ТИТУЛЬНЫЙ ВРЕМЕННЫЙ (ЧТОБ НЕ ЗАБЫТЬ НАЗВАНИЕ)

Диплом

Интегрированная система поддержки принятия решения о выборе режимов механической обработки тонкостенных деталей

Еремейкин Пётр Александрович

Москва, 2016 г.

СОДЕРЖАНИЕ

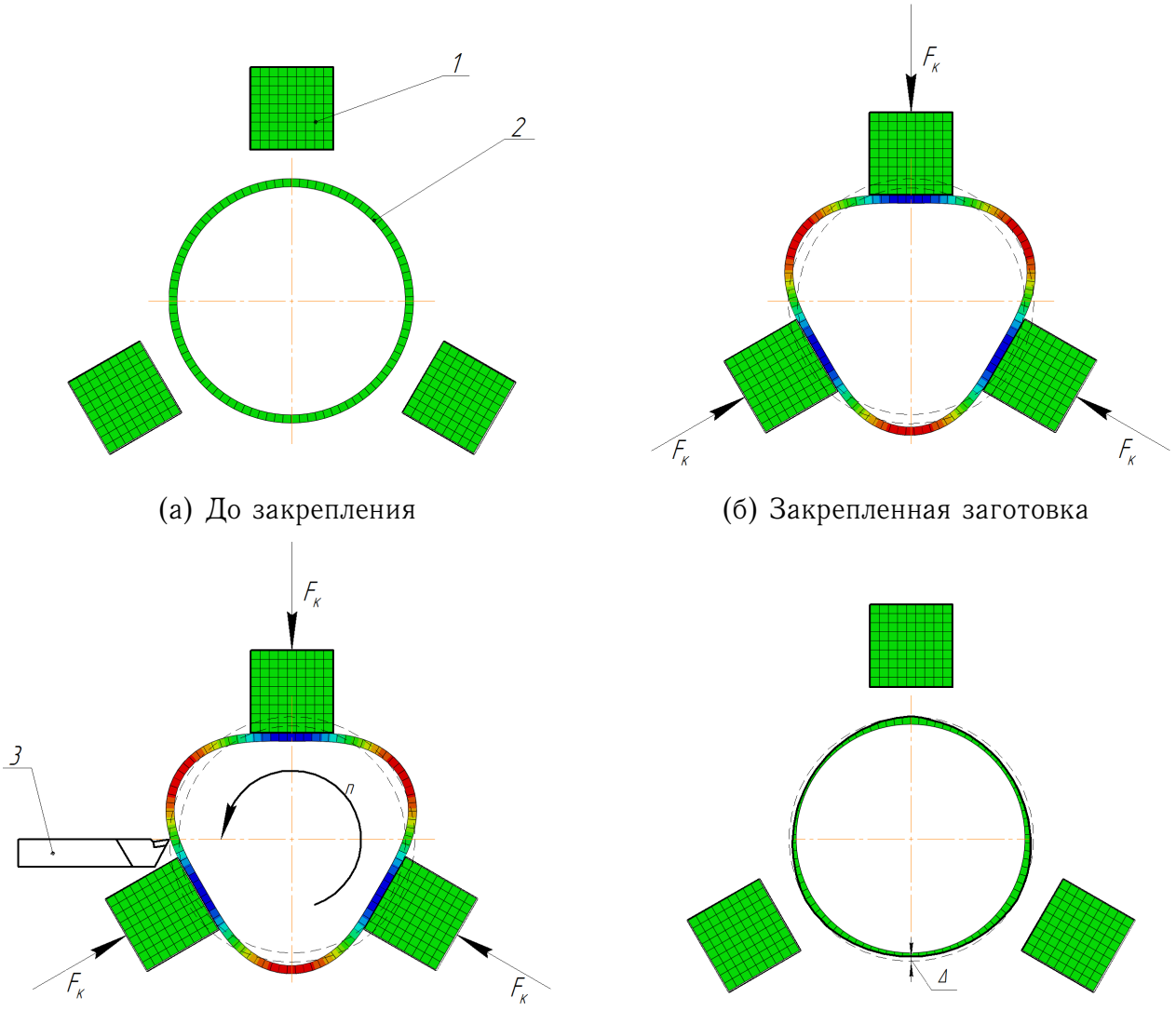
ГЛАВА 1
ВВЕДЕНИЕ

Эксплуатационные характеристики изделий машиностроения в большой мере зависят от качества изготовления деталей, составляющих изделие. Качество изготовления определяется степенью соответствия параметров готовой детали и параметров, достижение которых требуется документацией. В роли такого параметра может выступать точность изготовления геометрических размеров, определяемая допуском.

Погрешности различных видов препятствуют абсолютно точному воспроизведению заданных параметров. Например, на точность изготовления деталей влияют погрешности базирования, измерения, настройки инструмента и т.д. При обработке нежестких деталей на первый план выходят погрешности, связанные с деформацией самой детали.

Рассмотрим пример обработки тонкостенной детали в трехкулачковом патроне токарного станка. Схема обработки изображена на рисунке 1. Цифровыми сносками обозначены: 1 - кулачок патрона, 2 - цилиндрическая заготовка, 3 - токарный резец. До начала обработки деталь находится в недеформированном состоянии (рисунок 1а). После закрепления детали силы, действующие со стороны кулачков, деформируют заготовку таким образом, что профиль заготовки отличается от идеально цилиндрического (рисунок 1б). Дополнительно заготовка деформируется под действием силы резания, а также в связи с нагревом. В процессе резания инструмент неравномерно снимает припуск по окружности заготовки (рисунок 1в). После окончания обработки и снятия усилий закрепления отклонение от круглости заготовки достигает величин, сравнимых с величиной допуска и в зависимости от конкретных условий может превышать его (рисунок 1г).

На практике в случае обработки нежестких деталей применяется специальное технологическое оборудование, позволяющее уменьшить возникающие деформации. Использование сырых кулачков, растрачиваемых под диаметр заготовки - один из способов уменьшения деформаций закрепления. После растачивания кулачок охватывает деталь по большей площади.



