012 года, г. Санкт-Петербург. - Петрозаводск: Петропресс, 2012. - С.62-68.

6. Сергиенко Р.Б. Автоматизированное формирование нечетких классификаторов самонастраивающимися коэволюционными алгоритмами: Дис. канд. техн. наук: Красноярск, 2010. - 192 с.