УДК 004.94

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ КАК СРЕДСТВО ОПТИМИЗАЦИИ ПРОЦЕССА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Груданов Н.А.
аспирант,
НИЯУ МИФИ
Москва, Россия

Зоричев В.В.
магистрант,
НИЯУ МИФИ,
Москва, Россия

Груданова А.А.
магистрант,
НИЯУ МИФИ,
Москва, Россия

Аннотация

В статье проведено исследование подхода к оптимизации процесса разработки программного обеспечения посредством имитационного моделирования, в целях снижения стоимости разработки и сокращения сроков выполнения задач.

В статье рассмотрен структурный анализ процесса разработки ПО и составлено его описание с установленными особенностями подхода и определенным набором параметров. На основе описания процесса построена его имитационная модель с использованием программного обеспечения AnyLogic, встроенные функции которого позволяют решать задачи оптимизации параметров. В результате выполнена одна из возможных задач оптимизации с целью сокращения срока выполнения задач, затрат на разработку и сокращения простоя сотрудников. Успешное выполнение оптимизационного эксперимента на основе построенной модели и заданных

ограничений позволяет применять данную модель для оптимизации работы отдела разработки ПО.

Ключевые слова: имитационное моделирование, оптимизация, AnyLogic, моделирование, минимизация расходов, внутренний контроль.

SIMULATION MODELING AS MEANS OF OPTIMIZING THE SOFTWARE DEVELOPMENT PROCESS

Grudanov N.A.
graduate student,
NRNU MEPhI
Moscow, Russia
Zorichev V.V.
master's student,
NRNU MEPhI,
Moscow, Russia
Grudanova A.A.
master's student,
NRNU MEPhI,
Moscow, Russia

Abstract

The article studies an approach to optimizing the software development process through simulation modeling in order to reduce development costs and reduce task completion times.

The article discusses a structural analysis of the software development process and compiles its description with established features of the approach and a certain set of parameters. Based on the description of the process, a simulation model was built using AnyLogic software, the built-in functions of which allow solving parameters optimization problems. As a result, one of the possible optimization tasks was

completed in order to reduce task completion time, development costs and reduce employee downtime. Successful execution of an optimization experiment based on the constructed model and specified constraints allows the use of this model to optimize the work of the software development department.

Keywords: simulation modeling, optimization, AnyLogic, modeling, cost minimization, internal control.

Традиционный подход к выполнению проектов по разработке и поддержке ПО является сложной структурой, представляющую собой комплексную работу различных отделов компании. В ходе реализации проектов участвуют менеджеры проектов, дизайнеры, тестировщики, непосредственно сами разработчики ПО и другие сотрудники. Сложность оптимизации полного цикла разработки ПО, включающего в себя все отделы, состоит в различии структуры в зависимости от конкретного проекта, выбранной методологии управления проектом и других особенностях каждой отдельной компании. В связи с этим рассмотрим подход к оптимизации работы отдела разработки ПО, который является основополагающим отделом для всего процесса.

Рассмотрим функционирование отдела разработки независимо от работы других отделов. В таком случае работу отдела можно представить в качестве сервиса, принимающего на вход поставленные задачи и отдающий на выход выполненные задачи. Сама структура процесса в таком виде не зависит от выполняемого проекта и других внешних факторов, а основными входными параметрами процесса является интенсивность поступления задач, их трудоемкость и сложность [3]. Другие же параметры процесса заключаются внутри самого отдела.

Отдел разработки ПО представляет собой некоторый набор сотрудников разного уровня: team lead, senior-разработчики, middle-разработчики, junior-разработчики и стажеры. Каждый уровень характеризуется следующими параметрами:

1. Количество сотрудников уровня;

2. Заработная плата — почасовой ставкой оплаты труда сотрудников уровня;

3. Сложность задач — максимальная сложность задачи, которую способен выполнить сотрудник данного уровня;

4. Производительность — число выполняемых задач максимальной сложности сотрудниками данного уровня за единицу времени.

В результате процесс работы отдела разработки ПО можно описать в виде набора шагов следующим образом:

1. На вход процесса поступают задачи с некоторой заданной интенсивностью;

2. В зависимости от уровня сложности задача поступает в очередь на выполнение в один из потоков, связанных с соответствующим задаче уровнем разработчиков. Так задачи уровня сложности не выше установленной сложности задач стажеров поступает в очередь задач для стажеров, задачи уровня сложности выше установленной сложности задач стажеров и не выше установленной сложности задач junior-разработчиков поступает в очередь задач для junior-разработчиков и так далее;

3. При наличии в соответствующем пуле свободных разработчиков задача из очереди принимается к исполнению одним разработчиком из пула и выполняется в соответствии с установленной трудоемкостью. При этом в случае отсутствия свободных разработчиков в пуле соответствующего уровня

задача может быть выполнена свободным разработчиком из пула выше не более чем на один уровень, то есть задачи стажеров могут выполнить junior-разработчики, но не выше;

4. В результате выполнения задачи сотрудник возвращается в пул свободных сотрудников и ожидает следующую задачу.