(а) До закрепления

(б) Закрепленная заготовка

(в) Обработка

(г) Результат

Рисунок 1 – Обработка тонкостенной заготовки

Также используется обработка с технологическим заполнителем (например, легкоплавким материалом), который увеличивает жесткость детали на время обработки и выплавляется после окончания обработки. Вариант обработки на нормативных режимах резания в специальных приспособлениях привлекателен тем, что он позволяет применять апробированные технологические процессы. Однако, необходимость создания при этом специальных приспособлений требует значительных материальных, производственных и временных затрат. Это удорожает технологическую подготовку производства и увеличивает её сроки.

Ввиду указанных ограничений на существующие способы обработки, предлагается рассмотреть еще один метод - подбор режимов обработки и условий закрепления таким образом, чтобы обеспечить изготовление размеров в рамках допуска. Предполагается, что такой метод может найти применение прежде всего в единичном и мелкосерийном производстве, располагающим, как правило, только универсальным оборудованием. Суть подхода заключается в том, чтобы заранее, на этапе разработки технологической документации, определить режимы резания, менее эффективные с точки зрения производительности, но оптимальные с точки зрения точности изготовления. С уменьшением сил резания уменьшаются и деформации заготовки, а также требуемые усилия закрепления, но с другой стороны, увеличивается время обработки. Подобрав баланс между скоростью и точностью обработки, технолог имеет возможность назначить режимы резания, приемлемые для изготовления заданной детали без привлечения дополнительной технологической оснастки. Достоинством такого подхода является меньшая ресурсоёмкость и продолжительность технологической подготовки производства. Однако, данный подход в настоящее время ещё не получил достаточного научного обоснования и, как следствие, не поддержан методическими рекомендациями, необходимыми для его применения в промышленных масштабах [1].

Для поддержки решения о назначении режимов резания предлагается разработать информационную систему, позволяющую анализировать деформации заготовки при заданных геометрических параметрах и режимах резания. Такой инструмент можно использовать для последовательной проверки ряда значений параметров процесса и выбора наиболее рациональных.

Система, спроектированная в соответствии с современными тенденциями в разработке ПО станет эффективным инструментом для назначения режимов обработки. Объектно-ориентированная модульная архитектура такой системы позволит расширять её функционал в случае возникновения новых требований к применению программы, а независимость от конкрет-

ной операционной системы позволит легко внедрить такие системы в процесс разработки технологической документации на предприятиях безотносительно конкретной существующей информационной среды.

ГЛАВА 2
ТЕХНИЧЕСКОЕ ЗАДАНИЕ

2.1 Основания для разработки

Основанием для разработки системы являются документы:

1. Задание на выполнение дипломного проекта
2. Календарный план на выполнение дипломного проекта

2.2 Назначение разработки

Функциональным назначением разработки является обеспечение интегрированной рабочей среды для гибкого моделирования процессов деформирования тонкостенных заготовок в процессе их токарной обработки. Под гибкостью понимается возможность изменения параметров процесса в зависимости от расчетного случая и конфигурации заготовки. Эксплуатационным назначением является обеспечение инструментального средства для последовательного определения рациональных режимов резания тонкостенных заготовок и автоматизация сопутствующих расчетов.

2.3 Требования к функциональным характеристикам

Система информационной поддержки должна обеспечивать выполнение следующих функций:

1. Расчет деформаций тонкостенной заготовки при токарной обработке с закреплением в кулачковом патроне.
2. Автоматизация необходимых сопутствующих расчетов (режимов резания)

3. Изменение параметров рассматриваемого процесса: геометрии заготовки и оснастки и значений силовых факторов (силы резания и закрепления)

4. Представление результатов расчета в виде графиков или таблиц деформаций

5. Генерация отчета по результатам расчета

6. Вывод дополнительных информационных отладочных сообщений

2.4 Требования к надежности

Разрабатываемая система должна обеспечивать надежную работу в условиях ошибочного пользовательского ввода. При возникновении системных исключительных ситуаций необходимо обеспечить их обработку чтобы не допустить аварийного завершения программы.

2.5 Условия эксплуатации

Аппаратные средства должны эксплуатироваться в помещениях с выделенной розеточной электросетью 220В $\pm$ 10%, 50 Гц с защитным заземлением при следующих климатических условиях:

– температура окружающей среды – от 15 до 30 градусов С;

– относительная влажность воздуха – от 30% до 80%;

– атмосферное давление – от 630 мм. р.с. до 800 мм. р.с.

2.6 Требования к составу и параметрам технических средств

Программный продукт должен работать на компьютерах со следующими характеристиками:

– объем ОЗУ не менее 2 Гб;

– объем жёсткого диска не менее 40 Гб;

– микропроцессор с тактовой частотой не менее 1.5 ГГц;

– монитор с разрешением от 1024*768 и выше.

2.7 Требования к информационной и программной совместимости

Windows, Linux, и т.д.

2.8 Требования к маркировке и упаковке

Не предъявляются

2.9 Требования к транспортированию и хранению

Не предъявляются

2.10 Требования к программной документации

Не предъявляются

2.11 Технико-экономические показатели

? ? ?

2.12 Стадии и этапы разработки

? ? ?

2.13 Порядок контроля и приемки

? ? ?

2.14 Приложения

? ? ?

2.15 Предпроектное исследование

? ? ?

ГЛАВА 3
РАЗРАБОТКА КОНЦЕПЦИИ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ

3.1 Принцип моделирования

Согласно техническому заданию система должна обеспечить расчет деформаций тонкостенной заготовки в процессе токарной обработки. Токарная обработка характеризуется большим разнообразием возможных вариантов осуществления в зависимости от обрабатываемой поверхности и способа закрепления. Разработать и запрограммировать адекватную модель, основанную исключительно на математическом представлении известных теоретических закономерностях сопротивления материалов для каждого из возможных вариантов - чрезвычайно сложная задача с технической точки зрения.

Известно, что современные CAE системы позволяют успешно решать задачи определения деформаций и напряжений для объектов практически неограниченной сложности благодаря использованию метода конечных элементов (МКЭ). После разбиения рассматриваемых объектов на конечные элементы определенного типа и наложении ограничений на эти элементы согласно закономерностям предметной области задача сводится к решению системы из большого количества уравнений. Успех решения полученной системы уравнений как правило определяется только отведенным на решение временем работы ЭВМ.

Тем не менее, сама по себе реализация метода конечных элементов в разрабатываемом продукте не возможна из-за большой трудоемкости и требует огромной исследовательской и проектной работы целого коллектива профессионалов. Таким образом, наилучшим выходом из сложившейся ситуации представляется использование готовой CAE системы. Применение системы МКЭ расчета возможно, если она отвечает следующим трем основным требованиям:

1. Предоставляет возможность расчета заранее составленной модели под

управлением внешнего процесса операционной системы (в роли которого будет выступать разрабатываемая система анализа деформаций)

2. Имеет механизмы параметризации модели. В наилучшем случае параметризация должна достигаться благодаря выполнению пользовательских сценариев.

3. Позволяет получить результат расчета в виде графиков, диаграмм напряжений и деформаций а также в числовом виде для каждого элемента модели.

В рамках перечисленных соображений можно сформулировать концепцию системы следующим образом. На первом этапе система собирает информацию о значениях параметров модели через графический пользовательский интерфейс. По завершению ввода функция системы состоит в формировании задания для используемой готовой CAE системы. На этапе подготовки задания производится дополнительный расчет необходимых величин, если они не были непосредственно указаны на первом этапе и могут быть получены расчетным путем. Третий этап состоит в вызове расчетного ядра CAE системы по сформированному заданию. По завершению обработки модели разрабатываемая система агрегирует результаты и отображает их пользователю, при необходимости генерируя отчет.

Таким образом, основная идея состоит в расчете модели обработки с использованием сторонней CAE системы. Предполагается, что роль разработчика моделей будет выполнять эксперт, как правило не являющийся пользователем системы. В задачи эксперта входит описание нового расчетного случая при помощи инструментов, предоставляемых конкретной выбранной системой МКЭ расчета и другие необходимые работы по подготовке системы. Готовая модель предоставляется в распоряжению пользователю, в задачи которого входит выбор наиболее подходящей модели из множества имеющихся и её загрузка в разрабатываемую систему поддержки.

3.2 Принцип модульности

Уже на этапе формулирования концепции системы становиться видна её большая сложность. Для того чтобы сохранить контроль над расширяющейся по мере разработке системы и не допустить её деградации следует определиться с принимаемыми для этого мерами и также включить их в концепцию системы, так как принимаемые меры коренным образом повлияют на процесс проектирования.

В соответствии с накопленным мировым опытом в разработке программного обеспечения одним из наиболее жизнеспособных способов контроля сложности является разработка приложения согласно принципам объектно-ориентированного программирования. Во-первых, ООП методология позволяет добиться большого процента повторного использования кода благодаря принципам наследования и полиморфизма [2]. Во-вторых, принципы абстракции и инкапсуляции облегчают задачу программиста так как они существенно ограничивают область кода, который влияет на рассматриваемый участок программы и позволяют формулировать мысли в терминах предметной области.

Разрабатываемая система сложна не только по причине большой трудоемкости процесса разработки, но также и процесса поддержки готовой системы. В случае возникновения новых функциональных требований, не покрытых техническим заданием на первом этапе, потребуется совершить большой объем работ, если заранее не предусмотреть модульность системы. Модульность системы позволит “присоединить” к готовой системе недостающие функциональные элементы - модули, без переработки какой-либо существенной части самого приложения.

Как показано выше, сам по себе переход от структурного программирования к объектно-ориентированному - большой шаг вперед. Однако, такой шаг ещё не гарантирует структурированной модульной архитектуры приложения. Отдельный класс хоть и инкапсулирует некоторые данные, тем не менее не является модулем в смысле всего приложения. Для достиже-

ния этой цели потребуется применение специальных мер, зависящих от выбранного языка программирования. Такие меры могут представлять, например, использование определенного набора шаблонов проектирования, в случае реализации модульности системы своими силами, или использования одного из многочисленных фреймворков в другом случае.

3.3 Принцип комплексности

Принцип модульности оставляет широкие возможности для расширения функционала системы даже после выпуска готовой системы. Однако такой возможностью следует пользоваться в пределах разумного. Не следует исходить из этой концепции для оправдания отсутствия каких-либо востребованных функций. В контексте текущего проекта должен быть разработан исходный набор модулей, обеспечивающий выполнение большинства функций, которые могут понадобиться пользователю при работе в рамках рассматриваемой предметной области. Это означает что следует комплексно подойти к решению проблемы.

В рамках базового набора модулей согласно техническому заданию должны обязательно присутствовать модуль для расчета режимов резания, модуль графического отображения результатов и модуль генерации отчетов. Также следует включить в основной состав модули просмотра трехмерного изображения рассчитываемой модели и вывода текстовой информации, формируемой CAE системой.

Кроме самого факта наличия модулей в системе необходимо обеспечить взаимный обмен информацией между ними. Например, передачу значений, полученных в модуле расчета режимов резания в модуль параметризации модели. Этот прием позволит избавить пользователя от случайных ошибок при копировании и сделает работу с системой более удобной. Согласно идее комплексности, полезной особенностью может стать предоставление справочных данных для выбора в качестве значений вводимых параметров.

Таким образом, разрабатываемая система должна стать результатом

многосторонней проработки проблемы.

3.4 Принцип независимости

При разработке системы уже на самых ранних стадиях необходимо ориентироваться на возможное многообразие используемых на предприятиях программных и аппаратных средств. Известно, что для современного рынка информационных технологий характерна большая вариативность используемого программного обеспечения. В рамках проекта интерес представляют операционные системы и системы инженерных расчетов (CAE). Для разных предприятий сочетания конкретных решений в этих областях могут быть различными, например, роль операционной системы может выполнять какая либо версия Microsoft Windows или Linux. В роли CAE системы может использоваться Ansys, Abaqus, в некоторых случаях Siemens NX или SolidWorks.

В идеальном случае следует ориентироваться на независимость проектируемого приложения от индивидуальных особенностей какой-либо внешней системы. Это позволит расширить границы применения программы и облегчить задачу её освоения в рамках существующей информационной среды. Удовлетворить условиям концепции независимости поможет правильный выбор средств разработки а также определенные проектные меры, например, разработка межсистемного взаимодействия на основе соглашений (интерфейсов).

ГЛАВА 4
ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ — ПЕРЕИМЕНОВАТЬ?