Таким образом, в ходе выполнения задач отделом разработки возможно вычислить выходные параметры процесса, включающие стоимость разработки, время простоя сотрудников, а также время задержки выполнения задач.

В качестве средства имитационного моделирования описанного процесса использовалось программное обеспечение AnyLogic, имеющая удобный графический интерфейс и предоставляющая необходимый набор функций, в том числе для решения задач оптимизации [4].

В результате на основании описанной структуры построена имитационная модель функционирования отдела разработки ПО, представленная на рисунке 1.

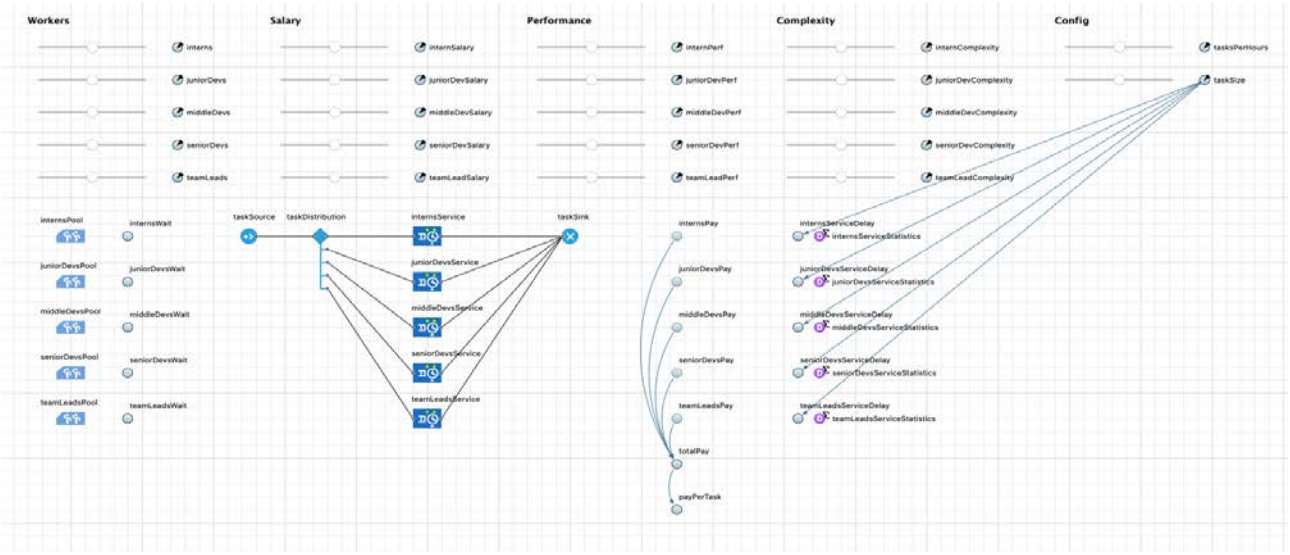

Рисунок 1 — Имитационная модель функционирования отдела разработки ПО

Построенная модель состоит из множества элементов, составляющих единую модель работы отдела разработки ПО, и содержит входные и выходные параметры, а также элементы самого процесса разработки.

Модель содержит 22 входных параметра, которые с помощью слайдера устанавливаются в зависимости от соответствующих составляющих для каждой отдельной компании или проектов и логически разделяются на 5 блоков:

1. *Workers* — блок параметров, определяющих количество сотрудников отдела каждого уровня: *interns*, *juniorDevs*, *middleDevs*, *seniorDevs*, *teamLeads*;

2. *Salary* — блок параметров, определяющих почасовую заработную плату сотрудников отдела каждого уровня: *internSalary*, *juniorDevSalary*, *middleDevSalary*, *seniorDevSalary*, *teamLeadSalary*;

3. *Performance* — блок параметров, определяющих производительность сотрудников отдела каждого уровня: *internPerf*, *juniorDevPerf*, *middleDevPerf*, *seniorDevPerf*, *teamLeadPerf*. Может принимать значения от 0 до 1;

4. *Complexity* — блок параметров, определяющих допустимую сложность задач сотрудников отдела каждого уровня: *internComplexity*, *juniorDevComplexity*, *middleDevComplexity*, *seniorDevComplexity*, *teamLeadComplexity*. Может принимать значения от 1 до 10.

5. *Config* — блок параметров, характеризующих поступающие задачи, где *taskPerHours* — интенсивность поступления задач (число задач в час), *taskSize* — средняя трудоемкость задач (человеко-часов).