4.1 Формулировка целей

Основной целью разработки является снижение трудоемкости назначения рациональных режимов резания. Эта цель может быть представлена в виде трех подцелей более низкого уровня:

1. Снижение трудоемкости предварительных расчетов
2. Снижение трудоемкости основных расчетов
3. Снижение трудоемкости анализа результатов

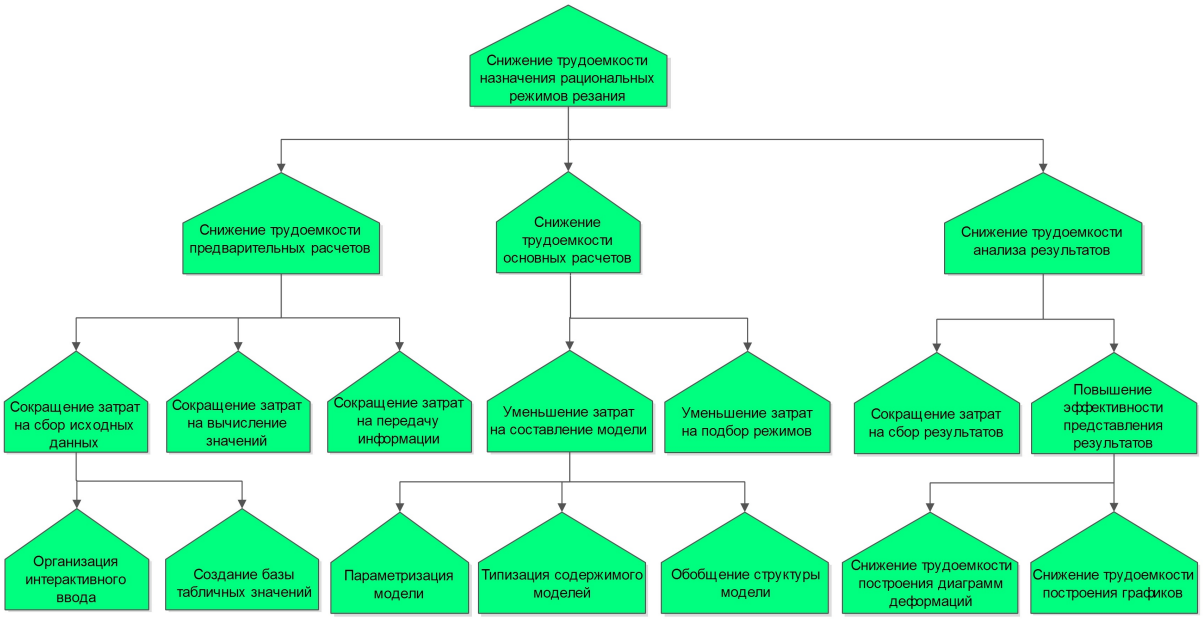


Рисунок 2 – Дерево целей

Снижение трудоемкости предварительных расчетов сводится к автоматизации вычисления режимов резания по традиционным методикам. Эта цель может быть достигнута при сокращении затрат на сбор исходных данных, вычисление значений и передачу информации. Под традиционными методиками понимается прежде всего таблично-аналитический подход,

описанный в [3]. Согласно указанной литературе, для вычисления параметров режима резания необходимо выбрать исходные данные из таблиц эмпирических коэффициентов и подставить их в приведенные в справочнике формулы. Снижение трудоемкости этих действий и образует первую подцель.

Вторая задача состоит в том, чтобы разработать общую структуру модели, а также типизировать её содержимое. Путем параметризации может быть достигнуто эффективное повторное применение одной модели в рамках разработки различных технологических процессов. Используя внутренний расчетный механизм системы планируется обеспечить уменьшение затрат на подбор рациональных режимов резания.

После проведения расчетов требуется сделать вывод о допустимости принятых параметров обработки. Автоматизация построения диаграмм и графиков деформаций позволит повысить эффективность анализа результатов.

Дерево целей изображено на рисунке 2. Оно систематизирует вышесказанное в виде иерархической структуры.

4.2 Выбор языка программирования

Выбору языка программирования следует уделить особое внимание. От правильного решения существенно зависит насколько трудоемким будет процесс написания программы. Существующие языки программирования сильно различаются по возможностям, гибкости и области применения. На данном этапе необходимо рассмотреть основные варианты и выявить наилучший.

На стадии разработки концепции было выявлено, что следует опираться на подходы объектно-ориентированного программирования. Этот вывод соответствующим образом сужает область рассматриваемых решений. Для повышения пригодности продукта к обслуживанию также ограничимся общеизвестными и широко распространенными языками. Дополнительная

выгода от использования таких языков состоит в возможности применения готовых компонентов и библиотек.

Под указанные ограничения подходят такие языки как C++, C#, Java, Python. Сравним эти языки по существенным факторам методом непосредственной оценки. Для оценки будут приняты во внимание следующие характеристики: распространенность, скорость разработки, гибкость, безопасность. В рамках проекта производительность не рассматривается так как она не является ограничивающим фактором. По каждому критерию языку будет присвоен балл от 1 до 10. Наиболее предпочтительным будет признан язык с наибольшим суммарным баллом.

Распространенность может быть определена по числу проектов на информационном ресурсе github.com и числу заданных вопросов на stackoverflow.com. Эти ресурсы выбраны в качестве референтных из-за большой популярности в IT сообществе. Аналитическая фирма RedMonk опубликовала объективный рейтинг языков программирования в соответствии с указанными критериями [4]. Этот отчет использован в качестве основания для присвоения баллов, поэтому Java получает максимальный балл, а остальные языки оценены меньшим числом.

Скорость разработки в большей мере субъективный параметр, так как он зависит не только от конструкций языка, но и от умений и опыта конкретного программиста, хотя можно выявить и общие тенденции. Скорость разработки рассматривается с учетом замечаний, изложенных в [5]. Известно, что разработка на C++ требует больших временных затрат ввиду сложности конструкций и большого количества деталей, которые должны быть учтены программистом. По этому критерию предпочтение отдано Python, который относят к языкам сценариев. C# и Java в плане скорости разработки не выделяются и получили средний балл.

Гибкость также не поддается количественному определению, однако она зависит только от самого языка. Динамическая типизация позволила Python набрать максимальный балл. Также высоко оценены возможности ручного управления памятью в C++.

Таблица 1 — Оценка языков программирования

Фактор	C++	C#	Java	Python
Распространенность	9	9	10	9
Скорость разработки	8	9	9	10
Гибкость	10	9	9	10
Безопасность	8	10	10	8
Итого	36	37	38	37

Безопасность в текущем контексте следует понимать как вероятность возникновения не обрабатываемых ошибок во время выполнения программы. Ошибки такого типа негативным образом сказываются на качестве программного продукта и наиболее затратны для исправления. Низкий балл Python обусловлен уже упомянутой особенностью - динамической типизацией. В данном случае программист не имеет возможности воспользоваться описанием ошибок компиляции для устранения несоответствия типов переменных, это приводит к не обрабатываемым исключительным ситуациям уже во время выполнения. C++ также получил более низкую оценку ввиду общей сложности разработки и возможных ошибок при использовании адресной арифметики.

Согласно таблице 1 наиболее рациональным выбором является язык Java. Следует отметить, что оценка произведена с учетом требований к конкретному проекту и для других случаев на первый план могут выйти другие факторы.

4.3 Выбор модульной платформы

При описании модульной архитектуры в рамках концепции был оставлен открытым вопрос о средствах реализации. После выбора языка программирования можно вернуться к этому вопросу и проработать его более детально.

Многие эксперты в области программирования рекомендуют не разра-

батывать сложные компоненты, если уже существуют достойные библиотеки, содержащие необходимый функционал [6]. В случае разработки на Java достичь модульной компоновки приложения можно при помощи одного из двух фреймворков: NetBeans Platform или Eclipse RCP. Детальный анализ показывает, что оба продукта в большей степени предоставляют эквивалентные возможности. Тем не менее были выделены следующие существенные различия: платформа NetBeans использует библиотеку Swing в качестве инструмента реализации пользовательского интерфейса, в то время как в Eclipse используется SWT. Немного более предпочтительным выглядит Swing, так как она является стандартной для Java и поставляется в комплекте разработчика (JDK - Java Development Kit) и поэтому имеет большое количество всевозможных сторонних расширений. Второе отличие в том, что в Eclipse модульная система реализована в соответствии со стандартом OSGi (Open Service Gateway Initiative) в то время как в NetBeans используется специфичная модульная система, хотя документация NetBeans также заявляет о частичной поддержке OSGi [7]. В рамках проекта это отличие не представляется существенным.

В качестве программной платформы было принято использовать NetBeans. На это решение повлияло наличие большого числа доступных ресурсов для освоения платформы, таких как проекты с открытым исходным кодом и обучающие официальные материалы а также подробная документация.

4.4 Выбор системы моделирования

Принцип моделирования подразумевает использование системы МКЭ расчетов при вычислении значений деформаций. Наиболее подходящими для этой цели были признаны ANSYS и SIMULA Abaqus FEA. Обе программы предоставляют богатый функционал, позволяющий моделировать рассматриваемый процесс обработки, поэтому функциональные критерии отходят на второй план. И ANSYS и Abaqus отвечают трем сформулиро-

ванным ранее требованиям к системе моделирования.

Для выдачи расчетного задания в случае использования ANSYS имеется возможность организации взаимодействия через интерфейс командной строки. Модель может быть составлена на специальном предметно-ориентированным языке ANSYS Parametric Design Language (APDL). Однако, анализ официальной документации не выявил ни одного приемлемого способа передачи фактических значений параметров модели при вызове выполнения задания. Эта проблема может быть решена путем непосредственной замены значений в текстовом файле задания, но такой путь в большей степени подвержен вероятному возникновению ошибок, менее гибкий и более трудоемкий. Экспорт геометрии из ANSYS возможен только в единственном формате IGES. Хотя IGES широко используется в коммерческих CAD системах, для него не существует бесплатных Java библиотек визуализации, а реализация такой библиотеки (с учетом размера спецификации IGES около 700 страниц) не предоставляется возможной.

В плане возможностей интеграции с внешними системами выгодно отличается Abaqus. В Abaqus для описания модели используется язык общего назначения Python. При вызове расчетного задания через интерфейс командной строки есть возможность передачи значений параметров (например, геометрических размеров или силовых факторов). Также Abaqus поддерживает открытый формат OBJ для обмена геометрией. OBJ — это простой формат данных, для него существуют Java библиотеки, позволяющие отображать сохраненные модели.

Таким образом, для моделирования процесса будет использован Abaqus так как эта система более пригодна для взаимодействия с внешней информационной средой.

ГЛАВА 5
ЭСКИЗНОЕ ПРОЕКТИРОВАНИЕ

5.1 Определение функционала системы

На данном этапе необходимо определить какие функции должна выполнять разрабатываемая система. Для графического представления будет применена UML модель вариантов использования. Она описывает не только состав функций, но и их взаимосвязь с множеством действующих лиц.

Основная идея диаграммы вариантов использования состоит в отображении требований к системе через две базовые сущности - действующие лица и варианты использования. Пользователи и другие системы, которые могут взаимодействовать с продуктом называются действующими лицами, а вариант использования представляет собой спецификацию поведения [8]. В системе предварительно можно выделить три лица, взаимодействующих с системой.

Первое лицо представленно пользователем, который заинтересован в выполнении функций, связанных с получением значений деформаций заготовки и их анализом. Для этого пользователю должна быть предоставлена возможность загрузки описания расчетного случая (то есть модели) в систему. Перед запуском расчета необходимо применить заданные характеристики процесса к параметризованной модели. По результатам работы CAE системы должны быть получены графики и диаграммы деформаций. Также пользователь ожидает от системы выполнения вспомогательных функций, таких как расчет режимов резания.

Необходимость выделения такого лица как инженер по знаниям обусловлена результатами анализа литературы [3], который показал, что для расчета режимов резания по таблично-аналитической методике потребуется представление экспертных знаний в формальном виде. Решение этой задачи возлагается на инженера по знаниям, который должен иметь возможность корректировать содержимое базы знаний.