Процесс разработки от постановки задачи до ее реализации отделом разработки состоит из пула сотрудников каждого уровня и 4 ключевых этапов:

1. С заданной интенсивностью в параметре *taskPerHours* задачи с помощью элемента *taskSource* типа *Source* поступают в отдел разработки;

2. Элемент *taskDistribution* типа *SelectOutput5* распределяет задач по 5 потокам в зависимости от сложности задачи. Так задачи поступают в поток, соответствующий самому нижнему уровню разработчиков, которые способны выполнить задачи соответствующего уровня;

3. Элементы типа *Service* для каждого из потоков получают на вход задачи и ставят их в очередь. При наличии сотрудника (ресурса) в соответствующем пуле сотрудников (или на уровень выше) сервис получает задачу из очереди и захватывает сотрудника, которые задерживаются в нем на время выполнение задачи равное значению параметра *taskSize*. По истечению времени задержки задача переходит к следующему этапу, а сотрудник освобождается и возвращается в пул.

4. В результате выполнения задачи в сервисе элемент *taskSink* тип *Sink* поглощает задачи и выводит их из процесса, фиксируя количество выполненных задач.

В результате работы процесса на выходе из модели получаем 17 параметров, представляющих собой динамические переменные, изменяющиеся в ходе выполнения процесса и делящиеся на 3 логических блока:

1. *Pay* — блок параметров, определяющих суммарную стоимость выполнения задач сотрудниками отдела каждого уровня: *interns*, *juniorDevs*, *middleDevs*, *seniorDevs*, *teamLeads* (вычисляются как произведение числа задач выполненных разработчиками соответствующего уровня, среднего времени выполнения задачи и почасовой заработной платы сотрудников), а также суммарную стоимость затрат на разработку всего объема выполненных задач

totalPay (сумма по всем уровням) и среднюю стоимость одной выполненной задачи *payPerTask* (отношение *totalPay* к общему числу реализованных задач);

2. *Wait* — блок параметров, определяющих среднюю долю времени простоя сотрудников соответствующего пула: *internsWait*, *juniorDevsWait*, *middleDevsWait*, *seniorDevsWait*, *teamLeadsWait* (вычисляется из средней загрузки соответствующего пула);

3. *Delay* — блок параметров, определяющих задержку задач в очереди на исполнение: *internsWait*, *juniorDevsWait*, *middleDevsWait*, *seniorDevsWait*, *teamLeadsWait* (вычисляется как среднее значение числа задач в очереди сервиса из соответствующего потока).

Полученная модель содержит большое число входных и выходных параметров, характеризующих процесс разработки ПО. В целях повышения эффективности работы отдела разработки ПО необходимо выполнить оптимизацию данного процесса, что возможно осуществить с помощью проведения оптимизационного эксперимента в AnyLogic.

Для проведения такого эксперимента необходимо выбрать оптимизационные параметры, задать целевую функцию и ограничения на значения переменных. Так одним из вариантов использования данной модели может быть решение задачи поиска оптимального числа сотрудников каждого уровня с установленной заработной платой для достижения минимальной стоимости разработки, наименьшей задержки задач в очереди и сокращения времени простоя сотрудников.

В таком случае в качестве оптимизационных параметров указываются значения входных параметров *interns*, *juniorDevs*, *middleDevs*, *seniorDevs*, *teamLeads*. Целевая функция, которую требуется минимизировать, имеет следующий вид (1):

$$(1 + payPerTask) * \\ * (1 + internsServiceDelay + juniorDevsServiceDelay + \\ + middleDevsServiceDelay + seniorDevsServiceDelay + \\ + teamLeadsServiceDelay), \quad (1)$$

где первый множитель отвечает за минимизацию стоимости разработки, а второй множитель — за минимизацию задержки, а единица в множителях прибавляется для устранения случаев, при которых минимальное значение одного из множителей может быть равно нулю, например, при отсутствии задержек при неограниченном числе сотрудников. При этом необходимо также учитывать, что сотрудников должно быть достаточно для выполнения задач, но при этом сотрудники не должны оставаться без работы. Для этого установлены ограничения на значения переменных: *internsWait < 0.5* (так как стажеры зачастую совмещают работу с учебой и работают на половину ставки), *juniorDevsWait < 0.25* (junior-разработчики тоже часто совмещают работу и учебу, но уже реже чем стажеры), а *middleDevsWait*, seniorDevsWait и *teamLeadsWait* меньше 0.1, в которое включены отпускные дни сотрудников и возможные выходные.

В результате запуска оптимизационного эксперимента с генетическим оптимизатором и установленными параметрами удается найти оптимальные значения входных параметров, при котором достигается наилучший результат работы модели. Предложенный вариант оптимизации не единственный и может быть изменен в рамках решаемой задачи, однако показывает возможность его выполнения на основе построенной имитационной модели.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Кабаева И. И. Имитационное моделирование // International scientific review. -2016. - № 13 (23). - С. 19-20.

2. Антонова Г. М., Цвиркун А. Д. Оптимизационно-имитационное моделирование для решения проблем оптимизации современных сложных производственных систем // Проблемы управления. – 2005. - № 5. - С. 19-27.

3. Шумков Е. А., Видовский Л. А. Определение трудоемкости задач и оценка эффективности управления проектом // Научный журнал КубГАУ. – 2016. - № 124. - С. 425-434.

4. Документация AnyLogic // The AnyLogic Company [Электронный ресурс]. — Режим доступа: https://anylogic.help/.

Оригинальность 79%