Загружаемая в систему модель должна быть предварительно подготов-

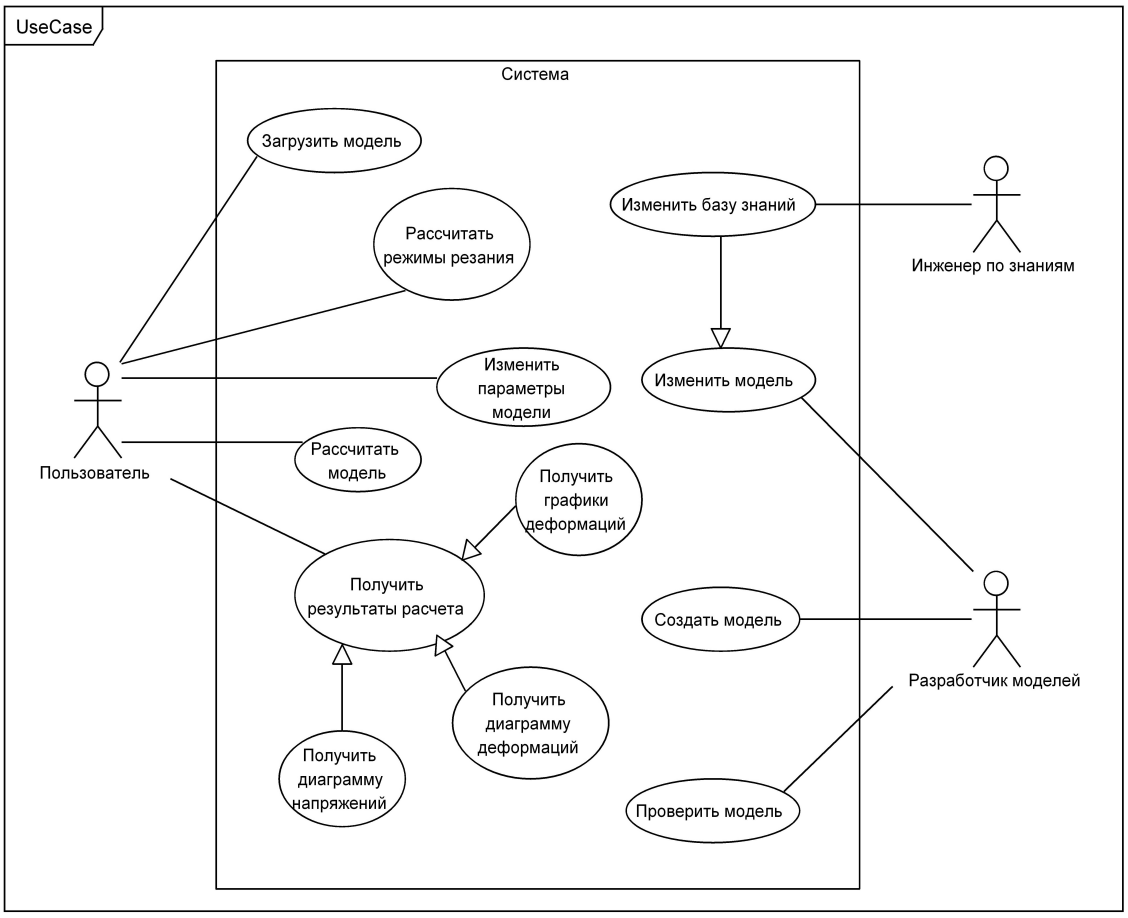


Рисунок 3 – Диаграмма вариантов использования

лена компетентным экспертом — разработчиком моделей. К обязанностям разработчика моделей относится также подготовка исходных данных в виде задания для выполнения CAE системой. После подготовки модели эксперт проверяет её работоспособность и при необходимости устраняет дефекты, после чего предоставляет пользователю.

Составленная диаграмма вариантов использования изображена на рисунке 3. Кроме описанных выше принципов на ней отображены зависимости обобщения между вариантами использования. Например, варианты использования “Получить диаграмму деформаций”, “Получить диаграмму напряжений” и “Получить графики деформаций” обобщены в “Получить результаты расчета”. Указанная диаграмма будет использована для выделения отдельных элементов системы в модули для последующей проработке.

ГЛАВА 6
ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКАЯ ЧАСТЬ

6.1 Введение

В данной главе рассматривается рассчет стоимости создания интегрированной системы поддержки принятия решения.

6.2 Организация и планирование процесса разработки

Разработка интегрированной системы поддержки принятия решения является сложным процессом, стадии которого определяются при помощи ГОСТа Единой Системы Программной Документации [9]. Согласно указанному документу всего выделяется 5 стадий, их подробное описание приведено в таблице **??**. **ДОБАВИТЬ ТАБЛИЦУ**

Основными факторами, определяющими трудоемкость создания интегрированной системы являются: степень новизны разрабатываемого программного комплекса, сложность алгоритма его функционирования, объем используемой информации, вид её представления и способ обработки, уровень языка программирования [10]. Определим принадлежность разрабатываемого продукта к группам по каждому из указанных факторов.

По степени новизны интегрированная система может быть отнесена к группе «Б»: она не имеет аналогов, но и не требует разработки принципиально новых методов создания.

По степени сложности систему можно отнести к группе продукции, реализующей оптимизационные и моделирующие алгоритмы. Хотя непосредственная реализация метода конечных элементов не входит в рамки разработки продукта, тем не менее в составе системы имеется модуль управления моделированием. Принимая во внимание необходимость создания модели процесса обработки можно заключить что программа относится к группе 1. К тому же дальнейшее развитие проекта предполагает создание модуля многокритериальной оптимизации.

Как показал анализ литературы [3], исходная информация представлена как в виде формул, так и в виде разнородных таблиц. При этом требуется учитывать взаимное влияние данных, содержащихся в этих таблицах. Таким образом, разрабатываемую программу по виду представления исходной информации относят к группе 11. По структуре выходных документов система принадлежит группе 22 - требуется вывод на печать одинаковых документов, представляющих собой отчеты о моделировании.

6.3 Расчет трудоемкости этапов

6.3.1 Расчет трудоемкости разработки технического задания

Трудоемкость разработки технического задания может быть расчитана по формуле:

$$T_{\text{ТЗ}} = T^{\text{З}}_{\text{РЗ}} + T^{\text{З}}_{\text{РП}}, \tag{1}$$

где $T_{\text{ТЗ}}$ – трудоемкость разработки технического задания на создание программного продукта, чел.-дни;

$T^{\text{З}}_{\text{РЗ}}$ – затраты времени разработчика постановки задачи на разработку ТЗ, чел.-дни;

$T^{\text{З}}_{\text{РП}}$ – затраты времени разработчика программного обеспечения на разработку ТЗ, чел.-дни.

Значения величин $T^{\text{З}}_{\text{РЗ}}$ и $T^{\text{З}}_{\text{РП}}$ расчитывают по формулам:

$$T^{\text{З}}_{\text{РЗ}} = t_{\text{З}} \cdot K^{\text{З}}_{\text{РЗ}}; \qquad T^{\text{З}}_{\text{РП}} = t_{\text{З}} \cdot K^{\text{З}}_{\text{РП}},$$

где $t_{\text{З}}$ – норма времени на разработку ТЗ на программный продукт, чел.-дни; $t_{\text{З}} = 67$ чел.-дни, согласно таблице 2, приведенной в [10]

$K^{\text{З}}_{\text{РЗ}}$ – коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком постановки задачи на стадии ТЗ; $K^{\text{З}}_{\text{РЗ}} = 0,65$ для случая совместной с разработчиком ПО разработки.

$K^{\text{З}}_{\text{РП}}$ – коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком программного обеспечения на стадии ТЗ;

$K^{З}_{РП} = 0,35$ для совместной с разработчиком постановки задачи.
После подстановки принятых значений в (1) получаем:

$$T_{ТЗ} = 67 \cdot 0,65 + 67 \cdot 0,35 = 67 \text{ чел.-дни}$$

6.3.2 Расчет трудоемкости выполнения эскизного проекта

Для расчета трудоемкости выполнения эскизного проекта (ЭП) используется формула:

$$T_{ЭП} = T^{Э}_{РЗ} + T^{Э}_{РП}, \quad (2)$$

где $T_{ЭП}$ – трудоемкость разработки эскизного проекта, чел.-дни;

$T^{Э}_{РЗ}$ – затраты времени разработчика постановки задачи на разработку ЭП, чел.-дни;

$T^{Э}_{РП}$ – затраты времени разработчика программного обеспечения на разработку ЭП, чел.-дни.
Для расчета значений $T^{Э}_{РЗ}$ и $T^{Э}_{РП}$ могут быть применены следующие формулы:

$$T^{Э}_{РЗ} = t_{Э} \cdot K^{Э}_{РЗ}; \qquad T^{Э}_{РП} = t_{Э} \cdot K^{Э}_{РП},$$

где $t_{Э}$ – норма времени на разработку ЭП программного продукта, чел.-дни; $t_{З} = 94$ чел.-дни, согласно таблице 3, приведенной в [10]

$K^{Э}_{РЗ}$ – коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком постановки задачи на стадии ЭП; $K^{Э}_{РЗ} = 0,7$ для случая совместной с разработчиком ПО разработки.

$K^{Э}_{РП}$ – коэффициент, учитывающий удельный вес трудоемкости работ, выполняемых разработчиком программного обеспечения на стадии ЭП; $K^{Э}_{РП} = 0,3$ для совместной с разработчиком постановки задачи разработки ЭП.

После подстановки принятых значений в (2) получаем:

$$T_{ТЗ} = 94 \cdot 0,7 + 94 \cdot 0,3 = 94 \text{ чел.-дни}$$

6.3.3 Расчет трудоемкости выполнения технического проекта

Трудоемкость разработки технического проекта $T_{ТП}$ зависит от функционального назначения программного продукта, количества разновидностей

форм входной и выходной информации и определяется по формуле:

$$T_{\text{ТП}} = (t^{\text{Т}}_{\text{РЗ}} + t^{\text{Т}}_{\text{РП}})K_{\text{В}} \cdot K_{\text{Р}} \tag{3}$$

где $t^{\text{Т}}_{\text{РЗ}}$, $t^{\text{Т}}_{\text{РП}}$ – норма времени, затрачиваемого на разработку ТП разработчиком постановки задач и разработчиком программного обеспечения соответственно, чел.-дни; по таблице 16 в [10] определено, что для принятой группы сложности алгоритма и степени новизны программной продукции нормы времени составляют $t^{\text{Т}}_{\text{РЗ}} = 89$ чел.-дни и $t^{\text{Т}}_{\text{РП}} = 64$ чел.-дни

$K_{\text{В}}$ – коэффициент учета вида используемой информации;

$K_{\text{Р}}$ – коэффициент учета режима обработки информации; по таблице 17 приведенной в [10] определено значение $K_{\text{Р}} = 1,45$.

Значение коэффициента $K_{\text{В}}$ определяют из выражения:

$$K_{\text{В}} = \frac{K_{\text{П}} \cdot n_{\text{П}} + K_{\text{НС}} \cdot n_{\text{НС}} + K_{\text{Б}} \cdot n_{\text{Б}}}{n_{\text{П}} + n_{\text{НС}} + n_{\text{Б}}} \tag{4}$$

где $K_{\text{П}}$, $K_{\text{НС}}$, $K_{\text{Б}}$ – значения коэффициентов учета вида информации для переменной, нормативно-справочной информации и баз данных соответственно. Согласно таблице 18 методических указаний [10] для группы новизны «Б»: $K_{\text{П}} = 1,20$; $K_{\text{НС}} = 1,08$; $K_{\text{Б}} = 3,12$.

$n_{\text{П}}$, $n_{\text{НС}}$, $n_{\text{Б}}$ – количество наборов данных переменной, нормативно-справочной информации и баз данных соответственно. Для разрабатываемой интегрированной системы $n_{\text{П}} = 2$; $n_{\text{НС}} = 1$; $n_{\text{Б}} = 0$.

После подстановки определенных значений коэффициентов учета вида информации и количества наборов данных в формулу (4) получаем:

$$K_{\text{В}} = \frac{1,20 \cdot 2 + 1,08 \cdot 1 + 3,12 \cdot 0}{2 + 1 + 0} = 1,16$$

Для рассчитанного значения $K_{\text{В}}$ по формуле (3) получаем значение трудоемкости выполнения технического проекта:

$$T_{\text{ТП}} = (89 + 64) \cdot 1,16 \cdot 1,45 = 257 \text{ чел.-дни}$$

6.3.4 Расчет трудоемкости выполнения рабочего проекта

Трудоемкость разработки рабочего проекта зависит от функционального назначения программного продукта, количества разновидностей форм входной и выходной информации, сложности алгоритма функционирования, степени использования готовых программных модулей и уровня алгоритмического языка программирования и определяется по формуле:

$$T_{\text{РП}} = K_{\text{К}} \cdot K_{\text{Р}} \cdot K_{\text{Я}} \cdot K_{\text{З}} \cdot K_{\text{ИА}}(t^{\text{Р}}_{\text{РЗ}} + t^{\text{Р}}_{\text{РП}}) \tag{5}$$

где $K_{\text{К}}$ – коэффициент учета сложности контроля информации; $K_{\text{К}} = 1,07$ в соответствии с таблицей 19 методических указаний [10].

$K_{\text{Р}}$ – коэффициент учета режима обработки информации; ранее определено что $K_{\text{Р}} = 1,45$.

$K_{\text{Я}}$ – коэффициент учета уровня используемого языка программирования; $K_{\text{Я}} = 1$ для языка высокого уровня.

$K_{\text{З}}$ – коэффициент учета степени использования готовых программных модулей; с учетом применения готового расчетного ядра МКЭ моделирования и большого количества стандартных библиотек $K_{\text{З}} = 0,6$.

$K_{\text{ИА}}$ – коэффициент учета вида используемой информации и сложности алгоритма;

Значение $K_{\text{ИА}}$ вычисляется по формуле:

$$K_{\text{ИА}} = \frac{K'_{\text{П}} n_{\text{П}} + K'_{\text{НС}} n_{\text{НС}} + K'_{\text{Б}} n_{\text{Б}}}{n_{\text{П}} + n_{\text{НС}} + n_{\text{Б}}} \tag{6}$$

где $K'_{\text{П}}$, $K'_{\text{НС}}$, $K'_{\text{Б}}$ – значения коэффициентов учета сложности алгоритма программного продукта и вида используемой информации для переменной, нормативно-справочной информации и баз данных соответственно; согласно таблице 22 методических указаний [10] для группы новизны «Б» и группы 1 сложности алгоритма $K'_{\text{П}} = 1,62$, $K'_{\text{НС}} = 0,97$, $K'_{\text{Б}} = 0,81$.

$t^{\text{Р}}_{\text{РЗ}}$, $t^{\text{Р}}_{\text{РП}}$ – норма времени затраченного на разработку программы на языке высокого уровня разработчиком постановки задач и разработчиком программного обеспечения соответственно, чел.-дни; ранее определено что

трудоемкость выполнения технического проекта для разработчика постановки задач $t^{\text{Т}}_{\text{РЗ}} = 89$ чел.-дни, тогда по таблице 35а $t^{\text{Р}}_{\text{РЗ}} = 107$ чел.-дни, а трудоемкость выполнения технического проекта разработчиком программного обеспечения $t^{\text{Т}}_{\text{РП}} = 64$ чел.-дни, следовательно $t^{\text{Р}}_{\text{РП}} = 683$ чел.-дни. Подставим табличные значения в формулу (6):

$$K_{\text{ИА}} = \frac{1,62 \cdot 2 + 0,97 \cdot 1 + 0,81 \cdot 0}{2 + 1 + 0} = 1,40$$

Тогда трудоемкость разработки рабочего проекта по формуле (5):

$$T_{\text{РП}} = 1,07 \cdot 1,45 \cdot 1 \cdot 0,6 \cdot 1,40(107 + 683) = 735 \text{ чел.-дни}$$

6.3.5 Расчет трудоемкости стадии внедрения

Трудоемкость выполнения стадии “Внедрение” $T_{\text{В}}$ может быть расчитана по формуле

$$T_{\text{В}} = (t^{\text{В}}_{\text{РЗ}} + t^{\text{В}}_{\text{РП}})K_{\text{К}} \cdot K_{\text{Р}} \cdot K_{\text{З}} \tag{7}$$

где $t^{\text{В}}_{\text{РЗ}}$, $t^{\text{В}}_{\text{РП}}$ – норма времени, затрачиваемого разработчиком постановки задачи и разработчиком программного обеспечения соответственно на выполнение процедур внедрения, чел.-дни; исходя из таблицы 48 $t^{\text{В}}_{\text{РЗ}} = 33$ чел.-дни, $t^{\text{В}}_{\text{РП}} = 98$ чел.-дни.

При подстановке норм времени и коэффициентов учета в формулу (7) получаем:

$$T_{\text{В}} = (33 + 98)1,07 \cdot 1,45 \cdot 0,6 = 122 \text{ чел.-дни}$$

СПИСОК ЛИТЕРАТУРЫ

[1] Метод определения условий механической обработки тонкостенных деталей. / Гаврюшин С.С., Жаргалова А.Д., Лазаренко Г.П. [и др.] // Известия высших учебных заведений. Машиностроение. 2015. № 11. С. 53–60. DOI: 10.18698/0536-1044-2015-11-53-61.

[2] Шилдт Г. Java. Полное руководство. М.: «Вильямс», 2012. 1104 с.

[3] А.Г. Косилова, Р.К. Мещеряков. Справочник технолога-машиностроителя. В 2-х т. М.: Машиностроение, 1986. Т. 2. 418 с.

[4] The RedMonk Programming Language Rankings: June 2015. URL: http://redmonk.com/sogrady/2015/07/01/language-rankings-6-15.

[5] Choosing Static vs. Dynamic Languages for Your Startup - DZone Web Dev. URL: https://dzone.com/articles/should-your-static-go-static.

[6] Bloch J. Effective Java. Addison-Wesley, 2008. 346 p.

[7] Böck H. The Define Guide to NetBeans Platform 7. Apress, 2012. 558 p.

[8] Object Management Group. Unified Modeling Language. March 2015. Version 2.5. URL: http://www.omg.org/spec/UML/2.5.

[9] ГОСТ 19.102-77. ЕСПД. Стадии разработки.

[10] В.В. Арсеньев, Ю.Б. Сажин. Методические указания к выполнению огранизационно-экономической части дипломных проектов по созданию программной продукции. Издательство МГТУ, 1994. 52 с.